Міністерство освіти і науки України
Національний технічний університет України «КПІ ім. І. Сікорського»

ВІСНИК

НАЦІОНАЛЬНОГО ТЕХНІЧНОГО УНІВЕРСИТЕТА УКРАЇНИ «КПІ»

Інформатика, управління та обчислювальна техніка

Заснований у 1964 р.
Випуск 64

Київ “ВЕК+” 2016

УДК 004

Вісник НТУУ «КПІ». Інформатика, управління та обчислювальна техніка: Зб. наук. пр. – К.:Век+, – 2016. – № 64. – 174 с.

Рекомендований до друку Вченою радою факультету інформатики та обчислювальної техніки, протокол № 10 від 23.05.2016

Головний редактор: Луцький Г. М.

Заст. гол. ред.: Стіренко С. Г.
Відповідальний секретар: Коган А. В.

Редакційна колегія: Павлов О. А.,
Теленик С. Ф.,
Бузовський О. В.,
Симоненко В. П.,
Жабін В. І.,
Кулаков Ю. О.,
Марковський О. П.,
Стенін Н. А.,
Томашевський В. М.,
Ляшевський С. Е.,
Гордієнко Ю. Г.

Описано результати дослідження і створення компонентів обчислювальних й інформаційних систем і комплексів, пристроїв автоматики та передавання даних, систем автоматизації програмування, контролю й діагностики, штучного інтелекту тощо.

Для аспірантів, студентів, фахівців з обчислювальної техніки, систем керування, автоматизації програмування, штучного інтелекту та інших інформаційно-обчислювальних систем.

Журнал індексується наступними наукометричними базами даних: Directory of Open Access Journals (DOAJ), Російський індекс наукового цитування (РІНЦ), Google Scholar.

Журнал входить до переліку наукових фахових видань МОН України.

ISSN2310-3620

Свідоцтво про державну реєстрацію № 16949-5719 Р від 17.06.2010

Збірник наукових праць українською, англійською та російською мовами
Web-ресурс – http://it-visnyk.kpi.ua

Підп. до друку 23.05.2016. Формат 60×84 1/8. Гарнітура Times. Папір офсетний № 1.
Наклад 150 прим.
Надруковано в ЗАТ “ВІПОЛ”, 03151 м.Київ, вул. Волинська, 60.

© Національний технічний університет України “КПІ”, 2016

ЗМІСТ

Бабенко О. А., Гіоргізова-Гай В.Ш. Балансування навантаження додатків у приватному «хмарному» середовищі…….. 4

Винничук С. Д., Максименко Е. В. Многократное прореживание для ускорения метода факторизации Ферма при неравномерных шагах для неизвестной…… 13

Куц В. Ю. Структура вычислителя для многошкальных фазовых измерителей на основе модулярной арифметики……. 25

Русанова О. В. Спосіб планування обчислень для мультиядерних кластерів…….. 31

Гломозда Д. К. Застосування методу інваріантів до аналізу кольорових мереж Петрі із дедлоками……... 38

Сергієнко А. М., Клименко І. А.,Сергієнко П. А. Реконфігурована багатопроцесорна обчислювальна система на ПЛІС…… 47

Костенко Ю. В., Марковский А .П., Русанова О .В. Метод защищенного модулярного экспоненцирования на удаленных компьютерных системах…… 51

Гергало А. М., Рудакова Г. В. Проблеми автоматизації процесу функціональної діагностики цифрових пристроїв обліку електричної енергії……...… 55

Марковський О. П., Семенюк Ю. В., Шевченко О. М., Радченко К. О. Метод корекції помилок передачі даних в одночіпових радіомодемах……..... 60

Рыбак В. А., Шокр Ахмад Методы и средства для анализа и прогноза эколого-экономических показателей ресурсоёмких инновационных проектов……... 68

Заковоротный А. Ю. Расширение возможностей геометрический теории управления за счет автоматизации аналитических преобразований в пакете MATLAB……..….. 76

Порєв В. М. Алгоритмічні аспекти автоматизованого проектування навчальних планів…….…… 84

Теленик С. Ф., Ролик А. И., Жариков Э. В. Управление распределением виртуальных машин в ЦОД…..... 90

Новотарський М. А., Бронфман Є. Б. Моделювання руху рідини в тривимірній області з рухомими границями……... 100

Долголенко А. О., Яцун В. О. Реалізація операційного пристрою суматора/віднімача з плаваючою крапкою для ядра суперскалярного процесора……. 106

Папуч Т. Ю. Інтеграція методів аналізу банкрутства та моделей оцінки кредитоспроможності юридичних осіб……..... 117

Буйбарова М. Ф., Виноградов Ю. М., Приймак В. Ю. Метод захищеної реалізації перетворень Фур'є на віддалених розподілених комп'ютерних системах…… 125

Гусев Е. И. Моделирование способов организации доступа к распределённым страницам памяти в системах облачных вычислений основанных на shared everything архитектуре…….. 133

Мурга М. О. Простір зі змінним базисом приналежностей ознакам для прогнозування показників фінансово-економічних процесів……..… 138

Диброва М. А., Коган А. В., Капорин Р. Н. Способы организации многопутевой маршрутизации……... 145

Жабин В. И., Жабина В. В., Скориченко А. В. Реализация неавтономных вычислений в избыточных системах счисления на ПЛИС……..…. 150

Ганжа І. М., Шемсединов Т. Г., Алєнін О. І., Стіренко С. Г. Модифікація та застосування алгоритму маркування зв'язних компонент на бінарному зображенні……...…… 156

Марковський О. П., Захаріудакіс Лефтерис., Федотов М. Ф. Метод строгої ідентифікації віддалених абонентів на основі стандартизованих шифроблоків та хеш-перетворень……..… 161

Теленик С. Ф., Троцький С. О Використання компонентного підходу при проектуванні інтелектуальних адаптивних сервісів……. 166

Відомості про авторів……..……...……..…. 173

Summary……..……..……..... 174

УДК 004.75

БАБЕНКО О. А.,
ГІОРГІЗОВА-ГАЙ В.Ш.

БАЛАНСУВАННЯ НАВАНТАЖЕННЯ ДОДАТКІВ У ПРИВАТНОМУ «ХМАРНОМУ» СЕРЕДОВИЩІ

У даній статті проведено порівняльну характеристику розповсюджених систем балансування навантаження Web додатків у відповідності до критеріїв, які враховують особливості роботи самих додатків і особливості інфраструктур обчислювальних комплексів, на яких вони розгортаються. Надано практичні рекомендації з налаштування системи балансування навантаження у «хмарній» платформі CloudStack на прикладі розгортання додатку дистанційного відео навчання. Приведено результати дослідження ефективності створених правил балансування.

This article shows comparative characteristics of the widespread load balancing systems for Web applications in accordance with criteria that take into account the characteristics of applications and computer systems infrastructure on which they are deployed. Practical recommendations for configuring the CloudStack platform load balancing system are given for example of remote video teaching application. The effectiveness of the created balancing rules investigated.

Ключові слова: балансування навантаження, Web додатки, CloudStack, хмарні технології

Вступ

Сьогодні все більше установ і організацій прагнуть створити власні Web сервіси як у мережі Інтернет так для корпоративних потреб на базі приватних або публічних «хмарних» інфраструктур. Оскільки такі «хмарні» системи належать до високонавантажених, то можливості масштабування і балансування навантаження при їх побудові мають особливе значення. Від вибору і налаштування системи балансування навантаження, які б найкращим чином відповідали задачам створення конкретного сервісу в конкретних умовах, напряму залежить ефективність роботи сервісу і безперебійність надання послуг.

Мета статті

Метою даної статті є надання практичних рекомендацій з вибору систем балансування навантаження Web додатків на основі критеріїв, які враховують особливості роботи самих додатків і особливості інфраструктур обчислювальних комплексів, на яких вони розгортаються, а також – з налаштування системи балансування навантаження у відкритій «хмарній» платформі CloudStack на прикладі організації сервісу дистанційного відео навчання. Для цього у статті проведено порівняльну характеристику ряду розповсюджених систем балансування навантаження Web додатків у відповідності до обраних критеріїв, та результати дослідження ефективності правил балансування, створених для «хмарного» сервісу.

Порівняльна характеристика розповсюджених систем балансування навантаження

Балансування (вирівнювання) навантаження, (англ. Load balancing) – метод розподілу завдань між декількома мережевими пристроями (наприклад, серверами) з метою оптимізації використання ресурсів, скорочення часу обслуговування запитів, горизонтального масштабування кластера (динамічне додавання / видалення пристроїв), а також забезпечення відмовостійкості (резервування).

При балансуванні навантаження «хмарних» додатків слід враховувати особливості роботи додатків, особливості інфраструктури обчислювальних комплексів і додаткові вимоги при організації сервісів.

Для порівняння можливостей систем балансування навантаження додатків в "хмарних" середовищах визначимо основні критерії такого порівняння.

Ефективність розподілу навантаження – характеризує можливості системи забезпечити рівномірність розподілу навантаження між обслуговуючими вузлами при різних вимогах до роботи додатків і до структур обчислювальних комплексів, які можуть бути враховані через наявні в сис-

темі алгоритми балансування навантаження і тип балансування (статичне і динамічне).

Основною перевагою статичного розподілу, який не змінюється у процесі роботи сервісу, є простота реалізації. Динамічне балансування є більш гнучким і передбачає перерозподіл навантаження на обчислювальні вузли під час роботи сервісу на підставі визначення прямих показників (завантаження центрального процесора, обсяг доступної оперативної пам'яті, доступного дискового простору тощо), або непрямих (кількість активних підключень до сервера чи час його відгуку).

Врахування гео-положення вузлів – характеризує можливість системи забезпечити зниження часу її відгуку через врахування географічної близькості певних груп клієнтів до певних груп обслуговуючих вузлів. Потрібно зауважити, що ефективність методу дуже залежить від актуальності даних про довжину маршрутів, пропускну спроможність і завантаження ліній зв'язку між потенційними клієнтами і серверами.

Незалежність від типу мережі – незалежність системи балансування від того, у локальній чи у глобальній мережі знаходяться обслуговуючі сервери, а також незалежність від типу серверної платформи.

Можливості автоматичного масштабування – характеризує наявність в системі механізмів горизонтального масштабування обслуговуючих вузлів (динамічне додавання/видалення) при балансуванні навантаження. Автомасштабування в «хмарних» платформах забезпечується спеціальними модулями, зв'язаними з системами балансування навантаження, які безпосередньо цю задачу не вирішують.

Автоматичне визначення і видалення непрацюючих вузлів – забезпечення системою балансування відмово стійкості сервісу на основі моніторингу доступності і працездатності його компонентів та автоматичного видалення непрацюючих вузлів з пулу розподілу навантаження.

Забезпечення довгострокових сесій – характеризує можливість системи балансування забезпечити додатку зберігання стану сесії користувача з декількох повторних запитів. Може реалізуватися в системі або через з'єднання користувача з одним і тим самим сервером, або через зберігання сесії в загальному для всіх серверів сховищі. Прив'язка клієнта до конкретного сервера зазвичай реалізується або за його IP-адресою (яка, однак, може змінюватись через певний час), або по cookies (в який записується ідентифікатор сервера).

Незалежність від прикладного протоколу – характеризує незалежність методу балансування навантаження від використовуваного протоколу прикладного рівня до будь-яких TCP- сервісів: HTTP, HTTPS, SMTP, IMAP, SSH, FTP, SQL і т.д. Це досягається за рахунок балансування запитів на 2-4 рівнях моделі OSI. Найчастіше використовується балансування на 4-му рівні OSI (IP- адреси + номери портів TCP і UDP), як найбільш інформативне і гнучке.

Контентно залежний аналіз, кешування – характеризує можливості системи забезпечити балансування за видом контенту по різних серверах (наприклад, до статичних сторінок, до динамічних сторінок, до медіа серверів і т.д.). Оскільки така можливість з'являється тільки при балансуванні на 7-му рівні моделі OSI за допомогою кешуючих проксі серверів, то можливості системи балансування розширюються за рахунок кешування відповідей, часто самостійної обробки HTTPS трафіку, стискання даних, що знижує навантаження обслуговуючих серверів, аналізу запитів для підвищення безпеки сервісу і т.д. До недоліків методу є високе споживання ресурсів і залежність від прикладного протоколу (в даному випадку розглядається тільки протокол HTTP).

Таким чином, при виборі системи балансування навантаження потрібно чітко розуміти задачі, які вона повинна вирішувати для конкретного сервісу, і наявність в ній засобів, які будуть для цього ефективними.

У в таблиці 1 наведено порівняльну характеристику ряду розповсюджених систем балансування навантаження.

Табл. 1. Порівняльна характеристика розповсюджених систем балансування навантаження

	Google Compute Engine	**Microsoft Azure Load Balancer / LoadMaster**	**Amazon EC2 Elastic Load Balancing**	**HAProxy**	**LVS**	**Nginx**	**Cloud-Stack**
Ефективнісь розподілу навантаження	+	-/+	+	+	+	+	+
Врахування гео-положення вузлів	+	-/+	+	-	+	-	+

	Google Compute Engine	Microsoft Azure Load Balancer / LoadMaster	Amazon EC2 Elastic Load Balancing	HAProxy	LVS	Nginx	Cloud-Stack
Можливості автомасштабування	+	+	+	-	-	-	+
Незалежність від прикладного протоколу (4-й рівень OSI)	+	+	+	+	+	-	+
Контентно залежне балансування, аналіз, кешування (7-й рівень OSI)	+	-/+	+	+	-	+	-
Забезпечення довгострокових сесій	+	-/+	+	+	+	+	+
Незалежність від типу мережі	+	+	+	+	+	-	+
Автоматичне визначення і видалення непрацюючих вузлів	+	+	+	+	-+	+	+

Система балансування Elastic Load Balancing, яка використовується в Amazon EC2 [1] є найбільш функціональною з розглянутих систем, в ній є все те, що зазвичай вимагається від системи балансування: автомасштабування, відмовостійкість, контроль за доступністю обслуговуючих серверів, простота використання, гнучкість, надійність, безпека, можливість міграції завдань, врахування явних параметрів завантаження обслуговуючих серверів. За рахунок великого рівня автоматизації, людське втручання в роботу системи зведено до мінімуму.

Система балансування Google Compute Engine [2] також має достатньо повний набір функціональних можливостей. Хоча система є менш гнучкою і універсальною, а також гірше документованою, у порівнянні з системою Amazon EC2, проте, її перевагами є більша простота використання і більш висока швидкодія програмно-апаратного рішення.

На цьому фоні дивує бідність можливостей вбудованої системи балансування Azure Load Balancer [3], хоча більшість її обмежень сьогодні дозволяє подолати рішення від Kemp Technologies LoadMaster-for-Azure [4]. Система LoadMaster забезпечує повноцінне балансування навантаження на 4-7 рівні моделі OSI для робочих навантажень в «хмарному» середовищі Azure, перевірку працездатності додатків і прискорення SSL, а також додаткові послуги, такі як запобігання вторгнень, кешування і стиснення даних для опублікованих служб. LoadMaster оптимізує потік трафіку для множинних віртуальних служб, багатокомпонентних додатків, розробка і розгортання яких в Microsoft Azure є перевагами цієї «хмарної» платформи так само як і висока продуктивність масштабних обчислень. Об'єкти LoadMaster, розміщені в хмарі Microsoft Azure і в середовищах локальної приватної хмари, працюють спільно для забезпечення безперервності доставки послуг на міжхмарному рівні.

Система HAProxy це безкоштовне, дуже швидке, ефективне (в плані використання процесора і оперативної пам'яті) і надійне рішення, що пропонує високу доступність і балансування навантаження для розподілених TCP і HTTP-додатків. Система HAProxy також є складовою частиною «хмарної» платформи Red Hat OpenShift і балансувальником по замовчанню у частиною «хмарній» платформі OpenStack [5].

Система Linux Virtual Server (LVS) - це безкоштовний популярний засіб управління кластерними системами для Linux. LVS дозволяє створити кластер з фізичних (або віртуальних) серверів, що розподіляє навантаження між машинами в залежності від їх стану, пріоритету та інших параметрів настроювання завдяки великій кількості алгоритмів балансування [6]. Система підтримує балансування навантаження на 4-му рівні моделі OSI, балансування на 7-му рівні на сьогодні не допрацьовано. До недоліків LVS можна віднести складність налаштування, необхідність встановлення додаткових пакетів для забезпечення високої готовності сервісу, відмінності в налаштуваннях для різних версій Linux. Найбільш докладно описано налаштування для RedHat Enterprise Linux Редакция 6, наприклад, у [7].

Система Nginx – це популярне безкоштовне ПЗ, що реалізує HTTP-сервер і зворотний проксі сервер, поштовий проксі-сервер, а також TCP проксі-сервер загального призначення [8]. Nginx забезпечує всі переваги, які надають

HTTP проксі сервери, і використовувати його доцільно при необхідності детального настроювання всіх, навіть найдрібніших аспектів балансування HTTP і HTTPS трафіку. Системам HAProxy та LVS Nginx програє по швидкодії та використанню ресурсів на балансування.

Вбудована система балансування відкритої «хмарної» платформи CloudStack 4 забезпечує досить повний набір функціонально можливостей, хоча і реалізує балансування навантаження тільки на 4-му рівні. Це обмеження, при необхідності, легко можна подолати, якщо прийняти до уваги, що платформа CloudStack підтримує зовнішні система балансування навантаження. В їх якості можна використовувати як комерційні балансувальники від Google, Microsoft, Amazon, Rackspace, NetScaler, так і відкриті, такі як HAProxy, LVS. Сьогодні CloudStack – досить потужна та надійна платформа з простим інтуїтивно зрозумілим інтерфейсом, яка багато чого запозичила у Amazon EC2, і дозволяє організувати роботу як публічного IaaS-сервісу, так і приватної «хмарної» інфраструктури [9].

Створення правил балансування в системі CloudStack

В якості прикладу розглянемо організацію сервісу дистанційного відео навчання для студентів заочної форми навчання і другої післядипломної освіти, активність яких сильно змінюється протягом періоду навчання. Сервіс розгортається на локальних серверах у приватному «хмарному» середовищі кафедри СП ІПСА КПІ на базі відкритого програмного забезпечення BigBlueButton [10].

Система BigBlueButton підтримує наявність декількох аудіодоріжок і обмін відео, можливість показу презентацій, документів Microsoft Office і OpenOffice, зображень, PDF документів, підтримуються розширені можливості дошки. Для зворотного зв'язку зі слухачами веб-конференції існують публічні та приватні чати. Сервер BigBlueButton може працювати в хмарному середовищі, як Amazon EC2, при його встановленні під ОС Ubuntu 14.04 64-бітної версії.

Встановлення сервісу складається з наступних кроків, детальний опис яких виходить за рамки даної статті:

- Створення шаблону з Ubuntu 14.04 64 біт;
- Створення віртуальної машини з шаблону і налаштування її;
- Встановлення сервера BigBlueButton на віртуальну машину;
- Створення шаблону з готовою віртуальну машину;
- Створення другої копії віртуальної машини.

На основі вимог додатку і у відповідності до умов розгортання сервісу сформулюємо правила для системи балансування, яка буде розподіляти трафік, отриманий на відкритій IP- адресі сервісу, на дві віртуальні машини. При створенні правила необхідно визначити алгоритм балансування, політику липкості, задати необхідні параметри і присвоїти правило множині віртуальних машин. Також треба визначити політику перевірки доступності серверів і авто масштабування.

Вибір алгоритму. Серед Round Robin (рівномірний статичний розподіл), Source Ip Hash (розподіл клієнтів по серверах на основі близькості) і Least Connection (динамічний розподіл на сервери з найменшою кількістю з’єднань) вибираємо алгоритм Least Connection, так як всі сервери знаходяться у одній локальній мережі, а при роботі з додатком передбачається зберігання стану сесій користувачів, що може призвести до нерівномірного навантаження при використанні статичних алгоритмів.

Політика липкості. Ця політика потрібна для ідентифікації користувача при повторних з’єднаннях і пошуку збережених ним даних на тому ж самому сервері. З двох методів – за cookie (згенерованим системою балансування або додатком) або IP-адресою джерела, вибираємо за IP-адресою користувача. Оскільки не передбачається сесія довше одного робочого дня, то у переважній більшості випадків динамічна IP-адреса користувача за цей час не зміниться.

Оскільки у період активного використання сервісу потрібно забезпечити його високу готовність, задаємо перевірку «стану здоров'я» серверів з опитуванням їх доступності кожні 10 секунд.

Налаштуємо політику авто масштабування, оскільки передбачається, що активність користувачів буди сильно змінюватись у часі. Масштабування вгору будемо здійснювати, коли використання CPU або RAM серверів почне перевищувати 80%, а вниз – коли використання CPU або RAM буде менше ніж 20%.

Для додавання правила балансування навантаження виконаємо наступні кроки:

1. У лівій панелі навігації графічного інтерфейсу CloudStack вибираємо вкладку "Мережа".

2. Натискаємо на назву нашої мережі, де ми хочемо балансувати трафік. Назва - default.

3. Натискаємо кнопку "Перегляд IP-адрес".

4. Натискаємо IP-адресу, для якої хочемо створити правило (192.168.1.125), і вибираємо вкладку "Конфігурація".

5. У вузлі балансування навантаження, натискаємо "Показати всі".

У базовій зоні, також можна створити правило балансування навантаження без вибору IP-адреси. CloudStack сам призначить IP- адреси серверам при створенні правила балансування навантаження, які будуть вказані на сторінці IP-адреси створення правила. Щоб зробити це, треба вибрати ім'я мережі, а потім натиснути вкладку "Додати балансувальник навантаження".

6. У вікні, що відкриється (рис. 1), потрібно заповнити наступні поля:

- Ім'я (Name): web;
- Публічний порт (Public Port): 80;
- Приватний Порт (Private Port): 80;
- Алгоритм (Algorithm): Least connection;
- Липкість (Stickiness): Метод SourceBased.

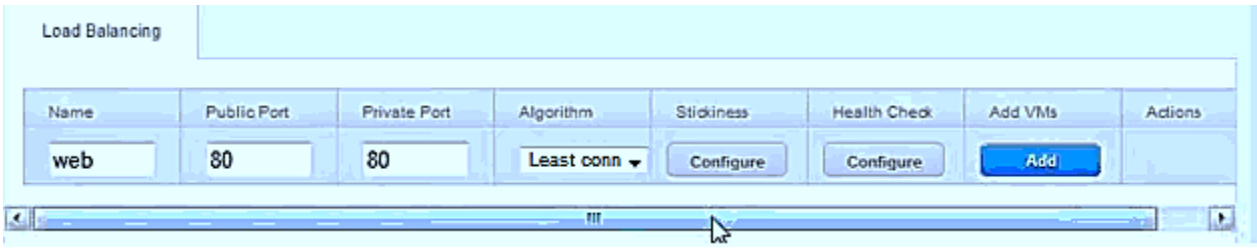


Рис. 1. Налаштування правила балансування CloudStack

7. На вкладниці Stickiness, потрібно задати метод політики липкості (load balancer-generated cookie, application-generated cookie, чи source-based), ім'я політики та додаткові параметри в залежності від методу (розмір таблиці IP адрес (200 тис. записів), строк їх дії (50 годин)) (рис. 2).

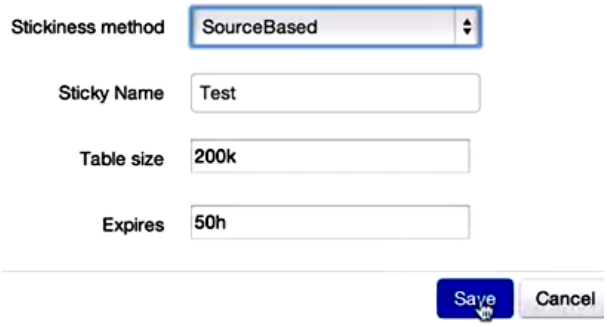


Рис. 2. Вікно конфігурації Stickiness CloudStack

8. На вкладниці Health Check (Перевірка здоров'я), натискаємо кнопку "Налаштування" (рис. 3) і заповняємо визначені нами параметри перевірки доступності серверів:

- Ping path: Послідовність адрес, до яких слід відправити запити перевірки. Залишаємо значення за замовчуванням: / (all).
- Response time: Час очікування відповіді від адрес. Задамо 10 сек.
- Interval time: Час між перевірками здоров'я. Задамо 10 секунд.
- Healthy threshold: Число послідовних успіхів перевірок здоров'я, які необхідні перш ніж оголосити екземпляр здоровим. Задамо 1.
- Unhealthy threshold: Число послідовних невдач перевірки здоров'я, які необхідні, перш ніж оголосити екземпляр нездоровим. Задамо 1.

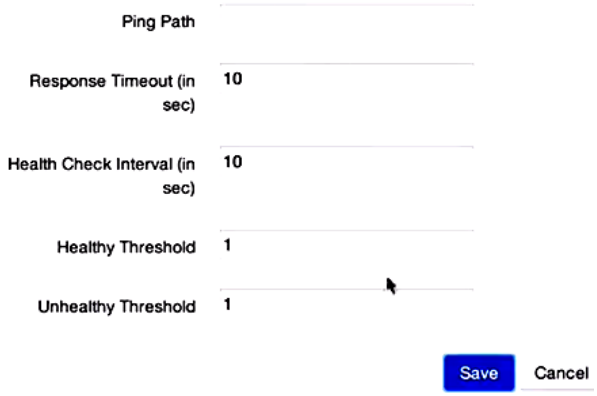


Рис. 3. Вікно конфігурації перевірки здоров'я CloudStack

9. Створюємо правила автомасштабування.

Налаштуємо масштабування вгору при використанні CPU або RAM більше 80% та масштабування вниз при використанні CPU або RAM менше ніж на 20% (рис. 4). Заповнюємо наступні поля:

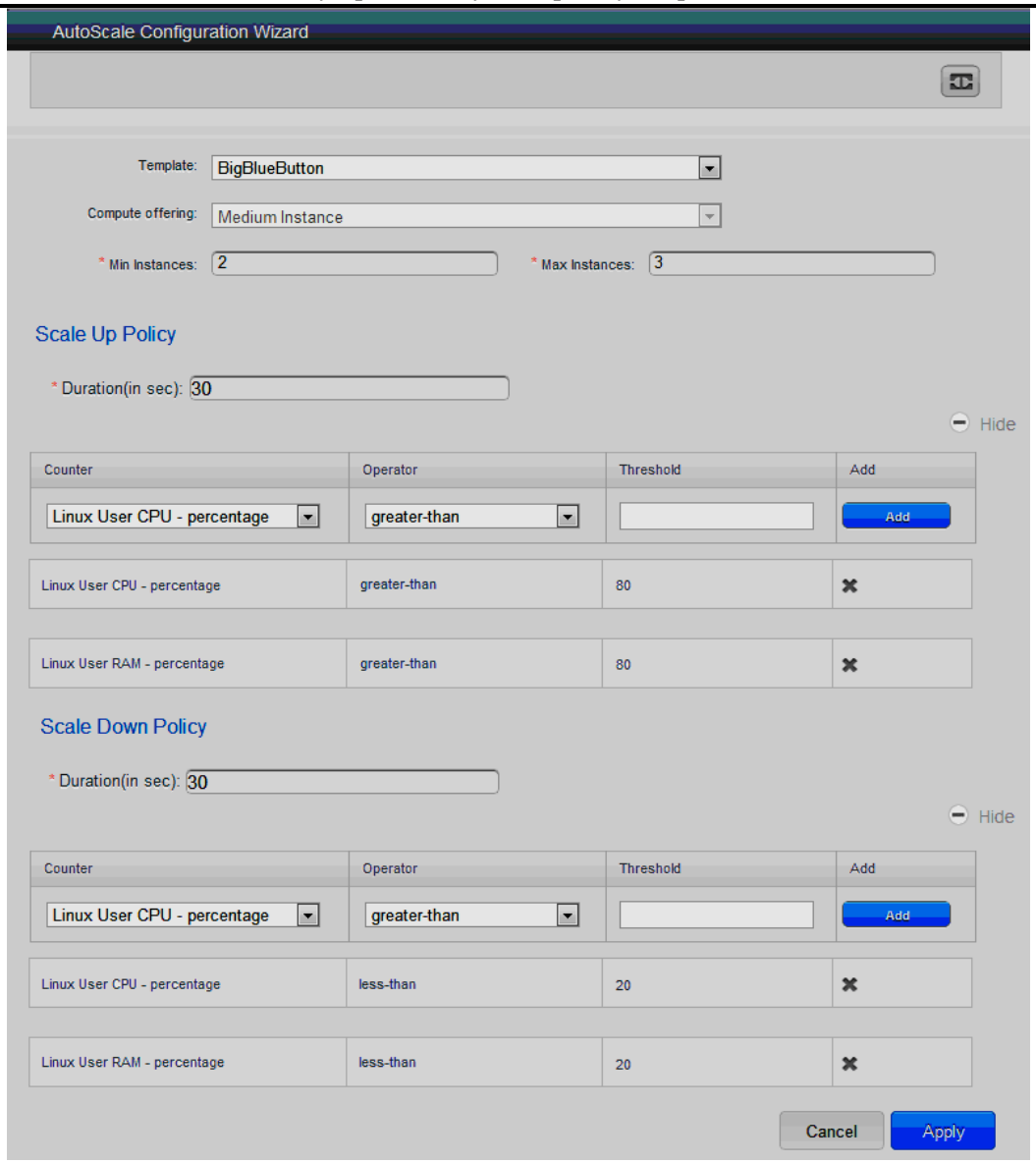


Рис. 4. Правила автомасштабування CloudStack

10. Натискаємо кнопку "Додати віртуальні машини", а потім вибираємо наші дві віртуальні машини зі встановленим BigBlueButton, які будуть розділяти навантаження вхідного трафіку, і натискаємо кнопку "Застосувати".

У вікні на рис. 5 бачимо, що нове правило балансування навантаження з'явилося в списку.

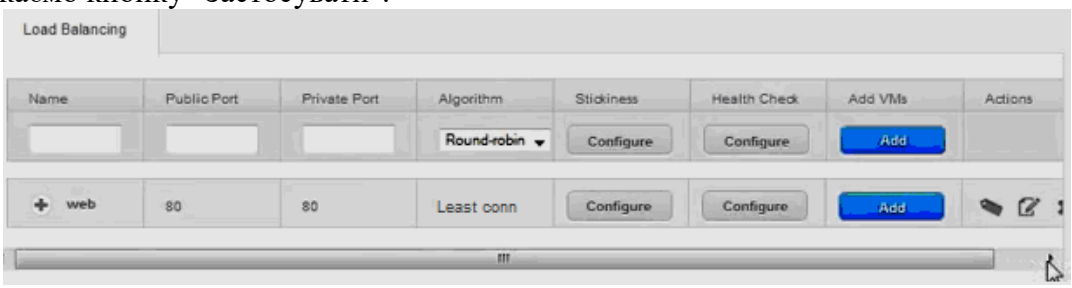


Рис. 5. Список створених правил балансування CloudStack

Навантажувальне тестування

Основна мета навантажувального тестування полягає в тому, щоб, створивши певне очікуване в системі навантаження (наприклад, за допомогою віртуальних користувачів) і використавши ідентичне програмне і апаратне забезпе-

чення, спостерігати за показниками продуктивності системи.

В якості засобу для навантажувального тестування було обрано WAPT (Web Application Performance Testing), який є засобом навантажувального і стресового тестування і надає можливості для детального та ефективного тестування веб-сайтів і Інтернет додатків з веб-інтерфейсом [11].

Продукт WAPT може емулювати типову активність тисяч користувачів, що працюють з сайтом одночасно. Він наділений низкою спеціальних можливостей, які дозволяють тестувати RIA-додатки з динамічними даними, що змінюються прямо під час тесту. WAPT підтримує всі типи авторизації на сервері. Результати тестування представляються у наочній формі у вигляді звітів і графіків, які дозволять проаналізувати характеристики продуктивності сайту під навантаженням різних типів і рівнів, виявити і усунути вузькі місця у роботі сайту, оптимізувати конфігурацію обладнання та програмного забезпечення.

Для налаштування тестування в WAPT нам необхідно:

- створити сценарій тесту за допомогою внутрішнього браузера програми, який записує дії користувача у веб-додатку;
- налаштувати кількість користувачів та час роботи тестування;
- додати два наших сервера в засіб моніторингу;
- додати агент для тестування.

При кожному тестуванні будемо імітувати роботу користувача (логін, переміщення в інтерфейсі системи) з використанням веб-камери і мікрофону.

В першому тесті перевіримо ефективність балансування. Для нього задамо в налаштуваннях тестування кількість користувачів, що збільшується від 0 до 40 з додаванням по 2 користувача кожні 10 секунд. Результати можна побачити на рис. 6 (для першої віртуальної машини) та на рис. 7 (для другої віртуальної машини).

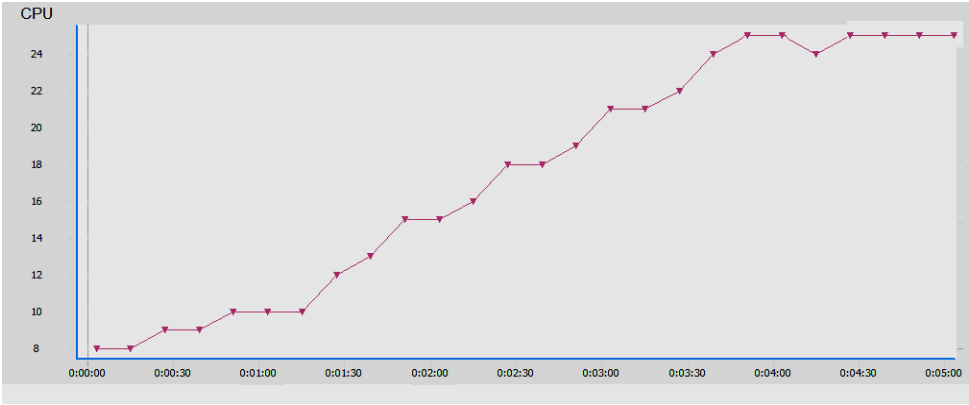


Рис. 6. Графік навантаження на CPU для першої віртуальної машини

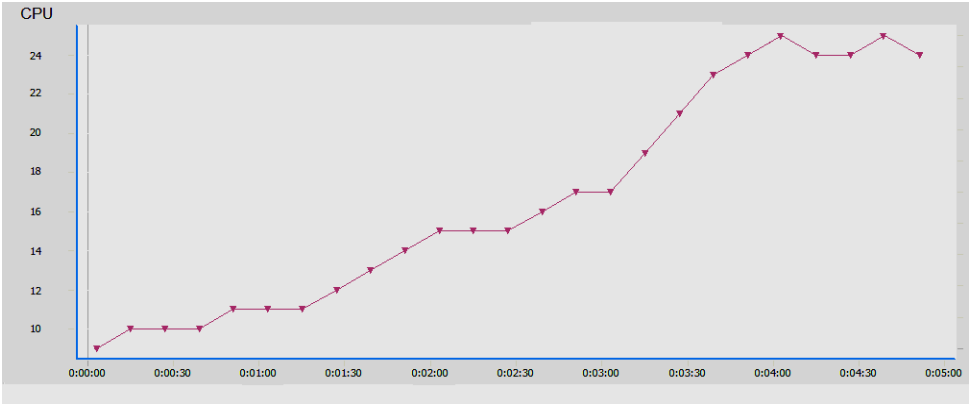


Рис. 7. Графік навантаження на CPU для другої віртуальної машини

З результатів тестування видно, що система балансування справляється з навантаженням і досить рівномірно розподіляє навантаження на обидві віртуальної машини. В момент часу 0:30:20 надходять останні користувачі, після чого навантаження вирівнюється. Продивившись логи CloudStack, бачимо, що на кожну машину прийшло по 20 користувачів.

Далі проведемо тестування роботи засобу автомасштабування CloudStack. Для цього зробимо два сценарії тесту, які будуть виконуватись послідовно. В першому сценарії кількість користувачів буде збільшуватися від 0 до 160 з додаванням 2 користувачів кожні 6 секунд. В другому – ті ж користувачі, створені в першому сценарії будуть завершувати сесію і їх кількість буде зменшуватись від 160 до 0 по 2 користувача кожні 6 секунд. Результати наведені на рис. 8 (для першого сценарію) і на рис. 9 (для другого сценарію).

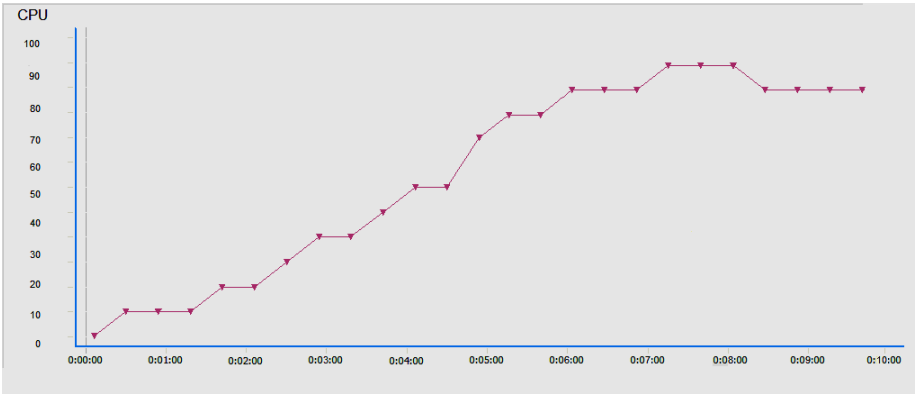


Рис. 8. Графік навантаження на CPU першої віртуальної машини при виконанні першої половини сценарію

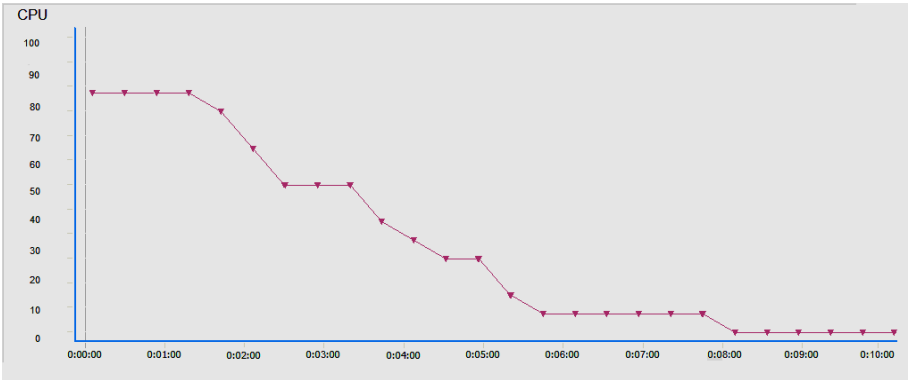


Рис. 9. Графік навантаження на CPU першої віртуальної машини при виконанні другої половини сценарію

Продивившись логи CloudStack, бачимо, що на наші дві машини прийшло по 69 користувачів. В момент часу 00:05:43 навантаження на CPU перевищило 80%, після чого через 30 секунд в CloudStack було ініційоване створення нової віртуальної машини, яка почала працювати в момент часу 00:06:35. Новий список віртуальних машин можна побачити на рис. 10.

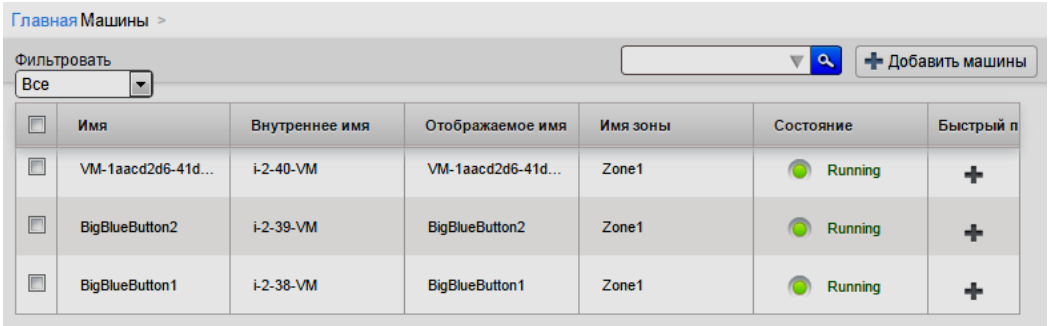


Рис. 10. Список віртуальної машини після активації засобу автомасштабування в CloudStack

Всіх наступних користувачів, відповідно до алгоритму Least Connections, балансувальник відправляв на нову машину. Кількість користувачів на ній досягла 22. Як можна побачити з

графіка (рис. 8) після створення нової машини, навантаження на першій машині вирівнюється.

Наступна фаза тестування почалась після моменту часу 00:10:00 (з онуленням таймера). Під час тесту навантаження на першій машині спадає, поки всі сесії користувачів не будуть завершені, а потім вирівнюється. В момент часу 00:05:16 навантаження на CPU стало нижче 20%, після чого через 30 секунд було ініційоване знищення третьої віртуальної машини (коли всі сесії її користувачів були завершені). Список віртуальних машин після завершення тестування знову складається з двох машин, які були створені вручну (рис. 11).

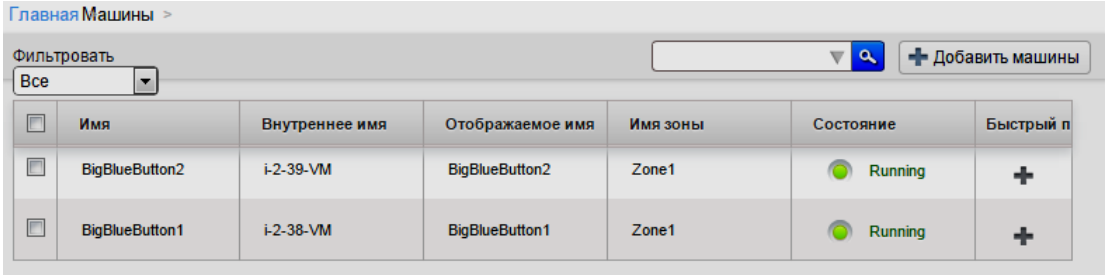


Рис. 11. Список ВМ CloudStack після завершення тестування

Висновки

Таким чином, результати навантажувального тестування розгорнутого «хмарного» сервісу показали ефективність створених правил балансування та роботи засобу автомасштабування. Система CloudStack показала себе як досить надійна та потужна платформа для створення приватної "хмари" корпоративного рівня та балансування навантаження всередині неї.

Результати проведених досліджень можуть бути корисні для вибору і налаштування систем балансування навантаження при створенні "хмарних" сервісів корпоративного рівня.

Список посилань

1. AWS | Elastic Load Balancing– сетевой балансировщик нагрузки в облаке. – Режим доступу: http://aws.amazon.com/ru/elasticloadbalancing/ – Дата доступу: 04.05.2015
2. Load Balancing – Compute Engine – Google Cloud Platform. – Режим доступу: https://cloud.google.com/compute/docs/load-balancing/ – Дата доступу : 04.05.2015
3. Azure Load Balancer overview | Microsoft Azure. – Режим доступу: https://azure.microsoft.com/ru-ru/documentation/articles/load-balancer-overview/ – Дата доступу: 04.05.2015
4. LoadMaster For Azure. – Режим доступу: http://kemptechnologies.com/solutions/microsoft-load-balancing/loadmaster-azure/ – Дата доступу: 04.05.2015
5. Load Balance OpenStack API – RDO. – Режим доступу: https://www.rdoproject.org/Load_Balance_OpenStack_API – Дата доступу: 04.05.2015
6. LVS Introduction – Load Balancing Server Cluster. – Режим доступу: http://www.linuxvirtualserver.org/whatis.html – Дата доступу: 05.05.2015
7. Комплект распределения нагрузки для Red Hat Enterprise Linux Редакция 6 – Режим доступу: http://adm-lib.ru/books/sergey/RHEL/docs/Virtual_Server-6.pdf - Дата доступу: 25.06.2015
8. nginx. – Режим доступу: http://nginx.org/ru/ – Дата доступу: 05.05.2015
9. Apache CloudStack 4 – News.stfw.ru. – Режим доступу: http://news.stfw.ru/716-apache-cloudstack-4.html – Дата доступу: 05.05.2015
10. BigBlueButton. – Режим доступу: http://docs.bigbluebutton.org – Дата доступу: 02.05.2015
11. WAPT – Инструмент для Нагрузочного Тестирования Сайтов и Веб-приложений. – Режим доступу: http://www.loadtesting.ru/product.shtml – Дата доступу: 26.05.2015

УДК 511:003.26.09

ВИННИЧУК С. Д.,
МАКСИМЕНКО Е. В.

МНОГОКРАТНОЕ ПРОРЕЖИВАНИЕ ДЛЯ УСКОРЕНИЯ МЕТОДА ФАКТОРИЗАЦИИ ФЕРМА ПРИ НЕРАВНОМЕРНЫХ ШАГАХ ДЛЯ НЕИЗВЕСТНОЙ

Предложен способ ускорения метода Ферма факторизации чисел за счет прореживания пробных значений неизвестной X при использовании множества оснований модуля, в котором существенная роль принадлежит выделяемому базовому основанию модуля bb. Базовое основание выбирается из условия максимума отношения bb к числу тех из Xmodbb, для которых разница $X^2 – N$ является квадратичным остатком по модулю bb (допустимые Xmodbb) и доступной оперативной памяти ЭВМ. На основании bb формируется список шагов переменной длины между ближайшими по величине допустимыми Xmodbb. Описан алгоритм MP, реализующий прореживание пробных значений. Для наиболее часто выполняемых шагов алгоритма MP представлен фрагмент программного кода на языке C. Представлены результаты численных экспериментов по разложению чисел на множители, являющихся произведением двух простых таких, что отношение большего к меньшему не превосходит 4. Показано, что при использовании только процедуры прореживания пробных значений временные затраты при решении задачи разложения на множители всегда пропорциональны числу $(p+q)/2 – N^{1/2}$, где N=pq. Они существенно зависят от выбора базового основания модуля и множества других оснований.

The way for accelerating Fermat's method of numbers factorization was offered due to the decimation of trial values of unknown X using a set of the bases of the module in which the essential role belongs to the allocated basic basis of the bb module. The basic basis is selected from the condition of maximum bb relationship to the number of those Xmodbb, for which the difference X^2 - N is a quadratic residue by mod bb (permissible Xmodbb) and available RAM of computer. Based on bb the list of steps of variable length between the the nearest largest permissible Xmodbb is formed. The algorithm of MR realizing the decimation of trial values is described. For the most frequently performed steps of the algorithm of MR the fragment of a program code in language C is presented. The results of numerical experiments for the decomposition of numbers on the factors which are the product of two primes such that the ratio of highest to lowest does not exceed 4. It is shown that using only the decimation procedure of trial values, the time expenses in solving the problem of decomposition into factors is always proportional to the number $(p + q)/2 - N^{1/2}$, where N = pq. They significantly depend on the choice of the basic basis of the module and a set of other bases of modules.

Ключевые слова: факторизация, метод Ферма, прореживание, ускорение.

Вступление

В информационно-телекоммуникационных системах для целей криптографической защиты информации применяется RSA алгоритм, что вызывает интерес к его криптоанализу. В работе [1] показано, что известные примеры компрометации RSA алгоритма не являются эффективней задачи факторизации. Основные используемые методы решения решения задачи факторизации представлены в [2]. Известно, что идеи классического алгоритма Ферма положены в основу таких современных методов факторизации как метод квадратического решета и решета числового поля. Поэтому можно предположить, что усовершенствования метода Ферма факторизации чисел может послужить основой для ускорения и других методов факторизации. Поэтому разработка способов ускорения метода Ферма факторизации чисел вида N = pq представляется актуальной. Одному из таких подходов, связанному с прореживанием пробных значений с неравномерным шагом при использовании множества оснований модуля, посвящено данное исследование.

1. Постановка задачи

Пусть задано составное нечетное число N=pq, которое следует разложить на множители, где p и q некоторые нечетные числа, не обязательно являющиеся простыми. Согласно исходному варианту метода Ферма для определения p и q решают уравнение

$$X^2 = N + Y^2, \qquad (1)$$

где x и y – целые положительные числа.

В уравнении (1) неизвестную X представим в виде

$$X = (\lfloor\sqrt{N}\rfloor + 1) + x = x_0 + x. \quad (2)$$

Тогда решение уравнения (1) получают перебором значений x = 0, 1, 2, …, до тех пор, пока остаток $X^2 - N$ не окажется полным квадратом целого числа, т.е.

$$X^2 - N = Y^2, \quad (3)$$

где $N = X^2 - Y^2 = (X - Y)\cdot(X + Y) = pq$, а переменные X и Y определяются через p и q согласно соотношений $X = (p+q)/2$ и $Y = |p - q|/2$. Поскольку число операций в методе Ферма пропорционально числу значений x, то временная сложность алгоритма метода пропорциональна величине $x = X - x_0 \approx (p+q)/2 - \sqrt{N}$, из-за чего при малых значениях множителя p (например, p=3, 5) с ростом N число x будет расти пропорционально N, т.е. временная сложность метода Ферма $T(N) = O(N)$. Хотя при $Y^2 \le 2X$ решение уравнения (2) получается уже при x = 0, т.е. метод Ферма эффективен при близких значениях p и q. На идее использование близости множителей числа основан метод, предложенный в работе [3], где использовано два метода разложения: метод пробного деления на простые числа до $\sqrt[3]{N}$, а для больших делителей N используется метод Ферма для чисел kN. Временная сложность такого комплексного метода $T(N) = O(N^{1/3})$.

Алгоритм метода Ферма можно интерпретировать как цикл, в котором перебираются значения x от 0 до $(p+q)/2 - \lfloor\sqrt{N}\rfloor - 1$, определяется разность $X^2 - N$ и из нее извлекается корень. Процесс заканчивается тогда, когда корень является целым числом. Для ускорения поиска корня на основании решения уравнения (3) предлагались ряд способов, описанных в работе [4].

Первый их них (алгоритм C) предполагает использование только операций сложение и вычитание, что исключает необходимость выполнения операций умножения и извлечения корня, которые являются более сложными в вычислительном смысле. Но при большом их количестве вычислительная сложность алгоритма C оказывается слишком большой.

Другой способ ускорения состоит в том, чтобы как можно меньше использовать извлечение квадратного корня. Как отмечается в [4], еще Ферма использовал для этого сравнение последних цифр разности $X^2 - N$ с возможными квадратичными остатками по модулю 100. Дальнейшее ускорение на основе такого подхода возможно при сравнении остатков $(X^2 - N) \bmod m_i (i = 1 \div r)$ с квадратичными остатками для соответствующего основания модуля, для чего формируется таблица просеивания. Такой подход принято называть методом решета (алгоритм D согласно [4]). В соответствии с алгоритмом D извлечение корня будет только тогда, когда для всех оснований модуля m_i остатки $(X^2 - N) \bmod m_i (i = 1 \div r)$ являются квадратичными остатками. Данный подход допускает ускорение, если существует основание модуля, для которого допустимые X меняются с равномерным шагом. Так, например, для основания модуля 3 при $N \bmod 3 = 2$ должно выполняться условие $X \bmod 3 = 0$. Тогда шаг по X можно выбрать равным 3. При наличии аналогичных соотношений для других оснований модуля процесс может еще ускорится.

Третий способ состоит в том, что по таблице просеивания можно сформировать конечный набор слов и использовать их для анализа. На этой идее были разработаны ряд машин. Машина, построенная в 1995 году, могла анализировать $6.144 \cdot 10^9$ чисел в секунду.

Назовем допустимыми те значения x, при которых разность $X^2 - N$ может быть полным квадратом согласно некоторых сформированных условий, и недопустимыми остальные. А процесс исключения недопустимых значений x назовем прореживанием.

Особенностью всех перечисленных выше способов ускорения для метода Ферма является то, что анализируются X на предмет того, являются ли они допустимыми, осуществляется с некоторым постоянным шагом. Идея использования при прореживании только тех X, для которых остаток от деления на некоторое основание модуля разности $X^2 - N$ также будет квадратичным остатком при использовании одного основания модуля предложена в работе [5]. В этом случае определено значение только среднего ускорения метода в зависимости от делителей основания модуля. В работе [6] получены оценки только для величины среднего ускорения в случае использования двух оснований модуля. Но не рассматриваются случаи использования множества оснований и не описан алгоритм их применения при решении уравнения (3). Нет также

количественных оценок ускорения, не совпадающих со средним значением. Такие задачи решаются в настоящей статье.

2. Прореживание возможных X для метода Ферма при одном фиксированном основании модуля b

Если выполняется соотношение (3), то для произвольного b имеет место равенство $\left(X^2 - N\right)\bmod b = Y^2 \bmod b$, что эквивалентно выполнению соотношения

$$\left((X \bmod b)^2 \bmod b - N \bmod b\right)\bmod b = \\ = (Y \bmod b)^2 \bmod b \quad (4)$$

Обратное неверно, т.е. из выполнения (4) не следует выполнение (3). Но **если** не выполняется соотношение (4), **то** не будет выполняться и соотношение (3). Поэтому те X, при которых не выполняется (4), исключаются из рассмотрения. В этом суть прореживания.

Обозначим через M2(b) – множество квадратичных остатков по модулю b. Тогда выполнение условия (4) означает, что существует квадратичный остаток $w \in M2(b)$, что

$$(w - N \bmod b)\bmod b \in M2(b). \quad (5)$$

Следовательно, X допустимо, если для $(X \bmod b)^2 \bmod b = w$ выполняется условие (5).

Приведем пример использования такой процедуры прореживания.

Пример 1. Пусть b = 48. Будем определять допустимые X за несколько шагов согласно следующих правил.

А). Определение квадратичных остатков по модулю b. Для случая b = 48 квадратичные остатки приведены в табл.1.

Табл.1. Квадратичные остатки по модулю b = 48

$i \bmod b$	**0**	**1**	**2**	**3**	**4**	**5**
$i^2 \bmod b$	0	1	4	9	16	25
$i \bmod b$	6	7	8	9	10	11
$i^2 \bmod b$	36	1	16	33	4	25
$i \bmod b$	12	13	14	15	16	17
$i^2 \bmod b$	0	25	4	33	16	1
$i \bmod b$	18	19	20	21	22	23
$i^2 \bmod b$	36	25	16	9	4	1
$i \bmod b$	24	25	26	27	28	29
$i^2 \bmod b$	0	1	4	9	16	25
$i \bmod b$	30	31	32	33	34	35
$i^2 \bmod b$	36	1	16	33	4	25
$i \bmod b$	36	37	38	39	40	41
$i^2 \bmod b$	0	25	4	33	16	1
$i \bmod b$	42	43	44	45	46	48
$i^2 \bmod b$	36	25	16	9	4	1

Б). Определение квадратичных остатков $w \in M2(b)$, для которых выполняется соотношение (5).

Для проверки выполнения условия (5) необходимо знать значение $N \bmod b$. Будем исходить из того, что N и b не имеют общих делителей. Пусть $N \bmod b$=25. Для всех $w \in M2(b)$ определим разности $(w - 25)\bmod b$ и выясним, являются ли они квадратичными остатками по модулю b. Результаты вычислений приведены в табл.2.

Табл.2. Результаты определения остатков $(w - 25)\bmod b$

Элементы w множества M2(b) квадратичных остатков по модулю b	**0**	**1**	**4**	**9**	**16**	**25**	**33**	**36**
$s = (w - 25)\bmod b$	23	24	27	32	39	0	8	11
$s \in M2(b)$? (да – (+), нет – (-))	-	-	-	-	-	+	-	-

Согласно данных табл.2 единственным квадратичным остатком, для которого выполняется соотношение (5), является $w = X^2 \bmod b = 25$.

В) Определяем значения $X \bmod b$, для которых $X^2 \bmod b = 25$. Такому равенству удовлетворяют значения $X \bmod b = 5, 11, 13, 19, 29, 35, 37, 43$.

Г). Поиск неизвестных X и Y.

Пусть N=2329. $_{N \bmod b}$=25. С учетом того, что $X \bmod b = 5, 11, 13, 19, 29, 35, 37, 43$, а $x_0 = \lfloor\sqrt{N}\rfloor + 1 = \lfloor\sqrt{2329}\rfloor + 1 = 49$, возможными допустимыми значениями X будут числа, большие или равные x_0, а именно 53, 59, 61, 67, 77, 83…. В табл. 3 приведены результаты проверок будет ли разница $X^2 - N$ полным квадратом для допустимых X.

Табл.3. Значения $X^2 - 2329$ ***для допустимых значений X.***

X	53	59	61	67	77
$X^2 - 2329$	480	1152	1392	2160	$3600 = 60^2$
($X^2 - 2329$) - полный квадрат (+) или нет (-)	-	-	-	-	+

Как видно из табл. 3, полный квадрат разности $X^2 - 2329$ получен для X = 77 за 5 вычислений разности $X^2 - 2329$ и извлечения корня из нее вместо 29 аналогичных операций для случая, когда не использовалось бы прореживание. По результатам вычислений определено, что 2329 = (77+60)*(77-60) = 137*17.

Рассмотрим еще два варианта значений $N \bmod b$ при b = 48 и определим множества допустимых X.

Пусть $N \bmod b$=11. Для всех вариантов квадратичных остатков $w \in M2(b)$ определим разности $(w - N \bmod b) \bmod b$ и выясним, являются ли они квадратичными остатками по модулю b. Результаты вычислений приведены в табл.4.

Табл.4. Результаты определения остатков $(w - 11) \bmod b$ ***для всех вариантов квадратичных остатков по модулю b = 48***

Элементы w множества M2(b) квадратичных остатков по модулю b	0	1	4	9	16	25	33	36
$s = (w - 11) \bmod b$	37	38	42	46	5	14	22	**25**
$s \in M2(b)$? (да – (+), нет – (-))	-	-	-	-	-	-	-	+

Согласно данных табл.4 единственным квадратичным остатком, для которого выполняется условие (8), является $w = X^2 \bmod b = 36$. Согласно данных табл.1 такое равенство выполняется при значениях $X \bmod b = 6, 18, 30, 42$.

Рассмотрим еще один вариант значения $_{N \bmod b}$. Пусть $_{N \bmod b}$=35. Для всех вариантов квадратичных остатков $w \in M2(b)$ определим разности $(w - 35) \bmod b$ и выясним, какие из них принадлежат множеству $M2(b)$. Результаты вычислений приведены в табл.5.

Табл.5. Результаты определения остатков $(X^2 \bmod b - 35) \bmod b$ ***для всех вариантов квадратичных остатков по модулю b = 48***

Элементы z множества M2(b) квадратичных остатков по модулю b	0	1	4	9	16	25	33	36
$s = (z - 35) \bmod b$	13	14	17	22	29	38	46	**1**
$s \in M2(b)$? (да – (+), нет – (-))	-	-	-	-	-	-	-	+

Согласно данных табл.5 единственным квадратичным остатком, для которого выполняется условие (6), является $w = X^2 \bmod b = 36$. На шаге В) определяем значения $_{X \bmod b}$, для которых

$X^2 \bmod b = w = 36$. Такому равенству удовлетворяют значения $X \bmod b = 6, 18, 30, 42$, аналогично как и в случае $N \bmod b$ =11.

Приведенные примеры дают основание сделать два следующих заключения:

- одно и то же множество допустимых X может соответствовать разным вариантам значений $N \bmod b$;
- для разных $N \bmod b$ число элементов r в соответствующих им множествах допустимых X может не совпадать.

Пусть $|r|_{min}$ – минимальное число элементов множества допустимых X среди всех возможных значений $N \bmod b$, не имеющих общих делителей с b, включая $N \bmod b = 1$, $|r|_{max}$ – максимальное, а $|r|_{ср}$ – среднее такое значение. Определим коэффициенты ускорения:

$z_{min}(b) = b/|r|_{max}(b)$ – коэффициент минимального ускорения,

$z_{max}(b) = b/|r|_{min}(b)$ – коэффициент максимального ускорения,

$z_{ср}(b) = b/|r|_{ср}(b)$ – усредненное значение коэффициента ускорения.

В работах [М1, м2] для коэффициента $z_{ср}(b)$ экспериментально установлено следующие наиболее важные его свойства:

1. $z_{ср}(2^2) = 2$, $z_{min}(2^4) = z_{ср}(2^4) = z_{max}(2^4) = 4$, при $k > 4$ $z_{ср}(2^k) < 6$,
2. $z_{ср}(cb) = z_{ср}(c) \cdot z_{ср}(b)$, если c и b взаимно простые,
3. $z_{ср}(cb) = 2 \cdot z_{ср}(b)$, если c простое число, взаимно простое с b.

Известно также, что такие свойства, как правило, не выполняются для $z_{min}(b)$ и $z_{max}(b)$. В то же время при оценках вычислительной сложности алгоритмов необходимо исходить из максимально сложного варианта, который для случая метода факторизации Ферма будет при тех значениях $N \bmod b$, при которых коэффициент ускорения будет минимальным. Поэтому при выборе основания модуля b важно иметь оценку для $z_{min}(b)$.

В работе [М1] показано, что для случая простых b $z_{min}(b) = 2b/(b+1)$, т.е. при увеличении простого b коэффициент ускорения $z_{min}(b)$ стремится к 2. Следовательно, увеличения минимального значения коэффициента ускорения можно добиться при умножении исходного основания b на простые числа, которые еще не являются множителями b. Элементарные вычисления показывают, что число таких простых множителей не может быть большим, поскольку очень быстро растет их произведение. Поэтому альтернативой такому подходу является использование многократного прореживания для ускорения метода факторизации Ферма, когда оценка допустимости X проверяется не только для одного основания модуля, а для некоторого их числа. Описание такого метода прореживания и исследование его свойств приводятся ниже.

3. Алгоритм метода многократного прореживания МР

Пусть задано значение базового основания модуля bb и множество других оснований модулей $\{b_i\}_{i=1}^m$, где m > 0, где все основания являются попарно взаимно простыми числами. Алгоритм **МР** метода многократного прореживания представим следующей последовательностью шагов.

Шаг 0. Предварительная подготовка.

Шаг 0.1. Определение неотрицательных остатков от деления $N \bmod bb$ и $\{N \bmod b_i\}_{i=1}^m$

Шаг 0.2. Для основания bb определение $|r|_{max}(bb)$.

Шаг 0.3. Определение квадратичных остатков для основания bb (формируется массив Mbb[bb]), а также информации о допустимых $X \bmod bb$, т.е. тех, для которых выполнено условие (5). Все такие значения $X \bmod bb$ определены в диапазоне [0, bb-1] и могут быть представлены списком допустимых значений X, число которых не превышает $bb/z_{min}(bb) = |r|_{max}(bb)$. Список допустимых X будем хранить в массиве $Xbb[(|r|_{max} + 1)]$. С учетом того, что при разных значениях $N \bmod bb$ в диапазоне [0, bb-1] число допустимых X разное, в ячейке Xbb[0] указывается фактическое значение числа допустимых X, а сами значения записываются в порядке роста в ячейках от первой до $|r|_{max}(bb)$, а на их основании формируется последовательность шагов по X переменной длины между допустимыми X массива $Xbb[Xbb[0]]$ по правилам:

$$\begin{cases} a = Xbb[1]; \\ Xbb[k] = Xbb[k+1] - Xbb[k]; \\ Xbb[Xbb[0]] = a - Xbb[Xbb[0]] + bb. \end{cases} \quad (5)$$

$(k = 1 \div (Xbb[0] - 1))$

Шаг 0.4. Определение допустимых $x \bmod b_i$ для оснований $\{b_i\}_{i=1}^{m}$. Для каждого из оснований $\{b_i\}_{i=1}^{m}$ формируется информация о квадратичных остатках по модулю b_i а по ней массив Xb[i][*], в котором в ячейке t соответствует значение 1, если $(t^2 - N) \bmod b_i$ – квадратичный остаток по модулю b_i, ($(t^2 - N) \bmod b_i$ принадлежит множеству квадратичных остатков $M2(b_i)$) а иначе 0, т.е.

$$Xb[i][t] = \begin{cases} 1, (t^2 - N) \bmod b_i \in M2(b_i) \\ 0, (t^2 - N) \bmod b_i \notin M2(b_i). \end{cases} \quad (6)$$

$$(i = 1 \div m,\ t = 0 \div b_i - 1)$$

Шаг 0.5. Определение $x_0 = \lfloor \sqrt{N} \rfloor + 1$ и стартового X, равного xx, являющегося ближайшим большим или равным x_0 числом, для которого $(xx^2 - N) \bmod bb$ будет квадратичным остатком по модулю bb. Определение номера ячейки i_{xx} в массиве $Xbb[Xbb[0]]$, соответствующей $xx \bmod bb$.

Шаг 0.6. Суммарный прирост X, число kx, принимается равным 0. Значение kx будет ограничиваться сверху значением kxmax. Текущий прирост x, число dx, принимается равным 0. Присваиваем X = xx, i = i_{xx}, где i – текущее значение индекса для массива Xbb.

Шаг 1. Определить остатки $xx \bmod b_k$ и сохранить их в массиве Mx[(m+1)]. Присвоить kx=0. Определить остатки $(xx^2 - N) \bmod b_k$ и сохранить их в массиве My[(m+1)]. Если все числа $(xx^2 - N) \bmod b_k$ $(k = 1 \div m)$ являются квадратичными остатками по соответствующему модулю, перейти к шагу 6, а иначе к шагу 2.

Шаг 2. i=i+1. Если $i = Xbb[0]$, то присвоить i = 1.

Шаг 3. $dx = Mbb[i]$.

Шаг 4. kx = kx + dx.

Шаг 5. В цикле по k от 1 до m проверить выполнение условия

$$Xb[k][(mx[k] + kx) \bmod b_k] = 1. \quad (7)$$

Если при некотором k окажется, что условие (5) не выполняется, т.е. X является квадратичным остатком по модулю b_k, перейти к шагу 7.

Шаг 6. Найдено значение $X = xx + kx$, для которого, при всех анализируемых вариантах оснований модуля, разности $(X^2 - N) \bmod b = ((X \bmod b)^2 \bmod b - N \bmod b) \bmod b$ $(b = bb,\ b = b_k (k = 1 \div m))$ являются квадратичными остатками, т.е. X может быть решением уравнения (3).

Проверка: является ли разность $X^2 - N$ квадратом целого числа Y. Если да, то получено решение уравнения (3) p=X-Y, q=X+Y и работа алгоритма завершена, а иначе перейти к шагу 1.

Шаг 7. Если kx > kxmax перейти к шагу 1, присвоив xx значение xx + kx , а иначе перейти к шагу 2.

Алгоритм **MP** ориентирован на использование в стандартных компьютерах, где переменные могут иметь тип long и double. Для работы с бо'льшими числами их предлагается представлять в некоторой системе счисления. В практических реализациях алгоритма использовались основания счисления b_0 = 10, b_0 = 1000, b_0 = 1024, b_0 = 3600 и другие их варианты. Для всех результатов численных экспериментов, представленных в настоящей статье, принято b_0 = 1000.

Проиллюстрируем эффект использования нескольких оснований модуля при решении уравнения (3), а именно работу шага 5 алгоритма **MP** для приведенного выше примера малого числа N = 2329, где bb = b = 48, а одно дополнительное основание модуля b_1 = 5.

Ранее для N = 2329 было определено, что $N \bmod b$=25, $x_0 = \lfloor \sqrt{N} \rfloor + 1 = \lfloor \sqrt{2329} \rfloor + 1 = 49$, а возможными допустимыми значениями X будут числа 53, 59, 61, 67, 77, 83, …. Квадратичными остатками по модулю 5 будут числа 0, 1, 4, а допустимыми значениями X для основания модуля 5 являются значения $X \bmod 5 = 0, 2, 3$. Результаты вычислений представлены в табл.6.

Табл.6. Результаты проверок на полный квадрат разностей $X^2 - 2329$ для допустимых значений X с учетом использования дополнительного основания модуля b1 = 5.

X	**53**	**59**	**61**	**67**	**77**
$X \bmod 5$	3	4	1	2	2
$Xb[1][X \bmod 5]$	1	0	0	1	1
$\sqrt{X^2 - 2329}$ целое число? Да (+), нет (-), пропуск – не проверяется	-			-	+

Как видно из табл. 6 полный квадрат разности $X^2 - 2329$ получен для X = 77 за 3 извлечения корня вместо 5, поскольку при X = 59 и X = 61 числа $(X^2 - 2329) \bmod 5$ не являются квадратичным вычетом по модулю 5.

Сравнивая данные табл. 36 и табл. 6 легко увидеть, что полученные в них результаты совпадают абсолютно, но сравнение $Xb[1][X \bmod 5]$ с единицей явно более простая операция, чем вычисление каждый раз значения $X^2 - 2329$. При этом корень из $X^2 - 2329$ ищется не каждый раз.

2. Разработка компьютерного приложения на основе алгоритма MP

2.1. Представление больших чисел

Предполагается, что предложенный алгоритм может быть использован для факторизации чисел, превышающих ограничения для стандартных описаний long (а также long long). С точки зрения представления таких чисел в компьютере они являются большими. При создании компьютерного приложения для работы с ними использовался способ их представления, основанный на разложении числа по основанию b_0.

Пусть N – некоторое число. Его можно представить в виде

$$N = \sum_{i=0}^{m(N)} c_i b_0^i , \qquad (9)$$

а информацию о числе представить массивом, в нулевой ячейке которого записано количество элементов массива m(N)+1, описывающих число N, а в ячейках i (i=1÷ m(N)+1) – числа c_{i-1}. Рассмотрим пример.

Пусть информация будет представлена в массиве Mas, число N = 235691, b_0 = 60. Тогда

$$N = 11 + 28 \cdot 60 + 5 \cdot 60^2 + 1 \cdot 60^3 = \\ = c_0 + c_1 \cdot b_0^1 + c_2 \cdot b_0^2 + c_3 \cdot b_0^3 ,$$

m(N) = 3, а элементами массива Mas будут числа: $Mas[0] = 4$; $Mas[1] = 11$; $Mas[2] = 29$; $Mas[3] = 5$; $Mas[4] = 1$.

В общем виде массив типа Mas формируется согласно правил

$$Mas[i] = \begin{cases} i = 0: & Mas[0] = m(N) + 1, \\ i = 1 \div m(N) + 1: & Mas[i] = c_{m(N)+1-i} \end{cases} . \qquad (10)$$

При таком представлении чисел их сложение (или вычитание) сводится к сложению (вычитанию) чисел, записанных в соответствующих ячейках массивов типа Mas с учетом переноса разрядов. Умножение чисел может быть выполнено в “столбик”. Такая организация данных о больших числах использована в разработанном компьютерном приложении, фрагмент которого будет представлен в статье ниже.

2.2. Структура компьютерного приложения

В алгоритме **MP** наиболее часто в цикле выполняются операции шагов 2-5 и 7. Поэтому при его программной реализации операции, предусмотренные в других шагах, выносились за пределы цикла.

При разработке приложения известными полагались следующие данные:

- число N;
- основание системы счисления b_0,
- базовое основание модуля bb,
- число дополнительных оснований модулей СВ и их значения $\{b_k\}_{k=1}^{CB}$.

При задании числа N вводится информация о m(N) и m(N) +1 значениях c_i $(i = 0 \div m(N))$, которая хранится в массиве nn[*].

Первоначально в компьютерном приложении выполняются операции, предусмотренных на шаге 0 алгоритма **MP**, т.е. с учетом формирования массива $Xb[k][*]\,(k=1\div CB)$, в котором тем из $X \bmod b_k\ (k=1\div CB)$, для которых $\left(X^2-N\right)\bmod b_k$ является квадратичным остатком, соответствует значение 1, а иначе 0. После определения $x_0=\left\lfloor\sqrt{N}\right\rfloor+1$ (массивы kr[*] = rx[*]), стартового X, равного xx= rx[*], номера ячейки i_{xx} в массиве $Xbb[Xbb[0]]$, соответствующей $_{xx \bmod bb}$, а также присвоений переменным kxmax (принято kxmax = 2144000000), kx=0, X= xx, и k00= i_{xx}, где k00 – текущее значение индекса для массива Xbb, следует блок операторов, реализующий шаги 1-5 и 7. Текст программы для этого блока на языке С приведен ниже, где более мелким шрифтом приведены комментарии.

```
d=0; // d – величина приращения для X,
  определяемая по данным массива Mbb.
  Для стартового X d=0
mj=2144000000; // mj = kxmax

mtt0: l=0;  // счетчик числа оснований
  модуля, для которых
  (X^2 – N)mod b_i ∈ M2(b_i) (i = 1÷CB) .
  for (i=0; i<CB; i++)
  {
    b=mb[i]; lll++;
    j =0; for (f=rx[0]; f>0; f--)
  j=(rx[f]+ j*b0)%b; // j=kr[]%b;
    rd[i]=j;  //  остатки  от  деления
  X=rx[*] на основание b=mb[i]
    if ( xb[i][j]>0 ) l++;
  }
  if ( l<kb)
  {   // X не формируется квадратичный
  остаток хотя бы по одному из основа-
  ний модуля (X не будет решением)
   n1=d;
mtt: l=0;
 n2=1;  // Признак того, что X форми-
  рует квадратичный остаток
 j=n1;   // n1 – значение суммарного
  приращения к xx = kr[*]
 for (f=0; f<CB; f++)
    {
b=mb[f]; // Значение основания модуля
  номер f из списка оснований, задан-
  ных в массиве mb[*]
k=n1+rd[f]; if ( k>=b ) j=k%b; else
  j=k;
  // Определение X mod b_f
if ( xb[f][j]>0 ) l++;  // X формирует
  квадратичный остаток по основанию b_f
else n2=0;  // X не формирует квадра-
  тичный остаток по основанию b_f
if ( n2==0 ) break;
    }
     if ( l<CB )
  { // Хотя бы для одного из оснований
    модулей X не формирует квадратичный
    остаток по его основанию
  k00++; if ( k00>dx[0] ) k00=1;
  d=xbb[k00];  // Определение прироста к
    X по данным массива xbb[*]
  if ( n1>mj )
  { // текущее значение прироста X пре-
    высило kxmax = mj
    j3=n1;
    for (f=1; f<=rx[0]; f++)
    {  //  учет  суммарного  прироста  в
rx[*]=X
   n=rx[f]+j3;
   rx[f]=n%b0;  // b0 – основание систе-
    мы счисления, (в программе b0=1000)
   j3=n/b0;
   if ( j3==0 ) break;
  }
     if ( j3>0 ) { n=rx[0]+1; rx[n]=j3;
    rx[0]=n;  }
     n1=d;
     goto mtt0; // Переход к шагу 1 ал-
гоpитма
   }
   n1=n1+d;
   goto mtt; // Переход к шагу 2 алго-
ритма MP
    }
     }
     else
     {
       // учет суммарного прироста в
     массиве rx[*]=X
       SQR(X²-N); // Процедура определе-
     ния корня из остатка X²-N
     }
```

Если по результатам определения корня из X^2-N окажется, что он является целым числом, то определяются множители p и q, а иначе переход по метке **mtt0 (**Переход к шагу 1 алгоритма **MP).**

При анализе приведенного фрагмента программного кода на языке С можно заметить, что в нем почти все операции ведутся с использованием не самого X, а приращений к нему (числа d и n1). И только в относительно редких случаях, когда n1 превышает mj или при допустимости X для всех оснований модуля, переопределяется X, представленное массивом rx[*].

3. Численные эксперименты

При проведении численных экспериментов ставилась задача определения предельных значений N = pq таких, что p и q являются простыми и q<4p, а время разложения на множители числа N минимально превышало 1000 секунд для ПК со следующими характеристиками: тактовая частота 2.4 GHz, ОЗУ 6 ГБ, 32-разрядная операционная система.

Для проведения численных экспериментов строилась базовая последовательность чисел $\{t_i\}_{i=0}^{120}$, а на ее основании последовательности простых чисел $p_{i,j} = t_i + \delta_{i,j}\ (i = 0 \div 120,\ j = 0 \div 8)$, где $\delta_{i,j}$ – прибавки к t_i такие, что при bb=277200 число элементов в множестве x_{bb} равно максимально возможному значению $|r|_{max} = 2880$ для $N_{i,j} = p_{i,0} \cdot (t_{i-j} + \delta_{i,j})\ (i > 7)$. При этом t_0 = 10111000; $t_0 \cdot 2^{0.25} \approx t_1$ = 12024073, $t_0 \cdot 2^{0.5} \approx t_2$ = 14299113, $t_0 \cdot 2^{0.75} \approx t_3$ = 17004607, а при i > 3 $t_i = 2t_{i-4}$.

Для базового основания модуля bb = 277200 определено минимальное ускорение равное $z_{min}(bb) = bb/|r|_{max}(bb) = 277200/2880 = 96.25$. Число дополнительных оснований принято m = 14. Каждое из оснований $b = b_k\ (k = 1 \div m)$ являлось произведением 2-х или 3-х простых чисел или простым числом. Значения оснований приведены в табл.11.

Табл.7. Дополнительные основания модуля bk (k = 1÷m) и их простые множители

k	1	2	3	4	5	6	7	8	9	10	11	12	13	14
b_k	4199	2461	2987	3131	3589	3649	3569	3713	3869	4189	4087	109	113	127
Множи- тели b_k	13 17 19	23 107	29 103	31 101	37 97	41 89	43 83	47 79	53 73	59 71	61 67	109	113	127

При численных экспериментах определены предельные значения $N_{i,j} = p_{i,0} \cdot (t_{i-j} + \delta_{i,j})$ для j = 1 ÷ 8. Результаты представлены в табл. 8 и 9.

Табл.8. Минимальное из Ni,j для j = 1 ÷ 8 для которого время расчета t > tmax = 1000

j	i_{max}	$N_{i,j}$	x	l_{kxmax}	l_{sqrt}	t
1	109	2 190 107 742 436 404 740 487 152 427 983	5 558 342 544 918	2245	732	1103
2	101	115 103 357 258 699 743 681 239 319 283	5 106 595 343 108	2013	750	1039
3	97	24 197 500 008 691 435 623 032 029 847	5 284 605 435 652	2013	996	1180
4	94	7 193 959 711 947 061 718 333 522 687	5 145 026 742 943	2047	777	1057
5	92	3 024 687 551 113 421 119 054 532 273	5 242 071 052 870	2030	839	1120
6	91	1 798 489 957 219 681 011 882 800 933	5 860 792 837 224	2250	1063	1260
7	89	756 171 875 272 834 283 163 615 947	5 214 582 582 945	2068	780	1061
8	89	449 622 502 710 882 990 030 860 543	5 301 075 967 839	2078	844	1099

Табл.9. Максимальное Ni,j для j = 1 ÷ 8 для которого время расчета t < tmax = 1000

j	i_{max}	$N_{i,j}$	x	l_{kxmax}	l_{sqrt}	t
1	108	1 548 640 026 044 167 252 739 717 338 763	4 673 991 914 759	1838	707	966
2	100	81 390 365 008 084 150 314 806 066 243	4 294 118 141 585	1742	554	854
3	96	17 110 216 737 864 187 826 162 121 677	4 443 805 821 938	1783	615	908
4	93	5 086 897 811 841 886 885 408 415 303	4 326 434 593 759	1708	657	892
5	91	2 138 777 028 997 712 373 241 359 173	4 408 038 568 624	1732	675	901
6	90	1 271 724 453 348 788 067 534 653 477	4 928 319 544 222	2000	634	976

j	i_{max}	$N_{i,j}$	x	l_{kxmax}	l_{sqrt}	t
7	89	534 694 273 087 056 334 501 735 193	4 384 923 852 940	1733	681	898
8	88	317 931 109 083 252 528 293 431 373	4 457 655 697 241	1846	492	877

В табл. 8 и табл. 9 указанные в заголовке переменные означают:

$x = (p+q)/2 - \left[\sqrt{N_{i,j}}\right] - 1$;

l_{kxmax} – количество переходов с шага 7 на шаг 1 при условии kx > kxmax;

l_{sqrt} – количество раз извлечения корня с остатком из многорозрядного числа;

t – время расчета (сек).

На основании данных о времени разложения на множители чисел $N_{i,j} = p_{i,0} \cdot (t_{i-j} + \delta_{i,j})\ (i > 7)$ было установлено:

1. Время расчета пропорционально $x = (p+q)/2 - \left[\sqrt{N_{i,j}}\right] - 1$ независимо от j;

2. Для каждого из j время разложения на множители числа $N_{i,j} = p_{i,0} \cdot (t_{i-j} + \delta_{i,j})\ (i > 7)$ примерно в 2 раза меньше чем для числа $N_{i+4,j} = p_{i,0} \cdot (t_{i+4-j} + \delta_{i,j})\ (i < 110)$.

Второй из выводов можно обосновать теоретически. Действительно, пусть при p<q $q = (1+\alpha)^2 p$, где α – некоторое действительное число. Тогда величину $x = (p+q)/2 - \left[\sqrt{N_{i,j}}\right] - 1$ можно оценить через p и α

$$x = X - x_0 \approx (p+q)/2 - \sqrt{N} = (p+q)/2 - \sqrt{pq} = \\ = 0.5\left(\sqrt{q} - \sqrt{p}\right)^2 = 0.5\left((1+\alpha)\sqrt{p} - \sqrt{p}\right)^2 = \frac{p\alpha^2}{2},$$

через q и α,

$$x = X - x_0 \approx 0.5\left(\sqrt{q} - \sqrt{p}\right)^2 = \\ = 0.5\left(\sqrt{q} - \sqrt{q}/(1+\alpha)\right)^2 = \frac{q\alpha^2}{2(1+\alpha)^2},$$

а также представить через N и α

$$x = X - x_0 \approx 0.5\left(\sqrt{q} - \sqrt{p}\right)^2 = \\ = \sqrt{\frac{p\alpha^2}{2} \cdot \frac{q\alpha^2}{2(1+\alpha)^2}} = \frac{\sqrt{N}\alpha^2}{2(1+\alpha)}. \qquad (11)$$

Из соотношения (11) следует, что в случае практически неизменного значения α значение x пропорционально корню квадратному из N. Это подтверждается и для данных о времени расчета при разных вариантах N. Общая информация представлена на рис.1.

В связи с полученными результатами о времени расчета при неизменном значении базового основания модуля bb, а также числа дополнительных оснований m > 0 и их значений $\{b_i\}_{i=1}^{m}$ в последующем исследовалось влияние на время расчета значения bb и числа оснований m при заданных их значениях в табл. 7.

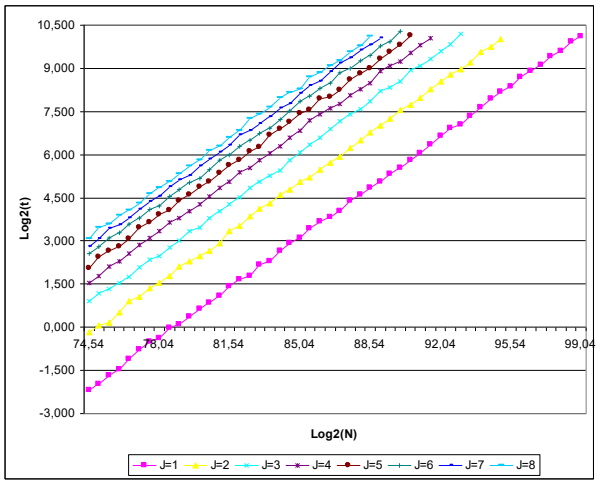


Рис.1. Диаграммы изменения времени расчета при росте i для j=1÷8

В случае использования значения базового основания модуля bb = 25200 (25200=277200/11) оказалось, что время расчета увеличилось в 1.74 ÷ 1.78 раза (в среднем в 1.77 раза). Для обеспечения сравнимости результатов суммарное число простых чисел в дополнительных основаниях модулей увеличилось на единицу за счет замены основания b_{12}=109 на $b'_{12} = 1199 = 109 \cdot 11$.

При использовании значения основания модуля bb = 3600 (3600=277200/11/7) время расчета увеличилось в 3.06 ÷ 3.17 раза (в среднем в 3.11 раза). Для обеспечения сравнимости результатов суммарное число простых чисел в дополнительных основаниях модулей увеличилось на два за счет замены оснований b_{12}=109 и b_{13}=113 на $b'_{12} = 1199 = 109 \cdot 11$ и $b'_{13} = 791 = 113 \cdot 7$. Следовательно, на время расчета существенно влияет выбор значения базового основания модуля bb, что требует дополнительного исследования.

Дополнительно были проведены численные эксперименты, в которых при bb = 277200 менялось число оснований модуля из их списка, приведенного в табл.7. Расчеты проводились для чисел

$N_{i,j} = p_{i,0} \cdot (t_{i-j} + \delta_{i,j})$ при i = 88 ÷ 94 для числа оснований m = 2, 3, 4, 6, 10 и 14. Средние значения коэффициентов отношений времени расчета для разных m (t_m) к аналогичному времени (t_{14} = 236 с) при m = 14 приведены в табл.10, где kp(m) – суммарное количество простых чисел в основаниях модуля в зависимости от их числа.

Табл.10. Средние значения отношений tm / t14 для ряда значений числа дополнительных оснований модуля.

m	**2**	**3**	**4**	**6**	**10**	**14**
t_m / t_{14}	4.937	1.969	1.257	1.090	1.032	1
kp(m)	5	7	9	13	21	26

Выше отмечалось, что увеличение числа простых множителей в основаниях модулей должно приводить к уменьшению количества извлечения корня в случаях, когда X оказывается допустимым для всех используемых оснований модуля. При этом добавление в основаниях одного простого числа u должно приводить к уменьшению количества обращений к процедуре извлечения корня в число раз от $2u/(u+1)$ до $2u/(u-1)$ (среднее значение равно 2). Когда число простых чисел в основаниях увеличивается на два, т.е. добавлены два простых числа u_1 и u_2, то количество обращений к процедуре извлечения корня уменьшается в число раз от $4u_1/(u_1+1)u_2/(u_2+1)$ до $4u_1/(u_1-1)u_2/(u_2-1)$ (среднее значение равно 4). Увеличение количества простых чисел в основаниях модуля на h должно приводить к уменьшению числа обращений к процедуре извлечения корня в среднем в 2^h раза. Для подтверждения истинности такого вывода были численно определены количества обращений к процедуре извлечения корня nq(m) из $X^2 - N_{i,j}$ в случае $N = N_{92,1}$ при количестве оснований модулей m = 2, 3, 4, 6, 10 и 14. Определялись также количество случаев (числа nkx(m)), когда n1 превышает mj (т.е. kx>kxmax). Такая информация представлена в табл.11.

Табл.11. Некоторые соотношения для количества раз обращения к извлечению корня при разном числе оснований модуля, заданных в табл.7

m	**2**	**3**	**4**	**6**	**10**	**14**
nq(m)	423652533	103255286	26386599	1678856	6447	180
nkx(m)	0	0	0	0	0	459
	kp(3)- kp(2)=2	kp(4)- kp(3)=2	kp(6)- kp(4)=4	kp(10)- kp(6)=8	kp(14)- kp(10)=5	
	$\frac{nq(2)}{nq(3)} = 4.1 \approx 2^2$	$\frac{nq(3)}{nq(4)} = 3.9 \approx 2^2$	$\frac{nq(4)}{nq(6)} = 15.7 \approx 2^4$	$\frac{nq(6)}{nq(10)} = 288.9 \approx 2^8$	$\frac{nq(10)}{nq(14)} = 35.8 \approx 2^5$	

Данные, приведенные в табл. 10 и табл. 11 позволяют сделать предположение о том, что количество оснований модуля, используемых при разложении на множители числа N, целесообразно увязывать с числом разрядов N и kxmax.

Выводы

Максимальное число N, которое можно разложить за фиксированное время, определяется величиной $x = (p+q)/2 - \lfloor\sqrt{N_{i,j}}\rfloor - 1$, которая не зависит от соотношения между p и q, что для случая прореживания пробных значений X подтверждается данными результатов расчетов, приведенных в табл. 8 и табл. 9. Следовательно, при оценках метода Ферма некорректно утверждать, что с его помощью возможна факторизация чисел, например, до 10^{25} ÷ 10^{30}, как это утверждается в работе [2] или 10^{30}, согласно работы [4]. Здесь важно дополнение «в худшем случае». Такой «худший случай» соответствует значению p=3. Если исходить из параметров прореживания (основания модулей bb и $\{b_i\}_{i=1}^{m}$,

число kxmax), то время расчета 1000 секунд приблизительно соответствовало $x=5\cdot10^9$, по значению которого при минимальном отношении p/q (p<q) из соотношения (4) определим максимальное N ≈60000046475809 $\approx6\cdot10^{13}$. В случае p = 3001 максимальное N ≈60021470437759827 $\approx6\cdot10^{16}$. Если процедура прореживания окажется более эффективной, то за 1000 секунд расчетного времени будут обработаны пробные значения x, превышающие $5\cdot10^9$, и максимальное N возрастет. Подтверждение этого получено на основании численного эксперимента в случае базового основания модуля bb = 61 261 200 и выбранных дополнительных оснований модулей, содержащих как множители простые числа от 19 до 173. Для ряда значений N было определено, что за 1 секунду времени работы ПК обрабатывается не менее $21.7\cdot10^9$ значений x, что более чем в три раза превосходит производительность специально созданной в 1995 году машины, которая могла анализировать $6.144\cdot10^9$ чисел в секунду [4].

Поскольку метод Ферма «плохо работает» для удаленных множителей, то для факторизации чисел можно использовать несколько методов, как в работе [3], за счет чего комплексный метод обладает временной сложностью $O(N^{1/3})$. Представляет интерес дальнейшая разработка способов разложения чисел на множители, в которых будет использован метод Ферма с прореживанием пробных X и другие методы, например, пробное деление на простые множители, как в работе [3], или известный ρ-метод Поларда.

Список литературы

1. Daniel R. L. Brown. Breaking RSA May Be As Difficult As Factoring. – [Электронный ресурс]. Режим доступа: http://www.pgpru.com/novosti /2005/1026vzlomrsabezfaktorizaciirealennoneeffektiven – Название с экрана.
2.5. Василенко О.Н. Теоретико – числовые алгоритмы в криптографии / О.Н. Василенко. – М.: МЦНМО, 2003. – 328с.
3. Lehman R. S. Factoring Large Integers // Math. Comp. 1974. V. 28. P. 637–646.
4. Кнут Д. Искусство программирования. Т. 2. Получисленные методы. 3-е изд. М.: Вильямс, 2007.
5. Винничук С.Д., Жилин А.В., Мисько В.Н. Алгоритм Ферма факторизации чисел вида N=pq методом прореживания. / Электронное моделирование - т.36, №2. – 2014. – с. 3-14.
6. Місько В.М. Прискорення методу Ферма методом проріджування з використання декількох баз. / Безпека інформації – Т. 21, №1. – 2015. – С. - 64 - 68.

УДК 621.396.96

КУЦ В.Ю.

СТРУКТУРА ВЫЧИСЛИТЕЛЯ ДЛЯ МНОГОШКАЛЬНЫХ ФАЗОВЫХ ИЗМЕРИТЕЛЕЙ НА ОСНОВЕ МОДУЛЯРНОЙ АРИФМЕТИКИ

Предложена структура процессора для обработки результатов измерений в многочастотной фазоизмерительной системе с представлением информации в системе остаточных классов. Рассмотрено использование особенностей модулярной арифметики для решения задачи устранения многозначности многошкальных фазовых измерений. Новый поход к получению и обработке численных данных позволяет выполнить контроль правильности устранения многозначности фазовых измерений и за счет этого обеспечить работоспособность таких систем при низких отношениях сигнал/шум.

The processor structure for processing the measurement results in a multi-frequency system fazoizmeritelnoy submitting information in residual classes system is proposed. We consider the use of the features of the modular arithmetic for solving the problem of eliminating ambiguity multiscale phase measurements. New trek to the preparation and processing of numerical data allows you to control the correctness of eliminating ambiguity phase measurements and thereby to ensure the efficiency of such systems at low signal / noise ratio.

Ключевые слова: система остаточных классов, фазовый метод, разрешение многозначности.

1. Введение

Область применения фазового метода измерений как чувствительного и информационно емкого инструмента исследования различных явлений и объектов, постоянно расширяется. Известны примеры его использования для измерения расстояний и плоских углов в астрономии, геодезии, радионавигации, радиотехнике, геологии, метеорологии, физике и т. п. [1, 2].

Фазовый метод измерения, особенного его многочастотный вариант реализации, связан с необходимостью использования значительных вычислительных ресурсов. Одной из наиболее ресурсозатратных задач является устранения многозначности фазовых измерений. Это обусловлено тем, что стремление обеспечить одновременно высокую чувствительность и широкий динамический диапазон требует измерения фазовых сдвигов гармонических сигналов вида $\Phi = 2\pi n + \varphi >> 2\pi$. Здесь n – количество целых фазовых циклов, $\varphi \equiv (\Phi) \bmod 2\pi$ – часть полного фазового сдвига сигналов по модулю 2π, $\equiv$ – обозначение операции сравнения двух чисел по некоторому модулю. Однозначное определение n и составляет суть задачи устранения фазовой многозначности (или неоднозначности).

Известны различные способы решения указанной задачи на основе измерений фазовых сдвигов $\varphi \in [0,\ 2\pi)$ сигналов на нескольких (двух и более) частотах. Например, для фазовых дальномеров на каждой *j*-той рабочей частоте определяется фазовый сдвиг между сигналами вида

$$\begin{aligned} u_{1j}(t) &= U_1 \sin(2\pi f_j t), \quad t \in [0, T], \\ u_{2j}(t, D) &= U_2 \sin[2\pi f_j t - \varphi_j(D)], \end{aligned} \tag{1}$$

при распространении измерительного сигнала вдоль дистанции D (здесь $T > f_j^{-1}$ – время измерения, а $\varphi(D) \equiv \Phi(D) \bmod 2\pi$).

В случае измерения на двух частотах $f_1 < f_2$ для фазового дальномера получают систему уравнений

$$\begin{cases} D = 0,5\lambda_1 (n_1(D) + \varphi_1(D)/2\pi), \\ D = 0,5\lambda_2 (n_2(D) + \varphi_2(D)/2\pi). \end{cases} \tag{2}$$

В уравнениях (2) λ_1, λ_2 – длины волн колебаний с частотами f_1, f_2. Система (2) для однозначного решения дополняется некоторыми начальными условиями. Например, в способе с качанием частоты определяют разность $n_2(D) - n_1(D)$ во время непрерывного изменения частоты сигнала от f_1 до f_2. В способах с использованием фиксированных частот система (2) дополняется условием $n_1(D) = 0$. Расширение динамического диапазона измерений влечет

необходимость увеличения числа рабочих частот и усложнения алгоритмов обработки данных. Это влечет повышение вероятности появления грубых ошибок в определении *n*.

В последнее время получил развитие другой способ устранения многозначности фазовых измерение, основанный на использовании особенностей модулярной арифметики [4]. Этот способ основан на том, что свойство цикличности фазовых сдвигов сигналов, которое аналитически можно представить выражением

$$\varphi(Q) \equiv \Phi(Q) \bmod 2\pi = 2\pi\left(\frac{\Phi(Q)}{2\pi} \bmod 1\right),$$

также присуще и остаткам α_i целого числа A при его представлении по модулю p_j, – $\alpha_j \equiv A \bmod p_j$, то есть в случае представления A в системе остаточных классов (СОК). В этом случае задача устранения многозначности многошкальных фазовых измерений сводится к задаче восстановления целого числа A из его представления остатками в СОК. Известно [3], что СОК относится к помехоустойчивым корректирующим кодам. Это свойство СОК распространяется и на процесс устранения многозначности фазовых измерений.

Известны примеры использования СОК для устранения многозначности в фазовых дальномерах [4], интерферометрах [5], фазовых пеленгаторах [6]. Однако вопросу построения специализированных вычислительных процессоров для таких устройств не уделено должного внимания. Целью статьи является разработка процессора для обработки данных измерений в многочастотных фазоизмерительных системах с представлением информации в СОК.

2. Восстановление целых чисел из их представления остатками в СОК

Модулярная арифметика, в основном, развивалась применительно к использованию в спецпроцессорах с целью повышения быстродействия, надежности и коррекции возможных сбоев во время вычислений. Наиболее важными свойствами СОК являются: 1) отсутствие связей между смежными остатками, 2) возможность организации эффективной системы обнаружения и исправления ошибок во время выполнения арифметических операций и восстановления чисел в позиционной системе счисления из их представления в СОК.

В СОК все целые числа A из некоторого рабочего интервала $A_{\text{р}} - 1$ отображаются множеством неотрицательных остатков α_j от деления A на другие целые числа – модули системы, которые образуют группу взаимнопростых чисел p_j, $j = \overline{1,n}$, то есть $A_{\text{СОК}} = (\alpha_1, ..., \alpha_j, ...\alpha_n)$, где $\alpha_j \in [0,\ p_j)$, $\alpha_j \equiv A \bmod p_j$, $j = \overline{1,n}$.

В отличие от позиционных числовых систем все остатки модульного представления независимы и являются малоразрядными числами. Это позволяет свести основные арифметические операции к параллельному модульному и быстрому выполнению одноименных операций для каждого модуля. Исключение составляют операции деления и сравнения, которые в общем случае выполняются не эффективно.

Для вычислителей фазовых многочастотных систем на основе модулярной арифметики наиболее значимой является операция восстановление целых чисел A из их представления $A_{\text{СОК}}$. Один из способов такого преобразования базируется на китайской теореме об остатках [7]. В соответствие с этой теоремой восстановление A возможно, если, во-первых, модули системы – взаимно простые числа, и во-вторых, максимальное восстанавливаемое число удовлетворяет неравенству

$$A_{\max} < P_{\text{р}} = \prod_{j=1}^{n} p_j .$$

При выполнении этих условий число A однозначно представляется в позиционной системе счисления, то есть существует обратное преобразование $A_{\text{СОК}} \Rightarrow A$. Число A вычисляют по формуле [3]

$$A = \sum_{j=1}^{n} a_j B_j \left(\bmod A_{\text{р}}\right), \qquad (4)$$

где $(B_1, ...B_j, ...B_n)$ – система ортонормированных базисов.

В работе [4] показано, что выбором квантов измерения фазовых сдвигов сигналов $\Delta\varphi_j$ и частот f_j обратно пропорционально модулям системы, задача устранения многозначности фазовых измерений сводится к задаче восстановле-

ния целого числа по его остаткам для максимального значения фазового сдвига $\Phi_{\max} = 2\pi \prod_{j=1}^{n-1} p_j$. При этом остатки вычисляются по фазовым сдвигам сигналов как $a_j = \left[\frac{\varphi_j}{\Delta\varphi_j}\right]^+$, где $[\square]^+$ – операция выделения целой части числа.

Проведение фазовых измерений в условиях реального физического эксперимента сопровождается действием шумов и помех. Последние возникают как в среде распространения сигналов, так и в электронных трактах измерительной аппаратуры. Уменьшение отношения сигнал/шум приводит к увеличению погрешности измерений фазовых сдвигов сигналов и, как следствие, к росту вероятности ошибок вычисления остатков.

Если основу СОК дополнить еще одним модулем $p_{n+1} > p_j$, $j = \overline{1,n}$ (или $k \geq 1$ модулями), образуется новая СОК, называемая полной, с иной системой ортонормированных базисов $(\tilde{B}_1, ... \tilde{B}_j, ... \tilde{B}_{n+1})$ и полным диапазоном представления чисел $[0,\ A_п)$, где $A_п = \prod_{j=1}^{n+1} p_j$. Любое искажение остатка в представлении $(\alpha_1, ..., \alpha_{n+1})$, например, первого – $\tilde{\alpha}_1$, приводит к тому, что число $\tilde{A} = \left(\tilde{\alpha}_1 B_1 + \sum_{i=2}^{n+1} a_i B_i\right)(\mathrm{mod}\, A_п)$ при его вычислении переходит из рабочего в запрещенный диапазон $[A_р,\ A_р p_{n+1})$. Такой результат является признаком ошибки и используется для ее поиска и устранения.

3. Организация процесса поиска остатков с ошибками в представлении числа $A_{СОК}$

Использование СОК для представления результатов многошкальных фазовых измерений позволяет увеличить вероятность правильного решения задачи устранения многозначности путем проведения измерений на одной или нескольких дополнительных шкалах, обнаружения остатков с ошибками и их удаления или исправления.

Наиболее простой способ поиска и коррекции искаженного остатка заключается в восстановлении числа A по его проекциям в СОК по разным подсистемам модулей [3] и последующему анализу полученных результатов восстановления числа в позиционной системе счисления. Новые подсистемы модулей получают из иссходной системы $(p_1, ..., p_j, ... p_{n+k})$ путем удасения произвольных k модулей. Количество таких подсистем равно количеству сочетаний $C_{n+k}^n = \frac{(n+k)!}{n!k!}$. Для $k = 1$ число таких систем $C_{n+1}^n = n+1$. Для каждой новой *g-той* подсистемы, $g \in \left[1,\ C_{n+k}^n\right]$, рассчитывается свой базис $Bg = (\tilde{B}g_1, ... \tilde{B}g_j, ... \tilde{B}g_n)$, определяется число

$$Ag = \left(\sum_{j=1}^{n} a_j \tilde{B}g_j\right)(\mathrm{mod}\, A_п) \qquad (5)$$

и проверяется выполнение условия $Ag \in [0,\ A_р)$ Это позволяет выявить искаженные ошибками остатки и исключить их при восстановлении числа *A*.

Такое свойство СОК объясняется тем, что ошибка в одной позиции вектора $A_{СОК}$ не влияет на расчеты в других позициях вектора. В отличие от позиционной системы счисления все элементы $A_{СОК}$ равнозначны и ошибка в одном из них ведет всего лишь к сокращению динамического диапазона. Этот факт собственно и позволяет проектировать устройства с коррекцией ошибок.

4. Моделирование процесса вычисления целого числа из его представления в СОК

Рассмотрим численные примеры, иллюстрирующие процесс восстановления числа и обнаружения остатков с ошибками.

В этих примерах вычисление базисов СОК для системы модулей произвольной размерности осуществлчлось с помощью созданной в среде Matlab файл-функции sokbaz.m:

```
function B=sokbaz(P)%P-вектор модулей
N=length(P);
PP=prod(P);
for l=1:N
   p=P(l);
   d=mod(PP/p,p);
   m=1;
   while mod(m*d,p)~=1
      m=m+1;
   end
```

```
   B(l)=m*PP/p;% базис
end
y=mod(sum(B),PP);
if y~=1
   B(1:N)=NaN;
   sprintf('%s','ОШИБКА!!!')
end
```

Пример 1. Зададим систему модулей (13, 14, 15, 17), для которой рабочий диапазон составляет $A_р = 13 \cdot 14 \cdot 15 = 2730$, а полный – $A_п = 13 \cdot 14 \cdot 15 \cdot 17 = 46410$, то есть n=3, k=1.

Рассмотрим представление числа $A = 2014$ в заданной системе модулей:

$$A_{СОК} = (12,\ 12,\ 4,\ 8).$$

Расчет базисов СОК дает результат:

$$B = (17850,\ 29835,\ 12376,\ 32760).$$

Вычисление позиционного числа A в соответствие с (5) дает значение

$$A = \sum_{j=1}^{4} a_j B_j \left(\bmod A_п\right) = \left(12 \cdot 17850 + 12 \cdot 29835 + \right.$$

$$\left. + 4 \cdot 12376 + 8 \cdot 32760\right) \bmod 46410 = 2014.$$

Таким образом, восстановление числа выполнено без ошибок.

Приведенная в примере 1 система модулей позволяет выявлять все одиночные ошибки в представлении $A_{СОК}$ в интервале $\left[0,\ A_р\right)$. Перейдем к рассмотрению этого корректирующего свойства СОК.

Пример 2. Изменим условия примера 1. Пусть в представлении $A_{СОК}$ числа $A = 2014$ искажен один, второй, разряд, например,

$$\tilde{A}_{СОК} = (12,\ 13,\ 4,\ 8).$$

Тогда процедура восстановления позиционного числа дает следующий результат

$$A = \sum_{j=1}^{4} a_j B_j \left(\bmod A_п\right) = \left(12 \cdot 17850 + 13 \cdot 29835 + \right.$$

$$\left. + 4 \cdot 12376 + 8 \cdot 32760\right) \bmod 46410 = 31849 > A_р.$$

Полученное неравенство свидетельствует о присутствии в $A_{СОК}$ остатка с ошибкой. Проверим корректность применения способа проекций для поиска искаженных ошибками остатков.

Пример 3. Выполним корректное восстановление искомого позиционного числа по его проекциям для данных примера 2. Количество таких проекций определяется числом сочетаний всех модулей по три: $C_4^3 = 4$. Выполним расчеты для четырех возможных подсистем модулей

А) для подсистемы модулей (14, 15, 17) имеем: $\tilde{A}1_{СОК} = (13,\ 4,\ 8)$,

$$B1 = (1275,\ 1666,\ 630),$$

а вычисленное по формуле (5) число –

$$\tilde{A}1 = (13 \cdot 1275 + 4 \cdot 1666 + 8 \cdot 630) \bmod 46410 =$$

$$= 28279 > A_р.$$

Б) для подсистемы модулей (13, 15, 17) –

$$\tilde{A}2_{СОК} = (12,\ 4,\ 8),$$

$$B2 = (1275,\ 2431,\ 2925),$$

$$\tilde{A}2 = 2014 = A < A_р\,!$$

В) для подсистемы модулей (13, 14, 17) –

$$\tilde{A}3_{СОК} = (12,\ 13,\ 8)$$

$$B3 = (2380,\ 1989,\ 1820)$$

$$\tilde{A}3 = 22567 > A_р$$

Г) для подсистемы модулей (13, 14, 15) –

$$\tilde{A}4_{СОК} = (12,\ 13,\ 4)$$

$$B4 = (1470,\ 2535,\ 1456)$$

$$\tilde{A}4 = 10009 > A_р.$$

Вывод: корректное знеачение восстановленного числа получено для подсистемы модулей (13, 15, 17). Єтот результат свидетельствует о том, что остаток $\tilde{a}_2 = 13$ по модулю $p_2 = 14$ получен с ошибкой.

Таким образом, вычисление искомого позиционного числа по различным подсистемам модулей с последующей проверкой выполнения неравенства $\tilde{A}j < A_р$ позволяет выявлять в представлении $A_{СОК}$ остаток с ошибкой и исключать его из процесса получения коректного результата.

5. Структура процессора для многошкальной фазовой системы с использованием модулярной арифметики

Выполненный выше анализ особенностей применения модулярной арифметики для обработки результатов многошкальных фазовых измерений позволил предложить специализированный процессор для реализации таких вычислений. Структура такого процессора приведена на рис.1.

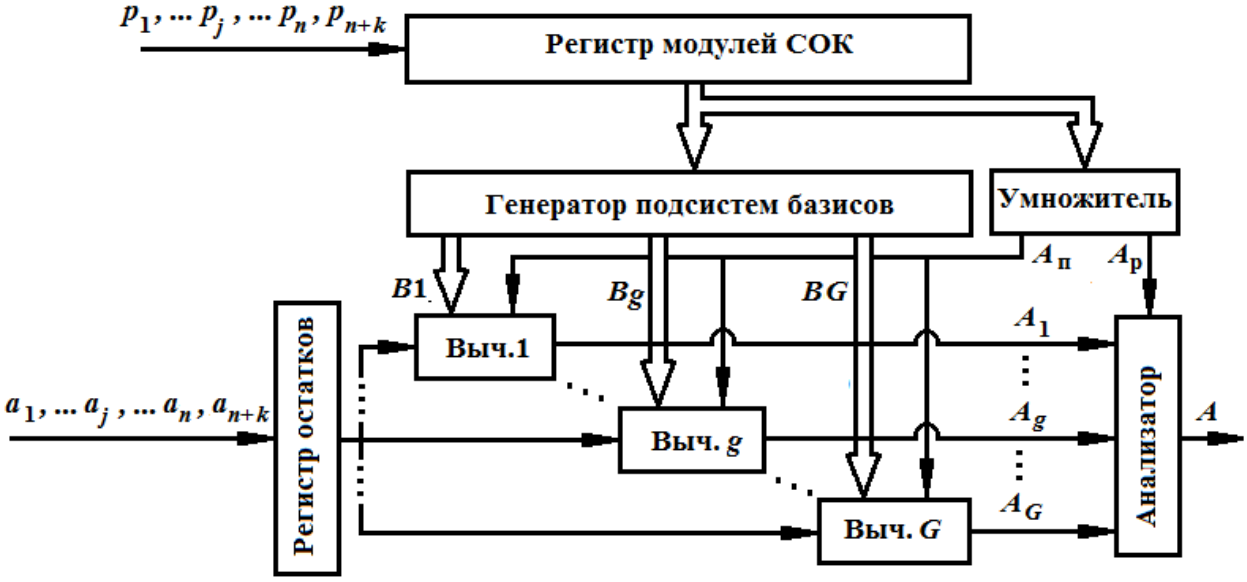


Рис. 1. Структура специализированного процессора для многошкальной фазовой измерительной системы с использованием модулярной арифметики.

Процессор работает следующим образом. Модули СОК $(p_1,...p_{n+k})$ и полученные остатки восстанавливаемого числа $(a_1,...a_{n+k})$ хранятся в соответствующих регистрах. Генератор подсистем базисов вычисляет значения последних для всех $g=\overline{1,G}$ возможных сочетаний базисов для рабочего диапазона СОК. Умножитель формирует границы рабочего $A_р$ и полного $A_п$ диапазонов представления чисел в позиционной системе счисления. Вычислители Выч.g, $g=\overline{1,G}$, реализуют вычисление чисел в соответствие с алгоритмом (5). Анализатор выполняет проверку неравенства $A<A_р$ и формирование корректного результата восстановления числа.

Ошибка в одной позиции вектора не влияет на расчеты в других позициях вектора. В отличие от позиционной системы счисления все элементы вектора равнозначны и ошибка в одном из них ведет всего лишь к сокращению динамического диапазона. Этот факт позволяет проектировать устройства с повышенной отказоустойчивостью и коррекцией ошибок

По данным [3] введение только одного дополнительного модуля позволяет выявлять все одиночне ошибки (ошибки в одном остатке) и ~95% двойных ошибок.

Заключение

Фазовый многшкальный метод имерения различных физических величин обеспечивает высокую точность и разрешающую способность измерения.

Задача устранения многозначности многошкальных фазовых измерениях сводится к задаче восстановления целых чисел из их представления в СОК по заданной системе модулей соответствующим выбором значений частот и ступеней квантов ания фазовых сдвигов сигналов.

Одним из наиболее важних свойств арифметики СОК является возможность наращивания числа модулей системы, что позволяет реализовать помехоустойчивый способ обработки результатов многошкальный фазових измерений для большого числа шкал и пониженного отношения сигнал/шум.

Использование многошкального фазового метода в сочетании с возможностями арифметики СОК позволяет существенно расширить функциональные возможности метода за счет выполнения фазовых измерений на дополнительных частотах и реализации способов обнаружения и исправления ошибок определения фазовых сдвигов сигналов, значительно превышающих величину 2π.

Предложенный процессор для обработки результатов многошкальных фазовых измерений с использованием модулярной арифметики. Введение только одной дополнительной шкалы за счет фазовых измерений на одной дополнитель-

ной частоте позволяет процессору выявить все одиночне ошибки и ~95% двойных ошибок при решении задачи устранения фазовой неоднозначности измерений. Это способствует уменьшению вероятности грубых погрешностей в условиях работы фазоизмерительной аппаратуры при низких соотношениях сигнал/шум. Развитие этого направления требует дальнейших исследований в части анализа эффективности выявления и корректирования боле сложных ошибок.

Список литературы

1. В.Б. Пестряков. Фазовые радиотехнические системы (основы статистической теории). М.: Советское радио, 1968, 468 с.
2. Ю.В. Куц, Л.Н. Щербак**.** Статистична фазометрія. Тернопіль: Тернопільський державний університет ім. І. Пулюя, 2009, 383 с.
3. И.Я. Акушский, Д.И. Юдицкий. Машинная арифметика в остаточных классах, М.: Советское радио, 1968, 440 с.
4. Ю.В Куц Вимірювання кумулятивних фазових зсувів. Технічна електродинаміка, 2001, №5, С.67-72.
5. В.И. Гужов, С.П. Ильиных. Компьютерная интерферометрия. Новосибирск: Изд-во НГТУ, 2004, 252 с.
6. В.Ю. Куц. Представлення і оброблювання даних вимірювань в системі залишкових класів. Матеріали V міжнар. наук.-практ. конф.«Комплексне забезпечення якості технологічних процесів та систем», 19-22 травня 2015 р.– Чернігів, 2015.–С.253.
7. Математическая энциклопедия: Гл. ред. И.М. Виноградов, т.2, с.862. М.: Советская Энциклопедия, 1979, 1104 с.

УДК 683.322

РУСАНОВА О. В.

СПОСІБ ПЛАНУВАННЯ ОБЧИСЛЕНЬ ДЛЯ МУЛЬТИЯДЕРНИХ КЛАСТЕРІВ

У роботі запропонований універсальний спосіб планування обчислень *HMTS* для мультиядерних кластерних систем із будь-якою топологією. Даний підхід є комбінацією спискового та кластерного способів планування. Розроблена програмна модель для реалізації запропонованого та найбільш відомих способів планування обчислень для мультиядерних кластерів. Наведено результати досліджень, які підтверджують більш високу ефективність підходу *HMTS* у порівнянні з відомими способами.

The universal scheduling approach *HMTS* for multi-core cluster systems with any topologies is proposed in paper. This approach is the combination of list and cluster scheduling methods. Software for implementation of the proposed and most well-known scheduling methods in multi-core cluster systems was developed. The results of researches, which are presented in the paper, confirm the higher efficiency of the approach *HMTS* in comparison with known methods, were presented.

Ключові слова: мультиядерні кластери, планування обчислень, топологія системи, списковий, кластерний, генетичний способи планування

1. Вступ

Сьогодні прогрес суперкомп'ютерів безпосередньо пов'язаний із розвитком кластерних систем. Аналіз останньої 46-ї редакції списку найпотужніших комп'ютерів світу TOP-500 показує, що підвищення продуктивності кластерних систем здійснюється за рахунок використання багатоядерних вузлів та збільшення загальної кількості ядер у системах. Так, за останній рік середня кількість ядер у системах зросло на 12308 (або на 26,6%) з 46288 до 58596. Таке збільшення кількості ядер призводить до суттєвого ускладнення вирішення задачі ефективного використання ресурсів кластерних систем. Наприклад, ефективність найбільш потужної системи у світі *Tianhe-2*, яка складається з 3120000 ядер, дорівнює 0,62. Підвищення ефективності використання ресурсів пов'язано з удосконаленням методів планування обчислень.

У даній роботі пропонується ефективний кластерно-списковий спосіб статичного планування обчислень для підвищення реальної продуктивності мультиядерних кластерних систем.

2. Існуючі підходи до планування в кластерних системах

Існуючі способи статичного планування для кластерних систем можна розділити на чотири основні групи: генетичні, спискові, кластерні та з дублюванням [1]. Розглянемо найбільш відомі з них.

Гібридний генетичний алгоритм *HGAHS*. У роботі [2] запропоновано алгоритм планування, який є гибридом генетичного та спискового підходів. Суть цього алгоритму полягає в наступному. Спочатку для підзадач завдання визначаються пріоритети з використанням спискового алгоритму *HEFT* [3]. Потім відповідно до отриманих пріоритетів виконується первинне призначення підзадач на ресурси. Номери ресурсів (процесорів) генеруються випадковим чином. Після цього виконується аналіз отриманого призначення та визначаються підзадачі, вибрані для мутації. Цей процес повторюється до тих пір, поки задана ймовірність якості призначення не буде досягнута. Розглянутий алгоритм добре зарекомендував себе для гетерогенних кластерних систем, проте він не враховує можливу мультиядерну архітектуру системи. Крім того, використання алгоритму припускає, що обчислювальна система є повнозв'язною, що робить неможливим застосування алгоритму в системах з будь-якою іншою топологією.

Списковий алгоритм планування *SHTSA (HCPPEFT)* з використанням дублювання. У роботі [4] запропоновано алгоритм, що є комбінацією спискового підходу та дублювання обчислень, суть якого полягає в наступному. На першому етапі, відповідно до спискового підходу, визначаються пріоритети підзадач та формується черга для їх призначення на проце-

сори. Для визначення пріоритетів використовуються такі характеристики: критичні шляхи до початку графу задачі, критичні шляхи до кінця графу задачі, кількість батьківських підзадач та приналежність підзадач до критичного шляху графа задачі. На другому етапі виконується призначення підзадач на процесори з використанням процедури дублювання обчислень. Дублювання виконується для мінімізації часу виконання задачі. Даний алгоритм, так само, як і алгоритм *HGAHS*, підтримує лише повнозв'язну топологію кластерної системи і не враховує мультиядерність вузлів.

Алгоритм планування для мультиядерних систем *SAMCS*

У роботі [5] запропоновано алгоритм для мультиядерних систем, суть якого полягає в наступному. Даний алгоритм включає два етапи. На першому етапі групи підзадач призначаються по вузлах кластерної системи, а на другому – окремі підзадачі призначаються відповідно по ядрах вузлів. Для реалізації даного алгоритму спочатку формуються групи підзадач з метою мінімізації витрат на комунікацію (пересилки даних). Для цього об'єднуються в групи найбільш зв'язані підзадачі, а також використовується процедура дублювання. Основним недоліком даного алгоритму є те, що він, як і попередні, розрахований лише на повнозв'язні кластерні системи.

У даній роботі пропонується універсальний спосіб планування обчислень для мультиядерних кластерних систем із будь-якою топологією.

3. Постановка задачі планування для мультиядерних кластерів

Вхідними даними для статичного планування є моделі задачі та кластерної системи.

Модель задачі, зазвичай, може бути представлена у вигляді направленого ациклічного графу *DAG (Directed Acyclic Graph)*. Вершинам такого графу відповідають підзадачі, а дугам – комунікаційні зв'язки між ними. Кожна вершина *DAG* характеризується номером та вагою, яка визначає її обчислювальну вартість. Дуги *DAG* також характеризуються вагою, що визначає комунікаційну вартість між відповідними вершинами. Комунікаційна вартість підзадач, призначених на одне й те саме ядро вузла системи, дорівнює нулю [6]. Підзадачі без вхідних дуг називаються вхідними, а підзадачі без вихідних дуг – кінцевими вершинами. Будь-яка підзадача вважається готовою до виконання, якщо всі її батьківські підзадачі виконані. Приклад графа задачі представлений на рис. 1.

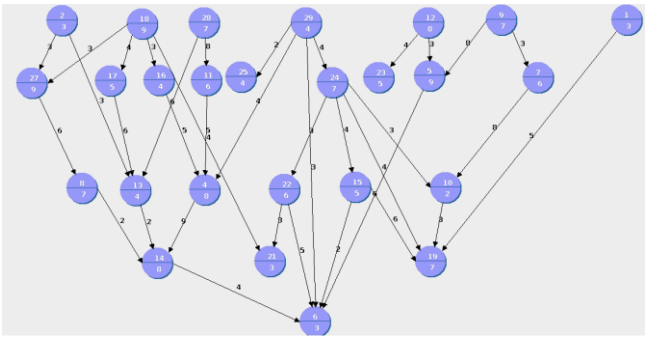


Рис. 1. Приклад графу задачі

У даній роботі кластерна система має ієрархічну структуру і складається з гомогенних (однорідних) вузлів. У свою чергу, кожен вузол системи є мультиядерним. Модель такої кластерної системи може бути представлена у вигляді ненаправленого графу системи. Вершинам такого графу відповідають вузли системи, а дугам – комунікаційні зв'язки між ними (топологія системи). Кожна вершина такого графу характеризується номером та кількістю ядер у вузлі системи. Крім того задається продуктивність усіх вузлів та кількість фізичних каналів для пересилки даних. Дуги графу системи характеризуються вагою, яка визначає їх пропускну спроможність щодо пересилки даних. Оскільки у даній роботі передбачається одна й та сама швидкість пересилки даних між сусідніми вузлами, тому і ваги дуг у графі системи мають одні й ті значення. Крім того передбачається, що всі процесори системи мають автономні контролери вводу-виводу, які дозволяють одночасно обчислювати підзадачі, а також виконувати обмін даних з іншими процесорами. Приклад графу системи представлений на рис.2. Модель цієї кластерної системи складається з 16 вузлів, кожен з яких має два ядра. У даному прикладі описана топологія гіперкуба 4-го порядку. Крім того, кожен вузол має продуктивність 1.0, пропускна спроможність усіх каналів також дорівнюють 1.0.

Результат планування може бути представлений за допомогою діаграми Ганта. Діаграма Ганта є розкладом роботи усіх обчислювальних вузлів системи і для кожного з них потактовий порядок виконання усіх завдань, призначених на цей вузол. Проте традиційна діаграма Ганта містить не усю інформацію про дані, що пере-

силаються, з одного вузла на інший. У роботі пропонується використати модифіковану діаграму Ганта [6], у якій розклад роботи кожного ядра вузлів системи включає не лише чергу виконуваних підзадач, але і порядок пересилки даних по усіх його каналах.

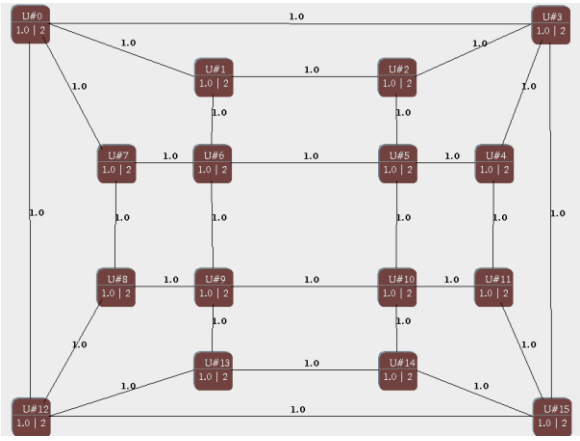


Рис. 2. Приклад графу системи

Загальна постановка задачі планування у даній роботі полягає в наступному. Нехай є кластерна система, що складається з k вузлів. Кожний j–й вузол включає p ядер. Топологія кластерної системи та її технічні характеристики задані за допомогою графа системи. Задана також задача користувача, що включає m підзадач, за допомогою графа задачі. Для кожної з m підзадач необхідно знайти таке z–е ядро кластерної системи, яке забезпечує мінімальний загальний час виконання заданої задачі (T_i). Математична модель цього завдання може бути записана таким чином. Знайти

$$\min_{i=1,R}\{T_i\}$$

$$T_i = \sum_{b=1}^{m}\sum_{j=1}^{k}\sum_{l=1}^{p} t_{bjl} * X_{bjl},$$

де R – кількість варіантів вирішення задачі;
t_{bjl} –час виконання b–ї підзадачі в l–му ядрі j–го вузла кластерної системи;
$X_{bjl} = 1$, якщо b–а підзадача виконується в l–му ядрі j–го вузла кластерної системи;
$X_{bjl} = 0$, у противному випадку.

4. Запропонований спосіб планування обчислень для мультиядерних кластерів

У роботі запропоновано новий ефективний спосіб планування обчислень для гомогенних мультиядерних кластерних систем з довільною топологією на основі комбінації спискового та кластерного підходів – *HMTS (Homogeneous Multicore Task Scheduling).* Спосіб планування *HMTS,* як і всі спискові підходи має етап визначення пріоритетів підзадач графу задачі та формування черги на призначення. Також даний спосіб включає етап вибору ядра вузла кластерної системи для підзадач. Проте, крім цього, в *HMTS,* використовується ще один додатковий проміжний етап. На цьому етапі виконується групування (кластеризація) підзадач. У даному способі планування не передбачається виконання кластеризації усього графа задачі, як це виконується в алгоритмі *SAMCS.* У запропонованому підході в групи об'єднуються лише вхідні вершини (підзадачі) графа задачі, а в процесі призначення можливі випадки групування підзадач за умови наявності непризначених вхідних підзадач і присутності вільних ядер у вибраному вузлі у даний момент часу. Цей підхід дає підстави збільшити ефективність планування. Таким чином, *HMTS* складається з наступних трьох етапів, які виконуються послідовно:

1. Визначення пріоритетів підзадач.
2. Групування вхідних підзадач графа задачі.
3. Призначення підзадач по вузлах та ядрах кластерної системи та додаткове групування.

На першому етапі пріоритет кожної підзадачі в графі задачі визначається довжиною її критичного шляху до кінця графа задачі з урахуванням тільки ваг вершин. Такий спосіб обчислення пріоритетів називають *b-level* [1]. Значення підзадачі n_i визначається за формулою:

$$b(n_i) = w_{n_i} + \max_{n_j \in imed_succ(n_i)} \{b(n_j)\},$$

де w_{n_i} – вага підздачі n_i;
imed_succ(n_i) – множина безпосередніх наступників (дочірніх підзадач) підзадачі n_i.
Значення *b-level* вихідних підзадач визначається їх власною вагою.

Етап групування підзадач (вершин) першого ярусу складається із наступних кроків:

1. Визначити множину вершин першого ярусу S_{1L}.

2. Відсортувати ці вершини у порядку спадання їх пріоритетів (*b-level*).

3. Визначити загальну кількість ядер у системі.

4. Якщо розмір множини вершин перевищує кількість ядер, зменшити розмір множини вершин до кількості ядер шляхом виключення вершин із найменшим пріоритетом.

5. Доки не порожня множина S_{1L}:

5.1. Створити нову групу вершин – множину G_i.

5.2. Виключити з S_{1L} першу вершину V_1 , включивши її в G_i.

5.3. Повторювати цикл *N-1* раз, де *N* – кількість ядер у вузлах:

5.3.1. Серед вершин із множини S_{1L} обрати таку вершину V_2, для якої коефіцієнт зв'язності пари $\{V_1, V_2\}$ буде максимальним. Коефіцієнт зв'язності *K* обчислюється за формулою:

$$K_{V_1,V_2} = \sum_{j=0}^{R} E_{i,j} * (L_{max} - L_i),$$

де $E_{i,j}$ - вартість (вага) *j*–ї спільної комунікації (дуги) між вершинами V_1 та V_2;

L_{max} - номер останнього ярусу графа задачі;

L_i - номер *i*–го ярусу, де починається спільна дуга.

5.3.2. Виключити з S_{1L} вершину V_2 та включити її в множину G_i.

Продемонструємо роботу етапу групування на прикладі графа задачі, зображенному на рис.1. Згідно алгоритму, визначимо множину вершин першого ярусу S_{1L} та впорядкуємо їх у порядку спадання пріоритетів. Значення *b-level* та пріоритети вершин першого ярусу зображені в табл.1.

Множина S_{1L} включає сім вершин і має вигляд: $S_{1L} = \{V_{18}, V_{28}, V_2, V_{29}, V_9, V_{12}, V_1\}$. Припустимо, що кластерна система складається з двох вузлів, кожен з яких має три ядра. Тоді загальна кількість ядер дорівнює шести, що перевищує розмір множини вершин S_{1L}. Необхідно з множини S_{1L} виключити вершину з найменшим пріоритетом, а саме V_1. Після цього множина S_{1L} має вигляд: $S_{1L} = \{V_{18}, V_{28}, V_2, V_{29}, V_9, V_{12}\}$.

Таблиця 1. Значення b-level та пріоритети вершин першого ярусу графа задачі

Вершина	*b-level*	Пріоритет
2	30	4
18	36	6
28	32	5
29	23	3
12	20	1
9	22	2
1	10	0

Виконаємо покроково операцію групування. Для цього створимо групу G_1, вершину V_{18} виключимо з множини S_{1L} і включимо в G_1. Тоді $G_1 = \{V_{18}\}$, $S_{1L} = \{V_{28}, V_2, V_{29}, V_9, V_{12}\}$. Тепер необхідно із множини S_{1L} знайти вершину, яка має максимальний коефіцієнт зв'язності з V_{18}. У табл.2 вказані коефіцієнти зв'язності вершини V_{18} з усіма вершинами з множини S_{1L}.

Таблиця 2. Коефіцієнти зв'язності вершин

	V_{28}	V_2	V_{29}	V_9	V_{12}
V_{18}	20	30	16	0	0

Із табл.2 видно, що максимальний коефіцієнт зв'язності з вершиною V_{18} має вершина V_2, тому вона має бути додана до множини G_1 і виключена з множини S_{1L}. У результаті $S_{1L} = \{V_{28}, V_{29}, V_9, V_{12}\}$, $G_1 = \{V_{18}, V_2\}$.

Для завершення формування групи G_1, необхідно повторити вищевказані кроки для пари V_{18}, V_2 та всіх вершин, що залишились у множині S_{1L}. Для визначення множини спільних дуг вершини V_x з групою вершин $\{V_y, V_z\}$ використовується співвідношення:

$$E = (E_y \cup E_z) \cap E_x ,$$

тобто перетин множини дуг графа з коренем у вершині V_x відбувається з об'єднанням множин дуг графів з коренями у вершинах V_y, V_z. Найбільший коефіцієнт зв'язності з парою вершин V_{18}, V_2 має вершина V_{28}. Таким чином, вершину V_{28} виключаємо з множини S_{1L} і включаємо її в G_1. У результаті $S_{1L} = \{V_{29}, V_9, V_{12}\}$ $G_1 = \{V_{18}, V_2, V_{28}\}$. Оскільки кількість вершин у множині S_{1L} співпадає із залишком ядер у кластерній системі, то група G_2 буде еквівалентною S_{1L}. Таким чином $G_1 = \{V_{18}, V_2, V_{28}\}$, $G_2 = \{V_{29}, V_9, V_{12}\}$. У результаті групування підзадач і призначення їх

на ядра одного й того самого вузла зменшуються витрати часу на взаємодію між вузлами, що дає можливість підвищити реальну продуктивність кластерної системи.

Етап призначення підзадач по вузлах та ядрах кластерної системи включає наступні кроки:

1. Для кожної групи, сформованої на другому етапі в порядку спадання пріоритету, виконати призначення на вузол системи. Вузол обрати із списку за пріоритетом. Пріоритети вузлів визначаються за їх зв'язністю у графі системи. Найбільший пріоритет мають вузли з максимальною зв'язністю.

2. Для кожної готової (доступної) підзадачі, у якої всі батьківські підзадачі вже призначені, у порядку спадання пріоритетів (*b-level*):

   2.1. Обрати вузол, який забезпечує мінімальний час старту підзадачі.

   2.2. Перевірити, чи не залишилось непризначенних підзадач з першого ярусу. Якщо такі підзадачі є, перевірити, кількість вільних ядер в даний момент часу у вибраному для призначення вузлі. Якщо в обраному вузлі більше одного вільного ядра, виконати групування обраної підзадачі з підзадачами першого ярусу за алгоритмом другого етапу.

3. Виконати призначення підзадачі або групи, в залежності від результату п.2.2.

Зазначимо, що при рівності пріоритетів, для призначення обирається підзадача з більшою зв'язністю, а при рівній зв'язності – з меншою вагою. Крім того, при виборі вузла для виконання враховуються усі вузли кластерної системи, незалежно від того зайняті вони, чи вільні в момент призначення.

Таким чином, описана процедура призначення дозволяє врахувати архітектурні топологічні особливості багатоядерної кластерної системи.

5. Порівняльний аналіз способу планування HMTS з існуючими підходами

Для підтвердження ефективності запропонованого підходу, розроблена програмна модель, за допомогою якої можна виконати порівняльний аналіз *HMTS* з широко відомими алгоритмами *HGAHS, SHTSA* та *SAMCS*.

При проведенні порівняльного аналізу використовувались наступні метрики:

- *Scheduling Length (SL)* або *makespan*. Визначається часом виконання задачі при плануванні обчислень.
- *Scheduling Length Ratio(SLR).* Головним індикатором потужності способу планування є *SL (makespan).* Оскільки при проведенні експериментів використовується велика кількість графів задач з різними властивостями, необхідно нормалізувати *SL* до меншої межі, якою є *SLR.* Значення *SLR* визначає у скільки разів *SL* більше $CPMIN$ і визначається за формулою

$$SLR = \frac{makespan}{\sum_{n_i \in CP_{MIN}} \min_{p_j \in Q}\{w_{i,j}\}}$$

Знаменником є сума мінімальних обчислювальних вартостей підзадач $CPMIN$. Спосіб планування обчислень, який забезпечує найменше значення *SLR* графа, є кращим з точки зору часу виконання графа задачі.

- *Speedup (SU)* або прискорення. Значення *SU* визначається обчислюється шляхом ділення послідовного часу виконання графа задачі на час виконання графа задачі *(SL – makespan)* у паралельній системі:

$$Speedup = \frac{\min_{p_j \in Q}\{\sum_{n_i \in V} w_{i,j}\}}{makespan}$$

- *Efficiency (E)* або ефективність. Визначається співвідношенням значення прискорення до кількості ядер кластерної системию
- *Running time (RT)* або час роботи самого алгоритму планування.

Тестування алгоритмів планування виконується з використанням випадково згенерованих графів задач та заданих графів кластерних систем.

За допомогою розробленої програмної моделі у даній роботі проведені наступні дослідження. Згенеровано 2400 типів графів задач із наступними вхідними параметрами:

V = {20, 30, 40, 50, 60, 70, 80, 90, 100} – кількість підзадач у графі задачі;

a = {0.5, 1.0, 2.0} – форма графа;

CCR = {0.1, 0.25, 0.5, 0.75, 1.0} – співвідношення сумарної комунікаційної вартості до сумарної обчислювальної вартості графа задачі;

Wmin = {5, 10, 40, 75, 100} – мінімальна вага вершини графу;

Wmax = {200, 500, 1000} – максимальна вага вершини графу.

Окрім цього, для кожного з типів генерувалось 10 графів задач і таким чином для

порівняння способів планування усього згенеровано 24000 графів задач.

В якості графа системи використано топологію гіперкуба 4-го порядку (рис.2), що складається з 16 вузлів, кожен з яких має два ядра.

Результати проведення експериментів, щодо показника *SL*, наведені у табл.3

Попарно порівнюючи середні значення *SL* досліджувальних способів планування між собою, можна зазначити наступне:

- Запропонований спосіб планування *HMTS* у 95% варіантів графів задач забезпечив мінімальне значення *SL*, що є кращим результатом серед усіх досліджуваних способів планування.
- Спосіб планування *SHTSA* в 78% варіантів графів задач забезпечив максимальне значення *SL*, що є гіршим результатом серед усіх досліджуваних способів планування.
- За результатами порівняння значень *SL*, досліджувані способи планування можна розташувати таким чином: *{HMTS, SAMCS, HGAHS, SHTSA}*, при цьому перший є кращим, а останній гіршим.

Середнє значення *SLR* на 24000 згенерованих графах завдань для *HMTS* = 2.09, для *SAMCS* = 2.36, для *HGAHS* = 2.48, для *SHTSA* = 2.68.

В усіх проведених тестах запропонований спосіб планування обчислень *HMTS* показав кращі результати, за винятком швидкості роботи, поступившись по цьому параметру підхорам *SAMCS* і *SHTSA*. По інших показниках досліджувані способи планування у порядку спадання ефективності можна розташувати наступним чином: *{HMTS, SAMCS, HGAHS, SHTSA}*. При цьому *SLR* запропонованого способу *HMTS* краще на 13%, ніж у *SAMCS*, на 19%, ніж у *HGAHS* і на 24%, ніж у *SHTSA*.

Таблиця 3. Результати попарного порівняння значень SL способів планування на множині випадково згенерованих графах завдань

		HMTS	***SAMCS***	***HGAHS***	***SHTSA***
HMTS	Краще Рівно Гірше	*	20657 197 3146	23776 114 110	22153 378 1469
SAMCS	Краще Рівно Гірше	3146 197 20657	*	21939 362 1699	19067 601 4332
HGAHS	Краще Рівно Гірше	110 114 23776	1699 362 21939	*	2152 340 21508
SHTSA	Краще Рівно Гірше	1469 378 22153	4332 601 19067	21508 340 2152	*

6. Висновки

У роботі запропоновано новий ефективний спосіб планування *HMTS* для мультиядерних кластерних систем із довільною топологією. Цей підхід є комбінацією спискового та кластерного способів планування. Розроблена програмна модель для виконання порівняльного аналізу запропонованого способу із найбільш відомими підходами, які можуть бути використані для планування обчислень мультиядерних кластерів. Приведені результати досліджень, що підтверджують більш високу ефективність способу *HMTS* у порівнянні з відомими підходами.

Запропонований підхід *HMTS* може бути використаний для підвищення реальної продуктивності як гомогенних мультиядерних кластерів, так і мультиядерних систем із масовим паралелізмом. Майбутні роботи у цьому напрямку пов’язані з адаптацією *HMTS* до гетерогенних кластерних систем.

Список посилань

1. Young Choon Lee, «Problem-Centric Scheduling for Heterogeneous Computing Systems», September 2007. – P.162.
2. Kamaljit Kaur. Heuristics Based Genetic Algorithm for Scheduling Static Tasks in Homogeneous Parallel System // Department of Computer Science & Engineering, Guru Nanak Dev University, Amritsar- 143001, Punjab, India, IJCSS, vol.4, pp.183-198, 2010.
3. H. Topcuoglu, S. Hariri and M. Wu, «Performance-Effective and Low-Complexity Task Scheduling for Heterogeneous Computing», IEEE Trans. Paralleland Distributed Systems, vol. 13, no. 3, pp. 260–274, March 2002.
4. Yanyan Dai. A Synthesized Heuristic Task Scheduling Algorithm // Institute of Information and Communication, Guilin University of Electronic Technology, Guilin 541004, China, 2014. – P. 67.
5. Xiaozhong Geng, Gaochao Xu, Xiaodong Fu, Yuan Zhang, «A Task Scheduling Algorithm For Multi-Core-Cluster Systems», Journal of Computers, vol.7,no11, pp.2797-2804, November 2012.
6. G. Loutsky, O. Rusanova, «Scheduling Problems on the Parallel and Distributed Systems - an Overwiew» – Poland, Rzeszow, tom 1, pp.101-105 (Engl), 2000.

УДК 681.3

ГЛОМОЗДА Д. К.

ЗАСТОСУВАННЯ МЕТОДУ ІНВАРІАНТІВ ДО АНАЛІЗУ КОЛЬОРОВИХ МЕРЕЖ ПЕТРІ ІЗ ДЕДЛОКАМИ

В роботі описано застосування методу інваріантів, що використовує алгоритм пошуку зрізаної множини розв'язків рівняння стану мережі Петрі, яке записується у вигляді систем лінійних однорідних діофантових рівнянь, до аналізу кольорових мереж Петрі, які містять дедлоки, на прикладі моделі функціонування багатопоточності в мові програмування Java за шаблоном «виробники/споживачі» з використанням комбінації методів `wait()/notify()`.

Application of invariants method, which uses Truncated Set of Solutions finding algorithm for Petri net state equations expressed through systems of linear homogenous Diophantine equations to analysis of colored Petri nets containing deadlocks is described in the article based on the models of multithreading in Java according to producers-consumers patter using `wait()/notify()` methods combination is covered.

Ключові слова: кольорові мережі Петрі, S-інваріанти, T-інваріанти, діофантові рівняння, багатопоточність.

Вступ

Мережі Петрі (МП) та їх розширення, зокрема кольорові мережі Петрі (КМП), є потужним та популярним засобом моделювання та дослідження властивостей складних динамічних розподілених та паралельних систем [1], бізнес-процесів [2], високопродуктивних IT-інфраструктур [3], навчальних та колаборативних середовищ [4, 5]. Інтерактивні автоматизовані системи їх побудови та аналізу на кшталт зв'язки графічного редактора PM Editeur [6] з Petri Nets Toolbox [7] для класичних (або чистих) МП та CPN Tools [8] для КМП користуються великою популярністю, що створює попит на ефективні алгоритми визначення їхніх властивостей.

Одним з таких є алгоритм визначення мінімальної породжуючої множини S- та T-інваріантів чистих МП за допомогою алгоритмів дослідження систем лінійних однорідних діофантових рівнянь (СЛОДР) та нерівностей (СЛОДН), зокрема їх використання для пошуку зрізаної множини розв'язків (Truncated Set of Solutions, TSS) записаного у вигляді СЛОДР рівняння стану мережі Петрі [9, 10]. Обчисливши S- та T-інваріанти для заданої МП, можна верифікувати такі її властивості як досяжність, обмеженість, повторюваність, консервативність, несуперечливість тощо [9, С. 199; 10, С. 157—170].

Доказом ефективності цього підходу є приклади його застосування до дослідження МП, які моделюють взаємодію абонентів у телефонній мережі з використанням різноманітних потенційно конфліктуючих сервісів [11, 12], організацію руху залізничного транспорту на залізничному вузлі взаємопов'язаних спеціалізованих станцій [13], функціонування обчислювального вузла Grid-системи [14]. Проте в цих роботах використовувалися чисті та компонентні МП, які поступаються потужністю та виразністю кольоровим МП. Водночас автоматизований аналіз КМП за допомогою методу інваріантів ускладнений через його функціональну, а не цілочисельну, як у класичних МП, природу [15]. В свою чергу, традиційні прямий та зворотній аналіз досяжності передбачають побудову дерев досяжності, кількість вершин яких вимірюється сотнями. Тому задача знаходження способу застосування до КМП методу знаходження S- та T-інваріантів є актуальною.

Саме вирішенню цієї задачі присвячена стаття [16], в якій підхід на основі S- та T-інваріантів було застосовано для дослідження властивостей двох КМП-моделей. Першою була КМП-модель, побудована на основі МП-моделі взаємодії абонентів у базовій телефонній мережі (Plain Old Telephone Service, POTS), запропонованої С. Л. Кривим та Л. Є. Матвєєвою [11, 12]. Другою була побудована автором КМП-модель взаємодії потоків у багатопоточних Java-застосуваннях за шаблоном «виробники-споживачі» з використанням комбінації методів wait()/notifyAll(). Було описано спосіб переведення КМП-моделі з базової форми з якісними, тобто семантично навантаженими, фішками, у форму з кількісними, тобто семан-

тично ненавантаженими, фішками, характерними для класичних мереж Петрі. Аналіз двох форм моделі засвідчив збереження основних її властивостей.

Утім, стаття [16] розв'язує задачу лише частково, оскільки використання комбінації `wait()/notifyAll()` принципово усуває вірогідність виникнення дедлоків. Отже, залишився недослідженим цілий пласт властивостей МП- та КМП-моделей.

Запропонована стаття є логічним продовженням статті [16]. В ній побудовано КМП-модель взаємодії потоків у багатопоточних Java-застосуваннях за шаблоном «виробники-споживачі» з використанням комбінації методів `wait()/notify()`. Використання цієї комбінації може привести до дедлоку. Отже, її модель дасть можливість перевірити, як перехід від якісних до кількісних фішок впливає на дослідження моделі на активність. Таким чином, задача застосування до КМП методу знаходження S- та T-інваріантів буде розв'язана повністю.

Метою роботи є виявлення взаємозв'язків і взаємовідповідностей між МП- та КМП-моделями, що стануть основою створення засобів автоматизованого аналізу КМП та автоматизованого перетворення МП у КМП. Для досягнення цієї мети автором побудовано і досліджено КМП-модель взаємодії потоків з використанням комбінації методів `wait()/notify()` із якісними фішками. Далі на її основі побудовано і досліджено модель із кількісними фішками. Доведено, що у випадку цих моделей перехід з однієї форми до іншої не призводить до зміни властивостей мереж та спотворення результатів аналізу.

КМП із якісними фішками

Побудована нами КМП-модель взаємодії потоків в багатопоточних застосуваннях за шаблоном «виробники-споживачі» з використанням комбінації методів `wait()/notify()` із якісними фішками складається з 6 місць та 10 переходів (рис. 1). Місце *Threads* містить активні потоки, представлені фішками типу TYPE, для якого визначено два можливі значення: PROD («потік-виробник») і CONS («потік-споживач»). Потоки-виробники збільшують на 1 значення спільної змінної типу INT (представлена місцем *Shared Variable*), якщо воно рівне 0, а потоки-споживачі зчитують його, якщо воно відмінне від 0, і зменшують його на 1. Необхідний для коректної роботи системи взаємовиключний доступ до спільної змінної регулюється монітором. Він складається з множини претендентів (місце *Entry Set*), зони власника (місце *Owner Zone*) та множини очікуючих (місце *Wait Set*). При зверненні до спільної змінної потік потрапляє до множини претендентів, де знаходиться доти, доки монітор не впустить його у зону власника. Потрапивши у зону власника, потік або виконує свою задачу, якщо для цього є умови, або відправляється у множину очікуючих, де чекає на свій наступний шанс. Наступний шанс випадає тоді, коли потік, що успішно виконав свою задачу, залишає монітор. При цьому викликається метод `notify()`, який пробуджує один обраний випадковим чином очікуючий потік (якщо хоча б один такий є). Детальніше про функціонування монітора у віртуальній машині Java див. [17].

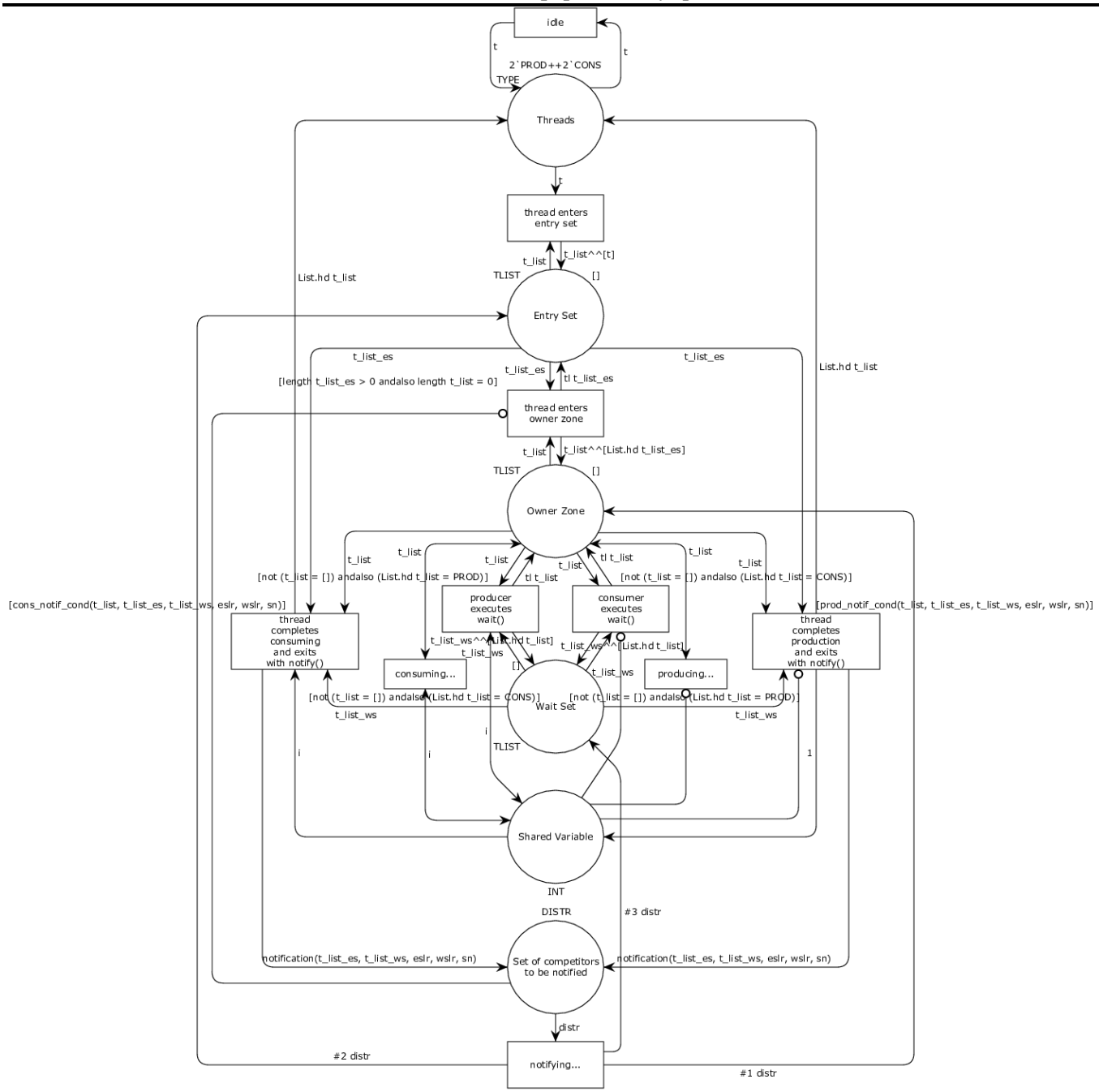


Рис. 1. КМП-модель із якісними фішками

Місця *Entry Set, Owner Zone* та *Wait Set* мають тип TLIST, тобто список об'єктів типу TYPE. В початковому стані всі ці списки порожні, а в подальшому до них додаються та прибираються елементи, що позначають окремі потоки. Розглянемо випадок, коли їх у системі чотири, при чому два з них є виробниками, а інші два – споживачами. Початкове маркування $\mathbf{M_0}$ = (2'PROD + 2'CONS, 1'[], 1'[], 1'[], -, -).

Ця модель складніша за попередню через потребу забезпечити випадковий вибір потоку для пробудження і переходу у зону власника з-поміж тих, які перебувають у множині претендентів та множині очікуючих. При роботі зі списковими типами даних це означає потребу генерування випадкових чисел – індексів елементів списків претендентів та очікуючих, з-поміж яких обирається один для переміщення у зону власника. Але вбудовані у CPN Tools функції генерування випадкових чисел, такі як `discrete()` та `uniform()`, при переході у простір станів генерують не набір різних станів, кожний з яких відповідає одному з імовірних варіантів, а тільки один стан, який відповідає єдиному обраному випадковим чином варіанту. Тому для здійснення повного перебору можливих варіантів було введено проміжкові типи: RANGE_L = int with 0..3 (тобто цілі числа від 0

до 3 включно для перебору індексів елементів списку, що може містити максимум три елементи одночасно) та RANGE_N = int with 1..2 (тобто цілі числа від 1 до 2 включно для перебору варіантів вибору між списком претендентів та списком очікуючих).

Дія методу `notify()` моделюється шляхом створення трьох списків. Перший з них складається з потоку, який до цього перебував у множині претендентів або очікуючих, але на наступному кроці прокинеться і перейде у зону власника, другий — з потоків, які були у множині претендентів і там і залишаться, третій — з потоків, які були у множині очікуючих і там залишаться. Ці списки зводяться в місці *Set of competitors to be notified* в одну фішку типу DISTR = TLIST * TLIST * TLIST, після чого розподіляються по місцях *Owner Zone, Entry Set* та *Wait Set* відповідно.

Аналіз КМП із якісними фішками

Аналіз графа досяжності даної КМП-моделі, побудованого засобами CPN Tools, дозволяє дійти таких висновків щодо її властивостей:

- обмеженість (граф досяжності включає в себе 210 станів, що є скінченною величиною, а отже, досліджувана КМП є обмеженою);
- повторюваність (існує початковий стан, починаючи з якого, можна побудувати таку послідовність переходів між станами σ, що всі переходи зустрічатимуться у σ нескінченну кількість разів);
- несуперечливість (початковий стан є досяжним із самого себе через таку послідовність переходів між станами σ, що всі переходи входять до σ принаймні один раз);
- взаємовиключність (в місці *Owner Zone* не може перебувати більше одного потоку одночасно);
- живучість (мертві переходи відсутні, мертві маркування: $\mathbf{M_{195}}$ = (-, 1’[], 1’[], 1’[CONS,CONS,PROD,PROD], 1’1, -) та $\mathbf{M_{209}}$ = (-, 1’[], 1’[], 1’[PROD,PROD,CONS,CONS], -, -)).

Результати дослідження підтверджують, що використання комбінації методів `wait()/notify()` у багатопоточних Java-застосуваннях, які функціонують згідно шаблону «виробники-споживачі», є небезпечним через можливість виникнення тупикової ситуації.

Перехід до КМП із кількісними фішками

Для застосування алгоритму визначення мінімальної породжуючої множини S- та T-інваріантів необхідно переробити побудовану нами КМП із якісними фішками на КМП із кількісними фішками типу UNIT та без інгібіторних дуг. Для цього треба замінити якісні місця КМП (*Threads, Entry Set, Owner Zone, Wait Set*) наборами місць, кожне з яких відповідатиме одному з допустимих значень відповідного якісного типу TYPE. Замість перших трьох місць матимемо шість (*Producers/Consumers, Producers/Consumers In Entry Set, Producer/Consumer In Owner Zone*). Місце *Wait Set* треба розбити на шість місць: *No Producers/Consumers In Wait Set, One Producer/Consumer In Wait Set, Two Producers/Consumers In Wait Set.* Також нам знадобиться місце *Shared Variable Is Null,* яке замінить інгібіторні дуги, що виходили з місця *Shared Variable.*

Для моделювання всіх можливих варіантів спрацювання методу `notify()` знадобляться переходи *prod/cons_wakes_none, prod/cons_wakes_cons/prod_none_left, prod/cons_wakes_cons/prod_one_left, prod_wakes_prod, cons_wakes_cons.* Переходи *prod/cons_wakes_prod/cons_none_left* та *prod/cons_wakes_prod/cons_one_left* не потрібні, оскільки виробників та споживачів тільки двоє, і ці переходи будуть мертвими. Утім, в моделях з більшою кількістю потоків ці переходи будуть потрібні.

Одержана нами в результаті перетворення КМП, яка складається з 17 місць та 22 переходів, наведена на рис. 3. Початкове маркування $\mathbf{M_0}$ = (2, 2, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1).

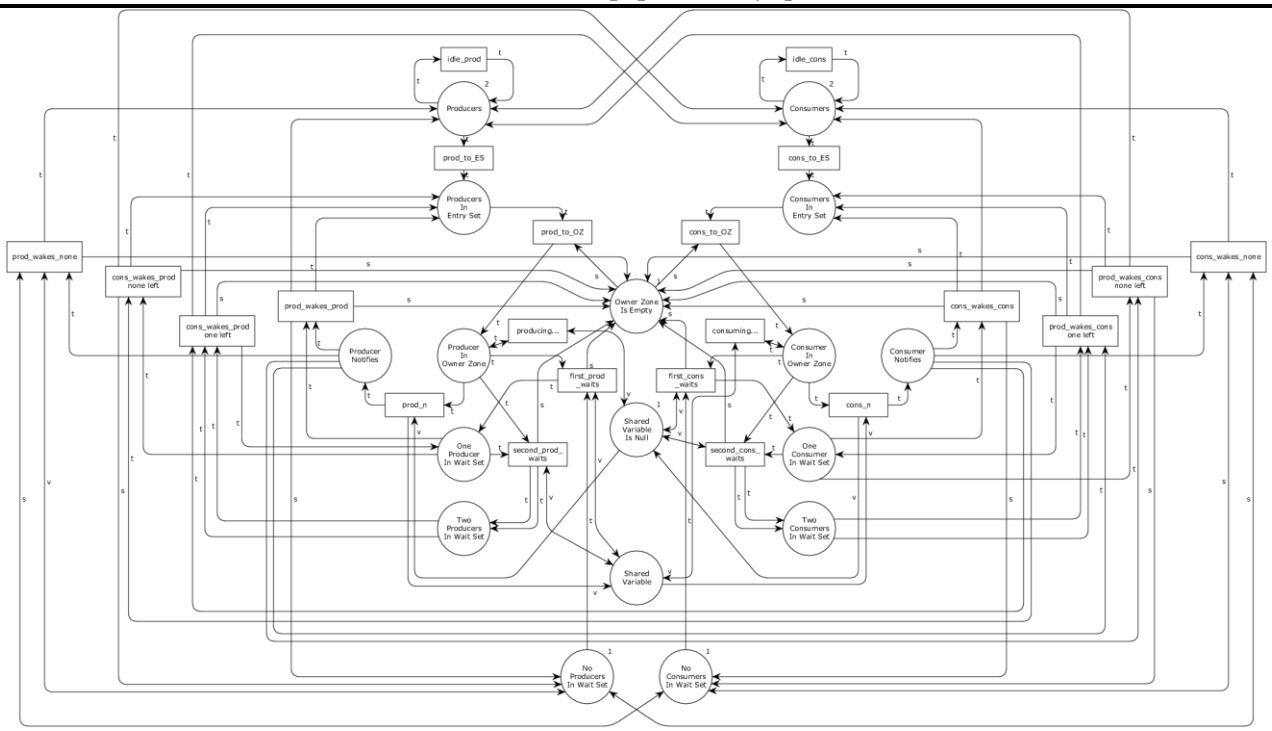


Рис. 3. КМП-модель із кількісними фішками

Аналіз КМП із кількісними фішками

Аналіз графа досяжності даної КМП-моделі, побудованого засобами CPN Tools, дозволяє дійти таких висновків щодо її властивостей:

- обмеженість (граф досяжності включає в себе 176 станів, що є скінченною величиною, а отже, досліджувана КМП є обмеженою);
- повторюваність (див. вище);
- несуперечливість (див. вище);
- взаємовиключність (в парі місць *Producer In Owner Zone / Consumer In Owner Zone* може перебувати не більше однієї фішки одночасно, і якщо вона є, місце *Owner Zone Is Empty* обов'язково порожнє);
- живучість (мертві переходи відсутні, мертві маркування: $\mathbf{M_{167}}$ = (0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 2, 2, 0, 0, 0) та $\mathbf{M_{176}}$ = (0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 2, 2, 1, 0, 0)).

Отже, при переході від КМП-моделі із якісними фішками до КМП-моделі із кількісними фішками властивості початкової моделі збереглися.

Матриця інцидентності **A** розмірності 18×17 (переходи *idle_prod, idle_cons, producing…* та *consuming…*, в яких вхідна позиція співпадає із вихідною, не враховуються) має такий вигляд:

$$A = \begin{pmatrix}
-1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & -1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & -1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -1 & -1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 \\
0 & 0 & 0 & 0 & -1 & 0 & 0 & -1 & -1 & 0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & -1 & 0 & -1 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & -1 & 0 & -1 \\
0 & 0 & 0 & 0 & 0 & -1 & -1 & 0 & 0 & -1 & -1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & -1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & -1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & -1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & -1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -2 & 0 & 0 & 0 & 0 & 0 & 2 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -2 & 0 & 0 & 0 & 0 & 0 & 2 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0
\end{pmatrix}$$

Для перевірки даної КМП на наявність дедлоків побудуємо систему лінійних однорідних діофантових нерівностей над множиною {0, 1}, еквівалентну системі логічних рівнянь, яка описує властивість дедлоків. А саме: якщо довільний перехід має вихідне місце, яке належить дедлоку Q, то дедлоку Q має належати і його вхідне місце [10, С. 166].

Для даної КМП матимемо такі множини вхідних та вихідних місць переходів (номери переходів і місць відповідають номерам відповідно стовпчиків і рядків матриці інцидентності):

$\bullet t1 = \{P1\}$;	$\bullet t2 = \{P2\}$;	$\bullet t3 = \{P3, P5\}$;
$t1\bullet = \{P3\}$;	$t2\bullet = \{P4\}$;	$t3\bullet = \{P7\}$;
$\bullet t4 = \{P4, P5\}$;	$\bullet t5 = \{P6\}$;	$\bullet t6 = \{P8\}$;
$t4\bullet = \{P8\}$;	$t5\bullet = \{P1, P5\}$;	$t6\bullet = \{P2, P5\}$;
$\bullet t7 = \{P9, P11\}$;	$\bullet t8 = \{P6, P12\}$;	$\bullet t9 = \{P6, P11\}$;
$t7\bullet = \{P2, P3, P5, P16\}$;	$t8\bullet = \{P1, P4, P5, P17\}$;	$t9\bullet = \{P1, P3, P5, P16\}$;
$\bullet t10 = \{P9, P12\}$;	$\bullet t11 = \{P9, P13\}$;	$\bullet t12 = \{P6, P14\}$;
$t10\bullet = \{P2, P4, P5, P17\}$;	$t11\bullet = \{P2, P3, P5, P11\}$;	$t12\bullet = \{P1, P4, P5, P12\}$;
$\bullet t13 = \{P7, P16\}$;	$\bullet t14 = \{P8, P17\}$;	$\bullet t15 = \{P7, P10\}$;
$t13\bullet = \{P5, P11\}$;	$t14\bullet = \{P5, P12\}$;	$t15\bullet = \{P6, P15\}$;
$\bullet t16 = \{P8, P15\}$;	$\bullet t17 = \{P7, P11\}$;	$\bullet t18 = \{P8, P12\}$;
$t16\bullet = \{P9, P10\}$;	$t17\bullet = \{P5, P13\}$;	$t18\bullet = \{P5, P14\}$.

Звідси маємо такі логічні залежності для пошуку дедлоків:

$P1 \in Q \to P6 \wedge (P6 \vee P11) \wedge (P6 \vee P12) \wedge (P6 \vee P14) = P6 \in Q$;
$P2 \in Q \to P8 \wedge (P9 \vee P11) \wedge (P9 \vee P12) \wedge (P9 \vee P13) = (P8 \wedge P9) \vee (P8 \wedge P11 \wedge P12 \wedge P13) \in Q$;
$P3 \in Q \to P1 \wedge (P6 \vee P11) \wedge (P9 \vee P11) \wedge (P9 \vee P13) = (P1 \wedge P11 \wedge P9) \vee (P1 \wedge P6 \wedge P9) \vee (P1 \wedge P11 \wedge P13) \in Q$;
$P4 \in Q \to P2 \wedge (P6 \vee P12) \wedge (P9 \vee P12) \wedge (P6 \vee P14) = (P2 \wedge P6 \wedge P12) \vee (P2 \wedge P6 \wedge P9) \vee (P2 \wedge P12 \wedge P14) \in Q$;
$P5 \in Q \to P6 \wedge P8 \wedge (P9 \vee P11) \wedge (P6 \vee P12) \wedge (P6 \vee P11) \wedge (P9 \vee P12) \wedge (P9 \vee P13) \wedge (P6 \vee P14) \wedge (P7 \vee P16) \wedge (P8 \vee P17) \wedge (P7 \vee P11) \wedge (P8 \vee P12) = (P6 \wedge P8 \wedge P7 \wedge P9) \vee (P11 \wedge P9 \wedge P16) \vee (P7 \wedge P11 \wedge P12 \wedge P13) \vee (P11 \wedge P16 \wedge P12 \wedge P13) \in Q$;

$P6 \in Q \to P7 \vee P10 \in Q$;
$P7 \in Q \to P3 \vee P5 \in Q$;
$P8 \in Q \to P4 \vee P5 \in Q$;
$P9 \in Q \to P8 \vee P15 \in Q$;
$P10 \in Q \to P8 \vee P15 \in Q$;
$P11 \in Q \to (P9 \vee P13) \wedge (P7 \vee P16) \in Q$;
$P12 \in Q \to (P9 \vee P13) \wedge (P8 \vee P17) \in Q$;
$P13 \in Q \to P7 \vee P11 \in Q$;
$P14 \in Q \to P8 \vee P12 \in Q$;
$P15 \in Q \to P7 \vee P10 \in Q$;
$P16 \in Q \to (P9 \vee P11) \wedge (P6 \vee P11) \in Q$;
$P17 \in Q \to (P6 \vee P12) \wedge (P9 \vee P12) \in Q$.

Ці умови можна переписати у вигляді такої системи логічних формул:

$\neg P1 \vee P6$;
$\neg P2 \vee (P8 \wedge (P9 \vee P11) \wedge (P9 \vee P12) \wedge (P9 \vee P13))$;
$\neg P3 \vee (P1 \wedge (P6 \vee P11) \wedge (P9 \vee P11) \wedge (P9 \vee P13))$;
$\neg P4 \vee (P2 \wedge (P6 \vee P12) \wedge (P9 \vee P12) \wedge (P6 \vee P14)) \in Q$;
$\neg P5 \vee (P6 \wedge P8 \wedge (P9 \vee P11) \wedge (P6 \vee P12) \wedge (P6 \vee P11) \wedge (P9 \vee P12) \wedge (P9 \vee P13) \wedge (P6 \vee P14) \wedge (P7 \vee P16) \wedge (P8 \vee P17) \wedge (P7 \vee P11) \wedge (P8 \vee P12))$;
$\neg P6 \vee P7 \vee P10$;
$\neg P7 \vee P3 \vee P5$;

$\neg P8 \vee P4 \vee P5$;
$\neg P9 \vee P8 \vee P15$;
$\neg P10 \vee P8 \vee P15$;
$\neg P11 \vee ((P9 \vee P13) \wedge (P7 \vee P16))$;
$\neg P12 \vee ((P9 \vee P13) \wedge (P8 \vee P17))$;
$\neg P13 \vee P7 \vee P11$;
$\neg P14 \vee P8 \vee P12$;
$\neg P15 \vee P7 \vee P10$;
$\neg P16 \vee ((P9 \vee P11) \wedge (P6 \vee P11))$;
$\neg P17 \vee ((P6 \vee P12) \wedge (P9 \vee P12))$.

Ці формули можна записати у вигляді системи лінійних однорідних діофантових нерівностей (далі СЛОДН) з коефіцієнтами із множини {-1, 0, 1}. Через велику кількість нерівностей в цій системі (41) наведемо тільки перші 9 із них (для ¬P1, ¬P2 та ¬P3):

$$A * x = \begin{cases} -1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \geq 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \geq 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & \geq 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & \geq 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & \geq 0 \\ 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \geq 0 \\ 0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & \geq 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & \geq 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & \geq 0 \\ & & & & & & & & \dots & & & & & & & & & \end{cases}$$

Розв'язавши цю систему (наприклад, за алгоритмом Аджілі-Контежан [18]), отримаємо множину із 3638 векторів, до якої входить, зокрема, вектор v_{max} = (1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1), який означає, що всі місця даної КМП складають дедлок. Найбільше нас цікавлять мінімальні дедлоки, тобто такі, до яких не входять інші дедлоки. Таких буде 10:

- v_1 = (0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0),
- v_2 = (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0),
- v_3 = (0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0),
- v_4 = (1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0),
- v_5 = (0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0),
- v_6 = (1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0),
- v_7 = (0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0),
- v_8 = (0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0),
- v_9 = (0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0),
- v_{10} = (1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0).

Аналіз інваріантів для даної КМП дозволяє дійти таких висновків щодо її властивостей:

- обмеженість. Множина S-інваріантів складається з таких векторів:
  - s_1 = (0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0);
  - s_2 = (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 1, 0, 0, 2, 0);
  - s_3 = (1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0);
  - s_4 = (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 1, 0, 0, 2);
  - s_5 = (0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0);
  - s_6 = (0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0).

Усі місця покриваються ненульовими координатами з множини S-інваріантів, а отже, дана КМП обмежена. Позбувшись інгібіторних дуг, ми позбулися неоднозначності у вигляді нульових стовпців, які вказували на потенційну наявність необмежених місць.

- Повторюваність. Множина T-інваріантів (неочищена від векторів, які є лінійними комбінаціями інших елементів множини) налічує 1 547 052 векторів, тож наводити її ми не будемо. Утім, ненульові координати векторів з цієї множини покривають всі переходи КМП, тож вона є повторюваною.
- Несуперечливість. Початкове маркування $\mathbf{M_0}$ є досяжним із самого себе через таку послідовність переходів σ, що кожен перехід входить до σ принаймні один раз.
- Взаємовиключність. Для перевірки на взаємовиключність слід переконатись, що 1) зона власника не може бути одночасно зайнята і вільна, та 2) в зоні власника може перебувати тільки один потік одночасно. Для доведення істинності твердження 1 треба показати недосяжність таких маркувань:
  - M_{11} = (2, 2, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1),
  - M_{12} = (1, 2, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1),
  - M_{13} = (2, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1)
  - та інших з фішкою у позиції *Owner Zone Is Free* і однією фішкою в одній з позицій *Producer/Consumer In Owner Zone.*

Для доведення істинності твердження 2 треба показати недосяжність таких маркувань:

- M_{21} = (0, 2, 0, 0, 0, 0, 2, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1),
- M_{22} = (2, 0, 0, 0, 0, 0, 0, 2, 0, 1, 0, 0, 0, 0, 0, 1, 1),
- M_{23} = (1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1)
- та інших з однією або більше фішками одразу в обох позиціях *Producer/Consumer In Owner Zone.*

Неважко переконатися, що при $\mathbf{M_0}$ = (2, 2, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1) жодне невзаємовиключне маркування, зокрема жодне із наведених вище в якості прикладу, не є досяжним, оскільки їх добуток на щонайменше один із S-інваріантів не дорівнює добутку $\mathbf{M_0}$ на цей самий S-інваріант.

Отже, заміна місць, які можуть містити типізовані фішки, такою кількістю місць, що можуть містити безтипові фішки, яка відповідає

кількості можливих значень для даного типу даних, та заміна інгібіторних дуг додатковими місцями не вплинули на основні властивості моделі, зокрема на наявність дедлоків. Це дає підстави припустити, що описані перетворення вихідної КМП-моделі із якісними фішками на КМП-модель із кількісними фішками є еквівалентними.

Висновки

В статті продовжено роботу з вироблення принципів застосування S- та T-інваріантів до аналізу КМП з метою поширення існуючих методів аналізу МП на КМП та розробки специфічних методів аналізу КМП на основі методів аналізу МП. Для цього побудовано та проаналізовано КМП-моделі взаємодії потоків з використанням комбінації методів `wait()/notify()` із якісними та кількісними фішками. Показано, що побудовані моделі є еквівалентними з точки зору властивостей активності, живучості та взаємовиключності. Це дає підстави стверджувати про можливість переходу з однієї форми до іншої без зміни властивостей, а відтак і без спотворення результатів аналізу моделі. Це дає підстави припустити, що перетворення КМП із якісними фішками на КМП із кількісними фішками є еквівалентними, оскільки вони зберігають властивості початкових КМП.

У підсумку КМП-моделі із якісними та кількісними фішками, побудовані нами в роботі [16] та цій статті, та результати аналізу за допомогою алгоритму визначення мінімальної породжуючої множини S- та T-інваріантів, розв'язують задачу знаходження способу використання цього алгоритму до аналізу властивостей КМП. В якості об'єктів дослідження було обрано не умовні та якимось специфічним чином налаштовані МП, а реальні МП- та КМП-моделі, побудовані в рамках розв'язання практично значущих прикладних задач, а отже, є підстави стверджувати, що описані у статті підходи є універсальними. Утім, остаточне доведення цього твердження вимагає додаткових досліджень.

Також отримані результати показують, що КМП із кількісними фішками подібні до класичних МП. Встановлення цієї подібності є важливим кроком до створення засобів автоматизованого аналізу КМП та автоматизованого перетворення МП у КМП — потужнішу за МП мову, а тому перспективнішу з точки зору проведення подальших поглиблених досліджень.

Логічним подальшим розвитком запропонованої роботи вбачається строге доведення еквівалентності КМП-моделей із якісними та кількісними фішками, а відтак і універсальності запропонованих підходів. Це стане основою для розробки алгоритму та програмних засобів для автоматизованого перетворення КМП із якісними фішками на КМП із кількісними фішками. Також вбачається доцільним поширення описаного у статті методу на інші види мереж Петрі, зокрема на часові, стохастичні та структурні. Також перспективним видається застосування до МП та КМП із кількісними фішками засобів аналітичної комбінаторики та побудова їхніх узагальнених породжуючих функцій [19-21].

Список посилань

1. Jensen K. Coloured Petri Nets. Modelling and Validation of Concurrent Systems / K. Jensen, L. M. Kristensen – Berlin: Springer, 2009. – 384 p.
2. van der Aalst W. M. P. Modeling Business Processes – A Petri Net-Oriented Approach / W. M. P. van der Aalst, C. Stahl. – The MIT Press, 2011. – 400 p.
3. Методи та новітні підходи до проектування, управління і застосування високопродуктивних IT-інфраструктур / Бойко Ю. В., Волохов В. М., Глибовець М. М. та ін. – К.: ВПЦ «Київський університет», 2015. – 376 с.
4. Глибовець М. М. Моделі та методи створення і супроводу високопродуктивного розподіленого навчального середовища: Автореф. дис. на здобуття наук. ступеня д-ра фіз.-мат. наук : спец. 01.05.03 «Математичне та програмне забезпечення обчислювальних машин і систем» / М. М. Глибовець. – К., 2006. – 36 с.
5. Гломозда Д. К. Моделювання роботи колаборативного середовища за допомогою кольорових мереж Петрі / Д. К. Гломозда // Наукові записки. Т. 163: Комп'ютерні науки / Національний університет «Києво-Могилянська Академія». – К., 2014. – С. 77–85.
6. Godon A. PM Editeur 3.1e. / A. Godon, F. Bousseau. – LisaDev, 1992—96. – Режим доступу: http://www.cremona.polimi.it/dispense/Automazione%20Industriale/pm_us.zip

7. Svadova M. Matlab Toolbox for Petri Nets / M. Svadova, Z. Hanzalek // INOVACE 2000, December 5—9, 2000.: proc. – Praha, 2000. – P. 15.
8. Jensen K. Coloured Petri Nets and CPN Tools for Modelling and Validation of Concurrent Systems / K. Jensen, L. M. Kristensen, L. Wells // International Journal on Software Tools for Technology Transfer. – 2007. – Vol. 9. – No. 3-4. – P. 213–254.
9. Крывый С. Л. О вычислении минимального множества инвариантов сети Петри / С. Л. Крывый // Искусственный интеллект. – 2001. – №3. – С. 199–206.
10. Кривий С. Л. Лінійні діофантові обмеження та їх застосування / С. Л. Кривий. – Чернівці – Київ: Букрек, 2015. – 224 с.
11. Крывый С. Л. О применении сетей Петри к решению некоторых задач телекоммуникации / Крывый С. Л., Матвеева Л. Е. // Искусственный интеллект. – 2002. – №3. – С. 590–598.
12. Матвеева Л. Е. Автоматическая система анализа и верификации телекоммуникационной системы, описанной на языке MSC, с помощью формализма сетей Петри / Л. Е. Матвеева // Проблеми програмування. – 2004. – № 2-3. – С. 108-117.
13. Лукьянова Е. А. Исследование однотипных структурных элементов CN-сети в процессе компонентного моделирования и анализа сложной системы с параллелизмом / Е. А. Лукьянова, А. В. Дереза // Кибернетика и системный анализ. – 2012. – № 6. – С. 20—29.
14. Шелестов А. Ю. Моделирование Grid-узла на основе сетей Петри / А. Ю. Шелестов // Системні дослідження та інформаційні технології. – 2009. – №3. – С. 52-65.
15. Jensen K. Coloured Petri Nets and the Invariant Method / Kurt Jensen // Theoretical Computer Science. – 1981. – Vol. 14. – P. 317–336.
16. Гломозда Д. К. Застосування методу інваріантів до аналізу кольорових мереж Петрі / Д. К. Гломозда // Наукові записки. Т. 177: Комп'ютерні науки / Національний університет «Києво-Могилянська Академія». – К., 2015. – С. 44–52.
17. Venners B. Thread Synchronization [Electronic resource] / Bill Venners – Mode of access: https://www.artima.com/insidejvm/ed2/threadsynch.html. – Title from the screen.
18. Ajili F. Avoiding Slack Variables in the Solving of Linear Diophantine Equations and Inequations / Ajili F., Contejan E. // Theoretical Comp. Science. – 1997. – V. 173. – P. 183–208.
19. Flajolet P. Analytic Combinatorics / Philippe Flajolet, Robert Sedgewick. — Cambridge: Cambridge University Press, 2009. – 810 p.
20. Ландо С. К. Лекции о производящих функциях / Сергей Константинович Ландо. — М.: МЦНМО, 2007. – 144 с.
21. Ландо С. К. Введение в дискретную математику / Сергей Константинович Ландо. — 2012. – Режим доступа: http://vyshka.math.ru/pspdf/1213/discr/LandoBook.pdf.

УДК 004.383

СЕРГІЄНКО А. М.,
КЛИМЕНКО І. А.,
СЕРГІЄНКО П. А.

РЕКОНФІГУРОВАНА БАГАТОПРОЦЕСОРНА ОБЧИСЛЮВАЛЬНА СИСТЕМА НА ПЛІС

Запропоновано підхід до проектування реконфігурованих обчислювальних систем на програмованих логічних інтегральних схемах, який ґрунтується на відображенні графів синхронних потоків даних у багатопроцесорну систему, в якій реконфігурація виконується через перемикання потоків даних. Для реалізації процесорних елементів такої системи розроблено шістнадцятирозрядне ядро RISC-процесора, який має малі апаратні витрати та систему команд, що настроюється.

An approach to the reconfigurable computer design based on FPGAs is proposed. It is based on mapping the synchronous dataflow graphs into the many-core processor system in which the reconfiguration means the dataflow switshing. A soft core of the 16-bit RISC microprocessor is designed for this system, which has small hardware volume and configurable instruction set.

Ключові слова: ПЛІС, конфігурований комп′ютер, RISC.

1. Вступ

Впровадження програмованих логічних інтегральних схем (ПЛІС), як основи конфігурованих комп′ютерів, є сталою тенденцією. ПЛІС, в якій реалізовано високопродуктивний спеціалізований обчислювач, програмується один раз для виконання певної задачі або вона повністю чи частково перепрограмується кілька разів під час виконання цієї задачі, забезпечуючи оптимальну обчислювальну структуру для різних фрагментів алгоритму. В останньому випадку кажуть про динамічно реконфігурований комп′ютер (ДРК) [1,2,3].

Як правило, в ДРК реалізуються потокові алгоритми, для яких можна передбачити об′єми порцій обчислень та міжпроцесорні комунікації. При розробці ДРК алгоритм моделюють, виділяють множини обчислень, які відображають на процесорні елементи (ПЕ), визначають потрібні варіанти процесорних структур та виробляють набір конфігураційних файлів, які завантажуються динамічно у ПЛІС під час роботи ДРК [1].

Нові серії ПЛІС фірм Altera і Xilinx підтримують часткову реконфігурацію. Але, по-перше, мінімальна порція реконфігурації покриває прямокутну область ПЛІС, яка є доволі великою. Наприклад, в ПЛІС Xilinx ця область включає чотири блоки множення DSP48. Крім того, частина ПЛІС, яка не змінила конфігурацію і острівець з новою конфігурацією повинні мати незмінні спільні точки передачі даних. Все це призводить до складнощів проектування ДРК та обмежень до оптимізації апаратних витрат. По-друге, мінімальна часткова реконфігурація триває кілька мілісекунд, тобто, вона складає великі часові накладні витрати [2].

Ці накладні витрати можна мінімізувати завдяки кешуванню прошивки [3]. Але у ПЛІС на зберігання прошивки вже використовується близько 60% транзисторів кристалу. Тому додаткові апаратні витрати на збереження прошивки призводять як до нераціонального використання ресурсів ПЛІС, так і до зменшення швидкодії конфігурованих схем [2].

Нові серії ПЛІС мають ряд особливостей, що впливають на вибір архітектури модулів, які в них конфігуруються. Якщо число транзисторів у ПЛІС, збільшується учетверо, то об′єм ресурсів для трасування сигналів збільшується лише удвічі. Затримки у лініях зв′язку стали в 1–3 рази більшими за затримки у логічних таблицях (ЛТ) та інших логічних елементах [2]. Як результат, конфігуровані модулі мають бути компактними, щоби їх внутрішні сигнали далеко не поширювались.

Часові та апаратні накладні витрати на реконфігурацію ДРК можна скоротити, якщо при цьому взагалі не змінювати прошивку ПЛІС. У багатьох роботах, наприклад в [4], пропонується багатопроцесорна архітектура ДРК, яка складається з множини однакових компактних ПЕ, які зв′язані конфігурованою системою комутації. Така система може швидко перепрограмуватись динамічно. Недоліком цієї архітектури є той, що ПЕ є доволі спеціалізованими,

що накладає обмеження на множину алгоритмів, що реалізуються.

Як конфігурований ПЕ варто використовувати програмно кероване процесорне ядро зі спеціалізованою системою команд. При реалізації в ПЛІС такий процесор може динамічно змінювати виконуваний алгоритм, завдяки оверлейній структурі програм. Крім того, шляхом часткової реконфігурації можна динамічно змінювати систему команд, не зачіпаючи спільні точки передачі даних.

Вимозі компактності не відповідають ядра мікропроцесорів з поширеною архітектурою, такі як Nios, Microblaze, Mico32, OpenRISC, MIPSFPGA. Для досягнення прийнятної тактової частоти ці процесори мають багатоступеневий конвеєр (5 і більше ступенів). Для виконання кількох незалежних програмних потоків у такому ядрі реалізуються система переривання та зміни контексту, віртуальна пам'ять, кеш-ОЗП, швидкісний інтерфейс до зовнішньої динамічної пам'яті. Для реалізації мультипроцесорної архітектури необхідні комутатори з високою пропускною здатністю та система підтримки режиму когерентності доступу пам'яті [5].

Метою роботи є побудова багатопроцесорного ДРК у ПЛІС, який реалізує модель синхронних потоків даних, завдяки чому зменшуються апаратні витрати, збільшуються швидкодія системи та можливості динамічного реконфігурування. Для цього було розроблене компактне ядро RISC-процесора, яке грає роль ПЕ такої системи.

2. Система синхронних потоків даних

Граф синхронних потоків даних (ГСПД) є зручною моделлю для завдання багатьох алгоритмів з циклічною природою, таких як алгоритми цифрової обробки сигналів. ГСПД складається з вершин, які виконують обчислювальні функції або актори і які з'єднані між собою дугами у систему. Кожен потік-дуга може мати буферну пам'ять типу FIFO. Актор спрацьовує негайно, як тільки є дані на його входах і видає результати на свої виходи. На відміну від графу потоків даних, в ГСПД протягом циклу виконання алгоритму кожен актор споживає і видає на свої виходи одну і ту саму кількість даних.

Модель ГСПД має такі властивості, як відсутність блокувань, статичне складання розкладу, можливість виконання великої множини алгоритмів, простота побудови алгоритмів для неї. Завдяки цьому, вона використовується як для програмування багатопроцесорних систем [6], так і для проектування структур конвеєрних обчислювачів [7, 8, 9].

При одиничному відображенні алгоритму, який задано на ГСПД, структура системи є ізоморфною цьому графу. Кожен процесорний елемент (ПЕ) такої структури виконує єдиний процес обчислення відповідної функції-актора. При появі чергової групи даних у вхідних портах процес запускається, обчислює функцію, записує результати у вихідні порти і зупиняється.

При відображенні ГСПД з N акторів у систему з n ПЕ кожен з них виконує послідовно до $k =]N/n[$ процесів обчислення суміжних акторів. При цьому програмування ПЕ виконується за методами, які обґрунтовані в [6, 10]. Отже, ПЕ системи виконують лише по одному програмному потоку. Для них не потрібна операційна система, система переривань, віртуальна пам'ять, тощо. Тобто, такі ПЕ можуть мати невеликі апаратні витрати і вони будуть компактними.

Цикл обчислення алгоритму ГСПД у такій паралельній системі визначається критичним шляхом передачі і обчислення даних, який проходить через максимально завантажені ПЕ. Решта ПЕ, через які не проходить критичний шлях, будуть недовантаженими [6, 7]. Тому є задача балансування завантаженості ПЕ.

3. Динамічне реконфігурування системи

Задача, яка потребує динамічного реконфігурування, має бути представлена як набір різних ГСПД. Тоді ці ГСПД відображаються у кілька багатопроцесорних структур, які відрізняються кількістю ПЕ та графами їх з'єднань. Ці структури об'єднуються в одну загальну результуючу структуру ДРК, використовуючи відповідні комутаційні вузли.

Завдання, які виконуються у вузлах структури, описуються як програми на відповідній мові програмування і транслюються у машинні коди ПЕ. При цьому виділяється множина команд використаних і віртуальні модулі ПЕ налаштовуються під ці множини, як це запропоновано у роботі [10].

Одержаний проект багатопроцесорної системи компілюється у прошивку ПЛІС разом з програмними кодами за звичайною методикою, використовуючи САПР виробника ПЛІС.

При вирішенні задачі ПЕ у ДРК виконують свої програми, а його реконфігурація полягає лише у відповідній комутації комутаційних

вузлів та у перезавантаженні програм у ПЕ з зовнішньої пам'яті. Такий процес реконфігурації відбувається значно швидше, ніж при повній чи частковій зміні конфігурації ПЛІС.

4. Архітектура ПЕ багатопроцесорної системи

Як ПЕ багатопроцесорної системи синхронних потоків даних, було розроблене 16-розрядне процесорне ядро RISC-ST2. За основу було взяте ядро RISC-ST, яке описане у [9]. Довжина команди була збільшена до 18 розрядів, що дало змогу оптимізувати систему команд та краще задіяти об'єм вбудованої пам'яті ПЛІС.

До ядра додано множину спеціалізованих команд обробки окремих бітів слів, виділення бітових полів заданої довжини, злиття полів, зсуву слів, підрахунку числа нульових старших розрядів. Для реалізації швидкого доступу до асоціативної таблиці введено команду обчислення хеш-функції. Такі команди сприяють ефективній реалізації алгоритму компресії, такому як LZW.

Множина використаних команд програмується під час формування конфігураційного файлу, так як це запропоновано в роботі [10]. Як результат, одержуються невеликий ПЕ і програмні коди скороченого об'єму, які повністю вміщуються всередині ПЛІС.

Регістрова пам'ять процесора має 32 регістри, але командно доступними є 16 молодших регістрів. Щоби максимально задіяти можливості регістрової пам'яті, яка реалізована на ЛТ, у процесорі протягом одного такту виконується читання з трьох регістрів та запис у один регістр.

Використовуються такі види адресації, як регістрова, непряма регістрова, базова, індексна з преінкрементом. Пам'ять даних розбита на сторінки по 256 байтів і має максимальну ємність 16 мегабайт. Для доступу до пам'яті даних використовується стандартний відкритий інтерфейс Wishbone.

Адресно доступними є до 256 периферійних регістрів. Такі регістри слугують регістрами вводу-виводу спецпроцесорів, які виконують швидкісні обчислення, як наприклад, обчислення елементарних функцій, цифрова обробка сигналів, шифрування. Кілька ядер можуть бути об'єднані у систему через їхні регістри вводу-виводу та систему переривань.

Завдяки тому, що конвеєр команд є трьохступеневим, більшість команд виконується за один такт, а команди переходу та читання пам'яті – за два такти. Виклик підпрограм виконується також за два такти, під час чого адреса повернення та прапорці умов зберігаються у апаратному стеку.

Ядро мікропроцесора описане мовою VHDL і не має обмежень для синтезу та конфігурування у ПЛІС будь-якої серії. Модель процесора має вбудований дизасемблер, який спрощує тестування та відлагодження програм. Розроблено програму кросасемблера на мові Java, виходом якої є VHDL-файли блоків пам'яті даних та програм процесора.

5. Реалізація процесорного ядра у ПЛІС

У таблиці 1 показані результати синтезу процесора для ПЛІС Xilinx Kintex-7 та його найближчих поширених аналогів. У порівнянні з аналогами розроблене ядро має менші апаратні витрати і більшу швидкодію. Слід відмітити, що на затримку у лініях міжз'єднання у даному процесорі припадає 75% затримки критичного шляху. Компактність ядра дає змогу розмістити близько трьохсот ядер у ПЛІС середнього об'єму Xilinx xc7k480t, кожне з яких матиме власну пам'ять у 12 кілобайт

Табл. 1. Реалізація мікропроцесорних ядер у ПЛІС Xilinx Kintex-7

Ядро мікропроцесора	Розрядність	Апаратні витрати, ЛТ	Макс. тактова частота, МГц
RISC-ST2	16	653	217
OpenMSP430	16	1387	150
OpenRISC1200	32	4945	107

Ядро RISC-ST2 може бути зконфігуроване у ПЛІС різних серій. В таблиці 2 показані результати такого конфігурування.

Табл. 2. Реалізація ядра RISC-ST2 у ПЛІС

Серія ПЛІС	Апаратні витрати, ЛТ	Макс. тактова частота, МГц
Xilinx Zynq	634	206
Xilinx Spartan6	701	140
Xilinx Artix7	625	150
Altera StratixV	723	240
Altera Cyclone V	999	132
Altera Max10	1522	80
Lattice ECP5U	742	124

Висновки

Запропоновано підхід до створення багатопроцесорних систем на кристалі на основі відображення ГСПД, який дає змогу отримати програмовану високопродуктивну систему для об-

робки потоків даних зі зменшеними апаратними витратами.

Розроблено ядро RISC-процесора зі спеціалізованою системою команд. Завдяки тому, що процесор орієнтований на виконання лише одного обчислювального процесу та обробки переривань, він має невеликі апаратні витрати та високу швидкодію.

На основі цього ядра можлива розробка багатопроцесорних систем на кристалі для обробки потоків даних з продуктивністю у кілька десятків мільярдів операцій за секунду. Планується створення системи, яка виконуватиме швидку декомпресію файлів за алгоритмом LZW.

Список посилань

1. DeHon A. Reconfigurable Computing Architectures / R. Tessier, K. Pocek, A. DeHon // Proceedings of the IEEE. – V. 103. –No. 3. – 2015. – P. 332 –354.
2. Koch D. Partial Reconfiguration on FPGAs. Architectures, Tools and Applications / D. Koch // – New York: Springer. – 2013. – 295 P.
3. Клименко І. А. Оптимізація реконфігурації в динамічно реконфігурованих обчислювальних системах // Вісник НТУУ «КПІ», сер. Інформатика, управління та обчислювальна техніка. – 2015. – №63. – С.93-100.
4. Lanuzza M. MORA: A New Coarse-Grain Reconfigurable Array for High Throughput Multimedia Processing // M. Lanuzza, S. Perri, P. Corsonello // Proc. 7-th Int. Conf. on Embedded Computer Systems, SAMOS'07. – 2007. – P. 159 – 168.
5. Processor Design. System-on-Chip Computing for ASICs and FPGAs / Edited by J. Nurmi. – Springer. – 2007. – 525 p.
6. Lee E. A. Software Synthesis from Dataflow Graphs / S.S. Bhattacharyya, P.K. Murthy, E.A. Lee // – Springer. – 1996. – 190 p.
7. Bhattacharyya S. S. Mapping Parameterized Cyclo-Static Dataflow Graphs onto Configurable Hardware / H. Kee, C.-C. Shen, S. Bhattacharyya, I. Wong, Y. Rao, and J. Kornerup // Journal of Signal Processing Systems. – V. 66. – No 3. – 2012. – P. 285–301.
8. Сергиенко А.М. Алгоритмические модели обработки потоков данных / А.М. Сергиенко, В.П. Симоненко // Электронное моделирование. –2008. –Т.30, №6. –С. 49–60.
9. Сергиенко А.М. VHDL для проектирования вычислительных устройств. – К.: ДиаСофт. – 2003. – 210 с.
10. Sergyienko A. Configurable Microprocessor Array for DSP Applications / O. Maslennikov, Ju. Shevtshenko, A. Sergyienko // LNCS. – Berlin: Springer. – V. 3019. – 2004. – P. 36–41.

УДК 004.056.5

КОСТЕНКО Ю. В.,
МАРКОВСКИЙ А .П.,
РУСАНОВА О .В.

МЕТОД ЗАЩИЩЕННОГО МОДУЛЯРНОГО ЭКСПОНЕНЦИРОВАНИЯ НА УДАЛЕННЫХ КОМПЬЮТЕРНЫХ СИСТЕМАХ

В статье предложен способ удаленного модулярного экспоненцирования в облачных системах с защитой данных и кода экспоненты. Предложенный способ позволяет разделить процедуру модулярного экспоненцирования на две составляющие, одна, меньшая часть которых выполняется пользователем, а другая, большая - в удаленных мощных вычислительных системах. Подробно описаны математические принципы предлагаемого разделения вычислений, изложена методика организации вычисления. Приведены числовые примеры удаленного защищенного вычисления экспоненты.

In this paper a method for the performing the calculations required for modular exponentiation using remote or distant computational resources. The proposed method operates by separating the procedure for modular exponentiation in two components. The first component which is computationally simple is performed on the user terminal and the second and computationally complex component, is executed on powerful cloud computational resources. The details of the proposed distributed calculation are presented. Hence, the methodology for the organization of the calculations is analyzed. The calculation is illustrated by means of a simple numerical example.

Ключевые слова: розподілені віддалені обчислення, модулярне експоненціювання, хмарні обчислення, криптографія з відкритим ключем, захищені обчислення.

Введение

Одним из наиболее значимых событий текущего тысячелетия в области информационных технологий стало появление облачных вычислений. Новая технология позволяет предоставить значительные по объёму вычислительные ресурсы для решения различных пользовательских задач. Возможность удаленного доступа к таким ресурсам существенно повышает уровень информатизации во всех сферах человеческой деятельности. Вместе с тем, развитие облачных технологий порождает ряд проблем, одной из которых может быть возможность предоставления ресурсов для решения задач, связанных со взломом протоколов криптографической защиты. Таким образом, внедрение облачных технологий объективно требует повышения уровня защищенности существующих систем защиты информации [1].

Математической основой большинства сетевых проколов криптографической защиты информации является модулярное экспоненцирование, то есть вычисление $A^E \bmod M$ [1]. Уровень защищенности этих протоколов полностью определяется разрядностью используемых в операции модулярного экспоненцирования чисел. В настоящее время в большинстве протоколов используется разрядность 2048 [2], очевидным путем повышения уровня защищенности является увеличение разрядности до 4096 или 8192. Рост разрядности имеет следствием экспоненциальное возрастание вычислительной сложности реализации сетевых протоколов защиты информации. Эта проблема становится особенно ощутимой для маломощных терминальных и мобильных устройств, поддерживающих сетевые протоколы. Выполнение на таких устройствах модулярного экспоненцирования на разрядностях, больших 2048, может привести к нарушению временных рамок работы протокола. В таком случае, актуальной становится задача повышения производительности операции модулярного экспоненцирования на маломощных терминальных и мобильных устройствах, поддерживающих протоколы криптографической защиты. Одной из её возможностей является использование удаленных вычислительных ресурсов облачных технологий. Однако для реализации этой возможности необходимым условием является организация удаленных вычислений без передачи пользовательских данных в сеть. Другими словами, при вычислении $A^E \bmod M$ необходимо учесть: M является частью открытого ключа и может передаваться в явном виде, компоненты E и A являются секретным ключом и секретными данными соответственно, и их передача по сети в явном виде должна быть исключена. Таким образом, научная

задача организации удаленного вычисления модулярной экспоненты, которая исключает передачу секретных данных, является актуальной и важной на современном этапе развития информационных технологий.

Анализ известных методов защищенной реализации модулярного экспоненцирования в открытых удаленных системах

Одним из основных недостатков облачных технологий, существенным образом ограничивающим их применение, является возможность несанкционированного доступа к данным пользователя при передаче и их обработке на удаленных вычислительных мощностях. Проблеме защиты данных пользователя в системах удаленной обработки посвящено в последние годы значительное число работ. Основная проблема состоит в том, что не существует единого подхода к защите данных в процессе их обработки. Фактически, подавляющая часть выполненных исследований решает проблему защиты данных только для отдельных классов вычислительных задач, например, для линейной алгебры, обработки изображений и т.п. [2]. Широким фронтом ведутся исследования, направленные на создание эффективной защищенной организации выполнения в открытых удаленных системах операции модулярного экспоненцирования – базовой процедуры широкого класса протоколов защиты информации [3]. Их анализ показал, что решение задачи защищенного модулярного экспоненцирования состоит в разделении вычислений на две части – одна, большая часть, выполняется на удаленных вычислительных мощностях, другая, меньшая по объему – на компьютерной системе пользователя. В качестве критериев эффективности организации удаленного вычисления модулярной экспоненты логичным представляется использовать:

– уровень защищенности, мерой которого является объем ресурсов, которые нужно затратить для восстановления секретных компонент операции по кодам передаваемых в открытую систему данных;

– соотношение числа операций выполняемых на процессоре пользователя при использовании удаленных вычислений и числа операций при условии, что все модулярное экспоненцирование осуществляется пользователем.

Последний критерий позволяет оценить возможность ускорения реализации сетевых протоколов защиты информации за счет использования возможностей современных облачных технологий без ущерба для информационной безопасности.

В работе [4] предложен метод удаленного вычисления модулярной экспоненты на основе случайного разделения кода экспоненты E на группы разрядов. Это позволяет организовать вычисление $A^E \mathrm{mod} M$ в виде произведения частичных экспонент, которые могу вычисляться независимо на удаленных вычислительных мощностях облаков. Формирование произведения, то есть формирование результата $A^E \mathrm{mod} M$ осуществляется пользователем. Для защиты компоненты A данных используется выполнение нескольких первых шагов экспоненцирования на компьютере пользователя.

Существенным достоинством рассмотренного метода является то, что в полной мере могут быть использованы возможности многопроцессорных удаленных систем по параллельному вычислению частичных экспонент. В работе [4] на основе теоретических и экспериментальных данных установлено, что описанный метод позволяет примерно в три раза реально повысить производительность выполнения модулярного экспоненцирования.

В работе [3] предложен метод удаленного вычисления модулярной экспоненты на основе поэтапного разложения компоненты A. Такой подход позволяет разложить задачу вычисления $A^E \mathrm{mod} M$ на ряд экспонент, которые оперируют с модифицированными данными и могут также вычисляться параллельно на удаленных вычислительных мощностях. Метод позволяет примерно в 2-3 раза ускорить процесс модулярного экспоненцирования. Еще один интересный подход предложен в работе [2]. Основной акцент в этих исследованиях сделан на том, чтобы пользователь мог не только удаленно выполнить модулярное экспоненцирование в закрытом режиме, но и косвенно контролировать правильность выполненных операций.

Проведенный анализ современного состояния организации защищенной реализации модулярного экспоненцирования на открытых удаленных вычислительных системах показывает, что основным их недостатком является относительно малая эффективность в плане ускорения выполнения операции модулярного экспоненцирования.

Целью исследований является повышение эффективности использования удаленных вычислительных мощностей, предоставляемых облачными технологиями, для реализации сетевых

криптографических протоколов защиты информации.

Метод защищенного удаленного вычисления модулярной экспоненты

Для достижения поставленной цели предложен метод организации вычисления модулярной экспоненты с использованием возможностей удаленных вычислительных мощностей.

Предложенный метод вычисления $A^E \mathrm{mod} M$ предполагает, что A, E, M – n-разрядные двоичные числа, удовлетворяющие условиям $A<M$, $E<M$. Метод предусматривает нахождение пары чисел G и R таких, что $G^R \mathrm{mod} M=1$. В большинстве сетевых протоколов защиты информации используется алгоритм *RSA*, в котором модуль M образуется путем произведения двух простых чисел p и q. Например, если p=7 и q=11, то M=77. Соответственно, в рамках этого примера могут быть выбраны числа G=19 и R=30, такие, что 1930 mod 77=1. Подбор чисел G и R может производиться с использованием малой теоремы Ферма.

При необходимости вычисления $A^E \mathrm{mod} M$, например, при 1431mod77 (A=14, E=31, M=77) предлагается выполнять следующую последовательность действий:

1. Вычисляется значение $B=G\cdot A \mathrm{mod} M$, в частности, в рамках рассматриваемого примера B=19·14 mod 77 = 35.

2. Случайным образом выбирается число W, разрядность которого значительно меньше разрядности n, например W=3.

3. Вычисляется $V=E-W$; для рассматриваемого примера V=28

4. Числа B, V, M передаются в облако, где вычисляется $C=B^V \mathrm{mod} M$ и полученное значение возвращается пользователю. В рамках рассматриваемого примера в облако передаются B=35, V=28, M=77, где вычисляется $C=35^{28}$mod14=14 и результат – число 14, возвращается пользователю.

5. Одновременно пользователь вычисляет $F=B^{R-V} \mathrm{mod}\ M$; для рассматриваемого примера F=352mod 77=70.

6. Также пользователь независимо вычисляет $D=A^{E-R} \mathrm{mod} M$, значение которого для рассматриваемого примера равно $D=A^1 \mathrm{mod} M$=14.

7. По получении из облака значения С пользователь вычисляет значение модулярного произведения $R=C\cdot F\cdot D \mathrm{mod} M$; для рассматриваемого примера это значение определяется в виде R=14·70·14 mod 77=14. Очевидно, что полученное значение совпадает с 14^{31}mod 77. Конструктивность предложенного метода может быть доказана следующим образом. Учитывая что вычисляемое в облаке значение $C=B^V \mathrm{mod} M=(G\cdot A)^V \mathrm{mod}\ M=(G^V \mathrm{mod} M\cdot A^V \mathrm{mod} M)\ \mathrm{mod} M$, а вычисляемое пользователем значение F можно представить как $F=B^{R-V} \mathrm{mod} M=(G\cdot A)^{R-V} \mathrm{mod} M=(G^{R-V} \mathrm{mod} M\cdot A^{R-V} \mathrm{mod} M) \mathrm{mod} M$, то произведение R можно представить в виде $R=C\cdot F\cdot D\ \mathrm{mod} M=(G^{V}\cdot G^{R-V})\ \mathrm{mod M}\ (A^V\cdot A^{R-V}\cdot A^{E-R}) \mathrm{mod}\ M=(G^R \mathrm{mod} M\cdot A^E \mathrm{mod} M)\ \mathrm{mod} M$. В силу того, что $G^R \mathrm{mod} M$=1, то $R=A^E \mathrm{mod} M$, что и требовалось доказать.

Оценка эффективности

Важнейшим элементом проведенных исследований является оценка эффективности предложенного метода. Оценивание эффективности целесообразно проводить по двум критериям:

1) Достигаемое повышение производительности выполнения операции модулярного экспоненцирования пользователем

2) Влияние на уровень защищенности передачи в открытую систему значений B, V, M.

Повышение производительности проще всего оценить путем подсчета количества операций модулярного умножения. Классический алгоритм модулярного экспоненцирования состоит в последовательной обработке каждого из n разрядов кода экспоненты. Обработка одного разряда состоит из обязательного возведения в квадрат кода предыдущего результата и модулярного умножения результата на A, если текущий бит экспоненты равен 1. Если предположить примерно равное количество единиц и нулей в коде экспоненты, то очевидно, что среднее количество операций модулярного умножения составляет $1.5\cdot n$.

В предлагаемом методе пользователь выполняет следующие операции модулярного умножения: при вычислении $F=A^{E-W} \mathrm{mod} M$ выполняется $1.5\cdot z$ операций модулярного умножения, где z – разрядность R-V; при вычислении $D=A^{E-W} \mathrm{mod}\ M$ выполняется $1.5\cdot y$ операций модулярного умножения, где y – разрядность разности $U=E-W$. Согласно разработанному методу разрядность U малая, равно как и разрядность разности R-V, это означает, что число выполняемых пользователем $1.5\cdot(z+y)$ операций модулярного умножения существенно меньше, чем $1.5\cdot n$. В частности, на практике n=4096, U=32, z<=32, соответственно, при выполнении модулярного экспоненцирования полностью пользователем, число операций

модулярного умножения равно 4096·1.5, а при выполнении модулярного экспоненцирования по предложенному методу пользователю необходимо выполнить только 64·1.5 операций модулярного умножения. Таким образом, для реальной ситуации, количество операций, выполняемых пользователем, сокращается в 64 раза.

Проведенные экспериментальные исследования в целом подтвердили проведенные выше теоретические оценки: время выполнения операций модулярного экспоненцирования реально сократилось в 60 раз при разрядности операндов 4096 и в 28 раз при разрядности операндов 2048.

Для оценки уровня защищенности предлагаемого метода следует учесть, что в распоряжении потенциального злоумышленника могут находиться коды *B* и *V*. Злоумышленник знает, что код *V* отличается от кода *E* в небольшом числе разрядов (не более 32), и хотя перебор 2^{32} вариантов технически реализуем, злоумышленник лишен возможности оценки правильности выбранного варианта, поскольку он не знает кода *A*. Исходя из этого, потенциальной тактикой злоумышленника может быть подбор *A*, что требует перебора 2^{n-1} вариантов. Общее число вариантов, которые необходимо перебрать составляет 2^{n-1+z}. На практике это значение примерно равно разрядности *U* – 24095+32=24127. Перебор такого числа вариантов технически нереализуем.

Классическая задача вскрытия *RSA* методом перебора требует анализа 24096 вариантов. Предложенный метод требует 24127 вариантов, что усложняет задачу взлома и таким образом повышает уровень защищенности.

Выводы

В результате проведенных исследований предложен метод защищенного модулярного экспоненцирования на удаленных вычислительных системах. Защита компонент модулярного экспоненцирования достигается за счет специального маскирования компоненты, которая возводится в степень, а так же модификации кода экспоненты. Теоретически и экспериментально доказано, что предложенный метод позволяет за счет использования удаленных вычислительных ресурсов на порядки уменьшить время выполнения экспоненцирования. Доказано, что за счет введения маскирования предложенный метод позволяет увеличить уровень защищенности компонент модулярного экспоненцирования. Предложенный метод ориентирован в первую очередь на маломощные терминальные микроконтроллеры и мобильные устройства, поддерживающие протоколы защиты информации в сетях, поскольку позволяет значительно сократить время трудоемких операций модулярного экспоненцирования даже для больших разрядностей чисел. Проведенные исследования доказали, что для таких устройств, даже при разрядности 8192 реализация протоколов криптографической защиты не выходит за установленные временные границы.

Список литературы

1. Boroujerdi N. Cloud Computing: Changing Cogitation about Computing/ Boroujerdi N., Nazem S. // IJCSI International Journal of Computer Science Issues, – Vol. 9, – Issue 4. –2012. – No 3. – PP. 169-180.
2. Xiaofeng Chen. New Algorithms for Secure Outsourcing of Modular Exponentiations / Xiaofeng Chen, Jin Li, Jianfeng Ma, Qiang Tang, Wenjing Lou // ESORICS 2012, LNCS 7459, – 2012. – PP. 541-556.
3. Can Xiang. Verifiable and Secure Outsourcing Schemes of Modular Exponentiations Using One Untrusted Cloud Server and Their Application // IACR Cryptology ePrint Archive 2014: PP.500 . – https://eprint.iacr.org/2014/500.pdf
4. Markovskyi O.P. Secure Modular Exponentiation in Cloud Systems. / Oleksandr P. Markovskyi, Nikolaos Bardis, Nikolaos Doukas, Sergej Kirilenko // Proceedings of The Congress on Information Technology, Computational and Experimental Physics (CITCEP 2015), 18-20 December 2015, Krakow, Poland, C. 266-269.

УДК 621.396.6

ГЕРГАЛО А.М.,
РУДАКОВА Г.В.

ПРОБЛЕМИ АВТОМАТИЗАЦІЇ ПРОЦЕСУ ФУНКЦІОНАЛЬНОЇ ДІАГНОСТИКИ ЦИФРОВИХ ПРИСТРОЇВ ОБЛІКУ ЕЛЕКТРИЧНОЇ ЕНЕРГІЇ

У статті аналізуються методи контролю та процес функціональної діагностики пристроїв обліку електричної енергії. Наведено задачі технічної діагностики електронного обладнання. Розглянуто особливості побудови та функціонування сучасних пристроїв обліку електричної енергії. Визначено низку необхідних досліджень щодо створення систем дистанційної діагностики для контролю надійності функціонування сучасного цифрового обладнання в реальному часі.

In the article the methods of process control and functional diagnostics devices of electric energy are analyzes. Tasks of technical diagnostics for electronic equipment are described. Aspects of construction and operation of modern devices of electric energy are considered. Define a circle of research, which is necessary for the establishment of remote diagnostics to monitor the reliability of modern digital equipment in real time.

Ключові слова: функціональна діагностика, повірка лічильника, автоматизація, дистанційна діагностика

1. Вступ

У наш час до однієї з найбільш важливих задач електроенергетики відноситься задача зменшення втрат енергоресурсів. Для підвищення точності обліку електричної енергії на етапах виробництва, розподілення і споживання здійснюється заміна індукційних лічильників на більш сучасні електронні.

Своєчасне виявлення несправностей та неправильного функціонування обладнання можливо при здійсненні регулярного контролю за його роботою в реальному часі.

2. Процес функціональної діагностики електронної техніки

Функціональна діагностика (контроль роботоздатності) проводиться в сертифікованих метрологічних установах (органом державної метрологічної служби або іншою уповноваженою організацією) шляхом їх повірки на спеціалізованих лабораторних установках в заздалегідь визначені терміни. Метою функціонального контролю приладів обліку електричної енергії є визначення відповідності характеристик засобів вимірювання (електролічильників) встановленим у нормативних документах значенням, що свідчіть про придатність цих засобів до застосування.

Повірка приладів обліку енергії здійснюється шляхом визначення метрологічних характеристик на підставі експериментальних випробувань та підтвердження їх відповідності встановленим обов'язковим вимогам.

Організація і порядок повірки засобів вимірювальної техніки в Україні регламентується нормативною документацією ДСТУ 8.259:2007 [1]. Повірка включає в себе низку процедур: демонтаж і доставку лічильника в спеціалізовану акредитовану службу, призначену для цих цілей лабораторію, сам процес випробувань на перевірному стенді і подальшого його повернення на місце використання. При перевірочних випробуваннях здійснюється ряд операцій, в результаті яких визначається стан приладу: справний, несправний - ремонтопридатний, несправний - ремонту не підлягає.

Перевірку здійснюють періодично через міжповірочний інтервал, який передбачає працездатність приладів протягом усього терміну між повірками [2]. Проте вплив як внутрішніх, так і зовнішніх факторів випадкового характеру може призвести до передчасної втрати функціональних властивостей приладів, що не гарантує належну працездатність приладів на протязі певного часу.

Бурхливий розвиток телекомунікаційних технологій, інтегрованих автоматизованих систем управління відкриває можливість здійснення дистанційного контролю функціональності сучасних електричних приладів, яка дозволяє віддалено діагностувати стан електронного обладнання практично в реальному часі (згідно графіку або за запитом оператора) на місці установки без демонтажу.

3. Постановка задачі

Метою досліджень є аналіз конструкцій сучасних приладів обліку електроенергії, причин виникнення випадків їхньої неналежної роботи та перспективи розробки і впровадження систем дистанційної діагностики для віддаленого функціонального контролю цифрового обладнання.

4. Задачі технічної діагностики

Засобами технічної діагностики вирішуються задачі двох типів:

1) діагностування з метою оперативного виявлення і класифікації помилок, що виникають в обладнанні;

2) діагностування (пошук) причини несправності з метою її усунення [3].

При діагностуванні в електронних приладах виділяють перевіряючі і перевіряємі підсистеми з обов'язковим забезпеченням їх зв'язності. Для спрощення процесу перевірки природним є використання для організації перевірок як існуючих в системі зв'язків, так і організації різних блокувань і перемикань, в результаті аналізу яких формуються ознаки дозволяють оцінити стан приладу [4].

До організації функціонального діагностування пред'являють такі вимоги як своєчасність виявлення помилок і повнота охоплення. Тому виникає ряд проблем, обумовлених складністю ідентифікації різнотипних дефектів та виявлення причин їх виникнення.

5. Особливості сучасних пристроїв обліку електричної енергії

Електричні лічильники є спеціальними приладами, призначеними для обліку споживаної електричної енергії. За принципом дії вони класифікуються на два типи: електронні, перетворення сигналів в яких здійснюється апаратно, та цифрові, в яких застосовуються вузькоспеціалізовані мікроконтролери [5].

Сучасні цифрові пристрої обліку електричної енергії включають в себе всі необхідні апаратні компоненти (мікропроцесори, блоки живлення та інші) і програмні модулі, що працюють в реальному часі. Вони можуть виконувати різні обчислення – цифрові, аналогові, а також змішані цифро-аналогові. Типову структурну схему сучасного однофазного лічильника електричної енергії наведено на рис. 1.

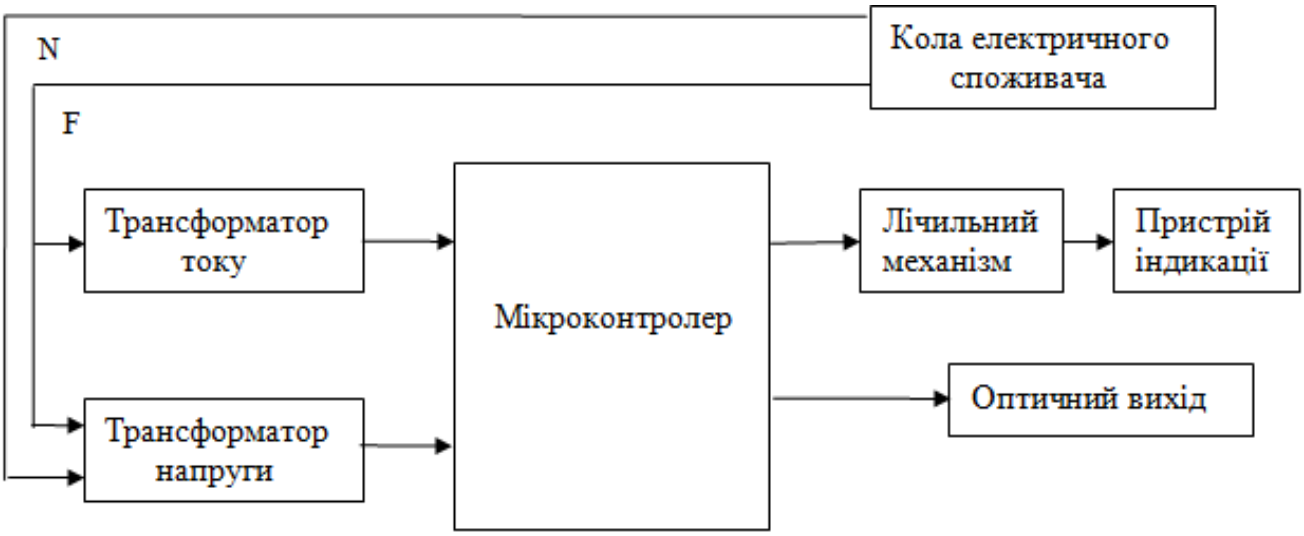


Рис. 1. Структурна схема лічильника електричної енергії

Обов'язковою вимогою при підключенні лічильника є дотримання полярності приєднання як по струму, так і по напрузі. Фіксація приладом обліку споживаної потужності здійснюється на основі одночасного вимірювання параметрів електричної мережі (струму і напруги) та подальшого перетворення її в частоту.

Найбільш часто в якості датчиків використовуються трансформатори струму і напруги. Сигнали пропорціональні напрузі і струму знімаються з датчиків і надходять на вхід мікроконтроле-

ра. Він у свою чергу перемножує вхідні сигнали, отримуючи миттєву споживану потужність, і перетворює їх відповідно рівню вимірюваної потужності у послідовність імпульсі з певною частотою надходження, яка підсумовується за допомогою лічильного механізму. Отримані показання лічильника у Вт·год відображаються на пристрої індикації.

Узгодження рівнів вхідних сигналів з датчиків та цифрових компонентів пристрою реалізується завдяки використанню додаткових мікросхем.

Найбільш часто в цифрових приладах обліку електричної енергії використовують відносно недорогі спеціалізовані мікроконтролери серії ADE7755. Вони містять два аналого-цифрових перетворювача (АЦП), джерело опорної напруги, і всі засоби обробки сигналів, необхідні для підрахунку активної потужності (рис. 2).

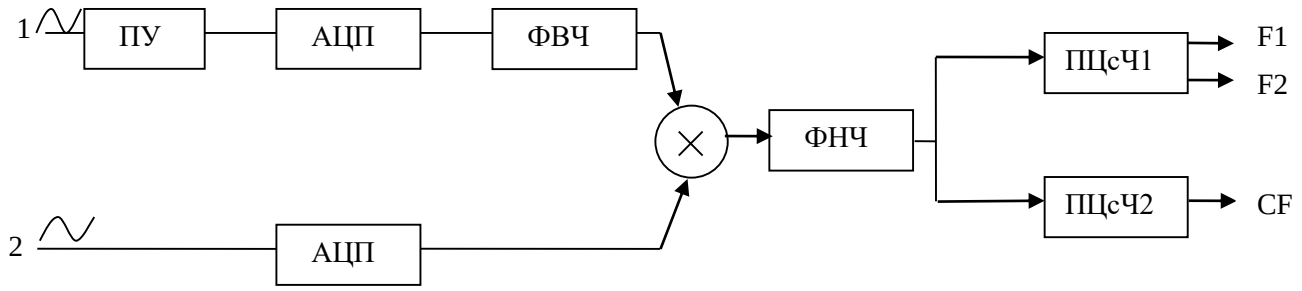


Рис. 2. Структурна схема мікроконтролера ADE7755

Складну функцію аналого-цифрового перетворення (АЦП) в цифровому лічильнику електричної енергії здійснюють типові мікросхеми, здатні перемножать сигнали і представляти отриману величину в зручній для мікроконтролера формі.

Перетворення активної потужності в частоту проходження імпульсів, здійснюється з точністю в межах декількох сотих часток відсотка. Використання аналогових входів 1 та 2 для сигналів напруги та струму спрощує взаємозв'язок перетворювачів, забезпечує широкий динамічний діапазон для прямого підключення до АЦП. Обидва входи мають внутрішню схему захисту від електростатичного розряду.

Програмований підсилювач (ПУ) у вхідному ланцюзі полегшує використання невеликих шунтів і резисторів. Фільтр високих частот (ФВЧ) видаляє будь-які компоненти постійного струму з сигналу. Усуває неточності в розрахунках активної потужності через зміщення в напрузі або в струмі сигналів. Фільтр низьких частот (ФНЧ) використається для згладжування сигналів з метою позбавлення від шумової компоненти.

Цифровий вихід, що формується блоком множення, містить сигнал активної потужності, який після фільтрації у ФНЧ передається на перетворювачі цифрового сигналу в частоту (ПЦсЧ1, ПЦсЧ2). Частота, яка генерується ADE7755 пропорційно середньої активної потужності, виводиться на виходах F1 і F2 у вигляді активних низьких імпульсів.

У своєму складі ADE7755 також має високочастотний імпульсний вихід CF з постійною величиною 3200 імпульсів/кВт, який підключається до світло діоду та й до віддалених мережевих комутаторів. Цей високочастотний вихід використовується для проведення прискореного калібрування і забезпечує можливість швидко перевірити працездатність і точність пристрою в реальному часі. ADE7755 також володіє можливістю безпосереднього управління електромеханічним лічильником.

Ці функціональні можливості спеціалізованого мікроконтролера свідчать про спроможність щодо віддаленого контролю та управління роботою сучасного приладу обліку електроенергії.

6. Контроль функціонування лічильників активної електричної енергії

Контроль та регулювання при перевірці однофазних лічильників активної електричної енергії в тому числі і лічильників з об'єднаними послідовними і паралельними ланцюгами, класу 1,0 і менш точних, що вимірюють енергію прямого і зворотного напрямків, відбувається за допомогою установки УАПС-2М.

При цьому можуть виконуватися такі операції як:

1) визначення статичних характеристик (похибки лічильника, порога чутливості, відсутності самоходу);

2) перевірка відповідних динамічних властивостей лічильників.

У результаті проведення цих операцій на основі аналізу показань, що реєструються при заздалегідь заданих умовах роботи, встановлюється стан тестового приладу обліку - справний або має неполадки.

Найтиповіші несправності лічильника, які виявляються в результаті повірки наведено в табл.1.

Табл. 1. Типові неполадки пристроїв обліку електричної енергії

№	Вид неполадки	Причина неполадки	Форма прояви
1	Вихід з ладу варистора призначеного для захисту від імпульсної напруги різного виду.	Перевищення допустимого рівня напруги на вході	Зупинка лічильника
2	Вихід з ладу вхідного металоплівкового резистору в трансформаторі напруги	Перевищення допустимого рівня напруги на вході	Зупинка роботи лічильника або коротке замикання
3	Некоректне функціонування мікроконтролера	Заводський брак Зовнішні наводки	Самохід або зупинка роботи лічильника
4	Номінал конденсатора в схемі фільтрації не відповідає паспортному значенню	Неякісний конденсатор Заводський брак	Сповільнюється робота лічильного механізму
5	Зміна товщини токового шунта	Зовнішнє втручання в прилад обліку	Лічильник рахує в зворотному порядку (зворотну сторону)

Особливістю сучасних цифрових вимірювальних систем є наявність всіляких сервісних функцій, таких як віддалений доступ до лічильника для дистанційного контролю показань з визначенням поточного значення потужності, що споживається, здійснення багато тарифного обліку споживання енергії, можливість віддаленого управління режимами роботи приладів обліку та інші. Наявність цифрового дисплея, керованого від мікроконтролера, дозволяє програмно встановлювати різні режими виведення інформації, наприклад, виводити інформацію про спожитої енергії за кожен місяць, за різними тарифами і т.д.

Включення до складу датчика ефективної системи автоматичного регулювання дозволяє надати йому задані властивості, тобто формувати необхідні амплітудну і амплітудно-частотну характеристики і адаптувати датчик до будь-яких вхідних впливів і дестабілізуючих чинників.

7. Перспективи створення систем дистанційного діагностування цифрових пристроїв

Об'єктом діагностування в цифрових приладах обліку електричної енергії є апаратура, лінії зв'язку, програмне та інформаційне забезпечення.

При організації систем діагностування можуть застосовуватися різні принципи, а при їх реалізації – використатися програмні і апаратні засоби та їх поєднання.

Процедура виявлення логічних або функціональних помилок у готових програмах та апаратному забезпеченні, які не відповідають своїм специфікаціям, в сучасній літературі називається функціональною валідацією [6]. Вона включає в себе генерацію тестових векторів, моделювання цифрового обладнання з використанням тестових векторів і порівняння вихідних сигналів, що генеруються, з очікуваними результатами.

Доцільно для зменшення загальної трудомісткості валідації використовувати підходи на основі моделювання та генерації спрямованих тестів. Скорочення загального часу генерації тестів

можна здійснити за допомогою методів кластеризації властивостей та процедури навчання підсистеми розпізнавання, які використовують так званні покриття моделей несправностей для формування належних властивостей моделі [6].

При створенні систем дистанційного діагностування для контролю стану сучасних пристроїв обліку електроенергії в реальному часі виникає необхідність проведення цілого комплексу досліджень:

1) Дослідження причин виникнення несправностей та невідповідного функціонування приладів.

2) Розробка та удосконалення математичних моделей, що повинні застосовуватися у системі контролю.

3) Розробка методів адаптивного керування процесом функціонального контролю електронного обладнання.

4) Створення та удосконалення заходів та засобів керування процесом контролю приладів обліку електричної енергії, які дозволяють реалізувати віддалене діагностування стану електронного обладнання в реальному часі.

5) Здійснення апробації моделей, методів та засобів, що розробляються, шляхом експериментальних досліджень на макетному встаткуванні.

Це потребує вирішення наступних першочергових задач:

- аналіз різноманітних неполадок, що виникають, статистичних характеристик їх появи;
- побудова математичної моделі процесу роботи цифрового устаткування (з використанням апарату теорії кінцевих автоматів, мереж Петрі) з можливістю урахування будь-яких умов функціонування (зовнішніх впливів або внутрішніх відмов);
- виявлення впливу чинників внутрішнього та зовнішнього характеру на роботу обладнання шляхом здійснення серії експериментів на комп'ютерній моделі.

8. Висновки

Вирішення проблем, що виникають при створенні системи дистанційної діагностики цифрових пристроїв обліку електроенергії, надає можливість своєчасно визначати їх стан, здійснювати усунення неполадок або заміну несправного обладнання. Що, в свою чергу, дозволяє підвищити надійність роботи автоматизованих систем комерційного обліку електроенергії.

Список литературы

1. Метрологія. Лічильники електричні індукційні активної та реактивної енергії. Методика повірки: ДСТУ ГОСТ 8.259:2007. – [Чинний від 2007-10-17]. – К.: Держспоживстандарт України, 2007. – 15 с. – (Національний стандарт України).
2. Ерёменко В.Т. Техническая диагностика электронных средств: учебник для высшего профессионального образования/ В.Т. Ерёменко [и др.]. – Орел: ФГБОУ ВПО «Госуниверситет - УНПК», 2012. – 157 с.
3. Гуляев В.А. Организация живучих вычислительных структур/ В.А. Гуляев, А.Г. Дронов, С.П. Пелехов. – Киев: Наук.думка, 1982 г. – 140 с.
4. Джейкокс Дж. Руководство по поиску неисправностей в электронной аппаратуре. Перевод с англ. - М.: Мир, 1989. - 176с.
5. Рощин В.А. Схемы включения счетчиков электрической энергии: производственно-практ. пособие/ В.А. Рощин. – М.: Изд-во НЦ ЭНАС, 2007. – 112 с.
6. Чэнь М. Валидация на системном уровне. Высокоуровневое моделирование и управление тестированием/ М. Чэнь, К. Цинь, Х.-М. Ку, П. Мишра. – Москва: Техносфера, 2014. – 296 с.

УДК 004. 056.5

МАРКОВСЬКИЙ О.П.,
СЕМЕНЮК Ю.В.,
ШЕВЧЕНКО О.М.,
РАДЧЕНКО К.О.

МЕТОД КОРЕКЦІЇ ПОМИЛОК ПЕРЕДАЧІ ДАНИХ В ОДНОЧІПОВИХ РАДІОМОДЕМАХ

Запропоновано ефективний метод корекції помилок передачі даних в одночіпових радіомодемах, особливістю якого є низька обчислювальна складність. Описано процедури кодування та корекції помилок, які також проілюстровано чисельним прикладом. Виконано порівняльний аналіз запропонованого методів корекції з позицій обчислювальної складності. Доведено, що запропонований метод забезпечує суттєве прискорення кодування та корекції, а також спрощення схеми апаратної реалізації. Проаналізовано результати практичного застосування методу для корекції помилок в мережі побудованій на одночіпових радіомодемах.

Proposed the effective method for data transmission errors correcting in single-chip radio modems, which feature is the low computational complexity. The procedure of coding and error correction are described in details and illustrated by numerical example. Performed comparative analysis of the proposed methods of correction from the standpoint of computational complexity. It is proved that the proposed method provides a significant acceleration of encoding and correction, as well as simplifying the hardware implementation scheme. The results of practical utilization of the method for correcting errors in a network built on single-chip radio modems has been analyzed.

Ключові слова: багатократні помилки передачі даних, канали передачі даних зі спектральною модуляцією, одночіпові радіомодеми корекція помилок, кодування помилок, зважені контрольні суми.

Вступ

Динамічний прогрес в області мікроефектроніки призвів до появи принципово нового класу пристроїв передачі даних — одночіпових радіо модемів. Низький рівень електроспоживання та дешевизна вказаних пристроїв зумовили їх широке застосування в різноманітних областях [1]. На сьогодні мережі побудовані на таких модемах(Low-Power Network та Low-Power Wide-Area Network) використовуються в найрізноманітніших сферах, наприклад:

- автоматизація виробництва;
- сільське господарство;
- концепція розумного будинку;
- концепція інтернету речей IoT;
- віддалений моніторинг та керування технологічними об’єктами.

Основними технологіями що використовуються для побудови LPN та LPWAN є ZigBee та 6LoWPAN [1]. В останні роки швидко набуває популярності протокол LoRaWAN.

На сьогодні більшість практичних систем використовують протоколи фізичного рівня, що базуються на використанні GFSK, O-QPSK або MPSK модуляції. Для ущільнення каналів та збільшення обсягу даних, які вони здатні передавати, використовують схеми модуляції з ефективним використанням наданого спектру. Прикладом таких схем модуляції може бути QAM (Quadrature Amplitude Modulation). Цим схемам притаманна висока швидкість передачі за умови високої якості каналу [2]. Данні передаються не окремими бітами як, під час використання MSK (Minimum Shift Keying) та BPSK (Binary phase-shift keying), а символами, які можуть складатись з багатьох біт, наприклад для QAM256 один символ складається з 8 біт. Схожим чином працює LoRa модуляція – кожному символу, що складається з багатьох біт співвідноситься один сигнал, що безпосередньо передається по каналу.

LoRa – це схема модуляції з розширенням спектра, в якій дані кодуються широкосмуговими ЛЧМ-імпульсами з частотою, що збільшується або зменшується на деякому часовому інтервалі. Цей вид модуляції має дві переваги: значне підвищення чутливості приймача за рахунок більш широкого спектру сиг-

налів та низька критичність до розладу по частоті між приймачем і передавачем.

Якщо в процесі передачі на сигнал діє зовнішня завада, то, відповідно, спотворюється весь символ, на який модулюється цим сигналом. В останні роки, в зв'язку зі багатократним зростанням швидкості передачі, за час дії зовнішніх завад по каналу передається декілька символів. Відповідно, типовою помилкою стає спотворення "пачки" символі – тобто послідовності суміжних символів на час передачі яких діє зовнішня завада [3].

Чим більше бітів в символі, тим більша довжина "пачки" помилок від однієї перешкоди. Для існуючих методів корекції символьних помилок збільшення їх кількості істотно впливає на зростання обчислювальної складності процесів виправлення. Вважаючи на те, що для переважної більшості систем корекція помилок передачі має виконуватися в реальному часі, об'єктивна тенденція збільшення довжини "пачки" спотворених символів потребує експоненційного зростання обчислювальних ресурсів потрібних для корекції в реальному часі.

Радіомодемі пристрої використовують батарейкове живлення, тому надзвичайно обмежені в обчислювальних ресурсах. З іншого боку, батарейкове живлення термінальних пристроїв накладає суттєві обмеження на потужність канальних сигналів, що негативно впливає на достовірність передачі даних і, відповідно, потребує використання більш ефективних корегуючих кодів.

Таким чином, проблема створення ефективних, в плані обчислювальної складності, методів для корекції "пачки" помилок, є актуальною для сучасного етапу розвитку мережевих технологій, що базуються на використанні одночіпових радіомодемів.

Аналіз методів корекції "пачки" помилок в каналах зі спектральним ущільненням

В сучасних комп'ютерних та телекомунікаційних системах використовують спеціальні види модуляції, які дозволяють ефективніше використовувати надані частоти.

Особливістю таких видів модуляції є те, що символ, який передається належить алфавіту з M символів, тому одночасно передається $k=\log_2 M$ бітів за один символьний інтервал. Це дозволяє в k раз збільшити швидкість передачі інформації.

На сьогодні, розроблено і активно використовується декілька видів такої модуляції, які орієнтовані на різні типи каналів. Параметри більшості з них фіксовані відповідними протоколами передачі цифрової інформації [1].

Під впливом зовнішньої завади може бути пошкоджено один або "пачка" символів, що залежить від часу дії завади та частоти слідування канальних сигналів. Виходячи з існуючої тенденції збільшення частоти слідування канальних сигналів має місце зростання довжини "пачки" пошкоджених символів.

В роботі розглядається передача блока m-розрядних символів, кожен з яких модулюється канальним сигналом. Вважається що за час передачі блока, можлива дія лише однієї зовнішньої завади, причому кількість спотворених нею символів лежить в інтервалі від одного до $2 \cdot q$-1 (параметр q залежить від типу каналу, умов та швидкості передачі).

Для виправлення помилок в каналах одночіпових радіомодемів використовується загорткове кодування. Для підвищення надійності передачі даних бажаним є використання блокових кодів разом із згортковими, як наприклад зроблено в GSM системі. Основною проблемою використання таких блокових кодів, як коди Файра, коди Міласа, або коди Ріда-Соломона [3] є їх висока обчислювальна складність, що обмежує можливості корекції помилок в темпі передачі даних. Тому більшість систем не використовує ці коди, так як для знаходження помилок необхідно розв'язувати систему нелінійних рівнянь. Аналітичного способу її розв'язання не існує, тому використовуються технології перебору. Технологія перебору відрізняється в залежності від коду. Саме необхідність виконувати перебір зумовлює високу обчислювальну складність.

Для систем, що використовують схожі типи модуляції, але не мають таких обмежень найбі-

льшого поширення для корекції пачок помилок набули коди Ріда-Соломона. Ці коди дозволяють виправляти h спотворених символів з використанням $2{\cdot}h$ контрольних символів. При виникненні пачки помилок коди Ріда-Соломона не враховують те, що спотворені символи є суміжними, тобто при корекції пачки значна частина процедури виправлення є надлишковою.

Інший підхід до корекції "пачки" помилок базується на використанні зважених контрольних сум. Добре відомо, що контрольні суми є найшвидшим методом контролю та виправлення помилок, оскільки вони базуються на використанні лінійних операцій [4]. На сьогодні запропонована спеціальна модифікація зважених контрольних сум для швидкого знаходження помилок у каналах з імпульсно кодовою модуляцією [4].

Особливістю використання зважених контрольних сум є те, що вони, на відміну від більшості інших корегуючи кодів, надають широкі можливості адаптації як до особливостей виникаючих помилок так до особливостей обчислювальних платформ, на яких виконуються процедури корекції.

Методу работи є модифікація зважених контрольних сум, яка забезпечує ефективне виправлення "пачки" помилок обмеженої кратності з урахуванням особливостей одночіпових радіомодемів.

Метод швидкої корекції пачки помилок

Запропонований метод дозволяє виконувати корекцію пачки помилок з максимальною довжиною $2{\cdot}q$-1 пошкоджених символів. В методі використовується операції множення без переносів та ділення чисел для пошуку частки і остачі. Вказані операції визначаються наступним чином.

Множення без міжрозрядних переносів, позначається символом '⊗' передбачає використання в якості операндів двох m-розрядних чисел X та Y, що визначаються як:

$$X = \{x_0, x_1, ..., x_{m-1}\} = x_{m-1} + 2 \cdot x_{m-2} + ... + 2^{m-1} \cdot x_0,$$
$$x_i \in \{0,1\} \forall i \in \{0,1,..., m-1\},$$
$$Y = \{y_0, y_1, ..., y_{m-1}\} = y_{m-1} + 2 \cdot y_{m-2} + ... + 2^{m-1} \cdot y_0,$$
$$y_i \in \{0,1\} \forall i \in \{0,1,..., m-1\},$$

їх добуток без міжрозрядних переносів $P = X \otimes Y$ являє собою $2{\cdot}m$-розрядне двійкове число, що визначається як:

$$P = \{p_0, p_1, ..., p_{m-1}\} = p_{2\cdot m-1} + 2 \cdot p_{2\cdot m-2} + ... +$$
$$+ 2^{2\cdot m-1} \cdot p_0 = X \cdot y_{m-1} \oplus 2 \cdot X \cdot y_{m-2} \oplus ... \oplus$$
$$\oplus 2^{m-1} \cdot X \cdot y_0, p_i \in \{0,1\} \forall i \in \{0,1,..., m-1\}.$$

Наприклад, якщо m = 4: $X = \{1,0,1,0\} = 10_{10}$ $Y = \{1,0,0,1\} = 9_{10}$, тоді $P = X \otimes Y = 1010 \oplus$ $\oplus 1010000 = 01011010 = 90_{10}$.

Операція ділення без міжрозрядних переносів визначається як ділення $2 \cdot m$-розрядного поліному на m-розрядний. При виконанні операції отримуються частка та остача, які позначаються як $Q(P/Y)$ та $R(P/Y)$ відповідно, де:

$$P = \{p_0, p_1, ..., p_{m-1}\} = p_{2\cdot m-1} + 2 \cdot p_{2\cdot m-2} + ... +$$
$$+ 2^{2\cdot m-1} \cdot p_0, p_i \in \{0,1\} \forall i \in \{0,1,..., m-1\}.$$
$$Y = \{y_0, y_1, ..., y_{m-1}\} = y_{m-1} + 2 \cdot y_{m-2} + ... + 2^{m-1} \cdot y_0,$$
$$y_i \in \{0,1\} \forall i \in \{0,1,..., m-1\}.$$

Наприклад, якщо m = 4: $P = \{0,1,0,1,1,1,1,0\} =$ $= 94_{10}, Y = \{1,0,0,1\} = 9_{10}$, тоді $Q(P/Y) = 1010 =$ $= 10_{10}, R(P/Y) = 0100 = 4_{10}$.

Запропонований метод передбачає ділення інформаційного блоку B на n кадрів F, кожен довжиною q m-розрядних символів, де q — обраний параметр коду, отже:

$$B = \{F_0, F_1, ..., F_{n-1}\}, F_i = \{X_{i\cdot q}, X_{i\cdot q+1}, ..., X_{i\cdot q+(n-1)}\},$$
$$X_j = \{x_{j,0}, x_{j,1}, ..., x_{j,m-1}\}.$$

Для кожного кадру F, передавач обчислює контрольний код, який складається з $2 \cdot q$ компонент, перші q компонент контрольної суми обчислюються за формулою (1), наступні q компонент є складовими зваженої контрольної суми і обчислюються згідно наведеної нижче формули (2).

$$C_k = F_{0,k} \oplus F_{1,k} \oplus ... \oplus F_{n-1,k}, \quad (1)$$
$$k \in \{0,1,..., q-1\}$$

$$S_k = F_{0,k} \otimes 1 \oplus F_{1,k} \otimes 2 \oplus ... \oplus F_{n-1,k} \otimes n, \quad (2)$$
$$k \in \{0,1,..., q-1\}$$

Компоненти контрольного коду обчисленого передавачем позначаються $C_{Tk}, S_{Tk}, k \in \{0,1,..., q-1\}$. Після обчислення контрольного коду передавачем, інформація передається на приймач в наступній послідовності: $B, C_{T0}, S_{T0}, C_{T1}, S_{T1},...., C_{Tq-1}, S_{Tq-1}$.

В запропонованому методі корекції помилок, приймач за прийнятим блоком B обчислює контрольні коди кожного з кадрів $C_{Rk}, S_{Rk}, k \in \{0,1,..., q-1\}$ за формулами (1) та (2) відповідно. З обчислених та прийнятих контрольних кодів обчислюються різниці між кодами приймача і передавача за формулою (3).

$$\Delta_k = C_{Rk} \oplus C_{Tk}, \delta_k = S_{Rk} \oplus S_{Tk}, \quad (3)$$
$$k \in \{0,1,..., q-1\}$$

Зі значень різниць визначається наявність помилок та їх тип. Можливі наступні часткові випадки:

1) $\Delta_k = 0, \delta_k = 0, k \in \{0,1,..., q-1\}$, в цьому випадку блок передано правильно, корекція не потрібна.

2) $\exists k \in \{0,1,..., q-1\} : (\Delta_k = 0, \delta_k \neq 0) \vee (\Delta_k \neq 0 \ , \delta_k = 0)$ - означає, що помилка відбулась у контрольних кодах, блок даних передано правильно і корекція не потрібна.

Якщо попередні умови не виконались, то в даних є помилки, які необхідно виправити. Доцільно ввести означення помилок 1, 2 та 3 типів.

Помилка з позицією l вважається помилкою 1 типу якщо в блоці відсутні помилки з позиціями $l'=l \pm q \cdot n$.

Якщо існує лише одна помилка з позицією $l' = l \pm q \cdot n, n \in N$, то помилка вважається помилкою 2 типу.

Якщо таких помилок більше – то це помилка 3 типу.

В межах однієї пачки помилок довжиною $2 \cdot q - 1$ помилки 3 типу неможливі. Тому вони в запропонованому методі не розглядаються.

Для помилок 1 типу виконується умова $R(\delta_k / \Delta_k) = 0$, де k позиція помилки у кадрі, якщо $R(\delta_k / \Delta_k) \neq 0$ – вважається, що помилка на позиції k - помилка 2 типу.

Наступний крок – знаходження множин індексів помилок обох типів. Для цього, використовуючи обчислені значення Δ_k та δ_k, для помилок 1 типу будується множина A за формулою (4), для побудови множини B помилок 2 типу використовується формула (5).

$$A = \{k \mid \forall k \in \{0,1,..., q-1\} : \Delta_k \neq 0, \delta_k \neq 0, \quad (4)$$
$$R(\delta_k / \Delta_k) = 0\},$$

$$B = \{k \mid \forall k \in \{0,1,..., q-1\} : \Delta_k \neq 0, \delta_k \neq 0, \quad (5)$$
$$R(\delta_k / \Delta_k) \neq 0\}$$

З побудованих множин визначаються помилки. Розглядаються 2 випадки:

1) Множина $B = \varnothing$ – робиться висновок, що відбулись лише помилки 1 типу, які виправляються за формулою (6).

$$\forall k \in A : l(k) = (Q(\delta_k / \Delta_k) - 1) \cdot q + k, \quad (6)$$
$$X_{l(k)} = X_{Rl(k)} \oplus \Delta_k$$

Після корекції за формулою (6) блок даних вважається виправленим.

2) Якщо множина $B \neq \varnothing$ – робиться висновок, що присутні помилки обох типів. Важливо перевірити умову $|A| + |B| \neq q$, що означає не виконання умови належності всіх помилок до однієї пачки або умови максимальної довжини пачки і виправлення помилок запропонованим методом неможливе.

З умови $|A| + |B| = q$, слідує що в даних, які отримав приймач відбулась пачка помилок довжиною $|A| \cdot 2 + |B|$. Знаходження позиції p першої помилки в пачці відбувається за формулою (7).

$$p = \min((Q(\delta_k / \Delta_k) - 1) \cdot q + k) - |B|, \forall k \in A \quad (7)$$

Оскільки знайдено початок пачки, можлива корекція. Для виправлення помилок 1 типу ви-

користовується формула (6), для помилок 2 типу – формула (8).

$$\forall t \in N, t \geq p, t < p + |B|:$$
$$k(t) = t \bmod q,\ f(t) = \left\lfloor \frac{t}{q} \right\rfloor + 1,$$
$$y(t) = Q\left(\frac{\delta_{k(t)} \oplus (\Delta_{k(t)} \otimes (f(t)+1))}{f(t) \otimes (f(t)+1)}\right), \qquad (8)$$
$$X_t = X_{Rt} \oplus y(t),$$
$$X_{t+q} = X_{Rt+q} \oplus y(t) \oplus \Delta_{k(t)}$$

Після виконання всіх необхідних операцій блок даних вважаться виправленим або робиться висновок що пошкодження надто сильні і корекція неможлива.

Описана процедура корекції ілюструється наведеним нижче прикладом в якому параметр коду q = 4. Нехай початковий інформаційний блок складається з 12 символів по 4 біти кожен:

$$X_0 = \{1,0,0,1\}, X_1 = \{0,1,1,1\}, X_2 = \{0,1,1,0\},$$
$$X_3 = \{1,0,0,1\}, X_4 = \{0,1,0,0\}, X_5 = \{1,1,0,0\},$$
$$X_6 = \{0,1,1,1\}, X_7 = \{1,0,1,1\}, X_8 = \{0,1,0,1\},$$
$$X_9 = \{1,0,0,0\}, X_{10} = \{1,0,1,1\}, X_{11} = \{0,0,1,1\}$$

Контрольні коди, обчислені на стороні передавача згідно з наведеними вище формулами (1) та (2) дорівнюють:

$$C_{T0} = 1001 \oplus 0100 \oplus 0101 = 1000,$$
$$C_{T1} = 0111 \oplus 1100 \oplus 1000 = 0011,$$
$$C_{T2} = 0110 \oplus 0111 \oplus 1011 = 1010,$$
$$C_{T3} = 1001 \oplus 1011 \oplus 0011 = 0001,$$
$$S_{T0} = 1001 \otimes 1 \oplus 0100 \otimes 2 \oplus 0101 \otimes 3 = 1110,$$
$$S_{T1} = 0111 \otimes 1 \oplus 1100 \otimes 2 \oplus 1000 \otimes 3 = 0111,$$
$$S_{T2} = 0110 \otimes 1 \oplus 0111 \otimes 2 \oplus 1011 \otimes 3 = 10101,$$
$$S_{T3} = 1001 \otimes 1 \oplus 1011 \otimes 2 \oplus 0011 \otimes 3 = 11010,$$

В рамках прикладу вважається, під час передачі блоку під дією завади було спотворено 7 суміжних символів:

$$X_2 = \{0,1,0,1\}, X_3 = \{1,1,0,0\}, X_4 = \{0,0,0,1\},$$
$$X_5 = \{0,1,0,1\}, X_6 = \{0,0,0,0\}, X_7 = \{0,1,0,0\},$$
$$X_8 = \{1,1,1,1\}.$$

Контрольні коди обчислені на стороні приймача за формулами (1) та (2) мають вигляд:

$$C_{R0} = 1001 \oplus 0001 \oplus 1111 = 0111,$$
$$C_{R1} = 0111 \oplus 0101 \oplus 1000 = 1010,$$
$$C_{R2} = 0101 \oplus 0000 \oplus 1011 = 1110,$$
$$C_{R3} = 1100 \oplus 0100 \oplus 0011 = 1011,$$
$$S_{R0} = 1001 \otimes 1 \oplus 0001 \otimes 2 \oplus 1111 \otimes 3 = 11010,$$
$$S_{R1} = 0111 \otimes 1 \oplus 0101 \otimes 2 \oplus 1000 \otimes 3 = 10101,$$
$$S_{R2} = 0101 \otimes 1 \oplus 0000 \otimes 2 \oplus 1011 \otimes 3 = 11000,$$
$$S_{R3} = 1100 \otimes 1 \oplus 0100 \otimes 2 \oplus 0011 \otimes 3 = 00001,$$

За формулою (3) обчислюються різниці:

$$\Delta_0 = C_{T0} \oplus C_{R0} = 1000 \oplus 0111 = 1111,$$
$$\Delta_1 = C_{T1} \oplus C_{R1} = 0011 \oplus 1010 = 1001,$$
$$\Delta_2 = C_{T2} \oplus C_{R2} = 1010 \oplus 1110 = 0100,$$
$$\Delta_3 = C_{T3} \oplus C_{R3} = 0001 \oplus 1011 = 1010,$$
$$\delta_0 = S_{T0} \oplus S_{R0} = 01110 \oplus 11010 = 10100,$$
$$\delta_1 = S_{T1} \oplus S_{R1} = 00111 \oplus 10101 = 10010,$$
$$\delta_2 = S_{T2} \oplus S_{R2} = 10101 \oplus 11000 = 01101,$$
$$\delta_3 = S_{T3} \oplus S_{R3} = 11010 \oplus 00001 = 11011.$$

Аналіз отриманих результатів показує, що всі $\Delta_k \neq 0, \delta_k \neq 0, k \in \{0,1,2,3\}$ отже необхідно знайти остачі від ділення:

$$R(\delta_0 / \Delta_0) = R(10100/1111) = 0101,$$
$$R(\delta_1 / \Delta_1) = R(10010/1001) = 0000,$$
$$R(\delta_2 / \Delta_2) = R(1101/0100) = 0001,$$
$$R(\delta_3 / \Delta_3) = R(11011/1010) = 0101,$$

Зі знайдених остач, за формулами (4) і (5) на стороні приймача формуються множини $A = \{1\}$ і $B = \{0,2,3\}$. Оскільки $B \neq \varnothing$, робиться висновок, що у пачці є помилки обох типів. Наступним кроком є перевірка умови $|A| + |B| = q$. Оскільки $1 + 3 = 4$, то робиться висновок що помилки можна виправити; для цього за формулою (7) знаходиться позиція початку пачки помилок $p = (Q(10010/1001) - 1) \cdot 4 + 1 - 3 = 2$. Виконується корекція для помилки 1 типу, використовуючи формулу (6): $l(1) = (Q(10010/1001) - 1) \cdot 4 + 1 = 5$

$X_5 = X_{R5} \oplus \Delta_1 = 0101 \oplus 1001 = 1100$. Корекція для помилок 2 типу виконується за формулою (8):

$$
\begin{aligned}
&t \in \{2,3,4\}: \\
&t = 2, k(2) = 2, f(2) = 1, \\
&y(2) = 0011, \\
&X_2 = 0101 \oplus 0011 = 0110, \\
&X_6 = 0000 \oplus 0011 \oplus 0100 = 0111, \\
&t = 3, k(3) = 3, f(3) = 1, \\
&y(3) = 0101, \\
&X_3 = 1100 \oplus 0101 = 1001, \\
&X_7 = 0100 \oplus 0101 \oplus 1010 = 1011, \\
&t = 4, k(4) = 0, f(4) = 2, \\
&y(4) = 0101, \\
&X_4 = 0001 \oplus 0101 = 0100, \\
&X_8 = 1111 \oplus 0101 \oplus 1111 = 0101.
\end{aligned}
$$

Отже приймач повністю відновлює символи пошкоджені завадою.

Оцінка ефективності

Запропонований метод дозволяє виправляти від 1 до $2 \cdot q - 1$ суміжних символів, спотворених зовнішньою завадою, використовуючи $b = q\left(2 + \frac{\log_2 n}{m}\right)$ контрольних символів.

Для експериментальної перевірки ефективності запропонованого методу було використано радіомережу, що побудована на базі одночіпових радіомодемів SX1276 виробництва Semtech. Модеми SX1276 використовують модуляцію LoRa з програмованою швидкістю передачі від 24 до 37520 біт на секунду і працюють в діапазоні від 137 до 1020 МГц.

Мережа побудована за архітектурою точка-точка. Для експерименту радіомодеми було налаштовано на частоту 915 МГц що входить до ISM(Industrial, Scientific and Medical) діапазону, який є частиною радіочастотного спектра загального призначення, і може бути використаний без ліцензування.

Для перевірки якості роботи методу передавач було запрограмовано на трансляцію пакетів з порядковим номером та даними за якими можна визначити правильність прийому та коригування. Пакети формувались з використанням запропонованого методу та кодів Ріда-Солмона один через один. Для проведення експерименту передавач було зафіксовано на мачті висотою 3м.

Якість прийому перевірялась в 9 точках на різних відстанях від передавача на висоті 1м від поверхні землі. Для прийнятих пакетів визначався тип кодування після чого виконувалось виправлення помилок визначеним методом. Для кожного прийнятого пакету визналась категорія:

- пакет прийнятий без пошкоджень;
- пакет має пошкодження, але виконана корекція за допомогою запропонованого методу;
- пакет має пошкодження, але виконана корекція за допомогою кодів Ріда-Соломона.

Для кожної точки вимірювання приймач працював 250 секунд, що необхідно для передачі 500 пакетів. Кількість пакетів у кожній категорії приведена до відсотків від загальної кількості пакетів для даного вимірювання, результати наведені в таблиці 1.

Таблиця 1. Відсоток правильних пакетів, виправлених різними методами в різних точках

Запропонований метод	Коди Ріда-Соломона	Без корекції
100%	100%	98%
97%	100%	73%
95%	90%	67%
91%	78%	59%
81%	70%	51%
72%	68%	40%
65%	65%	31%
31%	28%	18%
5%	14%	0%

Дані наведені в таблиці 1. демонструють високу ефективність запропонованого методу в порівнянні з кодами Ріда-Соломона. Перевага запропонованого методу пояснюється тим, що в радіомодемі використовується згорткове коду-

вання 7/8, що проводиться за допомогою регістра зсуву, відводи від якого підсумовуються по модулю два. При декодуванні згорткового коду в радіомодемі використовується алгоритм Вітербі. Метод являє собою декодування по максимуму правдоподібності. Ідея алгоритму полягає в тому, що в декодері відтворюють всі можливі шляхи послідовних змін станів сигналу, зіставляючи отримані при цьому кодові символи з прийнятими аналогами по каналу зв'язку і на основі аналізу помилок між прийнятими і необхідними символами визначають оптимальний шлях. Якщо надлишковості коду недостатньо для виправлення помилки певної кратності, то в результаті декодування отримується "пачка" помилок.

Основною перевагою запропонованого методу є значне, в порівнянні з кодами Ріда-Соломона, зменшення обчислювальної складності при збереження ефективності декодування.

Також суттєвою перевагою є простота алгоритму, що дозволяє ефективну апаратну реалізацію, та мінімізує розмір пам'яті програм та оперативної пам'яті при програмній реалізації.

Зважаючи на низькі вимоги до потужності процесора та об'єму пам'яті програм і оперативної пам'яті, для не швидкісних систем(до 50 кілобіт за секунду), наприклад, одночіпових радіомодемів можливе використання 8 бітних мікроконтролерів PIC або AVR. Якщо в мережі необхідні швидкості до 2 мегабіт на секунду, для виконання алгоритму достатньо мікроконтролеру з ядром Cortex M0, з частотою від 32 МГц. Для швидкісних систем оптимальним є використання процесорів загального призначення або спеціалізованих обчислювачів створених на основі FPGA або ASIC.

З результатів експерименту можна зробити висновок про ефективність запропонованого методу, головною перевагою якого є низька обчислювальна складність та незначні вимоги до пам'яті, що є надзвичайно важливим для одночіпових систем або систем побудованих на базі мікроконтролерів. Запропонований метод є більш ефективним в порівнянні з кодами Ріда-Соломона в модемах де використовується загорткове кодування, так як в результаті роботи декодера виправляються помилки з низкою кратністю але залишаються "пачки" помилок.

Висновки

В результаті проведених досліджень, направлених на підвищення ефективності корекції помилок передачі даних в радіо модемах – термінальних пристроях передачі даних мереж отримані наступні результати.

Показано, що висока ефективність виправлення помилок може бути досягнута в рамках урахування особливостей їх виникнення та специфіки радіо модему, як обчислювальної платформи.

Встановлено, що основною причиною виникнення помилок при передачі даних з використанням радіомодемів є зовнішні завади, а домінуючим типом помилок є "пачки" – тобто спотворення послідовності суміжних символів.

Розроблено метод корекції однієї "пачки" помилок в блоці, в основу якого покладено використання зважених контрольних сум, що що забезпечує простоту реалізації та високу швидкодію виконання операцій корекції помилок.

Експериментально досліджено запропонований метод, в результаті чого практично доведено, що рахунок використання лінійної операційної бази, урахування особливостей виникнення помилок та особливостей обробки даних в радіо модемах, досягнуто підвищення ефективності виправлення помилок в порівнянні з традиційними кодами Ріда-Соломона. Зокрема, значно спростилася обчислювальна процедура локалізації помилок та відновлення спотворених символів. На 1-2 порядки (в залежності від довжини блоку) прискорено процес корекції помилок, що має велике значення для систем реального часу; підвищена корегувальна спроможність при фіксованій кількості контрольних символів.

Основним джерелом досягнутого підвищення ефективності корекції помилок стала спеціалізація розробленого методу на конкретному домінуючому типі помилок та урахування особливостей кодування даних в радіо модемах.

Запропонований метод може бути ефективно застосований для підвищення надійності віддаленого моніторингу та керування процесами реального часу з використанням мережевих технологій.

Список посилань

1. Ирвин Дж. Передача данных в сетях: инженерный подход. / Д ж. Ирвин., Д. Харль. – СПб.: БХВ-Петербург, 2002. – 448 с.
2. Скляр Б. Цифровая связь. Теоретические основы и практическое применение. Изд. 2-е, испр.: Пер. с англ. – М.: Издательский дом ”Вильямс”, 2003.– 1104 с.
3. Морелос-Сагароса Р. Искусство помехоустойчивого кодирования. Методы, алгоритмы, применение. Москва: Техносфера, 2005. – 320с.
4. Марковський О.П. Метод корекції подвійних помилок в каналах передачі цифрових даних зі спектральною модуляцією / Марковський О.П., Терещенко С.Ю. // Вісник Національного технічного університету України ”КПІ” Інформатика, управління та обчислювальна техніка, – Київ: ВЕК+ – 2015. – № 63. – С. 19-24.

УДК 004.7

РЫБАК В. А.,
ШОКР АХМАД

МЕТОДЫ И СРЕДСТВА ДЛЯ АНАЛИЗА И ПРОГНОЗА ЭКОЛОГО-ЭКОНОМИЧЕСКИХ ПОКАЗАТЕЛЕЙ РЕСУРСОЁМКИХ ИННОВАЦИОННЫХ ПРОЕКТОВ

В настоящей статье описаны компоненты системы поддержки принятия решений для анализа и прогнозирования эколого-экономических показателей ресурсоёмких инновационных проектов. В исследование также включены методы и средства автоматизации управления природоохранной деятельностью, информационный портал для сбора и обработки параметров, методика прогнозирования. Новым представляется как комплексность при подходе к решению поставленной задачи, так и применение теории нечётких множеств и модифицированного метода анализа иерархий.

The article describes the components of the decision support system for analysis and forecasting of ecological and economic parameters of industrial innovation projects including methods and automation of environmental management, information portal for the collection and processing parameters, forecasting methodology. It is presented as a new complexity in the approach to the task, and the application of the theory of fuzzy sets and modification of the method of analysis of hierarchies.

Ключевые слова: природопользование, искусственный интеллект, принятие решений, экология.

1. Введение

В настоящее время в соответствии с Постановлением Совета Министров Республики Беларусь от 26 мая 2011 г. № 669 «О Государственной программе инновационного развития Республики Беларусь (ГПИР РБ) на 2011-2015 годы» реализуется Государственная программа инновационного развития экономики страны.

Государственная программа направлена на достижение главного приоритета страны – перевода национальной экономики в режим интенсивного развития в рамках белорусской экономической модели и определяет цели и задачи инновационного развития экономики, направления, механизмы и средства их реализации. В основу государственной программы заложено поэтапное построение национальной инновационной системы (НИС) – современной институционной модели генерации, распространения и использования знаний, их воплощения в новых продуктах, технологиях, услугах во всех сферах жизни белорусского общества [1].

Вместе с тем, среди контролируемых показателей реализуемых инновационных проектов отсутствуют данные об экологичности работ, то есть эффективность потребления природных ресурсов, безотходность и энергоёмкость напрямую не учитываются. Хотя в современных условиях проект не может быть назван инновационным, если он не в полной мере соответствует требованиям охраны окружающей среды (ОС). Таким образом, обозначенное направление является достаточно актуальным.

2. Анализ существующих методов и средств автоматизации управления природоохранной деятельностью

Современный процесс управления природопользованием и охраной окружающей среды не относится к классу стандартных, теоретически обоснованных и практически отработанных автоматизированных процессов с использованием известных теоретических основ, принципов построения и технических средств, практикуемых в управлении технологическими процессами в любых областях промышленности, сельского хозяйства. Его можно характеризовать как организационно-административный метод управления, основанный на сложном комплексе мероприятий, методов и средств, реализуемых специалистами управленческих структур соответствующих министерств, ведомств и организаций. В качестве механизмов управления здесь применяются законы, нормативы, постановления, указы, инструкции, предписания, различные мероприятия. Тем не менее, использование информационных технологий (ИТ) на базе современной компьютерной техники является основной и наиболее перспективной методологией управления природопользованием и охраной окружающей среды.

Проблема разработки информационной технологии формирования и управления информацией о состоянии и использовании природных ресурсов в народном хозяйстве, анализа, оценки и прогнозирования экологического состояния основных природных сред, разработка и реализация на этой основе эффективных природоохранных мероприятий и методов их социально-экономической оценки является важной проблемой мирового уровня. Большое внимание ей уделяется в странах СНГ, в частности, в Российской Федерации и, в первую очередь, разработке комплексных территориальных кадастров природных ресурсов [2]. КТКПР можно рассматривать как аналог создаваемой в Беларуси системы кадастровых данных, отражающих природно-ресурсный потенциал (ПРП) административных территорий. Он является предметной частью территориальной автоматизированной системы и представляет собой государственный свод системно-организованных данных о природных ресурсах и природных объектах в границах административных территорий (субъект РФ, округ, район). АИС КТКПР предназначена и для обеспечения процесса принятия управленческих решений по вопросам охраны окружающей среды, рационального использования природных ресурсов и обеспечения экологической безопасности. Она создаётся на основе информационных и телекоммуникационных технологий, что означает включение в неё систем создания цифровых кадастровых карт и планов, баз данных по учёту природных ресурсов и загрязнения природных сред, а также современных технических и программных средств для работы в различных режимах удаленного доступа [3].

В Украине создаётся свой вариант комплексного территориального кадастра. В нём есть отличия. Для эффективного обеспечения стратегии устойчивого развития регионов информационный свод сведений о природных ресурсах будет не только (и не столько) комплексным территориальным кадастром, сколько многоцелевым комплексным кадастром-инвентаризатором всех наличествующих и используемых в регионе природных ресурсов (КТКПРом). Он будет включать все сведения относительно ценности, практической значимости, перспектив, целесообразности промышленного освоения и использования о каждом из кадастров, проявленных в регионе природных ресурсов, как в текущий момент, так и в обозримом будущем [2].

Среди западных стран обращает на себя внимание методический подход к экологическим исследованиям окружающей среды в Бельгии – одной из наиболее развитых стран ЕС. Здесь, наряду с комплексной оценкой экологического состояния ОС, обращено большое внимание на системную идентификацию практически всех источников загрязнения окружающей среды, включая основные ингредиенты загрязнения. При этом комплекс исследований проводился в течение 10 лет, что позволило выявить ряд важных показателей динамики загрязнения ОС и определить вклад (долю) каждого из основных источников загрязнения в общее экологическое состояние ОС. Если принять общее состояние ОС за 100%, то усреднённое за 10 лет антропогенное воздействие каждого из них следующее: население – 15%, промышленность – 25%, энергетика – 9%, сельское хозяйство – 12%, транспорт – 22%, торговля и бытовые услуги - 17%. На долю остальных факторов (туризм и отдых) приходится 5%. Доминирующим воздействием являются промышленность и транспорт (25% и 22%) [4].

В Беларуси в последнее время также уделяется большое внимание разработке кадастров природных систем, баз и банков данных, АИС и ГИС в области формирования и управления информацией о состоянии природных ресурсов и сред, состоянию ПРП и окружающей среды в городах и административных регионах. Так, эта проблема достаточно полно раскрыта в концепции Государственного кадастра территорий, разработанной РУП «Республиканский центр кадастра территорий» [5]. Здесь административные территории рассматриваются как комплексные территориальные природные ресурсы, ресурсы среды жизнедеятельности населения.

На данный момент в Беларуси лучшей интегрированной системой в рассматриваемой области является АИС «Кадастры», разработанная в учреждении «БЕЛНИЦ ЭКОЛОГИЯ». Хотя и она не лишена ряда недостатков: АИС «Кадастры» является клиент-серверным приложением и не может быть тиражирована; сложность обновления данной системы; доступ к базе данных не является мгновенным и постоянным для всех пользователей; система может применяться только на Windows-платформах.

Таким образом, приняв за основу принцип комплексности информации для целей управления инновационными проектами была создана Интернет-ориентированная версия АИС «Кадастры», которая позволяет получать дан-

ные об имеющихся природных ресурсах и их использовании. Однако для анализа и управления эколого-экономическими показателями было предложено отдельное решение.

3. Программная реализация средств обработки и отображения параметров экологичности инновационных проектов

Схема информационной структуры Портала изображена на рис. 1.

Страница «Вход в систему» предназначена для защиты Портала от несанкционированного доступа. Идентификация абонентов системы происходит с помощью ввода имени и пароля абонента, выданного ему администратором Портала. Если имя и пароль абонента будут введены правильно, открывается главная страница Портала. Без аутентификации гостю доступные только ограниченные ресурсы Портала.

Рисунок 1 – Схема информационной структуры Портала

В целом, разработанные и реализованные требования к Интернет-Порталу позволяют обеспечить надёжность и эффективность функционирования системы сбора, хранения, обработки и отображения распределённой информации о параметрах инновационных проектов.

Информационная и логическая структура Портала разработана с учётом поставленных задач и позволяет пользователям осуществлять обратную связь с Администратором, добавлять и редактировать новости, просматривать регламент и осуществлять статистическую обработку параметров инновационных проектов.

4. Методика прогнозирования параметров экологичности инновационных проектов

При обработке статистических данных, касающихся наличия природных ресурсов и числовых параметров инновационных проектов, часто возникает задача прогнозирования рассматриваемых числовых рядов на ближайшую перспективу. Сложность в получении адекватных прогнозов состоит в наличии неустранимой неопределённости, которая связана с рядом факторов.

В условиях рынка любая экономическая организация в своей деятельности, в том числе и инвестиционной, неизбежно сталкивается с неопределённостью. Фирма (компания) не обладает дос-

таточными (полными) данными о своём настоящем и будущем, она не в состоянии предугадать все изменения, которые могут произойти во внешней среде. Планирование как одно из составляющих контроллинга бизнес-процессов служит способом прояснения внутренних условий деятельности, уменьшения неопределённости и риска. Однако любой, даже самой крупной фирме, не по силам полностью устранить неопределённость и, следовательно, целиком спланировать свою деятельность.

Следует отметить, что еще одним источником неопределённости может быть и лицо, принимающее решение (ЛПР). Одна из проблем, связанных с ЛПР, – это нечёткость в понятиях, суждениях и предпочтениях, неопределённость временного промежутка, на котором сохраняется монотонность предпочтений и суждений ЛПР.

Как показано в работе [6] неопределённость может быть учтена с использованием теории нечётких множеств (ТНМ). При этом в случае использования экспертных оценок последние задаются в интервально-нечётком виде.

Для прогнозирования уровня обеспеченности природными ресурсами реализуемых инновационных проектов и их эколого-экономических параметров нами предлагается следующая новая методика:

1. Ретроспективный числовой ряд, содержащий фактические количественные показатели за истёкшие периоды (не менее 7), обрабатывается с целью построения наиболее адекватной аппроксимирующей кривой. При этом из возможных моделей (линейной, логарифмической, полиномиальной, степенной и экспоненциальной), выбирается та, у которой величина точности аппроксимации наибольшая.

Для приведённого на рис. 2 ряда наиболее адекватной оказалась полиномиальная модель (R^2=0,9593). Тут и далее под R^2 понимается оценка точности, рассчитанная по методу среднеквадратичного отклонения.

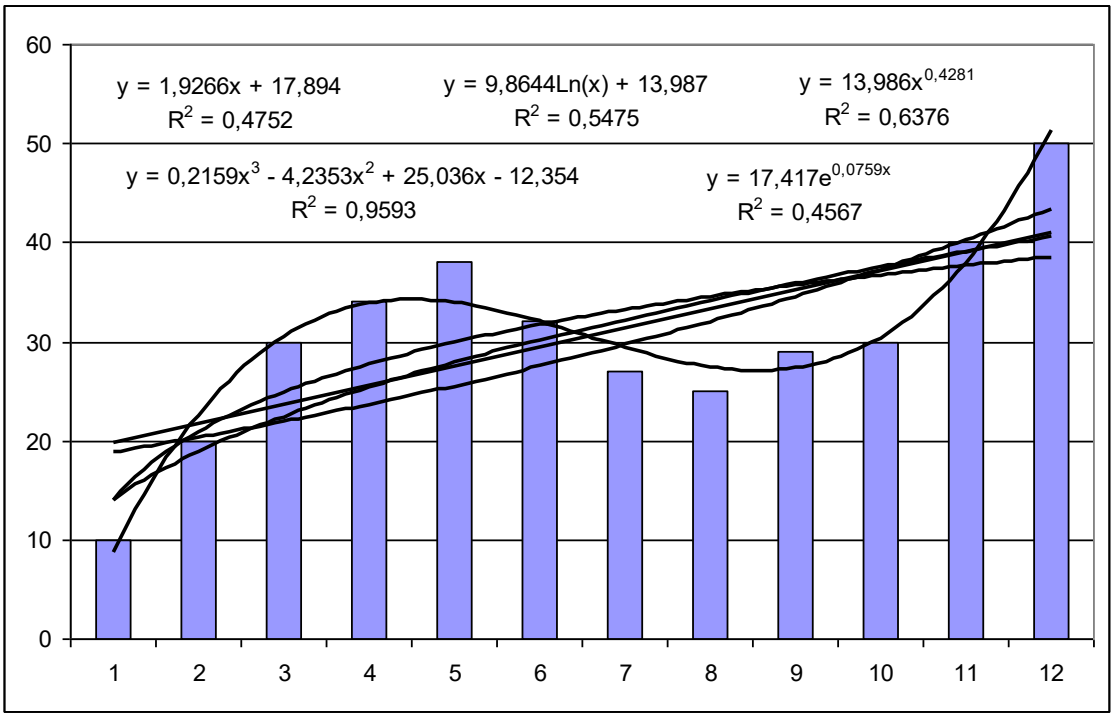


Рисунок 2 – Пример построения аппроксимирующей кривой

2. На основании полученного на предыдущем шаге уравнения рассчитываем прогноз для предстоящих временных интервалов (рис. 3). При x=13 прогнозное значение W=71,68.

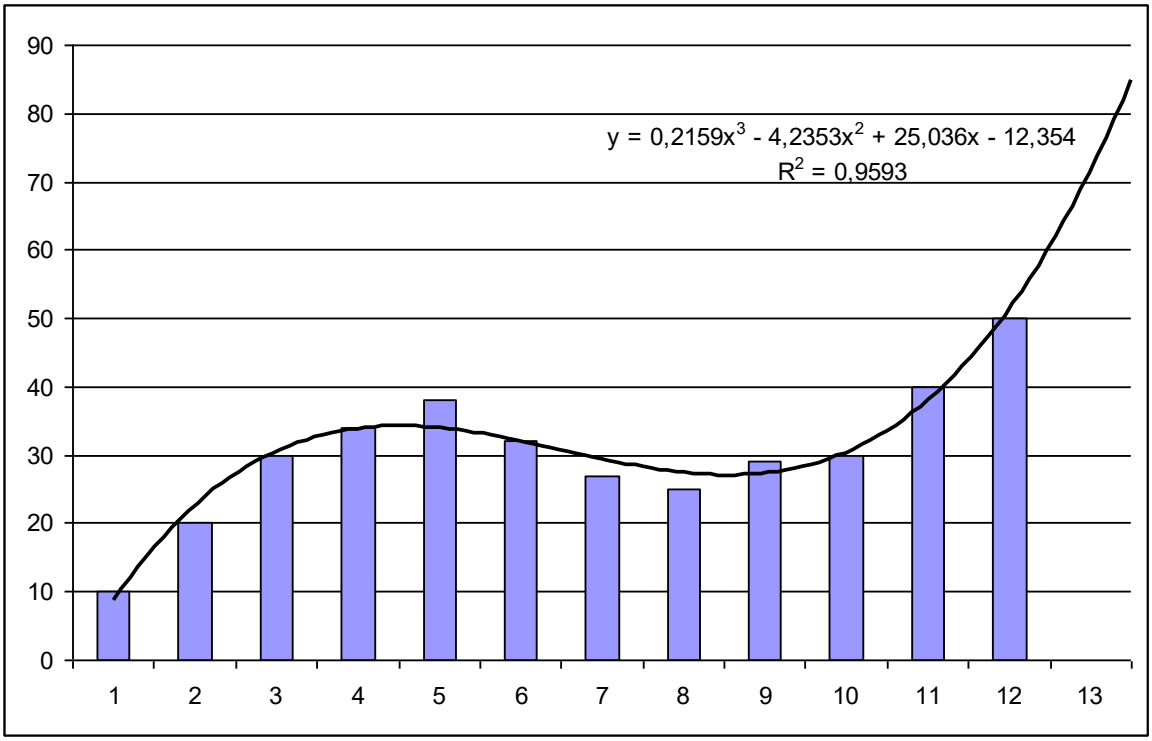


Рисунок 3 – Расчёт прогнозных значений

3. Рассчитанное прогнозное значение представляем в виде нечёткого множества: при $R^2<1$ форма функции принадлежности будет иметь треугольный вид (рис. 4а), при $R^2=1$ – вертикальная линия (рис. 4б). При этом основание треугольника будет равно произведению рассчитанного значения на удвоенную разность единицы и величины R^2. ($AB=2W(1-R^2)=5,83$)

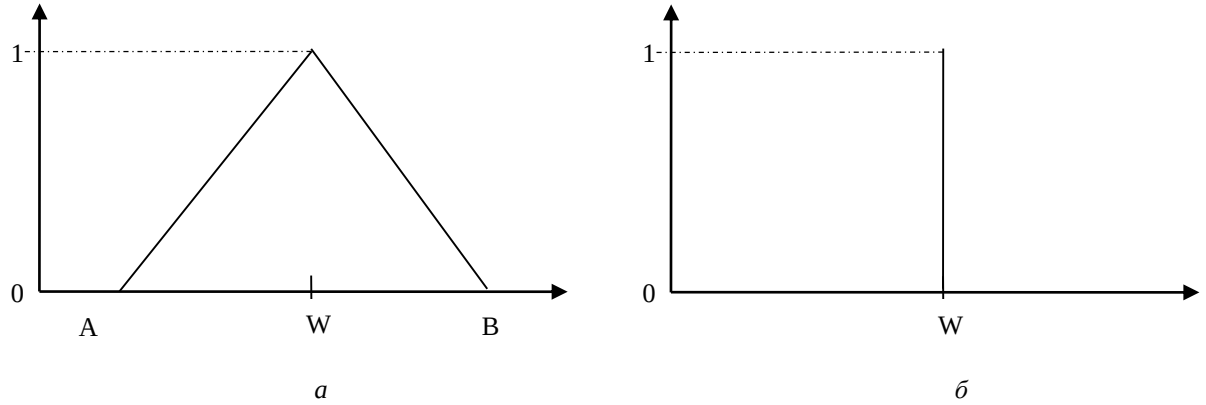


Рисунок 4 – Пример представления прогнозного значения в нечётком виде

4.Для учёта экспертной оценки формируем нечёткое множество, отражающее наиболее вероятное ожидаемое значение прогнозируемого параметра (рис. 5).

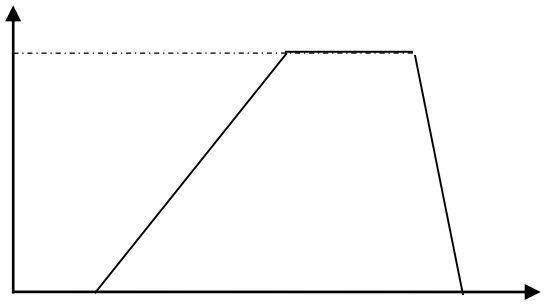


Рисунок 5 – Представление экспертной оценки в нечётком виде

5. Результирующее значение прогноза в виде нечёткого множества C находим как сумму двух нечётких множеств A и B: $C = A \cup B$. При этом функция принадлежности множества C имеет вид:

$$\mu_{A \cup B}(x) = \mu_A(x) \vee \mu_B(x) = \max(\mu_A(x), \mu_B(x)) \quad (1)$$

для каждого $x \in X$. Графическая интерпретация этой операции представлена на рис. 6.

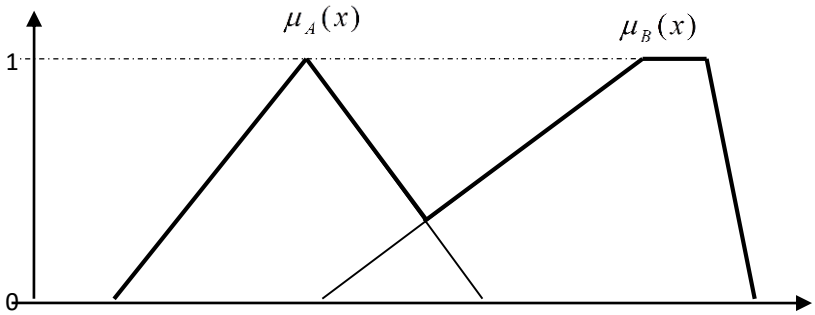


Рисунок 6 – Графическое представление операции суммирования нечётких множеств

Для получения чёткого числа используем методы дефуззификации – преобразования нечёткого множества к чёткой форме – методы центра тяжести и среднего центра. В случае если пересечением множеств A и B является не пустое множество $D = A \cap B \neq 0$, предпочтительнее использовать метод центра тяжести. Искомое значение $\bar{y}$ при этом рассчитывается как центр тяжести функции принадлежности $\mu_{B'}(y)$, т.е.

$$\bar{y} = \frac{\int_Y y\mu_{B'}(y)dy}{\int_Y \mu_{B'}(y)dy} = \frac{\int_Y y \max \mu_{\bar{B}^k}(y)}{\int_Y \max \mu_{\bar{B}^k}(y)} \quad (2)$$

при условии, что оба интеграла в приведённом выражении существуют.

В случае непересечения множеств A и B для дефуззификации используем метод по среднему центру. Значение $\bar{y}$ рассчитывается по формуле:

$$\bar{y} = \frac{\sum_{k=1}^{N} \mu_{\bar{B}^k}(\bar{y}^k)\bar{y}^k}{\sum_{k=1}^{N} \mu_{\bar{B}^k}(\bar{y}^k)}, \quad (3)$$

где $\bar{y}^k$ - это точка, в которой функция $\mu_{B^k}(y)$ принимает максимальное значение, т.е. $\mu_{B^k}(\bar{y}^k) = \max_y \mu_{B^k}(y)$. Точка $\bar{y}^k$ называется центром нечёткого множества B^k. На рис. 7 представлена идея этого метода. Обратим внимание, что значение $\bar{y}$ не зависит от формы и носителя функции принадлежности $\mu_{B^k}(y)$.

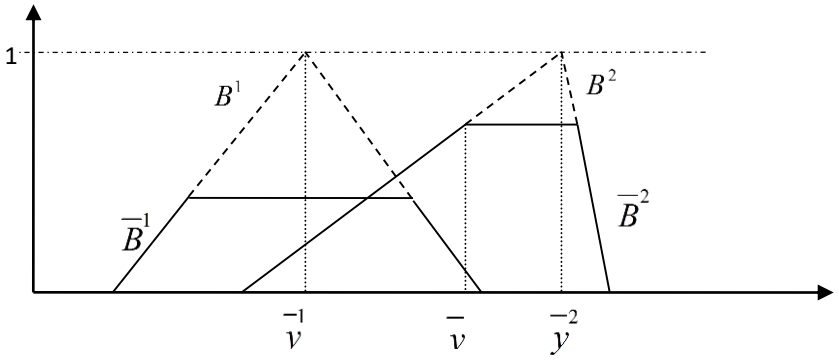


Рисунок 7 – Диффузификация по среднему центру

6. Для учёта значимости экспертной оценки в результирующем прогнозном значении будем использовать коэффициент $\alpha \in [0..1]$. С учётом этого формула (1) примет вид:

$$\mu_{A \cup B}(x) = \mu_A(x) \vee \mu_B(x) = \\ = \max([1-\lambda]\mu_A(x), \lambda\mu_B(x)) \quad (4)$$

Полученный таким образом прогноз лишён ряда недостатков (недостаточность статистиче-

ской информации для обоснованного применения вероятностных методов, высокая доля субъективизма при экспертном назначении оценок вероятностей, отсутствие полноты системы классификации неопределённости) и более адекватно отражает динамику изменения исследуемых параметров.

5. Система поддержки принятия решений для анализа эколого-экономических показателей ресурсоёмких инновационных проектов

Описанные выше компоненты информационной системы позволяют собирать, хранить, обрабатывать, прогнозировать и отображать данные, касающиеся эколого-экономических параметров инновационных проектов. Однако лицу, принимающему решение (например, для выбора одного, наиболее оптимального, проекта) необходимо представить инструмент, который позволит автоматизировать данный процесс. В основу системы поддержки принятия решений был положен модифицированный метод анализа иерархий с использованием теории нечётких множеств.

В отличие от строгой шкалы качественных сравнений альтернатив нами предложено использовать функции принадлежности. При этом и важность критериев, по которым производится выбор, определяется с учётом ТНМ. Такой подход представляется наиболее приближенным к реальности, когда в условиях выбора не всегда возможно чётко сравнить имеющиеся варианты и разграничить их по ценности.

В целом, технологическую цепочку описанной системы можно представить схематично (рис. 8). При этом в случае невозможности принятия решений на основании имеющейся информации необходимо вернуться на этап получения исходных данных и постановки задачи – в этом суть обратной связи.

В заключение следует отметить, что описанные выше методы и средства являются инструментами, предназначенными для лица, принимающего решение, и способны повысить эффективность его работы. Хотя разработанные компоненты не связаны единым пользовательским интерфейсом, по сути и назначению предложенные решения являются системой поддержки принятия решений.

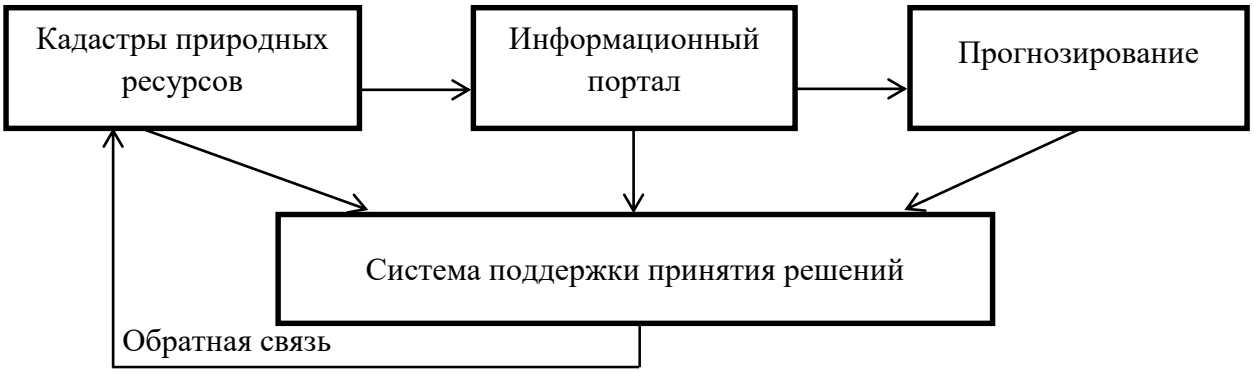


Рисунок 8 – Технологическая цепочка предложенной системы

6. Заключение

В результате выполненных исследований была разработана новая система для сбора, хранения, обработки и представления кадастровой информации. Информационная и логическая структура созданного интернет-портала разработана с учётом поставленных задач и позволяет пользователям осуществлять обратную связь с Администратором, добавлять и редактировать новости, просматривать регламент системы и осуществлять статистическую обработку параметров инновационных проектов. Разработанная методика прогнозирования параметров экологичности позволяет за счёт использования теории нечётких множеств учесть фактор неопределённости и использовать экспертные оценки для получения адекватных экстраполяционных моделей. Всё перечисленное позволило на основании полученных предметных данных предложить систему поддержки принятия решений на основании модифицированного метода анализа иерархий, что в свою очередь служит повышению эффективности выбора и реализации ресурсоёмких инновационных проектов.

Список литературы

1. Войтов И. В. Научно-аналитические подходы к оценкам состояния и повышения эффективности технологических процессов инновационных производств / И. В. Войтов, М. А. Гатих, В. А. Рыбак // Вестник БГУ. – 2009. – №1. – С. 100-106.
2. Дмитриев В. В. Многокритериальная оценка экологического состояния и устойчивость экосистем на основе метода сводных показателей / В. В. Дмитриев, Н. В. Мякишева, Н. В. Хованов // Вестник Санкт-Петербургского университета. – 1996. – Сер. 7. Геология, география. – Вып. 3 (21). – С. 52.
3. Рыбалов А. А. Качество окружающей среды: методы и подходы оценки / А.А. Рыбалов // Экологическая экспертиза. – М. : ВИНИТИ, 2001. – С. 12-67.
4. Парницкий Г. Основы статистической информатики / Г. Парницкий ; пер. с венг. Ю. А. Даниловой. – М. : Финансы и статистика, 1981. – 199 с.
5. Гатих М. А. Концепция развития информационной технологии формирования и управления информацией о состоянии и использовании природно-ресурсного потенциала регионов в Республике Беларусь / М. А. Гатих, О. А. Белый, И. И. Касьяненко ; Минприроды Республики Беларусь, РУП «БелНИЦ «Экология». – Минск : БелНИЦ «Экология», 2004. – 42 с.
6. Рыбак В. А. Методы принятия решений в экологии: от методологических основ к практическим алгоритмам / В.А. Рыбак. – Саарбрюкен: Lap Lambert Academic Publishing, 2014. – 376 с.

УДК 681.5

ЗАКОВОРОТНЫЙ А. Ю.

РАСШИРЕНИЕ ВОЗМОЖНОСТЕЙ ГЕОМЕТРИЧЕСКИЙ ТЕОРИИ УПРАВЛЕНИЯ ЗА СЧЕТ АВТОМАТИЗАЦИИ АНАЛИТИЧЕСКИХ ПРЕОБРАЗОВАНИЙ В ПАКЕТЕ MATLAB

Расширены возможности геометрический теории управления за счет автоматизации аналитических преобразований. Разработаны функции для пакета Matlab, позволяющие преобразовывать нелинейные модели объектов высокого порядка к эквивалентным линейным моделям с помощью инволютивных распределений геометрический теории управления. Выполнен синтез линейной математической модели движения вагонов дизель-поезда в форме Бруновского, которая учитывает параллельную работу двух эквивалентных тяговых асинхронных электроприводов обмоторенных вагонов. Сравнение процессов в исходной и линейной математических моделях в разных режимах работы подтвердило правильность линеаризации обратной связью исходной модели и работоспособность разработанных функций. Полученная модель может использоваться для поиска оптимальных управлений, а также для исследования процессов буксования и юза, а также параллельной работы двигателей.

New features of the geometric control theory by automating the analytical transformations. Designed functions for the package Matlab allow to convert the non-linear model object to the equivalent linear model using involutive distributions of geometric control theory. The synthesis of linear mathematical model of the motion wagons diesel train in the form of Brunovsky, which allows for parallel operation of two equivalent traction asynchronous electric motor carriages. Comparison of processes in the source and linear mathematical models in different modes of operation confirmed the correctness of the feedback linearization of the original model and the performance of the developed functions. The resulting model can be used to find the optimal controls, as well as for studying the processes of slipping and skidding as well as parallel operation of motors.

Ключевые слова: пакет Matlab, геометрическая теория управления, оптимальное управление.

1. Введение

Трудности анализа и синтеза нелинейных систем управления общеизвестны. Поэтому на протяжении многих лет ведется поиск более мощных теоретических средств для решения фундаментальных проблем теории управления. Одним их таких средств является современная геометрия, в частности, геометрический подход к теории управления на основе теории групп и дифференциальной геометрии. Успехи этого подхода привели к интенсивной разработке нового научного направления – единой геометрической теории управления [1, 2]. Преимущество нового научного направления состоит, с одной стороны, в создании математического аппарата, позволяющего описывать системы управления в пространствах состояний более общих, чем линейные пространства, что необходимо при решении целого ряда задач управления [1, 2], а с другой стороны, в реальной возможности преобразований нелинейных систем высокого порядка с несколькими управлениями к эквивалентным линейным, за счет разбиения исходных моделей на ряд подсистем меньшей размерности, в каждую из которых входит только одно управление. Такие преобразования открывают возможности для использования при решении задач разработки нелинейных систем управления методов и средств теории линейных систем [1, 3]. При этом линеаризация нелинейной системы выполняется не с помощью классического разложения в ряд Тейлора, а на основе использования линейной обратной связи в пространстве "вход – выход" или "вход – состояние". Теоретически линеаризация с помощью обратной связи позволяет преобразовать к линейному виду широкий класс нелинейных систем управления [1 – 8]. Однако в этом случае необходимо выполнять трудоемкие аналитические преобразования, которые не автоматизированы ни в одном из известных пакетов моделирования и которые стали причиной разрыва между теоретическими результатами геометрической теории управления и решением практических задач синтеза систем управления для нелинейных объектов не выше 3 – 5 порядка [9, 10].

2. Постановка задачи

Расширить возможности геометрический теории управления за счет разработки функций для универсального пакета моделирования Matlab, позволяющих автоматизировать сложные аналитические преобразования геометрической теории управления при линеаризации моделей объектов высокого порядка. Продемонстрировать работоспособности разработанных функций при синтезе линейной математической модели движение вагонов дизель-поезда форме Бруновского, которая учитывает параллельную работу двух эквивалентных тяговых асинхронных электроприводов обмоторенных вагонов.

3. Линеаризация нелинейных моделей

Для расширения возможностей применения геометрический теории управления при динамической линеаризации нелинейных систем высокого порядка по описанному в работе [10] алгоритму, были разработаны функции для универсального пакета моделирования Matlab, которые выполняют следующие действия: формируют векторные поля объектов по их моделям, проверяют выполнение условий инволютивности (условия возможности совместного интегрирования векторных полей), вычисляют производные Ли любого порядка, осуществляют преобразование всех переменных нелинейной модели объекта в переменные модели в форме Бруновского, а также выполняют интегрирование систем обыкновенных дифференциальных уравнений. Продемонстрируем эти функции в процессе синтеза линейной математической модели движения вагонов дизель-поезда в форме Бруновского, которая учитывает параллельную работу двух эквивалентных тяговых асинхронных электроприводов.

Математическая модель, которая описывает процесс движения трех вагонов дизель-поезда по ровному участку железнодорожного пути и эквивалентные тяговые асинхронные электроприводы двух обмоторенных вагонов поезда, может быть представлена следующей системой из шестнадцати нелинейных дифференциальных уравнений:

$$\begin{gathered}
\frac{dx_1}{dt} = a_{11}x_2; \\
\frac{dx_2}{dt} = a_{21}x_6x_8 + a_{22}x_4x_7 + a_{23}x_3 + a_{242}; \\
\frac{dx_3}{dt} = a_{31}x_2 + a_{32}x_{11}; \\
\frac{dx_4}{dt} = a_{41}x_4 + a_{42}x_6 + U_1; \\
\frac{dx_5}{dt} = a_{51}x_6x_8 + a_{52}x_4x_7 + a_{53}x_5 + a_{54}x_5^2 + a_{55}; \\
\frac{dx_6}{dt} = a_{61}x_6 + a_{62}x_4 - x_5x_7; \\
\frac{dx_7}{dt} = a_{71}x_7 + a_{72}x_8 - x_5x_6; \\
\frac{dx_8}{dt} = a_{81}x_8 + a_{82}x_7 + U_2; \qquad (1) \\
\frac{dx_9}{dt} = a_{91}x_{14}x_{16} + a_{92}x_{12}x_{15} + a_{93}x_{10} + a_{94}; \\
\frac{dx_{10}}{dt} = a_{101}x_{11} + a_{102}x_9; \\
\frac{dx_{11}}{dt} = a_{111}x_3 + a_{112}x_{10} + a_{113}; \\
\frac{dx_{12}}{dt} = a_{121}x_{12} + a_{122}x_{14} + U_3; \\
\frac{dx_{13}}{dt} = a_{131}x_{14}x_{16} + a_{132}x_{12}x_{15} + a_{133}x_{13} + a_{134}x_{13}^2 + a_{135}; \\
\frac{dx_{14}}{dt} = a_{141}x_{14} + a_{142}x_{12} - x_{13}x_{15}; \\
\frac{dx_{15}}{dt} = a_{151}x_{15} + a_{152}x_{16} - x_{13}x_{14}; \\
\frac{dx_{16}}{dt} = a_{161}x_{16} + a_{162}x_{15} + U_4,
\end{gathered}$$

где x_1 – расстояние, пройденное от начала перегона; t – время; $a_{11}, a_{21}, a_{22}, a_{23}, a_{24}, a_{31}, ..., a_{161}, a_{162}$ – постоянные коэффициенты; x_2, x_9, x_{11} – скорости движения, соответственно первого, третьего и второго вагонов дизель-поезда; x_6, x_{14} – проекции на ось α потокосцеплений ротора соответственно первого и второго эквивалентных двигателей; x_8, x_{16} – проекции на ось β потокосцеплений статора соответственно первого и второго эквивалентных двигателей; x_4, x_{12} – проекции на ось α потокосцеплений статора соответственно первого и второго эквивалентных двигателей; x_7, x_{15} – проекции на ось β потокосцеплений ротора соответственно первого и второго эквивалентных двигателей; x_3, x_{10} – силы, действующие между соответственно первым и вторым, и вторым и третьим вагонами дизель-поезда; U_1, U_2 – проекции соответственно на оси α и β напряжений

обмоток статора первого эквивалентного двигателя;

U_3, U_4 – проекции соответственно на оси α и β напряжений обмоток статора второго эквивалентного двигателя; x_5, x_{13} – угловые скорости вращения роторов соответственно первого и второго эквивалентных двигателей.

Обозначив $\frac{dx_i}{dt} = f_i,\ i = \overline{1,15},\ i \neq 4,\ 8,\ 12;$ $\frac{dx_j}{dt} = f_j + U_{j/4},\ j = 4,\ 8,\ 12{,}16,$ получим, что с системой дифференциальных уравнений (1) связаны следующие векторные поля:

$$\boldsymbol{X}(\boldsymbol{x}) = |\ f_1,\ f_2,\ f_3,\ f_4,\ f_5,\ f_6, \ldots,\ f_{16}\ |^{\mathrm{T}};$$

$$\boldsymbol{Y}_1 = |0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0|^{\mathrm{T}};$$

$$\boldsymbol{Y}_2 = |0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0|^{\mathrm{T}}; (2)$$

$$\boldsymbol{Y}_3 = |0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0|^{\mathrm{T}};$$

$$\boldsymbol{Y}_4 = |0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1|^{\mathrm{T}}.$$

Векторные поля (2) в пакете Matlab могут быть заданы следующим образом:

```
f1 = sym('a11 * x2');
f2 = sym('a21 * x6 * x8 + a22 * x4 * x7 + a23 *
x3 + a24');
f3 = sym('a31 * x2 + a32 * x11');
f4 = sym('a41 * x4 + a42 * x6');
……................................................

f10 = sym('a101 * x11 + a102 * x9');
f11 = sym('a111 * x3 + a112 * x10 + a113');
f12 = sym('a121 * x12 + a122 * x14');
f13 = sym('a131 * x14 * x16 + a132 * x12 *
x15 + a133 * x13 + a134 * x13^2 + a135');
f14 = sym('a141 * x14 + a142 * x12 - x13
* x15');
f15 = sym('a151 * x15 + a152 * x16 - x13
* x14');
f16 = sym('a161 * x16 + a162 * x15');
X = [f1; f2; f3; f4; f5; f6; f7; f8; f9; f10; f11;
f12; f13; f14; f15; f16];

Y1 = [0;0;0;sym('1');0;0;0;0;0;0;0;0;0;0;0;0];
Y2 = [0;0;0;0;0;0;0;sym('1');0;0;0;0;0;0;0;0];
Y3 = [0;0;0;0;0;0;0;0;0;0;0;sym('1');0;0;0;0];
Y4 = [0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;sym('1')];
x = [sym('x1') 'x2' 'x3' 'x4' 'x5' 'x6' 'x7' 'x8' 'x9'
'x10' 'x11' 'x12' 'x13' 'x14' 'x15' 'x16'];
```

Система уравнений (1) может быть преобразована к форме Бруновского только в случае, если инволютивны распределения $\boldsymbol{M}^0 = \mathrm{span}\{\boldsymbol{Y}_1, \boldsymbol{Y}_2, \boldsymbol{Y}_3, \boldsymbol{Y}_4\}$, $\boldsymbol{M}^1 = \mathrm{span}\{\boldsymbol{Y}_1, \boldsymbol{Y}_2, \boldsymbol{Y}_3, \boldsymbol{Y}_4, \boldsymbol{L}_{\boldsymbol{X}}\boldsymbol{Y}_1, \boldsymbol{L}_{\boldsymbol{X}}\boldsymbol{Y}_2, \boldsymbol{L}_{\boldsymbol{X}}\boldsymbol{Y}_3, \boldsymbol{L}_{\boldsymbol{X}}\boldsymbol{Y}_4\}$ и $\boldsymbol{M}^2$ для этой системы [2], где $\mathrm{span}\{\boldsymbol{Y}_1, \boldsymbol{Y}_2, \boldsymbol{Y}_3, \boldsymbol{Y}_4\}$ – линейная оболочка векторов $\boldsymbol{Y}_1, \boldsymbol{Y}_2, \boldsymbol{Y}_3, \boldsymbol{Y}_4$, $\boldsymbol{L}_{\boldsymbol{X}}\boldsymbol{Y}_k$ $(k = \overline{1,4})$ – производные Ли вдоль векторного поля $\boldsymbol{X}$ векторных полей $\boldsymbol{Y}_k$ $(k = \overline{1,4})$.

Для проверки возможности преобразования широкого класса нелинейных систем управления к канонической форме Бруновского для пакета моделирования *Matlab* была разработана функция *involutivity*(*M*, *x*), проверяющая выполнение условий инволютивности последовательности распределений, которая возвращает значение "1", если для распределения *M* условия инволютивности выполняются и значение "0" если нет [12]. Функция проверки условий инволютивности представляет собой следующую последовательность команд:

```
function involutive = involutivity(S, x)
saved_rank = rank(S);
lenght = size(S, 2);
for i = 1 : (lenght - 1)
for j = (i + 1) : lenght
S = [S difflie(S(:,i), S(:,j), x)]; end end
involutive = saved_rank == rank(S); end
```

Проверка инволютивности распределения $\boldsymbol{M}^0$ в пакете моделирования Matlab с использованием описанной выше функции осуществляется следующим образом:

```
M0 = [Y1, Y2, Y3, Y4];
involutive = involutivity(M0, x);
>> involutive = 1
```

Поскольку векторные поля $\boldsymbol{Y}_k$ $(k = \overline{1,4})$ постоянны, то распределение $\boldsymbol{M}^0$ – инволютивно и размерность распределения $\dim \boldsymbol{M}^0 = 4$.

Проанализируем распределение $\boldsymbol{M}^1$, для этого сначала осуществим вычисление производных Ли векторных полей $\boldsymbol{Y}_k$ $(k = \overline{1,4})$ вдоль векторного поля $\boldsymbol{X}$ с помощью разработанной для пакете моделирования Matlab функции *Dif_Li*(*X*, *Y*, *x*, *N*), которая возвращает *N*-ю производную Ли вдоль векторного поля *X* векторного поля *Y*, по элементам вектора *x* [13]. Функция вычисления производной Ли представляет собой следующую последовательность команд:

```
function U = Dif_Li(X, Y, variables, N)
U(:, 1) = sym(Y);
for i = 2 : N + 2
U(:, i) = difflie(X, U(:, i - 1 ), variables); end
end
```

Проверка инволютивности распределения $\boldsymbol{M}^1$ в пакете моделирования Matlab с использованием разработанных функций *Dif_Li* и *involutivity* осуществляется следующим образом:

```
C1_1 = Dif_Li(X, Y1, x, 2);
M1_1 = C1_1(:, 1 : (size(C1_1, 2) - 1));
C1_2 = Dif_Li(X, Y2, x, 2);
M1_2 = C1_2(:, 1 : (size(C1_2, 2) - 1));
……————————————————————————————
C1_4 = Dif_Li(X, Y4, x, 2);
M1_4 = C1_4(:, 1 : (size(C1_4, 2) - 1));
M1=[M1_1(:,1), M1_2(:,1), M1_3(:,1),
M1_4(:,1), M1_1(:,2), M1_2(:,2), M1_3(:,2),
M1_4(:,2)];
involutive = involutivity(M1, x);
>> involutive = 0
```

Проверка условий инволютивности показывает, что распределение $\boldsymbol{M}^1$ не является инволютивным, однако проверка всех его подраспределений $\boldsymbol{M}_k^1 = \mathrm{span}\{\boldsymbol{Y}_1, \boldsymbol{Y}_2, \ \boldsymbol{Y}_3, \boldsymbol{Y}_4, \ \boldsymbol{L}_{\boldsymbol{X}}\boldsymbol{Y}_k\}$, $k = \overline{1,4}$, показывает, что они являются инволютивными:

```
M11=[M1_1(:,1), M1_2(:,1), M1_3(:,1),
M1_4(:,1), M1_1(:,2)];
involutive = involutivity(M11, x);
>>involutive = 1
……————————————————————————————
M14=[M1_1(:,1), M1_2(:,1), M1_3(:,1),
M1_4(:,1), M1_4(:,2)];
involutive = involutivity(M14, x);
>>involutive = 1
```

Поэтому дополнительные переменные или интеграторы можно вводить в любой канал управления. Однако введение 1, 2, 3 или 4 интеграторов в любые каналы не позволяет решить проблему получения инволютивного распределения $\boldsymbol{M}^1$ для расширенной системы. Распределение $\boldsymbol{M}^1$ становится инволютивным только при введении одного интегратора в каждый канал объекта управления.

Для расширенной модели объекта управления введем следующие обозначения:

$$y_i = x_i, \ i = \overline{1,4}; \ \ y_{5*j} = U_j, \ \ \frac{dy_{5*j}}{dt} = U_j^*, \ j = \overline{1,4};$$

$$y_k = x_{k-1}, \ k = \overline{6,9}; \ \ y_g = x_{g-2}, \ g = \overline{11,14};$$

$$y_q = x_{q-3}, \ \ g = \overline{16,19}.$$

В этих обозначениях расширенная модель объекта записывается таким образом:

$$\frac{dy_1}{dt} = a_{11}y_2 = \varphi_1;$$

$$\frac{dy_2}{dt} = a_{21}y_7y_9 + a_{22}y_4y_8 + a_{23}y_3 + a_{24} = \varphi_2;$$

$$\frac{dy_3}{dt} = a_{31}y_2 + a_{32}y_{13} = \varphi_3;$$

$$\frac{dy_4}{dt} = a_{41}y_4 + a_{42}y_7 + y_5 = \varphi_4;$$

$$\frac{dy_5}{dt} = U_1^*; \ \ \varphi_5 = 0;$$

$$\frac{dy_6}{dt} = a_{51}y_7y_9 + a_{52}y_4y_8 + a_{53}y_6 + a_{54}y_6^2 + a_{55} = \varphi_6;$$

$$\frac{dy_7}{dt} = a_{61}y_7 + a_{62}y_4 - y_6y_8 = \varphi_7;$$

$$\frac{dy_8}{dt} = a_{71}y_8 + a_{72}y_9 - x_6x_7 = \varphi_8;$$

$$\frac{dy_9}{dt} = a_{81}y_9 + a_{82}y_8 + y_{10} = \varphi_9;$$

$$\frac{dy_{10}}{dt} = U_2^*; \ \ \varphi_{10} = 0;$$

$$\frac{dy_{11}}{dt} = a_{91}y_{17}y_{19} + a_{92}y_{14}y_{18} + a_{93}y_{12} + a_{94} = \varphi_{11};$$

$$\frac{dy_{12}}{dt} = a_{101}y_{13} + a_{102}y_{11} = \varphi_{12};$$

$$\frac{dy_{13}}{dt} = a_{111}y_3 + a_{112}y_{12} + a_{113} = \varphi_{13};$$

$$\frac{dy_{14}}{dt} = a_{121}y_{14} + a_{122}y_{17} + y_{15} = \varphi_{14};$$

$$\frac{dy_{15}}{dt} = U_3^*; \ \ \varphi_{15} = 0;$$

$$\frac{dy_{16}}{dt} = a_{131}y_{17}y_{19} + a_{132}y_{14}y_{18} + a_{133}y_{16} + a_{134}y_{16}^2 +$$

$$+ a_{135} = \varphi_{16};$$

$$\frac{dy_{17}}{dt} = a_{141}y_{17} + a_{142}y_{14} - y_{16}y_{18} = \varphi_{17};$$

$$\frac{dy_{18}}{dt} = a_{151}y_{18} + a_{152}y_{19} - y_{16}y_{17} = \varphi_{18};$$

$$\frac{dy_{19}}{dt} = a_{161}y_{19} + a_{162}y_{18} + y_{20} = \varphi_{19};$$

$$\frac{dy_{20}}{dt} = U_4^*; \ \varphi_{20} = 0.$$

С этой моделью объекта управления связаны векторные поля:

$$\boldsymbol{Y}(\boldsymbol{y}) = \left|\varphi_1, \varphi_2, \varphi_3, \varphi_4, \varphi_5, \varphi_6, \varphi_7, \varphi_8, \varphi_9, \varphi_{10}, \varphi_{11}, \varphi_{12}, \varphi_{13}, \varphi_{14}, \varphi_{15}, \varphi_{16}, \varphi_{17}, \varphi_{18}, \varphi_{19}, \varphi_{20}\right|^{\mathrm{T}};$$

$$\boldsymbol{Y}_1^* = \begin{vmatrix} 0\\0\\0\\0\\1\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0 \end{vmatrix}; \quad \boldsymbol{Y}_2^* = \begin{vmatrix} 0\\0\\0\\0\\0\\0\\0\\0\\0\\1\\0\\0\\0\\0\\0\\0\\0\\0\\0 \end{vmatrix}; \quad \boldsymbol{Y}_3^* = \begin{vmatrix} 0\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0\\1\\0\\0\\0\\0 \end{vmatrix}; \quad \boldsymbol{Y}_4^* = \begin{vmatrix} 0\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0\\1 \end{vmatrix},$$

которые в пакете Matlab могут быть заданы следующим образом:

```
f1 = sym('a11 * y2');
f2 = sym('a21 * y7 * y9 + a22 * y4 * y8 + a23 *
y3 + a24');
f3 = sym('a31 * y2 + a32 * y13');
f4 = sym('a41 * y4 + a42 * y7 + y5');
f5 = sym('0');
f6 = sym('a51 * y7 * y9 + a52 * y4 * y8 + a53 *
y6 + a54 * y6^2 + a55');
f7 = sym('a61 * y7 + a62 * y4 - y6 * y8');
f8 = sym('a71 * y8 + a72 * y9 - y6 * y7');

f9 = sym('a81 * y9 + a82 * y8 + y10');
f10 = sym('0');
……................................................

f16 = sym('a131 * y17 * y19 + a132 * y14 *
y18 + a133 * y16 + a134 * y16^2 + a135');
f17 = sym('a141 * y17 + a142 * y14 - y16
* y18');
f18 = sym('a151 * y18 + a152 * y19 - y16
* y17');
f19 = sym('a161 * y19 + a162 * y18 + y20');
f20 = sym('0');
Y_new = [f1; f2; f3; f4; f5; f6; f7; f8; f9; f10;
f11; f12; f13; f14; f15; f16; f17; f18; f19; f20];
Y1_new = [0; 0; 0; 0; sym('1'); 0; 0; 0; 0; 0; 0;
0; 0; 0; 0; 0; 0; 0; 0; 0];
Y2_new = [0; 0; 0; 0; 0; 0; 0; 0; 0; sym('1'); 0;
0; 0; 0; 0; 0; 0; 0; 0; 0];
Y3_new = [0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0;
sym('1'); 0; 0; 0; 0; 0];
Y4_new = [0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0;
0; 0; 0; 0; 0; sym('1')];
y_new = [sym('y1') 'y2' 'y3' 'y4' 'y5' 'y6' 'y7' 'y8'
'y9' 'y10' 'y11' 'y12' 'y13' 'y14' 'y15' 'y16' 'y17'
'y18' 'y19' 'y20'];
```

Поскольку вектора $\boldsymbol{Y}_k^*$ $(k = \overline{1,4})$ постоянны, то распределение $\boldsymbol{M}^{0*} = \mathrm{span}\{\boldsymbol{Y}_1^*, \boldsymbol{Y}_2^*, \boldsymbol{Y}_3^*, \boldsymbol{Y}_4^*\}$ – инволютивно, что подтверждает и программное обеспечение:

```
M0_new=[Y1_new,Y2_new, Y3_new,
Y4_new];
involutive = involutivity(M0_new, y_new);
>> involutive = 1
```

Так как производные Ли вдоль векторного поля $\boldsymbol{Y}$ векторных полей $\boldsymbol{Y}_k^*$ $(k = \overline{1,4})$ являются постоянными векторами, то распределение $\boldsymbol{M}^{1*}$ для расширенной системы является инволютивным:

```
C1_1_new =
Dif_Li(Y_new, Y1_new, y_new, 2);
M1_1_new =
C1_1_new(:, 1 : (size(C1_1_new, 2) - 1));
C1_2_new =
Dif_Li(Y_new, Y2_new, y_new, 2);
M1_2_new =
C1_2_new(:, 1 : (size(C1_2_new, 2) - 1));
C1_3_new =
Dif_Li(Y_new, Y3_new, y_new, 2);
M1_3_new =
C1_3_new(:, 1 : (size(C1_3_new, 2) - 1));

C1_4_new =
Dif_Li(Y_new, Y4_new, y_new, 2);
M1_4_new =
C1_4_new(:, 1 : (size(C1_4_new, 2) - 1));
M1_new = [M1_1_new(:,1), M1_2_new(:,1),
M1_3_new(:,1), M1_4_new(:,1),
M1_1_new(:,2), M1_2_new(:,2),
M1_3_new(:,2), M1_4_new(:,2)];
involutive = involutivity(M1_new, y_new);
>> involutive = 1
```

Проверка инволютивности распределения $\boldsymbol{M}^{2*} = \mathrm{span}\{\boldsymbol{Y}_1^*, ..., \boldsymbol{Y}_4^*, \quad \boldsymbol{L}_{\boldsymbol{Y}}\boldsymbol{Y}_1^*, \ ..., \boldsymbol{L}_{\boldsymbol{Y}}\boldsymbol{Y}_4^*, \quad \boldsymbol{L}_{\boldsymbol{Y}}^2\boldsymbol{Y}_1^*, \boldsymbol{L}_{\boldsymbol{Y}}^2\boldsymbol{Y}_2^*, ..., \boldsymbol{L}_{\boldsymbol{Y}}^2\boldsymbol{Y}_4^*\}$, где $\boldsymbol{L}_{\boldsymbol{Y}}^2\boldsymbol{Y}_{\boldsymbol{k}}$ $(k = \overline{1,4})$ – производные Ли второго порядка, показывает, что оно не является инволютивным:

```
M2_new = [M1_1_new(:,1), M1_2_new(:,1),
M1_3_new(:,1), M1_4_new(:,1),
M1_1_new(:,2), M1_2_new(:,2),
M1_3_new(:,2), M1_4_new(:,2),
M1_1_new(:,3), M1_2_new(:,3),
M1_3_new(:,3), M1_4_new(:,3)];
involutive = involutivity(M2_new, y_new);
>> involutive = 0
```

Однако инволютивными являются всех подраспределения $\boldsymbol{M}_k^{2*} = \mathrm{span}\{\boldsymbol{Y}_1, ..., \boldsymbol{Y}_4, \boldsymbol{L}_{\boldsymbol{Y}}\boldsymbol{Y}_1^*, \boldsymbol{L}_{\boldsymbol{Y}}\boldsymbol{Y}_2^*, ..., \boldsymbol{L}_{\boldsymbol{Y}}\boldsymbol{Y}_4^*, \boldsymbol{L}_{\boldsymbol{Y}}^2\boldsymbol{Y}_k^*\}$, $k = \overline{1,4}$, распределения $\boldsymbol{M}^{2*}$:

```
M21_new = [M1_1_new(:,1),
M1_2_new(:,1), M1_3_new(:,1),
M1_4_new(:,1), M1_1_new(:,2),
M1_2_new(:,2), M1_3_new(:,2),
M1_4_new(:,2), M1_1_new(:,3)];
involutive = involutivity(M21_new, y_new)
>> involutive = 1
……................................................

M24_new = [M1_1_new(:,1),
M1_2_new(:,1), M1_3_new(:,1),
M1_4_new(:,1), M1_1_new(:,2),
M1_2_new(:,2), M1_3_new(:,2),
M1_4_new(:,2), M1_4_new(:,3)];
involutive = involutivity(M24_new, y_new)
>> involutive = 1
```

Этого оказывается достаточно для осуществления динамической линеаризации и получения системы линейных дифференциальных уравнений в форме Бруновского. На основании теоремы о линейных эквивалентах для нелинейных аффинных систем с m управлениями [2], получим, что каноническая форма Бруновского имеет 4-и клетки (индекс управляемости $k_{\max}$ для данного объекта равен 4), т.е. расширенная модель объекта управления разбивается на 4-и подсистемы уравнений пятого порядка, в каждую из которых входит только одно управление. Математическая модель объекта управления в форме Бруновского в пространстве "вход – состояние" имеет следующий вид:

$$\begin{aligned} &\frac{dz_i}{dt} = z_{i+1}, \ i = \overline{1,19}, \ i \neq 5,\ 10,\ 15; \\ &\frac{dz_5}{dt} = v_1; \ \frac{dz_{10}}{dt} = v_2; \ \frac{dz_{15}}{dt} = v_3; \ \frac{dz_{20}}{dt} = v_4, \end{aligned} \tag{3}$$

где v_j ($j = \overline{1,4}$) – управления.

Поскольку модель объекта в форме Бруновского имеет 4-и клетки, то необходимо определить 4-и функции $T_j(\boldsymbol{y})$ ($j = \overline{1,4}$), преобразующих переменные расширенной модели объекта управления в переменные модели в форме Бруновского:

$$z_1 = T_1(\boldsymbol{y}); z_6 = T_2(\boldsymbol{y}); z_{11} = T_3(\boldsymbol{y}); z_{16} = T_4(\boldsymbol{y}).$$

Методика определения этих функций известна [2, 9 – 11]. В данном случае они являются однокомпонентными составляющими вектора $\boldsymbol{y} = (y_1, y_2, ..., y_{20})$. Из этих функций путем последовательного дифференцирования вдоль векторного поля $\boldsymbol{Y}^* = \boldsymbol{Y} + U_1\boldsymbol{Y}_1^* + U_2\boldsymbol{Y}_2^* + ... + U_4\boldsymbol{Y}_4^*$ можно получить выражения для определения соответственно z_2, z_3, z_4, z_5 (из функции $T_1(\boldsymbol{y})$), z_7, z_8, z_9, z_{10} (из функции $T_2(\boldsymbol{y})$), $z_{12}, z_{13}, z_{14}, z_{15}$ (из функции $T_3(\boldsymbol{y})$), $z_{17}, z_{18}, z_{19}, z_{20}$ (из функции $T_4(\boldsymbol{y})$).

В качестве примера рассмотрим получение зависимостей для определения z_2, z_3, z_4, z_5 с помощью функции $T_1(\boldsymbol{y})$.

Для вычисления функций перехода от переменных расширенной нелинейной модели объекта управления к переменным модели (3) в форме Бруновского применялась специально разработанная для универсального пакета моделирования Matlab функция *brunovsky*(*X, Y, T, x, N*), которая выполняет дифференцирование функции *T* вдоль векторных полей *X* и *Y*, по элементам вектора *x*, и возвращает для каждой клетки формы Бруновского массив из *N* выражений связывающих переменные в линейной и нелинейной моделях, а также новое управление для соответствующей клетки линейной математической модели в форме Бруновского [14]. Функция *brunovsky* представляет собой следующую последовательность команд:

```
function [Z, V] = brunovsky(X, Y, T, x, N)
Z(1, 1) = T;
for i = 2 : N
```

```
Z(i, 1) = diffvec(Z(i-1, 1), x, 1) * X; end
V = diffvec (Z(N, 1), x, 1) * X;
end
```

Для исследуемого объекта управления имеем: $T_1(\boldsymbol{y}) = y_1$, поэтому $z_1 = y_1$.

```
T1 = [sym('y1')];
[Z1_5, V1] = brunovsky(Y_new, [Y1_new], T1,
y_new, 5);
Z1_5 = simple(Z1_5)
```

Дифференцируя функцию $T_1(\boldsymbol{y})$ вдоль векторного поля $\boldsymbol{Y}^*$ и учитывая, что z_2, z_3, z_4 и их производные не зависят от управлений, получим функции перехода к канонической форме Бруновского:

$$z_2 = \frac{dz_1}{dt} = a_{11}y_2;$$

$$z_3 = \frac{dz_2}{dt} = a_{11}(a_{24} + a_{23}y_3 + a_{22}y_4y_8 + a_{21}y_7y_9);$$

$$z_4 = \frac{dz_3}{dt} = a_{11}a_{23}(a_{32}y_{13} + a_{31}y_2) + a_{11}a_{21}y_9(a_{62}y_4 + a_{61}y_7 - y_6y_8) + a_{11}a_{22}y_4(a_{71}y_8 + a_{72}y_9 - y_6y_7) + a_{11}a_{22}y_8(y_5 + a_{41}y_4 + a_{42}y_7) + a_{11}a_{21}y_7(y_{10} + a_{82}y_8 + a_{81}y_9);$$

$$z_5 = \frac{dz_4}{dt} = a_{11}(y_5 + a_{41}y_4 + a_{42}y_7)(a_{22}a_{41}y_8 + a_{21}a_{62}y_9 + a_{22}a_{71}y_8 + a_{22}a_{72}y_9 - a_{22}y_6y_7) - a_{11}(a_{22}y_4y_7 + a_{21}y_8y_9)(a_{54}y_6^2 + a_{53}y_6 + a_{55} + a_{52}y_4y_8 + a_{51}y_7y_9) + a_{11}(y_{10} + a_{82}y_8 + a_{81}y_9)\cdot(a_{21}a_{62}y_4 + a_{22}a_{72}y_4 + a_{21}a_{61}y_7 + a_{21}a_{81}y_7 - a_{21}y_6y_8) + a_{11}(a_{62}y_4 + a_{61}y_7 - y_6y_8)(a_{21}y_{10} + a_{22}a_{42}y_8 + a_{21}a_{61}y_9 + a_{21}a_{82}y_8 + a_{21}a_{81}y_9 - a_{22}y_4y_6) + a_{11}(a_{71}y_8 + a_{72}y_9 - y_6y_7)(a_{22}y_5 + a_{22}a_{41}y_4 + a_{22}a_{71}y_4 + a_{22}a_{42}y_7 + a_{21}a_{82}y_7 - a_{21}y_6y_9) + a_{11}a_{23}a_{32}(a_{113} + a_{112}y_{12} + a_{111}y_3) + a_{11}a_{23}a_{31}(a_{24} + a_{23}y_3 + a_{22}y_4y_8 + a_{21}y_7y_9).$$

Аналогичным образом, задавшись значениями функций $T_2(\boldsymbol{y}) = y_6 = z_6$, $T_3(\boldsymbol{y}) = y_{11} = z_{11}$, $T_4(\boldsymbol{y}) = y_{16} = z_{16}$, с помощью функции *brunovsky*(*X*, *Y*, *T*, *x*, *N*), могут быть получены соотношения для определения остальных переменных модели в форме Бруновского.

На рис. 1 и 2 приведены процессы, полученные с помощью математических моделей (1) и (3). На рис. 1 с помощью переменных x_1 (модель (1)) и z_1 (модель (3)) показано изменение во времени пройденного дизель-поездом расстояния при разгоне состава до 60 км/ч на ровном участке железнодорожного пути. Как следует из рисунка $x_1 \equiv z_1$.

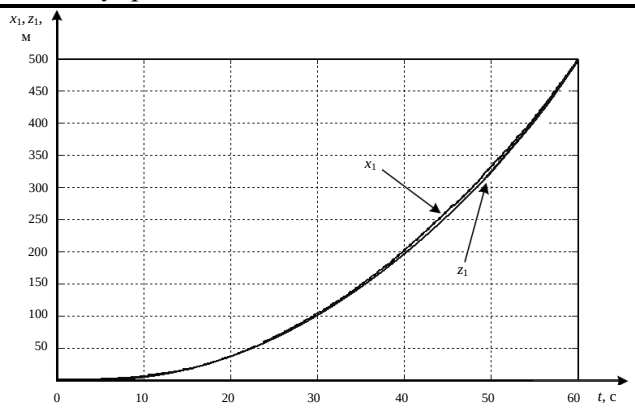


Рис. 1. Поведение переменных x_1 ***и*** z_1

На рис. 2 показаны изменения скорости первого вагона дизель-поезда, полученные с помощью модели (1), переменная x_2, и модели (3), переменная z_2. как видно из рисунка $x_2 \equiv z_2$. Таким образом, линейная математическая модель в форме Бруновского (3) эквивалентна исходной нелинейной модели объекта (1).

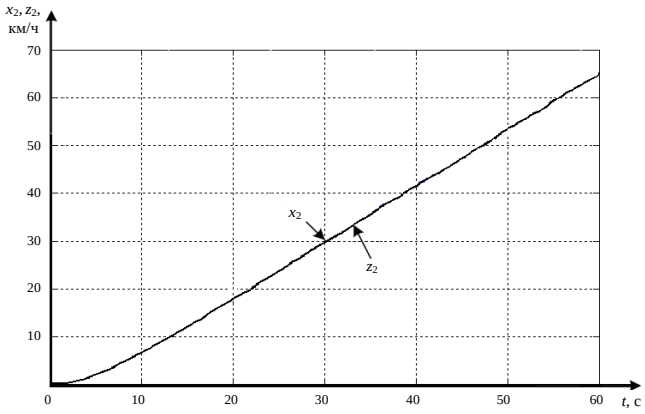


Рис. 2. Поведение переменных x_2 ***и*** z_2

4. Выводы

Таким образом, для универсального пакета моделирования Matlab разработаны функции, позволяющие расширить возможности применения геометрический теории управления. Если в известных работах [9 – 11] геометрическая теория управления применялась для синтеза управлений объектами, которые описываются 3 – 5 дифференциальными уравнениями, то автоматизация аналитических преобразований позволила применять геометрическую теорию управления к объектам существенно более высокого порядка (систем из 16-ти нелинейных дифференциальных уравнений с 4-мя управлениями, рассмотренные в статье). Программа тестировалась и показала свою работоспособность на объектах, описываемых системами из 30-ти нелинейных дифференциальных уравнений и содержащих 8-мь управлений. Таким

образом, за счет автоматизации аналитических преобразований при получении из нелинейных математических моделей объектов управления эквивалентных линейных моделей в форме Бруновского, существенно расширена область применения геометрический теории управления. В частности, с помощью разработанных функций получена линейная математическая модель движения вагонов дизель-поезда в канонической форме Бруновского, которая учитывает параллельную работу двух эквивалентных тяговых асинхронных двигателей. Полученная математическая модель может использоваться для поиска оптимальных управлений, исследования процессов буксования и юза, а также параллельной работы двигателей.

Список литературы

1. Методы классической и современной теории автоматического управления: Учебник в 5-и томах. Т. 5: Методы современной теории управления / Под ред. К.А. Пупкова, Н.Д. Егупова. – М.: Изд-во МГТУ им. Н.Э. Баумана, 2004. – 784 с.
2. Краснощёченко В.И. Нелинейные системы: геометрический метод анализа и синтеза / В.И. Краснощёченко, А.П. Грищенко. – М.: Изд-во МГТУ им. Н.Э. Баумана. – 2005. – 520 с.
3. Ким Д.П. Теория автоматического управления. Т.1. Линейные системы. Изд. 2. / Д.П. Ким. – М.: Физматлит, 2007. – 312 с.
4. Ким Д.П. Теория автоматического управления. Т.2. Многомерные, нелинейные, оптимальные и адаптивные системы. Изд. 2. / Д.П. Ким. – М.: Физматлит, 2007. – 440 с.
5. Kim D.P. Automatic Control. Theory Nonlinear and Multivariable System / D.P. Kim. – Seol: Harnol, 2000. – 558 p.
6. Marino R. Nonlinear Control Design / R. Marino, P. Tomei. – Prentice Hall Europe, 1995. – 396 p.
7. Краснощёченко В.И. Синтез регуляторов для нелинейных систем, приводимых к канонической форме Бруновского / В.И. Краснощёченко // Труды МГТУ им. Н.Э. Баумана. – 1997. – № 569. – С. 28-33.
8. Краснощёченко В.И. Нелинейные системы: геометрические методы анализа и синтеза / В.И. Краснощёченко, А.П. Крищенко. – М.: МГТУ им. Н.Э. Баумана. – 2005. – 520 с.
9. Дмитриенко В.Д. Линеаризация математической модели привода методами дифференциальной геометрии / В.Д. Дмитриенко, А.Ю. Заковоротный // Вісник НТУ "ХПІ". – Харків: НТУ "ХПІ". – 2007. – № 19. – С. 64-77.
10. Дмитриенко В.Д. Моделирование и оптимизация процессов управления движением дизель-поездов / В.Д. Дмитриенко, А.Ю. Заковоротный. – Х.: Изд. центр "НТМТ", 2013. – 248 с.
11. Дмитриенко В.Д. Синтез оптимальных законов управления движением дизель-поезда с помощью математической модели в форме Бруновского / В.Д. Дмитриенко, А.Ю. Заковоротный, Н.В. Мезенцев // Науково-технічний журнал "Інформаційно-керуючі системи на залізничному транспорті". – Харків. – 2010. – № 5-6 (84-85). – С. 7-13.
12. Zakovorotniy A. Involutivity of a vector fields set [Электронный ресурс]. – Режим доступа: http://www.mathworks.com/ matlabcentral/fileexchange/57786-involutivity-s--x-.
13. Zakovorotniy A. Calculation of derivatives Lee of vector fields [Электронный ресурс]. – Режим доступа: http://www.mathworks.com/matlabcentral/fileexchange/57788-dif-li-x--y--variables--n-.
14. Zakovorotniy A. Brunovsky canonical form [Электронный ресурс]. – Режим доступа: http://www.mathworks.com/matlabcentral/fileexchange/57787-brunovsky-x--y--t--x--n-.

УДК 378.1

ПОРЄВ В. М.

АЛГОРИТМІЧНІ АСПЕКТИ АВТОМАТИЗОВАНОГО ПРОЕКТУВАННЯ НАВЧАЛЬНИХ ПЛАНІВ

В статті запропонований підхід до автоматизованого проектування основних документів для планування навчального процесу: освітніх програм, навчальних планів та робочих навчальних планів. Розглянуто методологічні та алгоритмічні аспекти побудови інструментальних комп'ютерних засобів для підтримки планування. Наведені відомості щодо реалізації запропонованих методів та алгоритмів.

This paper proposes an approach to computer-aided design of key documents for the planning of the educational process: educational programs, curriculum and working curricula. The methodological and algorithmic aspects of the development of computer tools for planning support are presented. It gives information about the implementation of the proposed methods and algorithms.

Ключові слова: алгоритм, автоматизація, граф, дисципліна, документ, запити, зв'язки, конфлікт, методика, міждисциплінарні, модель, модуль, навчальний, освітня, план, програма, проектування, процес, цикл

1. Вступ

Проектування та планування навчального процесу є трудомісткою багатокроковою справою і, загалом, включає наступні стадії: розробка освітнього стандарту, освітньої програми, навчального плану, робочих навчальних планів, складання розкладу занять, розрахунки навантаження викладачів тощо. У даній статті увагу зосереджено на фрагменті ланцюжку розробки: освітня програма – навчальний план – робочі навчальні плани. Розробка таких документів має комбінаторно-ітеративний характер зі зворотними зв'язками; потрібно аналізувати багато варіантів та обирати оптимальний. Як здається, у цьому можуть суттєво допомогти комп'ютерні засоби і технології.

2. Аналіз досліджень і публікацій

Розробці та використанню комп'ютерних технологій для підтримки навчального процесу присвячено вже достатньо багато досліджень та публікацій. На Україні створено і відомо декілька діючих автоматизованих систем такого типу, зокрема, АІС "Навчальні плани НТУУ "КПІ" [1], технології фірми ПОЛІТЕК СОФТ [2] та інші [3].

В результаті аналізу літературних та рекламних джерел, присвячених системам підтримки автоматизації навчального процесу, можна зробити висновок, що ці системи будуються на платформах систем керування базами даних; і основна функція, яка реалізується у таких системах – зберігання та накопичення навчальних планів [4]. Особлива роль надається робочим навчальним планам, які є основою для розрахунків навантаження, штатів, складання розкладу занять [5].

У керівних документах та науково-методичних публікаціях [6,7], явно чи опосередковано виписана така послідовність: освітня програма (ОП) → навчальний план (НП) → робочий навчальний план (РНП). Загалом, така методологія "від загального до конкретного" – це достатньо популярний підхід, зручний для ієрархій керування, видання наказів. Проте, при складанні пакету планів для нових спеціальностей та спеціалізацій доцільнішою може бути зворотна послідовність: РНП → НП → ОП. Для цього можна навести такі аргументи:

- тільки після проектування на рівні кредитних модулів стають зрозумілими перелік, зв'язки та обсяги дисциплін. Від розуміння послідовності вивчення окремих модулів, їхніх обсягів, узгодженості з іншими спеціальностями та спеціалізаціями, які забезпечує кафедра та факультет, коректно виписуються міждисциплінарні зв'язки, а далі назви, обсяги та семестри викладання для усіх навчальних дисциплін;
- перехід від РНП до НП та ОП комп'ютерна програма може виконати повністю автоматично, що зменшує ризики багатьох людських помилок.

Циклічне, ітеративне повторення операцій проектування (фактично спіральна модель життєвого циклу розробки) дає можливості гарантовано розробити пакет з ОП, НП, РНП, які відповідають вимогам освітнього стандарту, вимогам сучасності та іншим вимогам.

Суттєвим недоліком багатьох існуючих автоматизованих систем є те, що користувачу надаються можливості лише для введення вже готових РНП та НП. Розробник створює документ-план у якомусь середовищі (найчастіше Microsoft Excel), друкує, погоджує паперову копію, хтось вводить з паперу план у базу, потім паперовий документ затверджується. Наявність такої буцімто автоматизованої системи призводить до того, що усі "паперові" служби залишаються, а ще й додатково залучаються значні обсяги людських та матеріальних ресурсів на забезпечення функціонування комп'ютерної системи. Керівництву закладу та розробникам цієї системи користь є, а розробникам РНП, НП (ОП також) майже жодної користі немає – хіба що з'являється додаткова можливість контролю помилок (але при вводі в базу також можливі помилки). Робити вигляд, що автоматизована система, у яку можна тільки ввести (часто вручну) готові РНП та НП, щось автоматизує – для розробників РНП та НП це є фікцією. Проектування РНП та НП є творчим трудомістким ітеративним процесом, який вимагає високої кваліфікації, і може та повинен бути забезпечений спеціальними інструментами. Наприклад, якщо при проектуванні потрібно перенести дисципліну, або її частину, в інший курс (можливо зі змінами обсягів), то потрібен інструмент, який допоможе швидко, зручно і без помилок перебудувати, як мінімум, два РНП з відповідними змінами НП (а, можливо, й ОП).

3. Постановка задачі

Навчальні плани містять перелік дисциплін, розподілених по різним семестрам. Для створення коректного плану необхідно загалом дотримуватися багатьох вимог та правил. Виділимо такі:

1. Сума обсягів дисциплін для кожного семестру повинна дорівнювати деяким визначеним кількостям годин (кредитів).

2. Повинна забезпечуватися послідовність викладання-вивчення дисциплін (модулів) згідно мети та основних завдань навчання.

Послідовність дисциплін у часі можна спробувати описати орієнтованим графом. Вершинами будуть дисципліни, а стрілками – зв'язки між ними. Стрілка від вершини А до вершини В означатиме, що дисципліна В базується на деяких знаннях дисципліни А і повинна викладатися пізніше дисципліни А.

Проектувати такий граф (структурно-логічну схему), можна двома способами. Перший спосіб: для кожної дисципліни намагатися уявити, які дисципліни потім можна буде вивчати. Другий спосіб: для кожної дисципліни визначати дисципліни, які необхідно вивчити до того. Другий спосіб на думку автора доцільнішій, оскільки набагато простіше шукати відповідь на питання "що необхідно для того, щоб зробити це" замість того щоб фантазувати на тему "де це може стати у нагоді".

Таким чином, для опису графу доцільно використовувати структури даних, які для кожної вершини містять список вершин-попередниць.

Одною з проблем при складанні таких графів є вірогідність циклів. Наприклад, викладач, який розробляє навчальну програму для дисципліни В, вважає, що для вивчення цієї дисципліни потрібні знання, навички, уміння, які є результатами засвоєння деяких інших дисциплін, зокрема, дисципліни А. Інший фахівець, який викладатиме дисципліну А, може вважати, що для вивчення цієї дисципліни необхідно у попередніх семестрах засвоїти декілька дисциплін, зокрема, дисципліну В. Це можна відобразити у вигляді графа зв'язків (рис. 1).

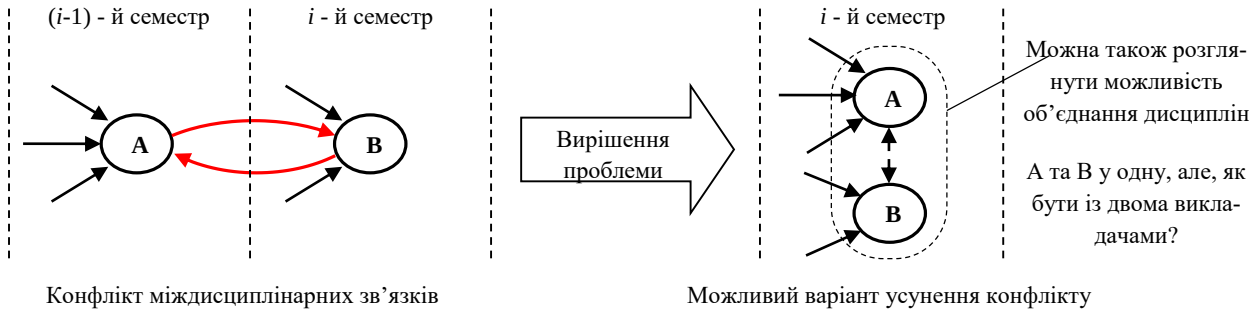


Рис. 1. Приклад простого конфлікту – мінімальний цикл на графі

Складніший варіант конфлікту, який важче помітити людині, яка планує навчальний процес – це замкнений шлях через багато дисциплін, розкиданих по різним семестрам.

Для виявлення та усунення таких конфліктів потрібно уважно аналізувати усі міждисциплінарні зв'язки, які містяться у пропозиціях викладачів і відкидати зайві, оскільки зазвичай викладачі не обмежуються тільки критично необхідними дисциплінами-попередницями, а вказують багато ще інших (трапляється і до десятка) дисциплін, які, на їхню думку, необхідно попередньо засвоїти. Декілька десятків дисциплін (модулів), кожна з яких може бути пов'язана з десятком інших, утворюють громіздке павутиння графа міждисциплінарних зв'язків опису проекту навчального (робочого) плану.

Необхідно зазначити, що наявність замкнених шляхів на графі проекту майбутніх НП-РНП зовсім не означає некомпетентність викладачів відповідних дисциплін. Цілком природно, що згодом дійсно може знадобитися знову вивчати якусь дисципліну з подібною назвою – по спіралі, вже на наступному вищому рівні навчання.

А тепер необхідно вказати на те, що деякі дисципліни можуть бути взагалі не пов'язані з іншими. Тоді – це не граф, а множина елементів. Загалом ця множина уявляється як декілька графів для груп зв'язаних елементів та декілька одиночних елементів. Як коректно розподілити у часі таку множину елементів?

У аналізі множин дисциплін, своєчасному виявленні проблем може допомогти комп'ютерна програма, яка втілює відповідні методи і алгоритми. Потрібно розробити такі алгоритми.

4. Виклад основного матеріалу

Для розподілу елементів множини (дисциплін) по стадіям часу (семестрам) можна запропонувати наступний алгоритм.

4.1. Алгоритм впорядкування

Нехай загальна кількість елементів буде N. Алгоритм складається з двох кроків.

Крок 1. Усі елементи розташовуються на першій стадії.

Крок 2. Виконується основний цикл впорядкування елементів у часі:

```
for (i=0; i<=N; i++)
  {
  res=MoveToNextStages();
  if (res == 0) break;
  }
```

У тілі циклу декілька разів виконується функція

```
int MoveToNextStages()
{
int done = 0;

for (i=0; i<N; i++)
  {
  mxstprev = MaxStagePrev(i);
  if (стадія i-го елементу <= mxstprev)
     {
     MoveToStage(i, mxstprev+1);
     done = 1;
     }
  }
return done;
}
```

Функція **MaxStagePrev(i)** знаходить для *i*-го елемента максимальну стадію елемента-попередника. Процедура **MoveToStage(i,s)** розташовує *i*-й елемент на стадії *s*.

Основний цикл кроку 1 наведеного вище алгоритму виконується максимум N+1 ітерацій. Чому N+1 а не N? Це зроблено для виявлення циклів у зв'язках елементів. Якщо елементів усього N, то їх можна розташувати максимум по N стадіям часу (це буде суцільний послідовний ланцюжок по одному елементу на кожну стадію). Якщо якийсь елемент займе N+1 стадію, то це вже буде помилкою – розрив у часі. Відповідно цей елемент належить до циклу і його зв'язки необхідно проаналізувати для розв'язання конфлікту. При наявності циклів у зв'язках елементи будуть безкінечно пересуватися на наступні стадії. Тому зроблене обмеження кількості ітерацій – остання N+1-а ітерація призначена для контролю можливості впорядкування елементів.

Якщо циклів немає, то цей алгоритм може завершитися суттєво раніше до N-ї ітерації. Це залежить від порядку розташування елементів у робочих масивах.

Складність наведеного вище алгоритму оцінюється як $O(N^2)$. Це можна вважати недоліком, проте для навчальних планів з кількістю дисциплін менше сотні, такий алгоритм у програмі для звичайного офісного комп'ютера відпрацьовує за долі секунди. Позитивною рисою алгоритму є поєднання впорядкування множи-

ни елементів з пошуком циклів у графах груп зв'язаних елементів.

4.2. Традиційна методика розробки пакету документів

Можна сказати, що традиційна методика складання документів є послідовним ланцюжковим процесом розробки-затвердження документів – документ на основі документа.

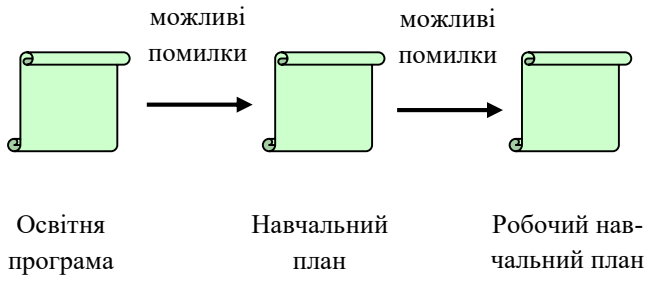


Рис. 3. Традиційне складання документів

Якщо людина бере, наприклад, освітню програму і на її основі складає навчальний план, то може припускатися помилок. На основі документа з помилками, які не помітні, далі може створюватися інший документ, у який можуть додатися нові помилки. Це може призвести до накопичення-розмноження помилок при створенні пакету документів. Досвід показує, що таке трапляється, проте, може бути мінімізовано на рівні виконавців та відповідних процедур контролю.

Набагато гірше, коли проектування пакетів документів пов'язане із частими змінами вихідних даних або(та) вимог. Наприклад, коли вже складені робочі навчальні плани, а потім хтось вимагає змінити назву дисципліни, обсяг, перемістити у інший семестр або курс, або навіть змінити набір варіативних дисциплін у освітній програмі. Це призводить до необхідності змінювати усі документи. З'являється стрес. Різко зростає вірогідність помилок. Розробка пакету документів у визначені терміни стає практично неможливою. Автор статті повною мірою відчув це на собі.

Як запобігти помилок, або хоча б зменшити ризики їхнього виникнення та розмноження? Традиційний шлях – фіксація документа-джерела, заборона внесення змін у нього, і тільки після цього дозвіл створювати підпорядковані документи. Іншими словами – каскадна (водоспадна) модель розробки. Проте, така модель непридатна для розробки чогось суттєво нового – люди не можуть одразу правильно спланувати на майбутнє усе, що треба.

Як створити гнучку технологію, менш чутливу до частих змін вихідних даних та вимог? Вирішити це можна у наступний спосіб.

4.3. Методика автоматизованої розробки пакету документів на основі єдиної моделі

Важливе правило: вихідна інформація повинна бути у єдиному екземплярі і вставлятися у багато документів шляхом посилань, а не копіюванням-повторенням. Тоді зміна назви дисципліни у освітній програмі буде автоматично відпрацьована у навчальному та робочому навчальному планах та в усіх інших підпорядкованих документах. Але зміна назви – це тривіальний випадок. А як бути зі змінами обсягів, термінів викладання дисциплін?

Потрібно створити таку модель навчального процесу, яка дозволить автоматично породжувати усі потрібні документи зі змінами. Іншими словами, усі документи повинні з'являтися шляхом відпрацювання запитів до бази даних. Якщо потрібно, вносяться зміни до моделі (редагування бази), а потім одною командою формуються (друкуються) усі потрібні оновлені документи (рис.4).

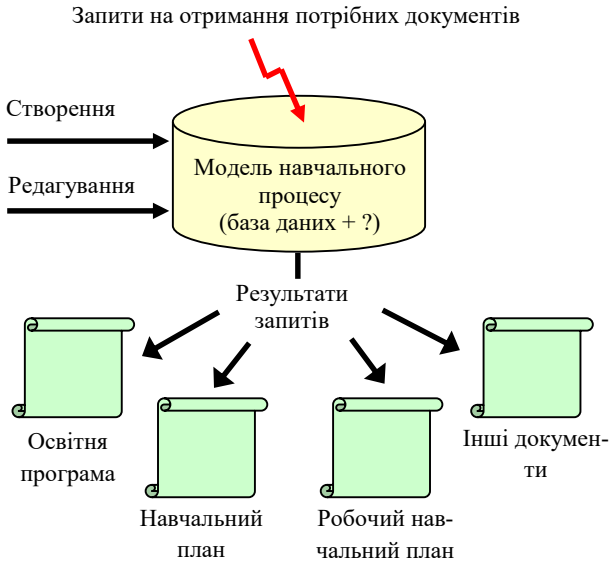


Рис. 4. Документи як результати запитів до бази (моделі)

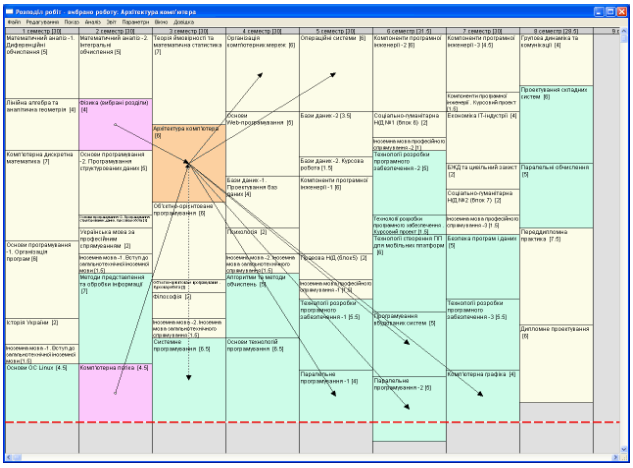
Якою має бути модель навчального процесу? Це може бути звичайна таблиця дисциплін плюс деяка структура даних, яка описує міждисциплінарні зв'язки. Більше того, окрім відповідних структур даних, модель може містити також і

набір процедур – методів обробки внутрішніх даних, які викликаються при редагуванні та запитах. Автор статті реалізував таку модель мовою C++ у відповідному програмному модулі.

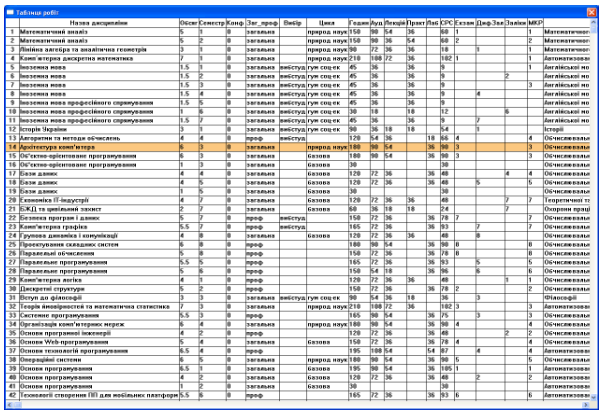
4.4. Реалізація

На рис. 5 проілюстровано, яка інформація є первинною для формування моделі навчального процесу. Для кожної дисципліни у першу чергу зберігається обсяг та зв'язки з іншими дисциплінами, які визначають розташування дисципліни на осі часу. Кожну з дисциплін також можна примусово розмістити у часі (відповідному семестрі), якщо це не призведе до протиріччя у міждисциплінарних зв'язках.

Решта інформації зберігається у таблиці атрибутів дисциплін.



У головному вікні діаграма розподілу дисциплін по семестрам. Виділено одну з дисциплін – показуються її зв'язки з іншими дисциплінами

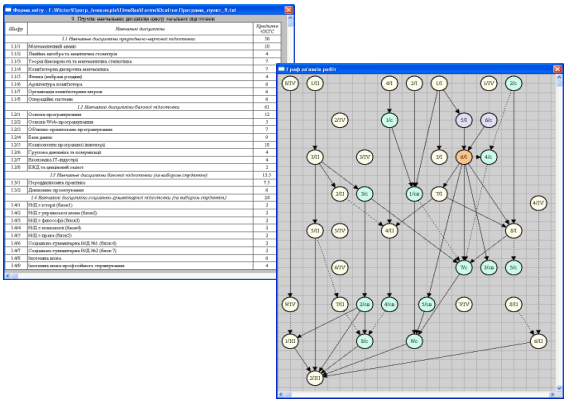


Вікно таблиці атрибутів дисциплін. Це вікно для вводу та редагування текстових та числових атрибутів дисциплін

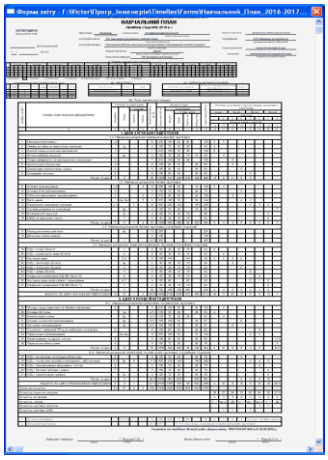
Рис. 5. Ілюстрація елементів моделі навчального процесу бакалаврів

Вторинною є інформація, яка надається у результаті виконання запитів до моделі.

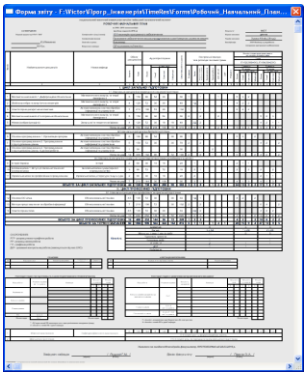
На рис. 6 проілюстровані деякі різновиди результатів запитів – документи-звіти



Елементи освітньої програми: перелік дисциплін та структурно-логічна схема



Навчальний план



Робочий навчальний план

Рис. 6. Документи – результати запитів до моделі навчального процесу

Комп'ютерна програма, яка усе це виконує, міститься у файлі формату EXE розміром декілька сотень кілобайтів. Програма має декілька рівнів. Внутрішній рівень – ядро, яке реалізує структури даних, що зберігають атрибути кредитних модулів дисциплін, міждисциплінарні зв'язки та розподіл у часі. До базових функцій ядра належать функції-примітиви створення моделі навчального процесу, редагування, контролю цілісності даних, відстеження конфліктів

зв'язків кредитних модулів тощо. На зовнішньому рівні реалізуються операції запитів для формування документів-звітів. Для того, щоб забезпечити максимальну гнучкість, щоб можна було б створити будь-який документ потрібного вигляду, запити оформлюються як виконання (інтерпретація ядром програми) скрипту, який є програмним кодом, записаним у текстовому файлі або модульно у декількох файлах. Автором розроблена проста мова скриптів для запитів. Обробку скриптів виконує вбудований парсер та ядро програми.

5. Висновки

Запропонований підхід до автоматизованого проектування освітніх програм, навчальних планів та робочих навчальних планів, який дозволяє побудувати інструментальні комп'ютерні засоби для підтримки планування навчального процесу, для ефективної розробки пакету документів для різноманітних спеціальностей та спеціалізацій. Висвітлені та проаналізовані деякі проблеми, які постають перед розробниками навчальних планів. Надані методичні рекомендації та алгоритми вирішення таких проблем. Одною з рекомендацій є складання моделі навчального процесу, яка враховує міждисциплінарні зв'язки та атрибути навчальних дисциплін. Така модель дозволяє зручно та ефективно проектувати навчальні плани та освітні програми в умовах часткової первісної невизначеності вихідних даних та змінах вимог.

Список посилань

1. АІС "Навчальні плани". Конструкторське бюро інформаційних систем НТУУ "КПІ". [Електронний ресурс]. – Режим доступу: http://kbis.kpi.ua/kbis/index.php?option=com_content&task=view&id=84&Itemid=91
2. ПОЛІТЕК СОФТ. Програмне забезпечення для вищих навчальних закладів України. [Електронний ресурс]. – Режим доступу: http://www.politek-soft.kiev.ua/
3. Медовая Е.В. Автоматизация процесса создания учебного плана в ОНАПТ // Автоматизація технологічних і бізнес-процесів. – 2014, Volume 6, Issue 4, С. 151-154. [Електронний ресурс]. – Режим доступу: http://www.journal-atbp.com/journals/atbp20.pdf
4. В.Г. Гриценко, Г.В. Луценко. UML-моделювання інформаційно-аналітичної системи "Навчальний план" // Інформаційні технології і засоби навчання. 2011. №2 (22). [Електронний ресурс]. – Режим доступу: http://www.journal.iitta.gov.ua
5. Мелкумян Е.Ю., Савицький А.Й., Стасюк Н. Розробка автоматизованої інформаційної системи роботи з робочими навчальними планами ВНЗ. Національний технічний університет України „Київський політехнічний інститут ", КБ інформаційних систем. [Електронний ресурс]. – Режим доступу: http://kbis.kpi.ua/kbis/index.php?option=com_content&task=view&id=29&Itemid=46&limit= =1&limitstart=2
6. Про вищу освіту. Закон України. Відомості Верховної Ради, 2014, № 37-38, ст. 2004.
7. Рашкевич Ю.М. Болонський процес та нова парадигма вищої освіти: монографія / Ю. М. Рашкевич. – Львів: Видавництво Львівської політехніки, 2014. – 168 с.

УДК 004

ТЕЛЕНИК С.Ф.,
РОЛИК А.И.,
ЖАРИКОВ Э.В.

УПРАВЛЕНИЕ РАСПРЕДЕЛЕНИЕМ ВИРТУАЛЬНЫХ МАШИН В ЦОД

Широкое внедрение парадигмы облачных вычислений привело к необходимости повышения эффективности управления ресурсами центров обработки данных. Учитывая частые изменения нагрузки на виртуальные машины и различные условия их функционирования, задачи размещения виртуальных машин и их миграции должны решаться в режиме онлайн. Для решения такого рода задач в статье предлагается подход к размещению новых виртуальных машин и миграции функционирующих. В работе оценена целесообразность использования адаптивного генетического алгоритма при управлении размещением виртуальных машин. Предложен адаптивный программно-определяемый подход к управлению виртуальными машинами центра обработки данных, который основывается на выборе политик управления ресурсами и виртуальными машинами с целью адаптации к влиянию возмущающих воздействий.

A widespread use of the cloud computing paradigm has increased the necessity and significance of improving the management efficiency of cloud data centers. Special attention is paid to solving cloud resource management problems. Due to the intensive changes of virtual machine workloads and different conditions of resource utilization the virtual machine placement and migration problems should be solved and optimized continuously in an online manner. To address such problems the authors present an approach to continuous new virtual machine allocation and virtual machine migration. The authors also evaluate a particular policy of the virtual machine allocation in a data center using an adaptive genetic algorithm. The proposed Adaptive Software Defined approach to the cloud data centers management is implemented by using the policy selector, that allows to select different algorithms or policies for resources and virtual machines management in order to adapt to the impact of disturbing influences.

Ключевые слова: облачные вычисления, программно-определяемая система, ИТ-инфраструктура, управление ресурсами, виртуализация, центр обработки данных.

Введение

Парадигма облачных вычислений кардинально меняет не только взгляд на предоставление ИТ-услуг, но и на понимание самих ИТ-услуг, ведь как ИТ-услуги в настоящее время рассматриваются ПО, платформы, ИТ-ресурсы и пр. Это, в свою очередь, потребовало изменений и принципов организации инфраструктуры предоставления ИТ-услуг. В современных центрах обработки данных (ЦОД), являющихся базой для предоставления облачных вычислений, используются технологии виртуализации, программно-определяемые сети, инновационные подходы в области обработки и хранения информации, а также другие решения, позволяющие предоставлять разнообразные ИТ-услуги пользователям и заказчикам. При этом существенно возрастает роль системы управления ИТ-инфраструктурой (СУИ) ЦОД, которая должна не только поддерживать качество ИТ-услуг на согласованном уровне при изменяющейся динамике запросов многочисленных пользователей, постоянно растущей сложности информационных процессов, непрерывно увеличивающихся объемах обрабатываемой информации, но и эффективно использовать дорогостоящие вычислительные и коммуникационные ресурсы ЦОД.

При решении задачи повышения эффективности управления СУИ должна учитывать следующие особенности ИТ-инфраструктуры ЦОД:

– большое количество ресурсов каждого типа или вида;

– на серверах развертываются виртуальные машины (ВМ), на которых устанавливаются приложения, поддерживающие сервисы;

– ИТ-инфраструктура является более статичной и менее подвержена изменениям, чем приложения и программные средства;

– изменения в ИТ-инфраструктуре инициируются необходимостью изменений на уровне приложений и бизнес-процессов.

– каждый сервер используется одновременно несколькими приложениями;

– когда поверх ИТ-инфраструктуры ЦОД функционируют приложения заказчиков, то возможен конфликт интересов администраторов серверов и администраторов приложений.

Исследования [1, 2] показывают, что современные подходы и методы управления ЦОД должны учитывать использование программно-определяемых технологий, непредсказуемую интенсивность нагрузок и рост потребления ИТ-ресурсов при появлении новых сервисов. Операционные затраты на управление ресурсами в современных ЦОД являются большими и постоянно возрастают [3], а применение парадигмы программно-определяемых устройств на всех уровнях управления позволяет уменьшить расходы на администрирование, расширить возможности интеграции и обеспечить масштабирование СУИ.

При создании СУИ необходимо разрабатывать модели и решать задачи предоставления, распределения, сопоставления, масштабирования и оценки ИТ-ресурсов. Это необходимо делать с учетом того, что провайдеры облачных услуг, с одной стороны, несут ответственность за соответствие текущих значений показателей качества предоставляемых услуг согласованным с заказчиком значениям (например, времени отклика, задержки, вероятности потери запросов и др.), с другой стороны, заинтересованы в предоставлении высококачественных услуг с минимальным использованием полосы пропускания каналов связи, дискового пространства для хранения данных и других задействованных для предоставления услуг ИТ-ресурсов также с минимальным расходом энергоресурсов.

В то же время, поставщики услуг обязаны предоставлять ИТ-услуги с показателями качества, зафиксированными в соответствующем соглашении об уровне услуг (SLA). Для выполнения требований SLA провайдеры услуг часто вынуждены резервировать ресурсы, запускать дополнительные экземпляры приложений и проводить другие мероприятия, предусматривающие избыточное использование ИТ-ресурсов. Оплату за такую избыточность провайдеры перекладывают на своих клиентов, что в свою очередь приводит к повышению стоимости услуг и делает таких провайдеров неконкурентоспособными. В данном случае в выигрыше оказываются те провайдеры, которые задействуют в СУИ механизмы, отслеживающие тенденции увеличения запросов пользователей и динамически изменяющие объемы ресурсов по результатам прогнозирования. Таким образом, планирование ресурсов и политика их распределения непосредственно влияют на стоимость облачных услуг.

Снижение качества ИТ-услуги, произошедшее, например, в результате увеличения количества запросов пользователей, может быть скомпенсировано увеличением объемов ресурсов, выделенных этой услуге. Поэтому при создании СУИ особое внимание уделяется решению задач управления ресурсами ЦОД с учетом частых изменений нагрузки на ВМ и различных условий функционирования ВМ. Причем задачи размещения ВМ и их миграции должны решаться в режиме онлайн.

Традиционные подходы к управлению ресурсами ЦОД и, в частности, к управлению ВМ не учитывают в полной мере динамику происходящих процессов, и обычно вырабатывают управляющие воздействия, когда инциденты и проблемы уже возникли. В работе предложен подход к управлению размещением и миграцией ВМ, который основан на учете динамики процессов, происходящих в ЦОД, при развертывании новых ВМ одновременно с миграцией уже работающих ВМ.

Значительное влияние на эффективность процесса управления ресурсами ЦОД оказывает использование исторических данных и предсказание потребности в ресурсах на краткосрочную или среднесрочную перспективу. В предложенном подходе используется среднесрочное предсказание в потребности ресурсов физического сервера и учитывается интенсивность потребления ресурсов каждой ВМ, что позволяет повысить качество принимаемых решений, связанных с миграцией ВМ и размещением новых ВМ.

Таким образом, целью статьи является разработка подхода к управлению виртуальными машинами с учетом прогноза потребностей в ресурсах.

Анализ публикаций

В работах [4, 5, 6] отмечается, что в настоящее время учеными и технологическими компаниями прилагаются значительные усилия для разработки методов, алгоритмов и систем управления ресурсами ЦОД, предоставляющих облачные услуги. Причем исследования сосредоточены в основном в направлениях предоставления, распределения, планирования емкости, сопоставления и масштабирования ресурсов облачных и виртуализированных ЦОД.

При этом предложено большое количество систем с использованием программно-определяемых технологий и программных платформ для предоставления и мониторинга ресурсов, включая точные методы, эвристики, механизмы предсказания нагрузки, методы учета разбалансированности нагрузки физического сервера, взаимовлияния ВМ, расположенных на одном физическом сервере и пр. Большинство из предложенных подходов для управления ресурсами в облачных и виртуализированных ЦОД основаны на использовании принципа консолидации ресурсов и ВМ, а также на использовании минимального количества физических серверов при обслуживании запросов клиентов. Причем физические серверы, на которых не осталось работающих ВМ, переводятся в режим минимального энергопотребления или отключаются.

Для управления ресурсами и ВМ в режиме онлайн с учетом динамики изменений среды в современных облачных решениях широко применяется механизм миграции ВМ [7]. В [8] предложена система для автоматизации мониторинга, выявления перегруженных элементов ИТ-инфраструктуры, определения нового плана размещения ВМ на физических серверах и выполнения миграции ВМ согласно этому плану. Однако применяемые в этой системе фиксированные пороги для определения степени утилизации ресурсов не позволяют эффективно управлять ресурсами в условиях смешанных нагрузок и нестационарных моделей использования ресурсов.

В работе [9] предложен алгоритм управления миграцией ВМ, минимизирующий число физических серверов, необходимых для обслуживания нагрузок на указанном в SLA уровне. Авторы используют оценочную модель для обычного предсказания необходимости будущего потребления ресурсов, поэтому их алгоритм не учитывает число миграций ВМ.

В работе [10] предложена автоматизированная система управления емкостью и нагрузкой, которая интегрирует несколько контроллеров ресурсов в трех разных областях и временных масштабах.

В [11] предложена система, позволяющая адаптивно подстраивать выделенные для различных ВМ ресурсы в условиях облачных вычислений. Система использует онлайн модель предсказания потребности в ресурсах для определения краткосрочных потребностей в ресурсах. Предложенный подход использует специальную программу внутри каждой ВМ, что недопустимо для многих систем и платформ облачных вычислений.

Анализ проблемы эффективной динамической консолидации ВМ проведен в работе [12]. Рассмотрены проблемы онлайн, офлайн, детерминированной и динамической консолидации ВМ и приведены адаптивные эвристики для реализации динамической консолидации ВМ. Показано, что для решения проблемы обеспечения энергоэффективности и достижения заданной производительности в процессе консолидации ВМ адаптивные эвристики работают лучше, чем оптимальный онлайн-детерминированный алгоритм. Предложенная модель управления включает в себя глобальный и локальный менеджеры. Разработанный алгоритм вырабатывает новый план размещения ВМ по физическим серверам с учетом недогруженных и перегруженных физических серверов. План миграции генерируется алгоритмом в дискретном режиме через заданные интервалы времени.

В исследовании [13] представлена мультиагентная система DRPM (Dynamic Resources Provisioning and Monitoring) которая предназначена для управления облачными ресурсами провайдера с соблюдением требований SLA. Система включает в себя алгоритм для определения ВМ, которая является претендентом на миграцию. При этом задействуются глобальный агент и множество локальных агентов. Модель системы DRPM разработана в среде моделирования CloudSim, которая широко используется для оценки работы алгоритмов и методов управления ресурсами в условиях виртуализации и облачных вычислений. Результаты моделирования показали, что система DRPM позволяет увеличить утилизацию ресурсов провайдера, снизить потребление электроэнергии и избежать нарушений требований SLA. Однако система работает в дискретном режиме, составляя план миграции через фиксированные промежутки времени путем реализации трех фаз: мониторинг, анализ, выполнение. Кроме того, использование локальных агентов внутри ВМ часто недопустимо в условиях облачных вычислений.

В предыдущих исследованиях авторов [14-17] разработаны теоретические и практические основы создания систем управления ИТ-инфраструктурой, включая декомпозиционно-компенсационный подход, двухуровневую модель системы управления ресурсами с координатором, адаптивный генетический алгоритм и

концепцию управления корпоративной ИТ-инфраструктурой. Статья продолжает теоретические исследования в области управления ресурсами ЦОД с акцентом на разработку алгоритмов и политик управления ресурсами и ВМ, обладающих свойством адаптации к возмущающим воздействиям, с учетом динамики развертывания новых ВМ и процессов миграции ВМ, которые происходят одновременно, а также учетом тенденций потребности в ресурсах для достижения энергетической эффективности.

Модель системы управления ЦОД

Провайдер облачных услуг предоставляет различные по содержанию и назначению сервисы для клиентов. Клиенты провайдера могут предоставлять некоторые сервисы для других клиентов путем реализации многопользовательского многозвенного ЦОД, который представлен сложной сетевой и вычислительной ИТ-инфраструктурой. Между провайдерами облачных услуг заключаются соглашения по предоставлению услуг на базе нескольких ЦОД. Для этого вырабатываются соответствующие политики, которые согласовывают предоставление ресурсов по требованию.

Для предоставления ИТ-сервисов провайдеры услуг обычно используют одну или несколько ВМ, которые размещаются на физических серверах. При ухудшении качества услуг вследствие увеличения нагрузки на ВМ используются методы горизонтального или вертикального масштабирования. Для такой схемы предоставления ИТ-услуг предлагается модель системы управления, приведенная на рис. 1.

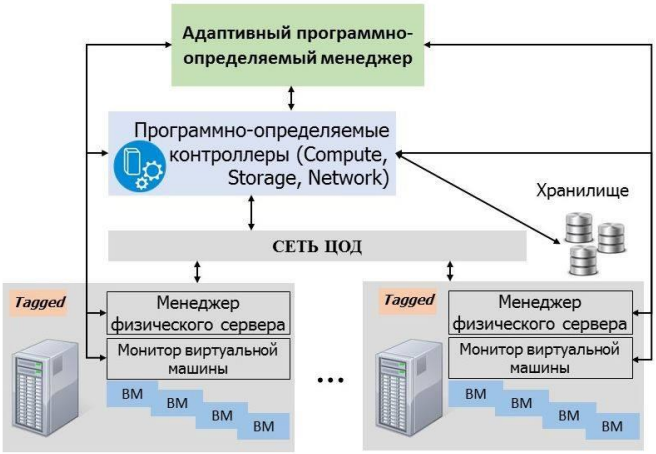


Рис. 1. Модель системы управления ЦОД

Объект управления представлен множеством физических серверов, каждый из которых характеризуется соответствующим аппаратным обеспечением с установленной операционной платформой и имеет фиксированную емкость ресурсов, которая характеризуется производительностью процессоров, объемом ОЗУ и ПЗУ, пропускной способностью сетевого подключения, производительностью работы дисковой подсистемы и др. На каждом сервере размещается одна или несколько ВМ. Серверы соединяются в группы и, в конечном итоге, образуют вычислительную базу ЦОД. Коммуникационная сеть ЦОД обеспечивает взаимодействие серверов и в настоящее время создается с использованием программно-определяемых технологий. Сервера физически группируются в виде стоек и подключаются к коммутатору стойки (ToR Switch).

ЦОД с поддержкой облачных услуг ориентирован на обработку пакетной и транзакционной нагрузки. Проблемы обеспечения гетерогенной нагрузки решаются путем использования моделей IaaS, PaaS and SaaS [18]. Модель IaaS является основой для предоставления базовых сервисов виртуализации и функционирования моделей PaaS и SaaS. Для хранения данных о сервисах и ВМ предлагается использовать централизованное хранилище наряду с распределенным хранилищем [19, 20, 21].

Предоставление дополнительных ресурсов осуществляется исходя из наличия свободных ресурсов. Запросы клиентов на увеличение объемов предоставляемых ресурсов должны удовлетворяться без ущерба для остальных клиентов. При этом владелец ЦОД всегда стремится избежать чрезмерного выделения ресурсов, которые в текущий момент времени не используются.

Существующая практика резервирования ресурсов с запасом для того, чтобы упредить нежелательные последствия, вызванные нехваткой ресурсов при растущих потребностях клиентов и росте рабочей нагрузки, приводит к увеличению эксплуатационных затрат и расходов на электроэнергию. Поэтому в СУИ следует использовать методы и алгоритмы, позволяющие сократить объемы зарезервированных ресурсов без снижения качества предоставляемых ИТ-сервисов и исключения случаев нарушения SLA.

Подход к адаптивному размещению виртуальных машин. Основной задачей при реализации управления облачным ЦОД является эффективное перераспределение ресурсов между пользователями услуг в условиях аварий,

неисправностей, непредсказуемых изменений нагрузки, увеличения численности клиентов и количества их запросов. Перераспределение ресурсов должно происходить так, чтобы не было случаев нарушения SLA при минимизации потребления электроэнергии

Для решения задачи управления ресурсами и ВМ предлагается использовать программно-определяемый подход с комбинацией централизованного и децентрализованного управления. Использование такого подхода к управлению ЦОД позволяет выбирать алгоритмы или политики управления ресурсами и ВМ таким образом, чтобы осуществлялась адаптация к воздействию таких факторов, как изменение нагрузки, установка дополнительных обновлений, развертывание нового программного обеспечения и аппаратных платформ, внесение изменений в структуру и производительность предоставляемых услуг.

Адаптивный программно-определяемый менеджер. В предлагаемой СУИ адаптивный программно-определяемый менеджер (АПОМ) осуществляет централизованное управление облачным ЦОД путем управления физическими и виртуальными ресурсами и выбором политик управления для адаптации к внешним факторам.

В АПОМ реализован адаптивный программно-определяемый подход, который основан на выборе политики управления в зависимости от текущего и предсказанного состояния ВМ, серверов и ЦОД в целом с учетом трендов использования ресурсов. АПОМ может взаимодействовать с аналогичными по функциональности модулями управления других ЦОД, обмениваясь управляющей информацией по глобальной сети или выделенным каналам связи. АПОМ также реализует политику допуска клиентов к сервисам. АПОМ реализуется на отказоустойчивом кластере с высокой доступностью.

АПОМ осуществляет управление ресурсами физических серверов. Для этого каждый сервер соединен с АПОМ с помощью виртуального канала, созданного на основе политик контроллера программно-определяемой сети (SDN). АПОМ периодически получает данные о текущем и предсказанном состоянии ресурсов путем получения сообщений от каждого сервера. Сообщения содержат метку, время отправки, общую емкость ресурсов сервера Res (для процессора, памяти, сетевого подключения и дискового пространства) и предсказанную утилизацию этих ресурсов, вычисленную на стороне сервера. Функции управления ресурсами отдельного сервера выполняет менеджер физического сервера (МФС). Причем Res записывается следующим образом:

$$Res = \{CPU, RAM, NET, STIO\} \quad (1)$$

Для обозначения серверов, которые задействованы в процессах управления, используются метки. Меткой “CONSOLIDATION” отмечаются серверы, которые в текущее время находятся в процессе консолидации. Меткой “PLACEMENT” отмечены серверы, используемые в данное время для размещения ВМ. Меткой “SCALING” отмечены серверы, находящиеся в процессе настройки внутренних ресурсов, предоставляемых локальным ВМ на этом сервере. Меткой “AVAILABLE” помечаются сервера, доступные для размещения и консолидации ВМ.

Подход к размещению новых и мигрирующих ВМ. Для управления виртуальными машинами АПОМ запускает отдельный экземпляр процесса управления для каждого запроса от сервера или для каждого запроса на создание ВМ. Первым выполняются запросы от серверов, на которых наблюдается (или прогнозируется) превышение использования имеющихся ресурсов, когда для ВМ требуются или потребуются дополнительные ресурсы, которые сервер принципиально не сможет выделить. Далее обслуживаются запросы клиентов на создание новых ВМ. После чего обслуживаются запросы серверов, которые, согласно прогнозу, будут недозагружены.

Разработанный алгоритм работы АПОМ размещения новых, а также управления функционирующими ВМ приведен на рис. 2. В результате работы алгоритма формируется множество ВМ G^{VM}, которые необходимо разместить на серверах ЦОД.

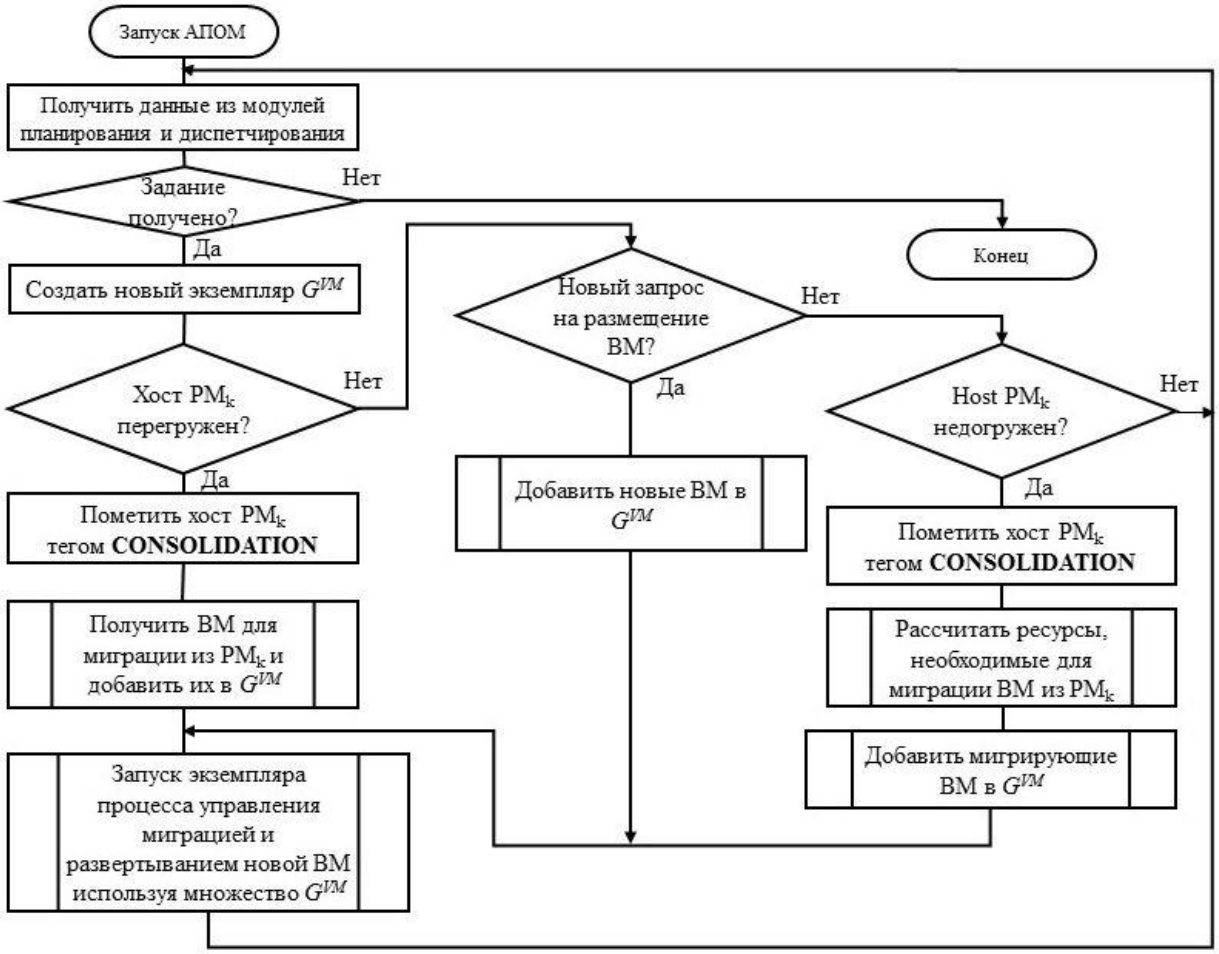


Рис. 2. Алгоритм управления миграцией и размещением ВМ

При обработке полученного списка G^{VM} АПОМ выбирает подходящие серверы из всех работающих в данный момент в ЦОД путем обработки локальных данных о прогнозируемом состоянии серверов. В результате АПОМ получает множество G^{PM} серверов, претендующих на размещение ВМ. Затем АПОМ рассылает сообщения, содержащие требования $CPU_K^{VM}, RAM_K^{VM}, NET_K^{VM}, STIO_K^{VM}$ к новой / мигрирующей ВМ, серверам из множества G^{PM}. Сервер, имеющий, согласно прогнозу, достаточно ресурсов для размещения этой ВМ, помечает себя меткой “PLACEMENT” и отправляет сообщение в АПОМ, в котором указывает прогнозируемый объем доступных ресурсов. После обработки ответов, полученных от серверов, АПОМ создает целевой список серверов GT^{PM} и реализует политику, базирующуюся на адаптивном программно-определяемом подходе. При этом из списка GT^{PM} выбирается сервер, наиболее подходящий для размещения заданной ВМ согласно выбранной в данный момент политики, например: выбор сервера с максимальной загрузкой или выбор сервера с такой загрузкой ресурсов, при которой при размещении новой ВМ будет минимальная разбалансировка ресурсов (отсутствие существенных перекосов в сторону использования одного из ресурсов сервера). Кроме того, на сообщения АПОМ могут отвечать серверы с достаточным прогнозируемым количеством ресурсов и помеченные меткой “AVAILABLE”.

Алгоритм работы экземпляра процесса управления показан на рис. 3.

Algorithm 2. Непрерывная миграция и размещение новых ВМ
Input: G^{VM}, N, P
Output: Решение о миграции или распределении VM на PM

```
1  K ← 0
2  VM ← G_K^VM
3  while VM is not NULL do
4      Создать новый экземпляр G^PM
5      Используя политику P, получить N подходящих PMs
       претендентов на размещение VM и добавить их в G^PM
6      GT^PM ← NULL
7      Res^VM ← G_K^VM {CPU_K^VM, RAM_K^VM, NET_K^VM, STIO_K^VM}
8      for i=1 to N do
9          Отправить требования Res^VM машины VM в PM_i
10         Получить ответ от PM_i
11         if ( PM_i имеет ресурсы для VM ) and ( PM_i помечен
           меткой AVAILABLE) then
12             Пометить PM_i меткой PLACEMENT
13             Получить значения Res^PM от PM_i
14             GT^PM ← PM_i
15         end if
16     end for
17     if GT^PM is NULL then
18         if idle PMs exists then
19             Используя политику P, включить подходящий PM
               и пометить его меткой AVAILABLE
20             Разместить VM на PM
21         else
22             Отправить требования Res^VM машины VM в
               АПОМ другого ЦОД DC
23             Получить ответ от АПОМ другого ЦОД DC
24             if response is negative then
25                 return невозможно распределить VM
26             else
27                 Распределить все ВМ из G^VM в другом ЦОД
                   DC
28                 return
29             end if
30         end if
31     else
32         Используя политику P, получить PM из GT^PM,
           подходящую для распределения VM
33         Распределить VM на PM
34         Пометить все PM в GT^PM, кроме PM, тегом
           AVAILABLE
35     end if
36 K ← K + 1
37 VM ← G_K^VM
38 end while
```

Рис. 3. Алгоритм работы экземпляра процесса управления непрерывной миграцией и размещением новых ВМ

Сервер с загрузкой процессора меньше установленного порога не отвечает на сообщения от АПОМ, так как, согласно прогнозу, работающие на нем ВМ следует переместить на другие серверы для переведения этого сервера в спящий режим или его выключения.

Если ответы на сообщения АПОМ не получены и подходящий для размещения ВМ из множества G^{VM} сервер не найден, то АПОМ принимает решение перевести из спящего режима или включить сервер в стойке, где находится максимальное количество работающих серверов. Затем АПОМ отправляет включенному серверу сообщение о создании новой ВМ или размещения у себя мигрирующей ВМ и помечает сервер меткой “AVAILABLE”.

АПОМ принимает решение, исходя из анализа таких данных: значениях индикаторов и метрик, указанных в SLA; загрузки сетевого интерфейса сервера и коммутатора стойки; чувствительности к перегрузкам, создаваемым процессом миграции; взаимного влияние ВМ друг на друга в пределах одного сервера; минимизации размещения ВМ с пульсирующими нагрузками на одном сервере. При этом АПОМ может учитывать информацию о прогнозе нагрузки на новую ВМ или не учитывать ее. Учет такой информации позволяет уменьшить число лишних миграций ВМ в будущем.

АПОМ анализирует выполнение условия

$$Res_{AVAIL}^{PM} \geq Res_{K}^{VM} \qquad (2)$$

где Res_{AVAIL}^{PM} – доступное количество ресурсов сервера, Res_{K}^{VM} – ресурсы, которые требуются для ВМ из списка G^{VM}.

Если условие (2) для каждого ресурса из множества (1) удовлетворены для некоторого множества серверов, то АПОМ обрабатывает это множество.

Когда экземпляр процесса управления по выбору целевого сервера закончен, АПОМ отправляет сообщение целевому серверу создать новую ВМ и помечает этот сервер меткой “AVAILABLE”. Затем отправляет сообщения серверам из списка GT^{PM} для пометки их как “AVAILABLE”. На стороне сервера локальным МФС проводятся несколько проверок перед тем, как сервер примет предложение о размещении ВМ.

Производится вычисление

$$Res_{AVAIL}^{PM} = Res_{Th}^{PM} - Res_{P}^{PM}, \qquad (3)$$

где Res_{Th}^{PM} – максимальный порог использования ресурса, Res_{P}^{PM} – прогнозируемое использование ресурса.

Выражение (3) для каждого ресурса из множества (1) вычисляется локально менеджером сервера. Если выполняется одно из условий $CPU_{AVAIL}^{PM} < 0$, $RAM_{AVAIL}^{PM} < 0$, $NET_{AVAIL}^{PM} < 0$, $STIO_{AVAIL}^{PM} < 0$, то сервер начинает процесс локальной консолидации.

Причем процесс консолидации ВМ среди серверов с загрузкой выше допустимой запускается с наивысшим приоритетом. Сервер, некоторое количество ВМ которого будут мигрировать, помечается меткой “CONSOLIDATION”. Таким образом, для размещения ВМ из G^{VM}, множество серверов GT^{PM} будет содержать серверы с меткой “AVAILABLE”. При размещении новой ВМ определяется, входит ли она в состав сервиса с масштабированием и балансированием нагрузки или в состав группы сервисов.

АПОМ периодически получает сообщения о прогнозе доступных ресурсов Res от всех серверов в ЦОД путем обмена сообщениями с локальными МФС. Для выбора соответствующей политики управления используется следующая метрика для каждого ресурса из (1)

$$CAP_{Res}^{DC} = \frac{\sum_{i=1}^{M} Res_i^{VM}}{\sum_{j=1}^{N} Res_j^{PM}}, \qquad (4)$$

где M – количество ВМ, работающих в определенный промежуток времени, N – число серверов, Res_i^{VM} – запрошенное количество ресурсов из (1) i-й ВМ, Res_j^{PM} – текущее использование ресурсов (1) j-м сервером.

Выражение (4) определяет текущую емкость ресурсов ЦОД с целью определения тренда использования каждого ресурса.

Менеджер физического сервера. Каждый сервер характеризуется использованием ресурсов процессора, памяти, сетевого подключения, загрузкой интерфейса устройства хранения и др. На каждом сервере выполняется мониторинг виртуальной машины (МВМ), например Xen [22], ES Xi [23] или Hyper-V [24], а также специальная ВМ с МФС для управления виртуальными машинами средствами МВМ. МФС также может быть модулем в МВМ.

МФС накапливает исторические данные об использовании локальных ресурсов сервера. На основе исторических данных МФС вычисляет прогноз использования ресурсов на среднесрочный период, сопоставимый со временем миграции работающей ВМ. Данные о прогнозе использования ресурсов вместе с меткой времени сохраняются локально и отправляются в АПОМ. МФС генерирует команды запуска, останова и паузы для ВМ при управлении МВМ или гипервизором. В результате мониторинга МФС выявляет одну или несколько ВМ, для которых не выполняются требования SLA. На основании политик, полученных от АПОМ, МФС принимает решение о выделении большего объема локальных ресурсов этим ВМ.

Экспериментальные исследования

Рассматривается случай применения политики размещения ВМ в ЦОД с гомогенной конфигурацией серверов. Предлагается решение задачи размещения ВМ на основе адаптивного генетического алгоритма (АГА), предложенного в [16]. Экспериментальные исследования проводились для решения задачи распределения ВМ с помощью трех алгоритмов: управляемого генетического алгоритма (УГА) [25], АГА [16] и классического генетического алгоритма (КГА). Исследования проведены для различного соотношения количества ресурсов сервера и количества ВМ.

Оценка качества работы КГА, УГА и АГА выполнена относительно количества высвобожденных серверов, которые могут быть выключены или переведены в спящий режим. Предполагается, что алгоритмы запускаются администратором, когда возникает необходимость оптимизации размещения ВМ.

На рис. 4 показана зависимость числа высвобожденных серверов от размерности задачи (числа работающих ВМ) для случая, когда запрошенные ресурсы процессоров и памяти для всех ВМ распределены случайным образом в диапазоне [0.05; 0.6].

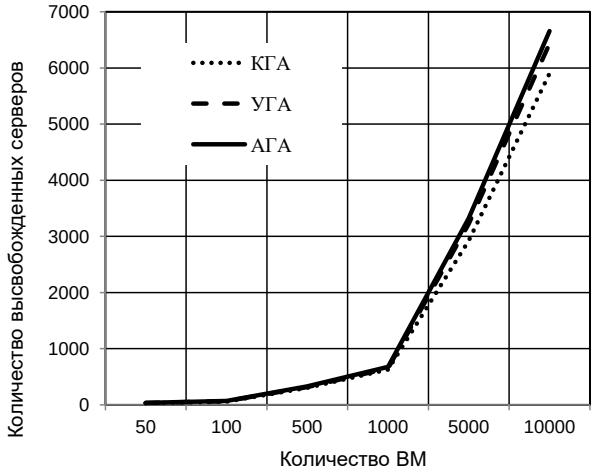


Рис. 4. Зависимость количества высвобожденных серверов от размерности задачи

Анализ экспериментальных исследований в широком диапазоне изменения запрашиваемых ресурсов позволил сделать следующие выводы: 1) использование УГА и АГА является более эффективным, чем использование КГА; 2) АГА всегда позволяет получить лучшие результаты по распределению ВМ на серверах, независимо от условий эксперимента; 3) в случае дисперсии в широком диапазоне требований [0,05; 0,6], использование АГА является предпочтительным.

Выводы

Для решения задачи размещения и миграции ВМ ЦОД предложен адаптивный программно-определяемый подход, основанный на использовании алгоритма распределения новых и миграции работающих ВМ. Адаптивный программно-определяемый менеджер использует комбинации эвристик, при этом учитываются следующие эвристики: индикаторы SLA, загрузка сетевого интерфейса сервера и коммутатора стойки, чувствительность к накладным расходам при миграции, взаимное влияние ВМ друг на друга при работе на одном сервере, а также минимизация размещения ВМ с пульсирующими нагрузками на одном сервере. АПОМ позволяет выбирать различные алгоритмы или политики управления ресурсами и ВМ для адаптации к воздействию внешних факторов, таких как: изменение нагрузки, установка обновлений программ и оборудования, использование новых программных и аппаратных платформ, изменение в структуре и производительности работы поставщика услуг.

Проведены экспериментальные исследования для оценки конкретной политики распределения ВМ с использованием адаптивного генетического алгоритма. Сравнение результатов работы алгоритмов АГА и УГА показало, что использование АГА эффективнее распределяет ВМ с различными начальными требованиями к ресурсам.

Далее авторы продолжают работу по таким направлениям как: разработка системы приоритетов для политик управления ресурсами ЦОД с целью адаптации к влиянию внешних и внутренних факторов; разработка методики и приоритетов применения эвристик при выборе целевого сервера для миграции или развертывания новых ВМ; посредством имитационного моделирования исследование непрерывного распределения новых и мигрирующих ВМ.

Список литературы

1. Buyya R., Calheiros R. N., Son J., Dastjerdi A. V., Yoon Y. Software-defined cloud computing: Architectural elements and open challenges/ Advances in Computing, Communications and Informatics (ICACCI) International Conference. – 2014. – pp. 1–12.
2. Jararweh Y., Al-Ayyoub M., Benkhelifa E., Vouk M., Rindos A. Software defined cloud: Survey, system and evaluation/ Future Generation Computer Systems. – 2015. – vol. 58, May 2016. – pp. 56–74.
3. Hariri S., Khargharia B., Chen H., Yang J., Zhang Y., Parashar M., Liu H. The autonomic computing paradigm/ Cluster Comput. – 2006. – vol. 9 (1). –¬ pp. 5–17.
4. Hameed A., Khoshkbarforoushha A., Ranjan R., Jayaraman P. P., Kolodziej J., Bala-ji P., Zeadally S., Malluhi Q. M., Tziritas N., Vishnu A. A survey and taxonomy on energy efficient resource allocation techniques for cloud computing systems/ Computing. – 2014. – pp. 1–24.
5. Madni S. H. H., Latiff M. S. A., Coulibaly Y. Resource scheduling for infrastructure as a service (IaaS) in cloud computing: Challenges and opportunities/ Journal of Network and Computer Applications. – 2016. – vol. 68. – pp. 173–200.
6. Singh S., Chana I. A Survey on Resource Scheduling in Cloud Computing: Issues and Challenges/ Journal of Grid Computing. – 2016. – pp. 1–48.
7. Clark C., Fraser K., Hand S., Hansen J. G., Jul E., Limpach C., Pratt I., Warfield A. Live migration of virtual machines/ Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation. – 2005. – vol. 2. – pp. 273–286.
8. Wood T., Shenoy P. J., Venkataramani A., Yousif M. S. Black-box and Gray-box Strategies for Virtual Machine Migration/ NSDI. – 2007. – vol. 7. – pp. 17–17.
9. Bobroff N., Kochut A., Beaty K. Dynamic placement of virtual machines for managing SLA violations/ Integrated Network Management, 2007. IM'07. 10th IFIP/IEEE International Symposium. – 2007. – pp. 119–128.
10. Zhu X., Young D., Watson B. J., Wang Z., Rolia J., Singhal S., McKee B., Hyser C., Gmach D., Gardner R. 1000 islands: Integrated capacity and workload management for the next generation data center/ Autonomic Computing (ICAC'08). International Conference. – 2008. – pp. 172–181.
11. Shen Z., Subbiah S., Gu X., Wilkes J. Cloudscale: elastic resource scaling for multi-tenant cloud systems/ Proceedings of the 2nd ACM Symposium on Cloud Computing. – 2011. – p. 5.

12. Beloglazov A. Buyya R. Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual ma-chines in Cloud data centers/ Concurrency and Computation: Practice and Experience. – 2012. – vol. 24, no. 13. – pp. 1397–1420.
13. Al-Ayyoub M., Jararweh Y., Daraghmeh M., Althebyan Q. Multi-agent based dynamic resource provisioning and monitoring for cloud computing systems infrastructure/ Cluster Computing. – 2015. – vol. 18, no. 2. – pp. 919–932.
14. Ролик А.И. Декомпозиционно-компенсационный подход к управлению уровнем услуг в корпоративных ИТ-инфраструктурах / А.И. Ролик // Вісник НТУУ «КПІ». Інформатика, управління та обчислювальна техніка: Зб. наук. пр. – К.: Век+, – 2013. – № 58. – С. 78–88.
15. Ролик А.И. Управление уровнем услуг корпоративной ИТ-инфраструктуры на основе координатора / А.И. Ролик //Вісник НТУУ «КПІ». Інформатика, управління та обчислювальна техніка: Зб. наук. пр. – К.: Век+. – 2013. – № 59. – С. 98–105.
16. Теленик С.Ф. Адаптивный генетический алгоритм для решения класса задач распределения ресурсов ЦОД / С.Ф. Теленик, А.И. Ролик, П.С. Савченко // Вісник НТУУ «КПІ»: Інформатика, управління та обчислювальна техніка. – К.: «ВЕК+», 2011. – № 54. – С. 164–174.
17. Ролик А.И. Концепция управления корпоративной ИТ-инфраструктурой / А.И. Ролик // Вісник НТУУ «КПІ»: Інформатика, управління та обчислювальна техніка. – К.: «ВЕК+», 2012. – № 56. – С. 31–55.
18. Mell P., Grance T. The NIST definition of cloud computing (draft),” NIST special publication. – 2011. – vol. 800, no. 145. – p. 7.
19. Apache.org. HDFS Architecture. http://hadoop.apache.org/core/docs/current/hdfs_design.html
20. Calder B., Wang J., Ogus A., Nilakantan N., Skjolsvold A., McKelvie S., Xu Y., Srivastav S., Wu J., Simitci H. Windows Azure Storage: a highly available cloud storage service with strong consistency. In Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles. ACM. – 2011. – pp. 143–157.
21. Ghemawat S., Gobioff H., & Leung, S.-T.. The Google File System. SOSP '03: Proceedings of the nineteenth ACM Symposium on Operating Systems Principles Bolton Landing, NY: ACM. – 2003. – pp. 29–43.
22. XenSource, Inc. [2008]. “TheXen™ virtual machine monitor”, http://www.cl.cam.ac.uk/research/srg/netos/projects/archive/xen/
23. VMware, Inc. [2016]. “vSphere Hypervisor,” http://www.vmware.com/products/vsphere-hypervisor/
24. Microsoft Corporation. [2016] “Hyper-V overview,” https://technet.microsoft.com/library/hh831531.aspx
25. Теленик С.Ф. Генетичні алгоритми вирішення задач управління ресурсами і навантаженням центрів оброблення даних / С.Ф. Теленик, О.І. Ролік, М.М. Букасов, С.А. Андросов // Автоматика. Автоматизація. Електротехнічні комплексі та системі. – 2010. – №1 (25). – С. 106–120.

УДК 681.3

НОВОТАРСЬКИЙ М. А.,
БРОНФМАН Є. Б.

МОДЕЛЮВАННЯ РУХУ РІДИНИ В ТРИВИМІРНІЙ ОБЛАСТІ З РУХОМИМИ ГРАНИЦЯМИ

В роботі запропоновано використання решітчастого методу Больцмана для моделювання руху рідин у тривимірних областях за умови, що він відбувається під дією перистальтичних процесів. Розглянуто основні принципи побудови решітчастої моделі Больцмана на тривимірній решітці D3Q27, описано спосіб задавання граничних умов на мезоскопічному та макроскопічному рівні. Дано короткий опис принципів побудови програмної моделі та наведено результати моделювання.

In this paper, we proposed using lattice Boltzmann method for modeling fluid flow in a three-dimensional area, if this flow occurs under the influence of peristaltic process. The basic principles of construction of lattice Boltzmann model for three-dimensional lattice D3Q27 were considered. Boundary conditions on mesoscopic and macroscopic levels were described. A brief description of the principles of design software model was done and some results of the simulation were presented.

Ключові слова: решітчастий метод Больцмана, математичне моделювання, перистальтичний процес.

1. Вступ

Відчутний прогрес у сфері високопродуктивних обчислень, який об'єктивно ґрунтується на широкому застосуванні на практиці багатоядерних структур, розподілених обчислень, хмарних обчислень та інших сучасних технологій, відкрив нові можливості у дослідженні складних фізичних процесів, зокрема процесів, які відбуваються у біологічних об'єктах [1]. Складність цих процесів зумовлена, перш за все, високою мінливістю та значною кількістю фізичних параметрів, від яких істотно залежить розвиток фізичного процесу. Зокрема, до процесів, які характеризуються згаданими властивостями, відносяться процеси руху рідин. Загальною характеристикою даних процесів є той факт, що рух рідин в біологічних об'єктах значною мірою залежить від руху обмежуючих рідину поверхонь. Такі процеси називають перистальтичними процесами [2]. Безпосереднє вимірювання параметрів перистальтичних процесів пов'язане з істотними труднощами, викликаними впливом засобів вимірювання на досліджуваний фізичний процес. Тому, на сьогодні, математичне моделювання перистальтичних процесів залишається одним з ефективних методів їх дослідження. Традиційний підхід при побудові математичних моделей перистальтичних процесів ґрунтується на дослідженні нелінійної нестаціонарної крайової задачі, в основі якої лежить система рівнянь, що містить рівняння Нав'є-Стокса та рівняння неперервності потоку. Розв'язування даної крайової задачі, за умови зміни в часі форми області, може бути реалізовано з достатньою точністю тільки з використанням чисельних методів і потребує великої кількості обчислень, оскільки швидка зміна параметрів потребує високого ступеня дискретизації [3].

Згадані проблеми, що виникають при моделюванні перистальтичних процесів, стали причиною пошуку альтернативних підходів, серед яких важливе місце займають нейронні мережі, кліткові автомати та інші засоби опису складних процесів. Серед них одне з чільних місць займає підхід, який ґрунтується на розв'язуванні решітчастого рівняння Больцмана у кожному з вузлів дискретизованої області і має назву решітчастої моделі Больцмана (РМБ).

2. Решітчаста модель Больцмана

Для опису руху рідини в області дослідження на мезоскопічному рівні будемо застосовувати рівняння Больцмана. У загальному вигляді це рівняння є рівнянням з частинними похідними, яке описує еволюцію функції розподілу елементарного об'єму рідини [4].

$$\frac{\partial f\left(x,v,t\right)}{\partial t}+v\nabla f\left(x,v,t\right)=\Omega\left(f\left(x,v,t\right)\right), \quad (1)$$

де $\Omega\left(f\right)$ – оператор колізій, $f\left(x,v,t\right)$ – функція розподілу густини, яка задає ймовірність того, що частинка в момент часу t перебуватиме в кубі з вершиною в точці x і довжиною

сторони dx, а її швидкість буде знаходитися в діапазоні від v до dv.

Оскільки рух елементарного об'єму рідини є наслідком певної послідовності зіткнень з такими ж сусідніми елементарними об'ємами, то цей факт описує оператор колізій, що стоїть в правій частині рівняння (1). З метою зменшення кількості обчислень часто використовують найпростіший оператор колізій, що представлений виразом:

$$\Omega\left(f\left(x,v,t\right)\right)=\frac{1}{\tau}\left(f\left(x,v,t\right)-f^{(eq)}\left(x,v,t\right)\right), \quad (2)$$

де τ – константа, яка визначає масштаб часу, необхідний для встановлення локальної рівноваги, $f^{(eq)}\left(x,v,t\right)$ – рівноважна функція розподілу густини Максвела-Больцмана.

Застосування до моделювання еволюції елементарного об'єму рідини рівнянь (1) та (2) утворює широко відому модель BGK [5]:

$$\frac{\partial f(x,v,t)}{\partial t}+v\nabla f\left(x,v,t\right)= =\frac{1}{\tau}\left(f\left(x,v,t\right)-f^{(eq)}\left(x,v,t\right)\right) \quad (4)$$

Наявною є певна проблема переходу від мезоскопічного рівня, на якому існує деяка множина еволюцій функцій розподілу для кожного елементарного об'єму, до макроскопічних параметрів поведінки рідини в кожній точці досліджуваної області. Такими параметрами найчастіше є розподіл тиску та поле швидкостей в області дослідження. На сьогодні не існує строгого доведення існування залежностей між мезоскопічними та макроскопічними параметрами руху рідини, але численні практичні дослідження показали, що така залежність існує і її точність значною мірою визначається способом дискретизації області. В загальному випадку мезоскопічний та макроскопічний рівні моделювання пов'язані виразами:

$$\rho=\int_{-\infty}^{\infty} f\left(x,v,t\right)d\mathbf{v}=\sum_{k=0}^{8} f_i=\sum_{k=0}^{8} f_k^{(eq)}, \quad (5)$$

$$u=\frac{1}{\rho}\int_{-\infty}^{\infty} v\cdot f\left(x,v,t\right)dv=\frac{1}{\rho}\sum_{k=0}^{8} v_k f_k = =\frac{1}{\rho}\sum_{k=0}^{8} v_k f_k^{(eq)} \quad (6)$$

де u – вектор швидкості потоку рідини, ρ – масова густина потоку рідини.

З рівнянь (5) та (6) видно, що макроскопічні параметри u та ρ формуються на основі попередньо обчислених швидкостей елементарних об'ємів у вузлах решітки та значень функції розподілу у відповідних вузлах. Для одержання згаданих мікроскопічних параметрів виконаємо дискретизацію області з урахуванням законів збереження [6], в результаті чого одержуємо систему, яка складається з Q рівнянь:

$$\frac{\partial f_k\left(x,t\right)}{\partial t}+v_k\nabla f_k\left(x,t\right)=\frac{1}{\tau}\left(f_k\left(x,t\right)-f_k^{(eq)}\left(x,t\right)\right),$$

$$k=0,1,2,...,Q-1, \quad (7)$$

де $Q-$ кількість вузлів дискретизації, $f_k\left(x,t\right)-$ функція розподілу густини, що пов'язана з напрямком вектора мезоскопічної швидкості v_k, $f_k^{(eq)}\left(x,t\right)-$ відповідна вектору v_k рівноважна функція розподілу густини.

Отже, решітчаста модель Больцмана представлена системою рівнянь (7), які, описуючи рух елементарних об'ємів рідини на мезоскопічному рівні, завдяки BGK-переходу дозволяють визначити макроскопічні параметри, які узгоджуються з результатами обчислень крайової задачі на основі рівняння Нав'є-Стокса.

Для обчислення системи рівнянь (7) виконаємо їх дискретизацію для кожного вузла решітки з часовим кроком Δt та просторовим кроком $\Delta x_k=v_k\Delta t$ [9]:

$$\frac{f_k\left(x_k+v_k\Delta t,t+\Delta t\right)-f_k\left(x_k+v_k\Delta t,t\right)}{\Delta t}+ +\frac{f_k\left(x_k+v_k\Delta t,t\right)-f_k\left(x_k,t\right)}{\Delta x_k}= =-\frac{f_k\left(x_k,t\right)-f_k^{(eq)}\left(x_k,t\right)}{\tau}. \quad (8)$$

З метою спрощення обчислень задаємо кроки дискретизації в часі та просторі рівними $\Delta x_k=\Delta t=1$. Тоді решітчасте рівняння Больцмана для $k-$го вузла решітки матиме вигляд:

$$f_k\left(x_k+v_k\Delta t,t+\Delta t\right)-f_k\left(x_k,t\right)= =-\frac{1}{\tau}\left(f_k\left(x_k,t\right)-f_k^{(eq)}\left(x_k,t\right)\right) \quad (9)$$

де x_k – точка в дискретизованому фізичному просторі.

Процес розв'язування рівняння (9) складається з двох кроків.

1. Зіткнення:

$$f_k'\left(x_k, t+\Delta t\right) = \\ = f_k\left(x_k, t\right) - \frac{1}{\tau}\left(f_k\left(x_k, t\right) - f_k^{(eq)}\left(x_k, t\right)\right) \quad (10)$$

2. Розповсюдження:

$$f_k\left(x_k + v_k \Delta t, t+\Delta t\right) = f_k'\left(x_k, t+\Delta t\right). \quad (11)$$

В (10) і (11) функція розподілу $f_k^{\%}$ описує стан елементарного об'єму рідини після чергового зіткнення в точці дискретного простору x_k. Значення елементів множини $\{v_k\}$ визначаються залежно від вимірності моделі та кількості зв'язаних вузлів, які утворюють базовий елемент решітки.

В рамках даної роботи використана тривимірна модель, основою якої є решітка D3Q27 з базовими елементами, що показані на рис. 1.

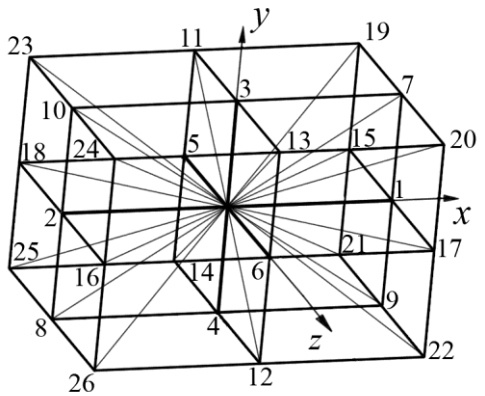


Рис. 1. Базовий елемент тривимірної решітки D3Q27 решітчастої моделі Больцмана

Для даної моделі рівноважна функція розподілу густини визначається з виразу:

$$f_k^{(eq)}(x,t) = \\ = \omega_k \rho\left[1 + \frac{3}{c^2} v_k \cdot u + \frac{9}{2c^4}\left(v_k \cdot u\right)^2 - \frac{3}{2c^2}\left(u \cdot u\right)\right] \quad (12)$$

де v_k – вектор швидкості k–го об'єму рідини, $c = \frac{\Delta x}{\Delta t}$ – константа швидкості розповсюдження активності в решітці, u – вектор макроскопічної швидкості, ω_k – ваговий коефіцієнт у вузлі k, $\Delta x, \Delta t$ – відповідно крок решітки та крок у часі.

Вектори решіткової швидкості масштабуються ваговими коефіцієнтами при визначенні рівноважної функції розподілу густини у вузлі решітки k. Коефіцієнти масштабування ω_k для кожного значення v_k визначаються з виразів:

$$v_k = \begin{cases} (0,0,0),\ k=0 \\ (\pm1,0,0)c,\ (0,\pm1,0)c,\ (0,0,\pm1)c, \\ \text{при } k = 1,2,3,4,5,6, \\ (\pm1,\pm1,0)c,\ (0,\pm1,\pm1)c,\ (\pm1,0,\pm1)c \\ \text{при } k = 7,8,9,\ldots 18, \\ (\pm1,\pm1,\pm1)c,\ \ k = 19,20,21,\ldots 26 \end{cases} \quad (13)$$

$$\omega_k = \begin{cases} 8/27,\ k=0, \\ 2/27,\ k=1,2,3,4,5,6, \\ 1/54,\ k=7,8,9,\ldots,18, \\ 1/216,\ k=19,20,21,\ldots,26. \end{cases} \quad (14)$$

Якщо локальний розподіл тиску в області дослідження виразити через функцію розподілу, враховуючи, що $p_k = c_m^2 f_k$, то рівноважне рівняння розподілу тиску матиме вигляд:

$$p_k = c_m f_k^{(eq)} = \\ = \omega_k\left\{p + p_0\left[3\frac{v_k \cdot u}{c^2} + \frac{9}{2}\frac{\left(v_k \cdot u\right)^2}{c^4} - \frac{3}{2}\frac{u^2}{c^2}\right]\right\}, \quad (15)$$

де $c_m = \frac{c}{\sqrt{3}}$ швидкість розповсюдження звуку у досліджуваному середовищі, p_0 – середній тиск, для якого характерне співвідношення $p = c_s^2 \rho$ за умови, що досліджувана рідина є нестисливою.

Отже, результуючий процес моделювання руху рідини дозволяє визначити локальний розподіл тиску у вузлах решітки за допомогою рівнянь, які визначають динаміку зміни локального тиску для кожного вузла решітки:

1. Зіткнення:

$$p_k'\left(x + v_k \Delta t, t+\Delta t\right) - p_k\left(x,t\right) = \\ = -\frac{1}{\tau}\left[p_k\left(x,t\right) - p_k^{(eq)}\left(x,t\right)\right], \quad (16)$$

2. Розповсюдження:

$$p_k\left(x_k + v_k \Delta t, t+\Delta t\right) = p_k'\left(x_k, t+\Delta t\right) \quad (17)$$

Макроскопічні параметри розподілу тиску та поля швидкостей визначимо з рівнянь:

$$p = \sum_k p_k \ , \ u = \frac{1}{p_0} \sum_k v_k p_k \qquad (17)$$

3. Формування граничних умов

Дослідження параметрів руху рідини під дією перистальтичних процесів проводились в трубковидній області, форма якої показана на рис. 2

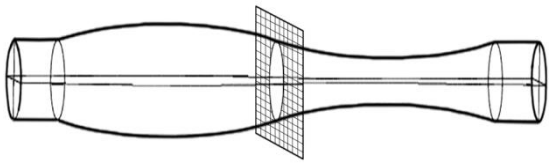


Рис. 2. Триплексна форма

Ця форма має назву триплексної форми, оскільки складається з прямолінійної випуклої та ввігнутої частин трубки. Вона використовується при реконструктивних операціях травного тракту, тому її гідродинамічні характеристики мають велике практичне значення. Розглянемо принцип задавання граничних умов у даній області на прикладі поперечного перерізу, який є однією з площин, сформованих міжвузловими зв'язками на решітці D3Q27.

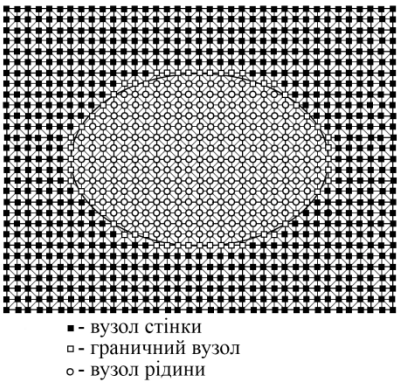


Рис. 3. Переріз з трьома типами вузлів

На рис.3 зображено переріз об'єкта моделювання у місці, яке показане на рис.2. Границя має форму еліпса відповідно до форми перерізу досліджуваного об'єкта. Довільний вузол решітки може належати стінці, границі або рідині. Приналежність вузла до множини граничних вузлів залежить від того, чи знаходиться фізична границя в зоні притягання вузла.

Така зона визначається для кожного вузла і геометрично є кубом з даним вузлом, розміщеним на перетині його діагоналей, і довжиною сторони, яка дорівнює кроку дискретизації решітки (рис.4).

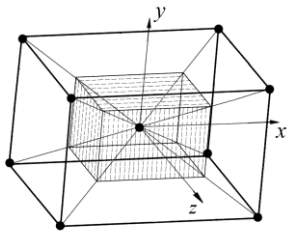


Рис. 4. Зона притягання вузла решітки

Оскільки математична модель передбачає наявність рухомої границі, то статус кожного з вузлів визначається динамічно в залежності від положення фізичної границі. Окремою задачею є перехід від граничних умов, які задані на макроскопічному рівні у термінах розподілу тиску та поля швидкостей на поверхні стінки об'єкта, до граничних умов на рівні вузлів решітки Больцмана.

Оскільки в приграничній зоні частина сусідніх з граничним вузлів не належить рідині, то для таких вузлів необхідно довизначити відповідну функцію розподілу з метою збереження термодинамічної рівноваги [7] та виконання закону розподілу Максвела-Больцмана.

4. Реалізація математичної моделі

При розв'язуванні задачі в описаній вище математичній постановці використано бібліотеку «Palabos» з відкритими кодами, яка орієнтована на моделювання тривимірних об'єктів решітчастим методом Больцмана. Для реалізації математичної моделі з рухомими границями розроблено програмний код, який забезпечує проведення послідовності моделювань з урахування зсуву границь обмежуючого рідину об'єкта.

Одним з важливих етапів постановки задачі в рамках створеної системи моделювання є задавання параметрів тривимірної області, що обмежує рух рідини. Для вирішення цієї задачі використано реалізовану в бібліотеці «Palabos» функцію імпортування файлів, які задають форму границі обмежуючої поверхні в форматі STL. Для створення такої поверхні застосовано пакет тривимірної комп'ютерної графіки "Blender 2.78". Відповідний STL-файл містить координати вузлів стінок поверхні з граничними умовами без ковзання зі зворотним відскоком (bounce-back).

Перший етап обробки вхідних даних полягає у перетворенні структури даних з STL-файлу в більш ефективну внутрішню структуру даних бібліотеки «Palabos». Така структура даних TriangleBoundary3D на наступному кроці у подальшому використана для задавання граничних умов.

Задача моделювання руху рідини в трубковидній області відноситься до класу задач з внутрішнім потоком. Тому вузли, які розміщуються всередині області обчислень, належать рідині, і їх динаміка має відрізнятися від динаміки вузлів, які знаходяться за обмежуючою поверхнею. Процес задавання динаміки вузлів та забезпечення відповідних зав'язків між локальними елементами області називають процесом вокселізації, в результаті якого формується структура даних бібліотеки «Palabos» MultiBlockLattice3D, яка підтримує зв'язки між елементарними об'ємами у тривимірному просторі відповідно до заданого дескриптора решітки D3Q27.

Процес моделювання виконується як послідовність колізій та розповсюджень, які ітеративно повторюються до досяжності збіжності відповідно до заданого критерію оцінки.

Модифікація бібліотеки «Palabos» полягає у розробці механізму перетворення структури MultiBlockLattice3D з результатами моделювання у структуру зі зміненою геометрією граничних вузлів та довизначенні нових вузлів, виходячи з умови підтримки термодинамічної рівноваги.

Для підвищення точності моделювання підтримується також механізм динамічної зміни дискретизації області досліджень з автоматичним перевизначенням граничних вузлів.

5. Результати моделювання

В рамках роботи побудована математична модель руху рідини в триплексній трубковидній формі під дією перистальтичних коливань її стінок. Макроскопічні коливання довільної точки поверхні просторової форми описуються модулем швидкості, вектор якої направлений по нормалі до осі просторової форми x і визначається функцією:

$$U_n(t,x,\delta,\varphi,\lambda,\gamma)=U_{\max}\cdot U_t(t,\lambda)\cdot U_x(x,\delta,\varphi,\lambda),$$

Зміну модуля швидкості у часі описує вираз $U_t(t,\lambda)=\sin^2\left(\frac{\pi t}{\lambda}\right)$, а просторова складова проходить три стадії, які визначаються умовами у виразі, в залежності від кроку δ:

$$U_x(x,\delta,\varphi,\gamma)=\begin{cases}\cos(\pi x/\varphi) \text{ при } \delta=0,\\ -\cos(\pi x/\varphi-\delta\gamma)+\\ +\cos(\pi x/\varphi-(\delta+1)\gamma)\\ \text{при } 0<\delta<\delta_{\max},\\ -\cos(\pi x/\varphi-\delta\gamma) \text{ при } \delta=\delta_{\max}\end{cases}$$

При $\delta=0$ відбувається зростання деформації поверхні. На проміжних кроках відповідна формула описує зменшення деформації попереднього кроку та зростання деформації наступного. Останній крок $\delta_{\max}$ характеризується поверненням деформованої поверхні до початкового стану.

Загальну оцінку продуктивності перистальтичного процесу будемо визначати через об'ємну швидкість потоку V [м^3/с], яка є інтегралом від границь перерізу на кінцях трубчатої форми.

Модель дозволяє відображати поточний стан границі та області, що дає можливість візуальної оцінки результатів моделювання. На рис.5 показано приклад відображення поля швидкостей руху границі при перистальтичному коливанні.

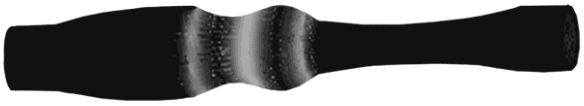


Рис. 5. Візуальне відображення руху границі при перистальтичному коливанні

В результаті прогонки моделі одержано залежності об'ємних швидкостей через початковий та кінцевий перерізи та об'єму потоку від часу, які показані на рис.6 та рис.7.

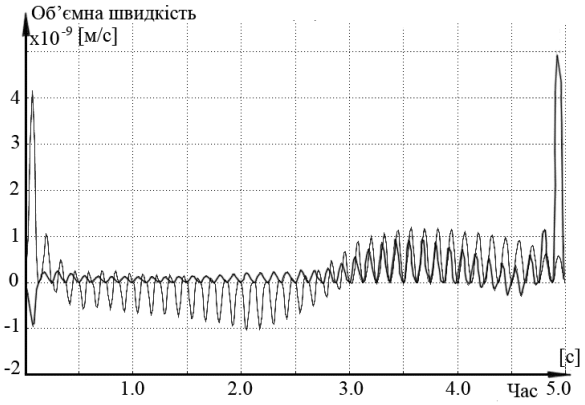


Рис .6. Об'ємні швидкості рідини через початковий та кінцевий перерізи триплексної форми

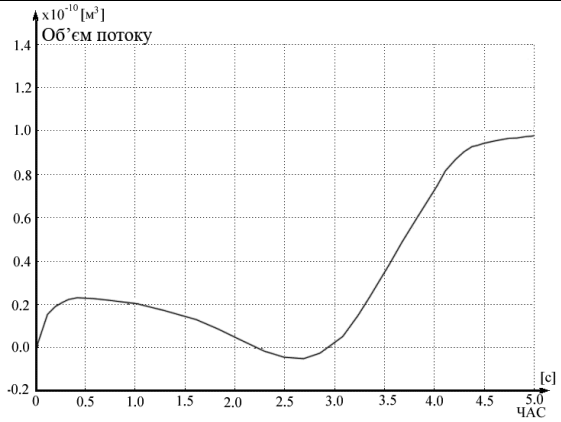


Рис.7. Об'єм потоку

5. Висновки

Доступність багатоядерних та розподілених обчислювальних структур привела до бурхливого розвитку нових обчислювальних методів, які характеризуються природною паралельністю. Серед таких методів у досліджені складних гідродинамічних процесів чільне місце займають решітчасті моделі Больцмана. В роботі застосовано даний метод для дослідження тривимірної нестаціонарної моделі руху рідини у триплексній трубковидній формі під дією перистальтичних коливань. Для побудови програмної реалізації моделі було модифіковано бібліотеку з відкритими кодами «Palabos» для одержання можливості динамічної зміни форми обмежуючої поверхні в ході моделювання.

В результаті моделювання одержано дані про поле швидкостей руху рідини та розподіл тиску в області моделювання. Інтегральні параметри моделі представлені графіками залежностей для об'ємних швидкостей через вхідний та вихідний переріз, а також залежністю об'єму потоку від часу.

Список посилань

1. Новотарський М.А., Нестеренко Б.Б. Штучні нейронні мережі: обчислення.– Праці Інституту математики НАН України.– Т51.– К.: Ін-т математики НАН України, 2004.– 408 с.
2. Новотарский М.А., Нестеренко Б.Б. Математическое моделирование перистальтических процессов на клеточных нейронных сетях // Искусственный интеллект.– 2012.– № 4.– С.578-587.
3. Novotarskiy M.A. Solving of boundary value problems for mathematical physics equations in cellular neural networks // Radio Electronics, Computer Science, Control.– 2003.– №1.– P.42-47.
4. Mohamad A.A. Lattice Boltzmann method / A.A. Mohamad.– London: Springer-Verlag.– 2011.– 178 p.
5. Bhatnagar P.L. A model for collision processes in gases. I: Small amplitude processes in charged and neutral one-component system / P.L. Bhatnagar, E.P. Gross, M. Krook // Physical Review.– 1954. – Vol.94, №3 – P.511–525.
6. He X., Luo L-S. Theory of the lattice Boltzmann equation: from Boltzmann equation to lattice Boltzmann equation / X. He, L-S. Luo // Physical Review E.– 1997.– Vol. 56, №6.– P.6811–6817.
7. Inamuro T. A non-slip boundary condition for lattice Boltzmann simulations // Physics of Fluids.– 1995.– Vol. 7, №12.– P.2928–293.

УДК 004.7

ДОЛГОЛЕНКО А.О.,
ЯЦУН В.О.

РЕАЛІЗАЦІЯ ОПЕРАЦІЙНОГО ПРИСТРОЮ СУМАТОРА/ВІДНІМАЧА З ПЛАВАЮЧОЮ КРАПКОЮ ДЛЯ ЯДРА СУПЕРСКАЛЯРНОГО ПРОЦЕСОРА

В цій статті описаний операційний пристрій суматора/віднімача з плаваючою комою для ядра суперскалярного мікропроцесора та спеціалізованих апаратних засобів на ПЛІС. Суматор розроблено як набір комбінаційних схем, без використання елементів пам'яті й мікропрограмного керування. Для реконфігурації суматора на обробку операндів потрібного формату не потребується ніяких додаткових дій, крім подачі на його керуючі входи сигналу оброблюваного формату.

This article describes addition/subtraction device operating on floating point numbers designed for cores of superscalar microprocessors and specialized hardware built using FPGA. The adder is designed as a set of combinational circuits without memory elements and without microprogram management. No additional actions are required for the adder to be reconfigured for processing operands of a required format but setting the required format into its control inputs.

Ключові слова: суматор, суперскалярний мікропроцесор, CISC, RISC.

Вступ

Починаючи з мікропроцесорів *Intel P6* (1995 р.) та *AmD K5* (1996 р.), для побудови мікропроцесорів з архітектурою *x86-64* використовується технологія *OoOE (Out-of-Order Execution)*, що заснована на реалізації обмеженої архітектури потоку даних [1]. Такі мікропроцесори отримали назву суперскалярних мікропроцесорів [2], або мікропроцесорів з архітектурою *CISC-RISC («CISC-outside RISC-inside»),* де: *CISC - Complex Instruction Set Computing* (набір команд в архітектурі *x86-64*), *RISC - Reduced instruction set computing* (скорочений набір команд, що реалізується множиною операційних пристроїв мікропроцесора.) В процесі роботи таких мікропроцесорів *CISC* – команди, що виконуються відповідно до лічильника команд, декодуються на множину *RISC*– команд [3-5] (їх також називають: *uop, micro-ops*, або подібними термінами).

Планування виконання *RISC* – операцій здійснюється, відповідно до архітектури потоку даних [6,1], на підставі готовності до виконання операндів виконуваних *RISC* – операцій. До набування *RISC* – операціями стану готовності до виконання, вони розміщуються в комірках найбільш швидкодіючої пам'яті мікропроцесора, що носять назву в різних джерелах: *Out-of-Order Window,* комірки універсального планувальника, або комірки станції резервування. Кількість цих комірок постійно збільшується, наприклад, у *Intel:* 168 в *Sandy Bridge* (2011 р.), 192 в *Haswell* (2013 р.), 224 в *SkyLake* (2015 р.). *RISC* – операція, котра набула стану готовності, може бути переданою з комірки станції резервування в вільний операційний пристрій, що може її виконати. Таким чином в сучасних мікропроцесорах організовується динамічний паралелізм на рівні *RISC* – операцій.

Метою роботи є розробка швидкодіючого операційного пристрою суматора/віднімача з плаваючою крапкою, який не потребує в своїй роботі ніяких додаткових мікропрограмних засобів і може використовуватися для динамічного розгалуження на рівні *RISC* – операцій як в роботі ядра сучасного суперскалярного мікропроцесора так і, наприклад, при побудові спеціалізованих апаратних засобів на ПЛІС.

Стандарт *IEEE Std 754*™ для арифметики з плаваючою крапкою

Представлення чисел з плаваючою крапкою схоже на широко використовувану форму представлення чисел в наукових обчисленнях і складається з двох частин: значущої частини числа (або мантиси) f і показника ступеня (експоненти, або порядку) e. Число з плаваючою крапкою x представляється у вигляді $\pm, e_x, f_x$ і має значення:

$x = \pm f_x r^{ex}$, де r – основа системи числення (база експоненти, або основа ступені). Стандарт для арифметики з плаваючою крапкою [7] припускає використання $r = 2$ і $r = 10$.

Деякі скорочення, що прийняті в цьому стандарті:

LSB – найменш значущий біт;

MSB – найбільш значущий біт;

NaN – не число.

Формат представлення чисел з плаваючою крапкою показаний на рис. 1.

±	*e+bias*	*f*

Рис. 1 – Формат представлення чисел з плаваючою крапкою

Абсолютне значення мантиси нормалізованого числа знаходиться в діапазоні [1, 2). При нормалізації вона зсувається, поки в $|f|$ *MSB* не буде дорівнювати 1. При кожному лівому зсуві виконується операція $e = e - 1$, при кожному правом зсуві $e = e + 1$. Таким чином, двозначна крапка в уявленні f завжди розташована після біта *MSB*.

Біт *MSB* мантиси нормалізованого числа $|f|$, який завжди дорівнює 1, з уявлення числа видаляється і в пам'яті зберігається тільки дрібна частина f. В арифметичному пристрої цей прихований біт відновлюється і тим самим сприяє підвищенню точності представлення операндів, не займаючи місця в пам'яті.

Число з плаваючою крапкою має два знаки: знак числа (*sign*), відображається окремим бітом; знак порядку, відображається зміщенням порядку (*bias*).

Для обчислень з плаваючою крапкою стандарт [7] передбачає використання типів даних (форматів), зведених у табл.1.

Так як число 0 не може бути представлено у формі з нормалізованою мантисою, для його представлення існує спеціальний код: $sign = 0 \vee 1, e + bias = e_{min}, f = 0$.

Спеціальний код також існує для представлення *NaN*: $sign = 0 \vee 1, e + bias = e_{max}, f \neq 0$.

Для представлення невизначених результатів, таких як 0/0, використовується значення ±∞: $sign = 0 \vee 1, e + bias = e_{max}, f = 0$.

Коли одне з цих спеціальних значень бере участь як операнд в арифметичній операції, результат операції формується відповідно до наступних правил:

число/(±∞) = ±0;

число±0 =число;

±0-число=-число;

±0(x∨/)число=±0; (1)

(число∨±∞∨±0)/±0 = ±∞;

±∞(+∨-∨x∨/)число = ±∞;

NaN(+∨-∨x∨/)(число∨±∞∨±0) = NaN,

не чекаючи завершення операції.

Денормалізоване число це число з плаваючою крапкою у якого експонента дорівнює нулю (e_{min}) і ненульова мантиса: $sign = 0 \vee 1, e+bias = e_{min}, f \neq 0$. При виконанні арифметичної операції з денормалізованим операндом відновлення прихованого біта в ньому не проводиться. Використання денормалізованних операндів дозволяє зробити ефект втрати значимості менш різким. Без використання таких операндів деякі малі значення, які не представимі як нормалізовані числа, округлюються до 0. Таке рішення погіршує, в деяких випадках, точність обчислень з плаваючою крапкою. Наприклад, число $(0.1)_2 2^{126}$ не має нормалізованого подання в одинарному форматі. Якщо ми замість цього числа в подальших розрахунках будемо використовувати 0, то матиме місце «*поступова втрата точності результату*». Однак, реалізація в обчислювальному пристрої можливості обробки денормалізованих чисел може викликати небажані апаратурні, часові і вартісні витрати. У зв'язку з цим, стандарт [7] не вимагає обов'язкової реалізації обробки денормалізованих чисел. Рис. 2 ілюструє область денормалізованих чисел в одиночному форматі стандарту. Вони всі знаходяться у відкритому інтервалі (0, $|min|$).

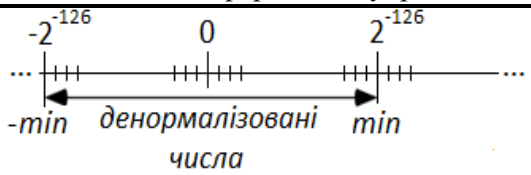


Рис. 2 – Область денормалізованих чисел в одиночному форматі стандарту IEEE

Табл. 1 – Типи даних (точність) для представлення чисел з плаваючою крапкою

Параметр	Половин-на (*SF*)	Одинарна (*F*)	Подвійна (*DF*)	Подвійна розширена (*DEF*)	Учетверена (*QF*)
Всього біт (*n* біт)	16	32	64	80	128
Знак числа (*sign*)	1	1	1	1	1
Порядок (кільк. бітів)	5	8	11	15	15
Зсув порядку (*bias*)	15	127	1023	16383	16383
Значення e_{max}	31	255	2047	32767	32767
Значення e_{min}	0	0	0	0	0
Мантиса (*k* біт)	10+1	23+1	52+1	64+1	112+1
Діапазон уявлення мантиси	$[1,2 - 2^{-10}]$	$[1,2 - 2^{-23}]$	$[1,2 - 2^{-52}]$	$[1,2 - 2^{-64}]$	$[1,2 - 2^{-112}]$
Нуль (±*0*)	$\pm, e_{min}, f=0$	$\pm, e_{min}, f=0$	$\pm, e_{min}, f=0$	$\pm, e_{min}, f=0$	$\pm, e_{min}, f=0$
Денормалізоване число	$\pm, e_{min}, f\neq 0$	$\pm, e_{min}, f\neq 0$	$\pm, e_{min}, f\neq 0$	$\pm, e_{min}, f\neq 0$	$\pm, e_{min}, f\neq 0$
Нескінченність (±∞)	$\pm, 31, f=0$	$\pm, 255, f=0$	$\pm, 2047, f=0$	$\pm, 32767, f=0$	$\pm, 32767, f=0$
Не число (*NaN*)	$\pm, 31, f\neq 0$	$\pm, 255, f\neq 0$	$\pm, 2047, f\neq 0$	$\pm, 32767, f\neq 0$	$\pm, 32767, f\neq 0$
Число	$e \in [-15, 16]$ $(\pm 1.f\ x2^e)$	$e \in [-127, 128]$ $(\pm 1.f\ x2^e)$	$e \in [-1023,1024]$ $(\pm 1.f\ x2^e)$	$e \in [-16383,16384]$ $(\pm 1.f\ x2^e)$	$e \in [-16383,16384]$ $(\pm 1.f\ x2^e)$
Еквівалентне число десяткових цифр	≈ 3.3	≈ 7.2	≈ 15.9	≈ 19.2	≈ 34.0

Стандарт [7] також визначає виконання чотирьох базових арифметичних операцій (додавання, віднімання, множення, ділення), а також кореня квадратного, що стосується очікуваної точності в їх результатах. В принципі результати обчислень повинні відповідати результатам, які були б отримані, при виконанні всіх проміжних розрахунків з нескінченною точністю.

Алгоритм виконання операцій додавання/віднімання з плаваючою крапкою

Виконання арифметичних операцій додавання/віднімання над операндами з плаваючою крапкою концептуально просте. Однак необхідно проявляти обережність у реалізації обладнання для забезпечення коректності та уникнення невиправданої втрати точності. Крім того має бути реалізована можливість обробки будь-яких винятків.

Розглянемо виконання додавання $c = a+b$, тому що віднімання зводиться до додавання додаткового коду зображення мантиси f_b від'ємника: $(\pm f_a\, 2^{ea}) + (\pm f_b\, 2^{eb}) = \pm f_c\ \ 2^{ec}$.

Ця операція вимагає, щоб порядки обох операндів були вирівняні до початку складання їх мантис. Тому спочатку:

begin {add}

if $e_a = e_b$ *then begin* $e_c = e_b$*; go to M1 end;*

if $e_a > e_b$ *then begin* $e_c = e_a$*;* $f_b = f_b 2^{-(ea-eb)}$ {арифметичне зрушення f_b на e_a-e_b розрядів вправо} *end else begin* $e_c = e_b$*;* $f_a = f_a 2^{-(eb-ea)}$ {арифметичне зрушення f_a на e_b-e_a розрядів вправо} *end;*

M1: $f_c = f_a + f_b$ {Обчислення f_c буде виконуватися в додатковому модифікованому коді (з двома знаковими бітами). Суматор матиме $2(k+1)$ розрядів, де k – розрядність мантис операндів}.

Тепер потрібно виконати нормалізацію f_c, забезпечивши її знаходження в діапазоні: $1 \leq |f_c| < 2$

і виконати корекцію e_c, якщо вона необхідна. Після нормалізації f_c, потрібно виконати округлення f_c і ще раз перевірити її нормалізацію. Для цього:

m2 = 0; M2: if $|f_c| \geq 2$ *then begin* $f_c = f_c 2^{-1}$; $e_c = e_c + 1$; *if m2 = 0 then* {округлення f_c} *begin* $f_c = f_c + [(not\ sign(c))2^{-(k+1)}]_{k+2}$ {після округлення f_c скорочується до $k+2$ розрядів}; *m2=1; go to M2 end; go to M3 end;*

m4 = 2k; M4: if $|f_c| < 1$ *then begin* $f_c = f_c 2$; $e_c = e_c - 1$; *m4 = m4-1; if m4>0 then go to M4 else go to M5 end;*

{округлення f_c} *if m2 = 0 then begin* $f_c = f_c + [(not\ sign(c))2^{-(k+1)}]_{k+2}$ {після округлення f_c скорочується до $k+2$ розрядів}; *go to M5 end;*

M3: if $e_c \geq e_{max}$ {переповнення формату результату} *then begin* $e_c = e_{max}$; $f_c = 0$ *go to M6 end;*

M5: if $e_c \leq e_{min}$ {втрата значущості, або нуль} *then begin* $e_c = e_{min}$; $f_c = 0$; *go to M6 end else if* $|f_c| \geq 2$ *then begin* $f_c = f_c 2^{-1}$; $e_c = e_c + 1$; *go to M3 end;*

M6: end {add}

Структурна схема суматора/віднімача з плаваючою крапкою, що працює за цим алгоритмом, приведена на рис. 3. Операнди спочатку розпаковуються. Розпакування включає в себе виділення: знака, порядка і мантиси, а також відновлення MSB для кожного з операндів. Під час розпакування також здійснюється конвертація форматів операндів до внутрішнього формату арифметичного пристрою (до формату *QF*). Розпаковані операнди тестуються на наявність серед них винятків: 0, *NaN*, ±∞. При наявності винятків, результат операції формується відповідно до (1) і відправляється на упаковку, минаючи додавання/віднімання.

Блок вирівнювання мантис операндів здійснює правий арифметичний зсув мантиси операнду з меншим порядком. Величина зсуву визначається різницею порядків операндів. Немає необхідності зрушувати мантису з меншим порядком більше ніж на *k* розрядів.

Таким чином, як тільки різниця порядків перевищує *k*, можна вважати, що результат виконуваної операції не зміниться, якщо число з меншим порядком замінити на ±*0*, відповідно. У зв'язку з цим блок управління може відразу ж ініціювати формування результату, відповідно до (1), в блоці упаковки результату.

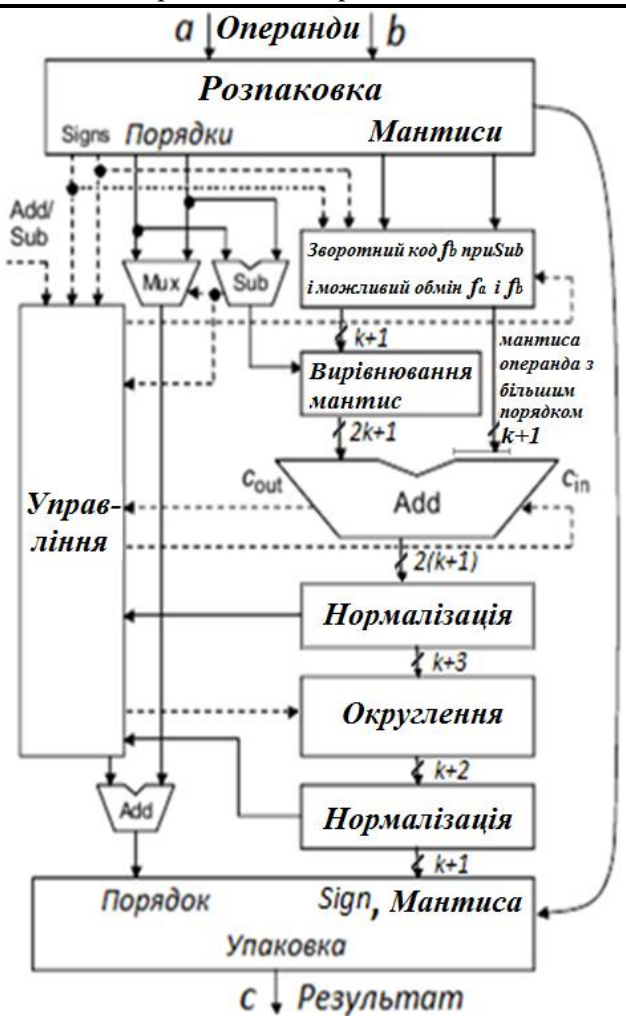


Рис. 3 – Суматор/віднімач з плаваючою крапкою

Різниця двох порядків використовується для визначення зрушуваної мантиси і кількості розрядів арифметичного зсуву вправо для здійснення вирівнювання порядків операндів.

В блоці вирівнювання мантис реалізується можливість зсуву тільки однієї мантиси. Якщо зрушенню підлягає мантиса іншого операнду, то блок можливого обміну забезпечує заміну місць мантис.

Обчислена в суматорі *2(k+1)* – розрядна мантиса результату f_c нормалізується в першому блоці нормалізації. При цьому f_c може зсуватися як ліворуч, так і праворуч відповідно до вищеприведеного алгоритму. Нормалізована *(k+3)* – розрядна мантиса f_c округлюється до *k+2* розрядів у блоці округлення. Після округлення вона знову може втратити нормалізацію. При цьому в другому блоці нормалізації буде виконана її повторна нормалізація до *k+1* розрядів.

Приклад: Обчислимо різницю чисел ***a*** = $[37.625]_{10}$ і ***b*** = $[3.852]_{10}$, представлених операндами з плаваючою крапкою половинної точності.

Рішення:

a = 2^5*1.0010110101

b = 2^1*1.1110110100

---

a = 2^5*01.0010110101 {Знак і *k* - розрядна f_a }.

b = 2^5*11.11100001001011100111 1 {Знак і *2k* – розрядна інвертована мантиса f_b на виході блоку вирівнювання мантис (в цьому блоці для вирівнювання порядків операндів був виконаний арифметичний зсув мантиси операнда з меншим порядком на 4 розряду вправо)}.

111.11100001001011100111 1{f_b на першому вході суматора, її знак продубльований}.

001.0010110100 {f_a на другому вході суматора, її знак продубльований}.

1{Вхід c_{in} суматора, для формування додаткового коду f_b}.

001.000011100010111010000 {2 знакові розряди та *2k* – розрядна мантиса *fc* на виході суматора}.

001.00001110001 {2 знакові розряди і (*k+1)* розрядна мантиса *fc* на виході першого блоку нормалізації (на вході блоку нормалізації *fc* вже була нормалізованою, тому в блоці тільки проведено її усічення без зміни порядку результату)}.

001.00001110001

+1 {Для округлення *fc* до *k* розрядів у блоці округлення}.

001.0000111001 {2 знакових розряду і *k* – розрядна мантиса *fc* на виході блоку округлення}.

01.0000111001 {Знаковий розряд і *k* – розрядна мантиса *fc* на виході другого блоку нормалізації (при виконанні округлення *fc* її нормалізація не порушилась, тому в цьому блоці тільки проведено видалення одного знакового розряду без зміни порядку результату)}.

c=2^5*1.0000111001 {Різниця ***c*** = $[33.781]_{10}$}.

Блок додавання/віднімання мантис з реконфігурованою структурою

Розроблюваний суматор орієнтований на обробку всіх форматів чисел з плаваючою крапкою, що передбачені стандартом [7] (див. табл. 1). Таких форматів п'ять: половинна точність (*SF*) – 16 розрядів, одинарна (*F*) – 32, подвійна (*DF*) – 64, подвійна розширена (*DEF*) – 80 та учетверина (*QF*) – 128. Для досягнення цієї мети блок додавання/віднімання мантис повинен мати можливість оброблювати, відповідно формату: 12, 25, 54, 66 та 114 бітні мантиси (разом із знаковим та прихованим бітом). Структурна схема такого блоку показана на рис. 4. В залежності від оброблюваного формату, мантиси *x* та *y*, що представлені в додатковому коді, надходять на інформаційні входи блоку, як це показано на рис 4, а на керуючий вхід однорозрядних мультиплексорів подається 0 тільки в тому мультиплексорі, що відповідає оброблюваному формату операндів. При виконанні віднімання на вхід вхідного перенесення суматора $\sum_i$ і керуючий вхід мультиплексорів додатково подається ***1***, що забезпечує передачу на вхід блоку додаткового коду від зображення мантиси ***y***.

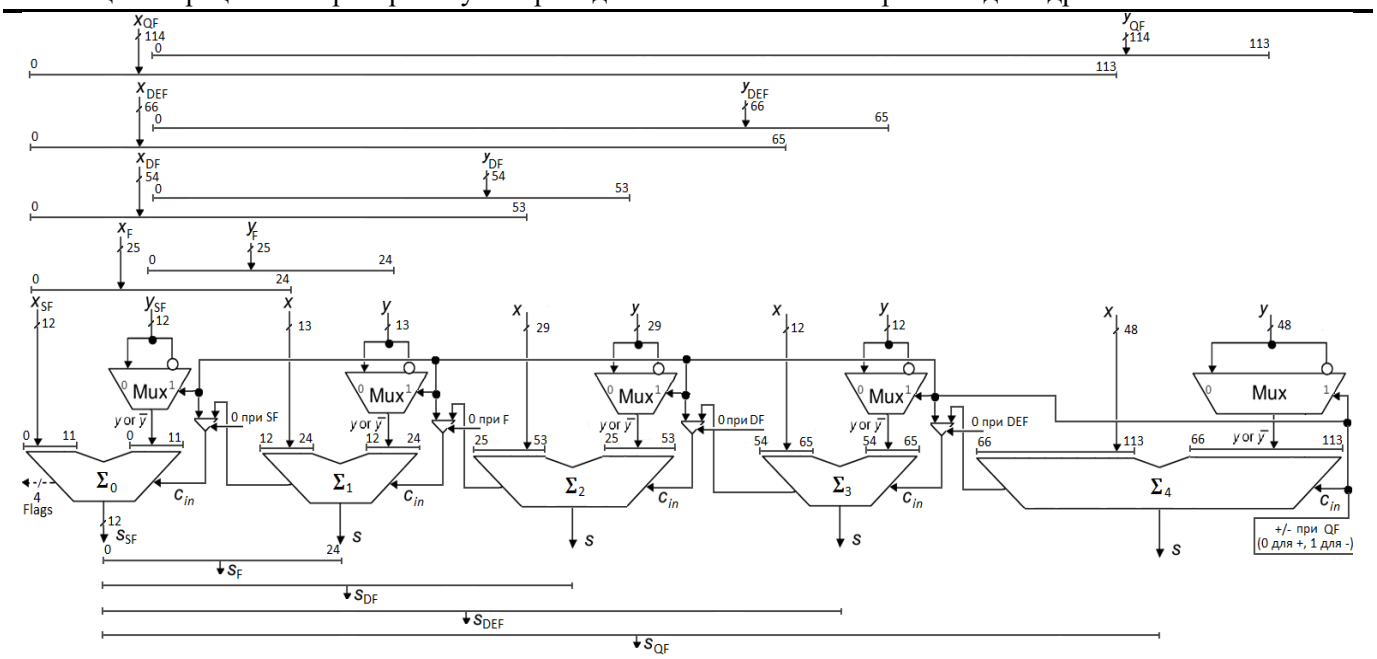


Рис. 4 – Структурна схема блоку додавання/віднімання мантис з реконфігурованою структурою

Мантиси *x* і *y*, при обробці операндів формату *QF*, подаються на входи всіх п'яти суматорів. При цьому на керуючі входи всіх однорозрядних мультиплексорів з блоку керування (див. рис. 3) на вищеописані входи подається код 1, що забезпечує передачу вихідного переносу суматора $\sum_i$, через перший вхід ***i***-го однорозрядного мультиплексора на вхід переноса суматора $\sum_{i-1}$.

Мантиси *x* і *y*, при обробці операндів формату *DEF,* через відповідні мультиплексори подаються на входи чотирьох старших суматорів ($\sum_0$ - $\sum_3$). При цьому на керуючий вхід однорозрядного мультиплексору, що зв'язаний своїм виходом із входом переноса $\sum_3$, з блоку керування подається код 0, що забезпечує блокування вихідного переносу суматора $\sum_4$ та передачу на 4 старших суматори схеми, в залежності від виконуваної операції, прямого чи додатного коду від зображення мантиси ***y***.

Мантиси *x* і *y*, при обробці операндів форматів: *DF, F* та *SF* подаються, відповідно, на входи суматорів: $\sum_0$ - $\sum_2$, $\sum_0$ - $\sum_1$ та $\sum_0$. При цьому з блоку керування подається код 0 на керуючий вхід тільки того однорозрядного мультиплексора, що зв'язаний своїм виходом із входом переноса суматора молодших розрядів операндів.

Для мінімізації затримки розповсюдження сигналу в блоці додавання/віднімання мантис був проведений аналіз [9] вибору конкретної схеми реалізації суматорів $\sum_0$ - $\sum_4$, на підставі відомих схем побудови суматорів [8].

Біти станів: *Zero*, *Sign flag*, *Overflow* та *Carry flag* в блоці додавання/віднімання мантис формуються одночасно з формуванням результату виконання операції, представленого додатковим кодом. Виходи суматора мають 4 додаткові виходи, призначені для передачі архітектурних станів процесора. Сформовані архітектурні стани, зберігаються в спеціально виділених бітах регістра станів. Так, наприклад, в архітектурі *x86-64* [2] як регістр станів використовується 64 – розрядний регістр *RFLAGS*.

Блок розпаковки операндів

Схема блоку розпаковки операндів приведена на рис. 5. Її основу складають дві вхідні 128-розрядні шини упакованих операндів: *pack bus x* та *pack bus y* та чотири вихідні шини розпакованих операндів: 114-розрядні шини мантис операндів *unpack bus* M_x та *unpack bus* M_y, а також 15-розрядні шини порядків операндів *unpack bus* E_x та *unpack bus* E_y. Незалежно від формату оброблюваних

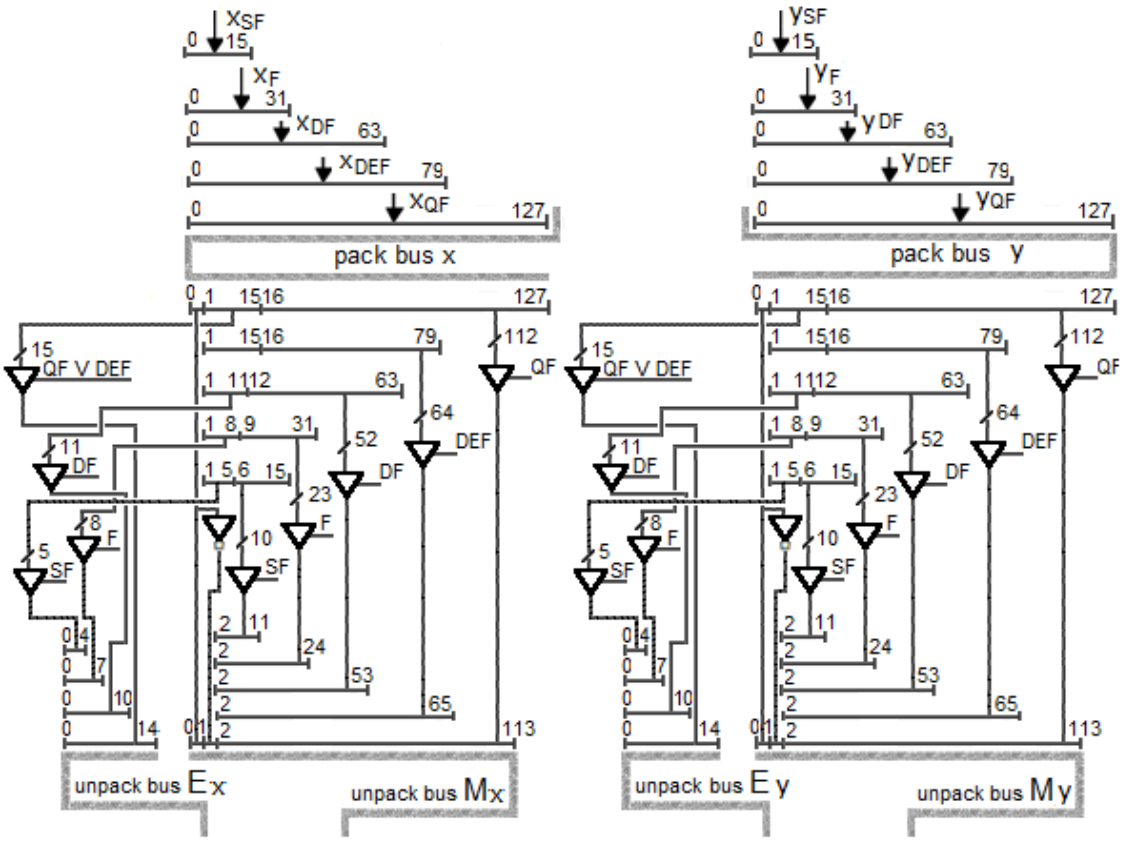


Рис. 5 – Функціональна схема блоку розпаковки операндів

операндів *x* та *y* вони завжди, на початку виконання операції додавання/віднімання, поступають із блоку регістрів з плаваючою крапкою загального призначення в старшу частину своєї вхідної шини. Розпаковка операндів здійснюється шляхом виконання потрібних зв'язків між вхідними шинами упакованих операндів та вихідними шинами порядків і мантис блоку розпаковки. Для створення потрібних зв'язків, в залежності від оброблюваного формату використовуються буфери з трьома станами, що створюють конструктивне «або» на вихідних шинах блоку. Для запобігання відмови блоку регістрів з плаваючою крапкою загального призначення потрібно забезпечити умову, що в кожний момент часу, в залежності від оброблюваного формату операндів, повинен бути відкритим тільки один з буферів, що поєднуються конструктивним «або» на вихідній шині блоку.

При розпаковці операндів в 0-ий розряд кожної з вихідних шин мантис з відповідної вхідної шини напряму заноситься знак операнда, а в 1-ий розряд цих же шин, за допомогою інверторів, із знаків операндів відновлюються приховані біти.

Блок нормалізації

Якщо в результаті виконання операцій в блоках обробки мантис та порядків блок керування суматором з плаваючою крапкою виявив виникнення наступних ситуацій: признак *Zero* (признак рівності нулю мантиси результату), *NaN* («не число»), ±∞ («нескінченність»), то блок керування зразу ж ініціює виконання операції упаковки числа відповідного результату, пропускаючи роботу блоку нормалізації. В іншому випадку відбувається нормалізація результату. Функціональна схема блоку нормалізації приведена на рис. 6.

Основу блоку нормалізації складають вхідна 116-розрядна шина *bus* M_s, що є вихідною шиною блоку обробки мантис і містить: в 0-му розряді сигнал *Overflow*, в 1-му розряді сигнал *Carry flag*, в 2-му розряді сигнал *Sign flag*, в 3 – 115 розрядах значення мантиси результату та 113-розрядна вихідна шина *bus normalize* M_s, що містить: в 0-му

розряді сигнал *Sign* ***s***, а в розрядах 1 – 112 нормалізовану мантису без прихованого біта.

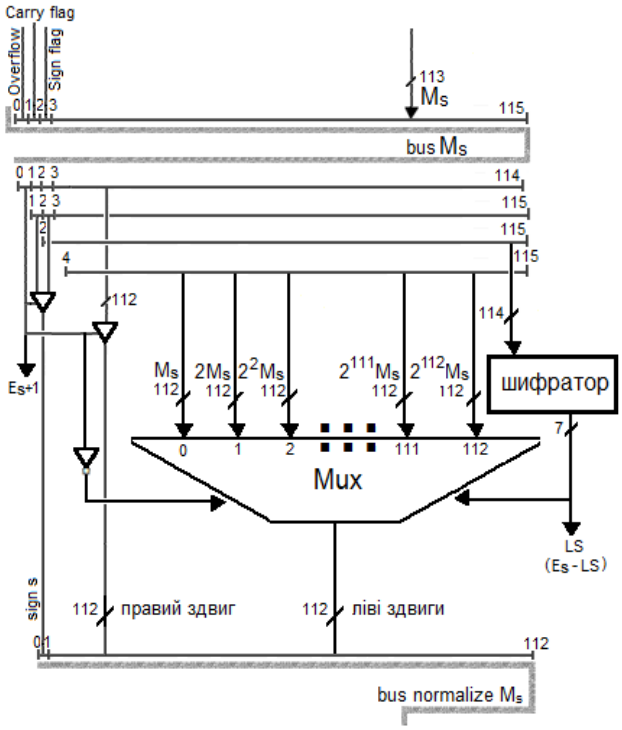


Рис. 6 – Функціональна схема блоку нормалізації

Всі операції нормалізації можливо розділити на 3 групи.

1. Мантиса результату є нормалізованою (мантиса результату передається із вхідної шини на вихідну без здвигу з видаленням прихованого біту).

2. Переповнення розрядної сітки (виконується правий здвиг мантиси на 1 розряд з видаленням прихованого біту та видається сигнал E_s+1 в блок керування суматором з плаваючою крапкою для збільшення порядку результату на 1).

3. Зменшення точності результату (виконуються ліві здвиги, що дорівнюють кількості «0» до першої правої значущої «1» в мантисі для додатного результату, або кількості «1» до першого правого значущого «0» в мантисі для від'ємного результату з видаленням прихованого біту та видається сигнал E_s - *LS* в блок керування суматором з плаваючою крапкою для зменшення порядку результату на число лівих здвигів).

Робота блока нормалізації ґрунтується на організації безперебійної роботи конструктивного «або» на вихідній шині. Для цього вивід з третього стану 112 – розрядного буферу, що виконує правий здвиг мантиси результату, та 112 – розрядного мультиплексора, що виконує ліві здвиги та безздвигову передачу, здійснюється відповідно, прямим та інверсним значенням одного й того ж сигналу *Overflow*. Така організація роботи схеми виключає одночасний вихід з третього стану буфера і мультиплексора.

Для формування знака результату його передачі використовується двухвходовий однорозрядний мультиплексор без третього стану, на керуючий вхід якого подається сигнал *Overflow*. При переповненні результату в якості його знаку передається *Carry flag*, в інших випадках – *Sign flag*.

Визначення кількості лівих здвигів, або безздвигової передачі здійснюється за допомогою шифратора. Виходи шифратора подаються на керуючі входи мультиплексора та в блок керування суматором з плаваючою крапкою, де вони використовуються для збільшення порядку результату на число лівих здвигів.

Видалення прихованого біта та формування здвинутої мантиси на потрібну кількість бітів вліво забезпечується зв'язками між вхідною шиною та відповідними розрядами на входах мультиплексора.

Блок обробки порядків

Схема блоку обробки порядків приведена на рис. 7. Основу блоку обробки порядків складають дві вхідні 15-розрядні шини порядків операндів: *unpack bus* E_x і *unpack bus* E_y та 15-розрядна вихідна шина порядку результату *bus* E_s. Незалежно від формату оброблюваних операндів *x* та *y* їх прядки, на початку виконання операції додавання/віднімання, поступають на вхідні шини блоку із блоку розпаковки операндів. Кожна з вхідних шин паралельно навантажена на чотири комбінаційні схеми: схему виявлення мінімального значення порядку, схему виявлення максимального значення порядку, віднімач порядків та мультиплексор порядків.

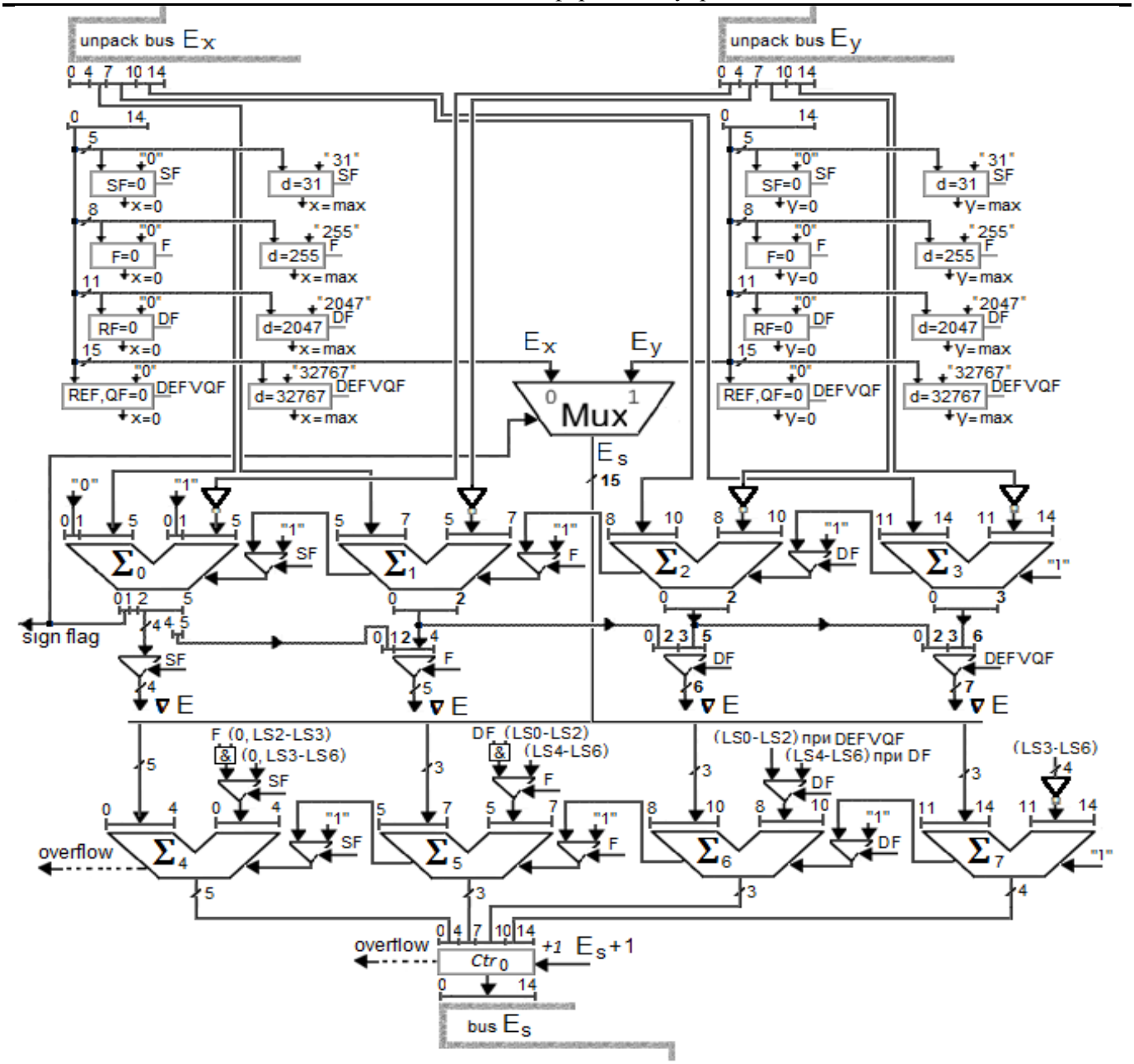


Рис. 7– Структурна схема блоку обробки порядків

Схема виявлення мінімального значення порядку, в залежності від формату оброблюваних операндів, порівнює відповідну кількість старших розрядів вхідної шини з нулем (*MSB* порядку, незалежно від формату представлення операндів, завжди знаходиться в нульовому розряді вхідної шини). П'ять оброблюваних форматів операндів мають чотири різних формати для представлення порядку (див. табл. 1.1). Схема виявлення мінімального значення порядку складається з чотирьох схем порівняння з трьома станами, виходи котрих поєднуються за допомогою конструктивного «або» разом в сигнал $x=0$ (або $y=0$) для блока керування виконанням операцією. Вивід кожної з цих схем із третього стану, здійснюється сигналом обробки відповідного формату операндів, що надходять з блока керування виконанням операцією (наприклад, при обробці подвійного формату $DF=1$).

Схеми виявлення максимального значення порядків порівнюють порядки операндів з константою максимального значення порядку, що для кожного з форматів порядків має своє значення (див. табл. 1.1). При виникненні сигналу $x=max$ (або $y=max$) число x (чи y) вважається за нескінченість без перевірки, відповідно, $M_x=0$ (чи $M_y=0$). Таке припущення призводить до

того, що якщо один з оброблюваних операндів був *NaN* (не числом), то воно буде автоматично прирівняне до ±∞. Наскільки таке рішення відповідає вимогам [7], потребує подальшого дослідження. При формуванні сигналу $x = max$ (або $y = max$) блок керування виконанням операцією, не очікуючи завершення виконання операції, ініціює перехід до упаковки результату відповідно до (1). Наприклад, якщо при виконанні операції x-y виникає сигнал $y = max$, то в якості результату буде упаковано: $s = -y$ (нескінченість із зміненим знаком).

Схема віднімача порядків складається з суматорів: $\sum_0 - \sum_3$, лінійок інверторів, що зв'язані з другим входом відповідного суматора та буферів з трьома станами, котрі своїми входами поєднані з виходами суматорів, відповідно до рис. 7, а їх виходи утворюють конструктивне «або», формуючи в підсумку семи розрядну шину $\blacktriangledown E$, та сигнал *sign flag*. За допомогою віднімача реалізується операція $E_x - E_y$. Віднімач реалізовано, як суматор, що оброблює кількість розрядів шин порядків відповідно до оброблюваного формату операндів. Для цього на других входах суматорів формується додатний код E_y. Схема віднімача реалізована таким чином, що без знакові порядки доповнюються знаковим розрядом. Знак $\boldsymbol{E_x}$ завжди є додатнім, а знак $\boldsymbol{E_y}$ – від'ємним. Завдяки цьому на віднімачі обраховується число $\blacktriangledown E$ в додатному коді, що представляє собою кількість розрядів на які виконується, в блоці вирівнювання мантис, правий здвиг мантиси числа з меншим порядком. Якщо $\blacktriangledown E \geq 0$ (при цьому *sign flag = 0*), то більшим за абсолютним значенням є число x. При цьому через нульовий вхід мультиплексора передається для подальшої обробки E_x в якості попереднього порядку результату E_s. Разом з цим в блок вирівнювання мантис передається число $\blacktriangledown E$, на яке в ньому буде виконано правий здвиг мантиси числа y. При $\blacktriangledown E < 0$ (*sign flag = 1*) більшим за абсолютним значенням є число y. При цьому, в якості попереднього порядку результату E_s. через перший вхід мультиплексора, передається для подальшої обробки E_y. Разом з цим в блок вирівнювання мантис передається число $\blacktriangledown E$, на яке в ньому буде виконано правий здвиг мантиси числа x.

З виходу мультиплексора E_s подається для подальшої обробки на перші входи другого віднімача, що зібраний з суматорів: $\sum_4 - \sum_7$. На другі входи цих суматорів, в залежності від оброблюваного формату операндів та за допомогою мультиплексорів й зв'язків з рис. 7, видається число *LS*. Це число формується в блоці нормалізації й несе інформацію про кількість лівосторонніх здвигів мантиси результату, які були виконані над нею в процесі нормалізації результату. В процесі цієї операції значення E_s зменшується на *LS*. З виходів другого віднімача E_s подається для подальшої обробки на входи комбінаційної схеми інкрементного лічильника.

Завдяки використанню лічильника значення E_s може збільшитися на одиницю у тому випадку, коли в результаті округлення мантиси результату до кількості розрядів, що відповідають оброблюваному форматові операндів, виникло переповнення її розрядної сітки. Після цієї корекції E_s, за допомогою схеми формування сигналу *overflow* здійснюється перевірка на переповнення, в процесі округлення результату, розрядної сітки порядку результату. Сигнал *overflow* видається в блок упаковки результату, де він при *overflow = 1* використовується для упаковки $y = \pm\infty$ (в залежності від $sign\ s$).

Висновки

В цій роботі описана структурна схема суматора з плаваючою крапкою. Перевагами розробленої схеми є те, що вона виконана як набір комбінаційних схем, без використання елементів пам'яті й мікропрограмного керування.

Операція сумування операндів починається й завершується за один цикл роботи суматора. На входи суматора з плаваючою крапкою операнди подаються з блоку регістрів загального призначення, а сигнал оброблюваного формату надходить з відповідної станції резервації, що зберігає *RISC* операцію виконуваної *CISC* команди типу сумування.

Результат виконаної операції заноситься в один з регістрів загального призначення та в буфер переупорядкування результатів. Таким чином, розроблений суматор з плаваючою крапкою орієнтований на використання в якості

одного із операційних пристроїв суперскалярного мікропроцесорного ядра з архітектурою *CISC- RISC.*

Для реконфігурування суматора на обробку операндів потрібного формату не потребується ніяких додаткових дій, крім подачі на його керуючі входи сигналу оброблюваного формату.

Виконана розробка буде корисною при проектуванні операційного пристрою сумування операндів з фіксованою чи з плаваючою крапкою, як при побудові ядра мікропроцесора з суперскалярною архітектурою, сумісною з сімейством x86-64, так і при побудові спеціалізованих апаратних засобів на ПЛІС.

Список посилань

1. Y. Patt, W. Hwu, et al, Experiments with HPS, a Restricted Data Flow Micro architecture for High Performance Computers, Digest of Papers, COMPCON 86, (March 1986), pp. 254-258.
2. John L. Hennessy, David A. Patterson. Computer Architecture. A Cuantitative Approach, USA, Morgan Kaufmann, 2012 – 497 p.+ add-ins.
3. Kanter, David (September 25, 2010). "Intel’s Sandy Bridge Microarchitecture" (http://www.realworldtech.com/sandy-bridge/).
4. Kanter, David (November 13, 2012). "Intel’s Haswell CPU Microarchitecture" (http://www.realworldtech.com/haswell-cpu/).
5. Луцький Г.М., Долголенко О.М., Аксьоненко С.В., Сторожук В.О. Моделювання обмеженої реалізації архітектури потоку даних в структурі суперскалярного процесора. Київ,- Вісник НТУУ «КПІ». Інформатика, управління та обчислювальна техніка: Зб. наук. пр. – К.: Век+, – 2014. – № 60. – с. 83-94.
6. J. Dennis: Data Flow Supercomputers; IEEE Computer, pp. 48-56, Nov. 1980.
7. IEEE 754: Standard for Binary Floating-Point Arithmetic [Електронний ресурс] / 3 апрель 2014. – URL: http://grouper.ieee.org/groups/754/.
8. Behrooz Parhami. Computer Arithmetic. Algorithms and Hardware Designs, New York, Oxford University Press, 2000– 491 p.
9. Яцун В.О., Долголенко О.М. Блок додавання та віднімання мантис з реконфігурованою структурою. Матеріали наукової конференції студентів, магістрантів та аспірантів «Інформатика та обчислювальна техніка – ІОТ-2016». Київ 25 – 27 квітня 2016, – с. 158-162 (http://fiot.kpi.ua/wp-content/uploads/2016/06/IOT-2016-OT.pdf).

УДК 519.8

ПАПУЧ Т. Ю.

ІНТЕГРАЦІЯ МЕТОДІВ АНАЛІЗУ БАНКРУТСТВА ТА МОДЕЛЕЙ ОЦІНКИ КРЕДИТОСПРОМОЖНОСТІ ЮРИДИЧНИХ ОСІБ

В даній статті розглянуто методи та моделі оцінки кредитоспроможності юридичних осіб. З метою покращення точності кредитного прогнозу запропоновано враховувати оцінку ризику банкрутства позичальника. Виконано математичний та прикладний аналіз методів інтелектуального аналізу даних для вимірювання кредитного ризику та ризику банкрутства. Аналіз кредитоспроможності підприємств здійснено за допомогою реалізованої інтелектуальної системи підтримки прийняття рішень, що використовує фінансові показники в якості вхідних параметрів.

This paper is devoted to methods and models adopted for estimating the creditworthiness of enterprises. To consider bankruptcy risk evaluation of the borrower were offered in order to improve accuracy of credit forecasting. Mathematical and applied analysis of intellectual data techniques for measuring credit and bankruptcy risk were done. Creditworthiness analysis of enterprises were employed by using implemented intellectual decision support system, which utilizes financial ratios as input parameters.

Ключові слова: ризик банкрутства, кредитоспроможність підприємства, фінансові показники, модель кредитного скорингу, модель Альтмана, моделі нечіткого логічного висновку.

1. Вступ

Кредитний ризик є широко досліджуваною темою в фінансовій сфері, особливо в періоди фінансової нестабільності. Під час кризи важливо оцінювати крім кредитоспроможності позичальника, ще й ризик його банкрутства. Саме тому, було запропоновано інтеграцію методів аналізу банкрутства та моделей оцінки кредитоспроможності юридичних осіб, що дозволить зменшити відсоток помилок 1-го і 2-го типу. Помилка першого типу виникає тоді, коли фірма-банкрут класифікувалася як фірма зі сприятливим фінансовим станом. Помилка другого типу виникає тоді, коли підприємство з нормальним фінансовим станом класифікується як потенційний банкрут. Обидві помилки можуть призвести до серйозних наслідків і збитків.

Для того, щоб відповідати вимогам Національного Банку України і рекомендаціям Базельского комітету по банківському нагляду, банкам необхідно мати ретельно розроблену політику по управлінню ризиками.

Одним з найважливіших складників даної політики являється набір професійно підібраних, з урахуванням економічного стану країни і специфіки діяльності банку, статистичних або математичних методів оцінки очікуваних та неочікуваних втрат.

2. Постановка задачі

Задано фінансові показники підприємств, які можуть бути отримані з щорічної фінансової звітності. Після попередньої обробки даних, що включає розрахунок фінансових факторів, що впливають на кредитоспроможність підприємства, необхідно оцінити ризик банкрутства позичальника. Ця оцінка у подальшому використовується для аналізу кредитоспроможності юридичної особи в якості вхідного параметру. Для оцінки ризику банкрутства запропоновано використовувати статистичну модель Альтмана та нечітко-множинний підхід Нєдосєкіна. Основні методи для вимірювання кредитного ризику, що використовуються, є кредитний скоринг та моделі нечіткого логічного висновку. Після застосування даних методів необхідно отримати клас кредитоспроможності, до якого відноситься позичальник. На основі цієї класифікації кредитний менеджер може прийняти рішення про надання кредиту або про відмову у його наданні.

3. Виклад основного матеріалу дослідження

Першим методом для оцінки кредитоспроможності позичальника був обраний метод кредитного скорингу. Кредитний скоринг може бути визначений як систематичний метод для оцінки кредитного ризику, що виконує послідо-

вний аналіз факторів, які були визнаними такими, що впливають на рівень ризику. Процес кредитного скорингу починається саме з визначення цих факторів і їх врахування у модель. Фінансовим факторам, так само як і не фінансовим компонентам надається кредитна оцінка або 1, або 2, або 3 (таблиця 1). Кредитна оцінка 1 означає високий рівень ризику банкрутства, 2 представляє середній ризик та кредитна оцінка 3 показує низький рівень ризику банкрутства. Таким чином, низька кредитна оцінка показує високий ризик і меншу кредитоспроможність корпорації, а висока кредитна оцінка представляє низький рівень ризику банкрутства і більшу кредитоспроможність корпорації.

Табл. 1. Модель кредитного скорингу для підприємств

	Скоринг	**Скоринг**	**Скоринг**	**Скоринг**
1. Фінансові фактори = 80%	1	2	3	
Коефіцієнти ліквідності				
Коефіцієнт поточної ліквідності	<1	1 – 1,5	>1,5	
Коефіцієнт миттєвої ліквідності	<0,75	0,75 – 1,25	>1,25	
Коефіцієнти рентабельності				
Чистий валовий прибуток	<1,5%	1,5 – 5%	>5%	
Рентабельність по операційному прибутку	<1,5%	1,5 – 5%	>5%	
Рентабельність по чистому прибутку	<1,5%	1,5 – 5%	>5%	
Рентабельність активів	<5%	5 – 15%	>15%	
Рентабельність власного капіталу	<10%	10 – 20%	>20%	
Зростання продажів (за минулі 2 роки)	<5%	5 – 20%	>20%	
Показники використання фінансових коштів				
Заборгованість до власного капіталу	>1,2	0,8 – 1,2	<0,8	
Загальна заборгованість до активів	>1,2	0,8 – 1,2	<0,8	
Коефіцієнт фінансової заборгованості	>5	1,5 – 5	< 1,5	
Коефіцієнти покриття				
Коефіцієнт покриття процентних ставок	<1	1 – 1,5	>1,5	
Коефіцієнт покриття обслуговування боргу	<1,2	1,2 – 2	>2	
Діяльність / Коефіцієнти ефективності				
Коефіцієнт оборотності дебіторської заборгованості – дні	>120	60 – 120	<60	
Період оборотності запасів	>180	90 – 180	<90	
Оборотність кредиторської заборгованості – дні	>90	30 – 90	<30	
Ринкові показники				
Прибуток на акцію	<10	10 – 50	>50	
Відношення ціни акції до доходу	<20	20 – 25	>25	

	Скоринг	Скоринг	Скоринг	Скоринг
Z оцінка Альтмана = 10%	<1,8	1,8 – 2,99	>2,99	
Загальна оцінка – фінансові фактори				80 (макс)
2. Не фінансові фактори = 20%				
Кредитний рейтинг	CCC-C	B-BBB	A-AAA	
Кредитна історія – дні	>=90	>=30 і <90	Дефолту не було	
Загальна оцінка – не фінансові фактори				20 (макс)
Загальна оцінка				100 (макс)

Існує чотири класи ризику, а саме A, B, C, D (таблиця 2). Клас ризику 'A', коли кредитна оцінка знаходиться в діапазоні 91-100%, показує відсутність ризику банкрутства та відповідає найвищій кредитній оцінці. Клас ризику 'B' показує найнижчий ризик банкрутства завдяки високій кредитній оцінці. Клас ризику 'C' представляє середній рівень кредитного ризику, так як має середній рівень кредитної оцінки. Клас ризику 'D' визначає високий рівень ризику і має кредитну оцінку
нижче середньої. Коли кредитна оцінка нижча за 55%, підприємство не отримує кредит [1].

Табл. 2. Діапазони кредитної оцінки та класи ризику

Діапазон кредитної оцінки, %	Оцінка фінансового стану підприємства	Клас ризику
91 – 100	Найвища	A
76 – 90	Висока	B
55 – 75	Середня	C
<55	Нижче середнього	D

Таким чином, при аналізі кредитоспроможності підприємства за методом кредитного скорингу було враховано оцінку ризику банкрутства, розраховану за допомогою статистичної моделі Альтмана. Модель Альтмана використовує класичний метод багатовимірного дискримінантного аналізу і має наступний вигляд:

$$Z = 1{,}2 * K_1 + 1{,}4 * K_2 + 3{,}3 * K_3 + 0{,}6 * K_4 + \\ + 1{,}0 * K_5,$$

де K_1 – власний оборотний капітал / сума активів;

K_2 – нерозподілений прибуток / сума активів;

K_3 – прибуток до сплати відсотків / сума активів;

K_4 – ринкова вартість власного капіталу / вартість позикового капіталу;

K_5 – об'єм продажів / сума активів.

В результаті після підрахунку Z-показника можна зробити наступний висновок:

якщо $Z < 1{,}8$ – дуже висока ймовірність банкрутства;

якщо $1{,}8 \le Z < 2{,}7$ – висока ймовірність банкрутства;

якщо $2{,}7 \le Z \le 2{,}99$ – банкрутство можливе;

якщо $Z \ge 3{,}0$ – ймовірність банкрутства дуже мала [4].

Другим інструментом оцінки кредитного ризику є модель нечіткого логічного висновку Мамдані. Аналіз кредитоспроможності діючого підприємства на основі цього підходу складається з наступних етапів:

Етап 1. Лінгвістичні змінні і нечіткі підмножини.

На цьому етапі необхідно ввести лінгвістичні змінні, що описують кредитоспроможність позичальників.

– Кредитна історія (бездоганна, погана, нема даних). Змінна – чітка і приймає значення від 0 до 1.

– Фінансово-економічний стан (стабільний, негативні тенденції, позитивні тенденції).

Етап 2. Визначення показників.

- Коефіцієнт незалежності.
- Коефіцієнт миттєвої ліквідності.
- Період оборотності запасів.
- Ймовірність несвоєчасного погашення.
- Ризик збитків.
- Стан документації.
- Банкрут (так або ні).
- Забезпечення (першокласне, ліквідне, неліквідне, можливі проблеми, відсутність). Вид змінної - нечітка. Функція приналежності для терму «першокласне забезпечення»:

$$\mu(d) = \begin{cases} 0, & x < 0{,}8, \quad x > 1; \\ 1, & x = 1; \\ 5x - 4, & 0{,}8 \le x \le 1. \end{cases}$$

Функція приналежності для терму «ліквідне забезпечення»:

$$\mu(d)=\begin{cases}0, & x<0{,}6, \quad x>0{,}8;\\ 1, & x=0{,}8;\\ 5x-3, & 0{,}6\le x\le 0{,}8.\end{cases}$$

Функція приналежності для терму «неліквідне забезпечення»:

$$\mu(d)=\begin{cases}0, & x<0{,}4, \quad x>0{,}6;\\ 1, & x=0{,}6;\\ 5x-2, & 0{,}4\le x\le 0{,}6.\end{cases}$$

Функція приналежності для терму «можливі проблеми з забезпеченням»:

$$\mu(d)=\begin{cases}0, & x<0{,}2, \quad x>0{,}4;\\ 1, & x=0{,}4;\\ 5x-1, & 0{,}2\le x\le 0{,}4.\end{cases}$$

Функція приналежності для терму «відсутність забезпечення»:

$$\mu(d)=\begin{cases}0, & x<0, \quad x>0{,}2;\\ 1, & x=0{,}2;\\ 5x, & 0\le x\le 0{,}2.\end{cases}$$

Етап 3. Формування бази правил системи нечіткого висновку.

На цьому етапі визначаються правила для класифікації позичальника з урахуванням рівня забезпечення по кредитній операції.

Етап 4. Фазифікація вхідних параметрів.

Для того, щоб визначити значення лінгвістичних змінних необхідно порівняти значення вибраних коефіцієнтів за поточний період і за минулий період, тобто необхідно визначити значення дельти для кожного коефіцієнта. Будемо розглядати дельту для кожного показника як нечітку змінну. Для неї необхідно ввести функцію приналежності.

1. Коефіцієнт незалежності. Сам коефіцієнт змінюється від 0 до 1. Дельта змінюється від -1 до 1.

Функція приналежності для терму «Стабільний»:

$$\mu(d)=\begin{cases}1+20d, & якщо -0{,}05\le d<0;\\ 1, & якщо \quad d=0;\\ 1-20d, & якщо \quad 0<d\le 0{,}05;\\ 0, & якщо \quad d\le -0{,}05, \quad d\ge 0{,}05.\end{cases}$$

Функція приналежності для терму «Негативні тенденції»:

$$\mu(d)=\begin{cases}-20d, & якщо -0{,}05\le d<0;\\ 1, & якщо \quad -1\le d<-0{,}05;\\ 0, & якщо \quad 0\le d.\end{cases}$$

Функція приналежності для терму «Позитивні тенденції»:

$$\mu(d)=\begin{cases}20d, & якщо \quad 0<d\le 0{,}05;\\ 1, & якщо \quad 0{,}05<d\le 1;\\ 0, & якщо \quad d\le 0.\end{cases}$$

2. Коефіцієнт миттєвої ліквідності. Сам коефіцієнт змінюється від 0 до 1. Дельта змінюється від -1 до 1.

Функція приналежності для терму «Стабільний» має наступний вигляд:

$$\mu(d)=\begin{cases}1+10d, & якщо -0{,}1\le d<0;\\ 1, & якщо \quad d=0;\\ 1-10d, & якщо \quad 0<d\le 0{,}1;\\ 0, & якщо \quad d\le -0{,}1, \quad d\ge 0{,}1.\end{cases}$$

Функція приналежності для терму «Негативні тенденції» виглядає наступним чином:

$$\mu(d)=\begin{cases}-10d, & якщо -0{,}1\le d<0;\\ 1, & якщо \quad -1\le d<-0{,}1;\\ 0, & якщо \quad 0\le d.\end{cases}$$

Функція приналежності для терму «Позитивні тенденції»:

$$\mu(d)=\begin{cases}10d, & якщо \quad 0<d\le 0{,}1;\\ 1, & якщо \quad 0{,}1<d\le 1;\\ 0, & якщо \quad d\le 0.\end{cases}$$

3. «Період оборотності запасів». Сам коефіцієнт змінюється від 0 до 180. Дельта змінюється від -1 до 1.

Функція приналежності термів «Стабільний», «Негативні тенденції» та «Позитивні тенденції» має аналогічний до коефіцієнту миттєвої ліквідності вигляд.

4. Ймовірність несвоєчасного погашення (низька, середня, висока, дуже висока). Ймовірність змінюється від 0 до 1.

Функція приналежності дельти до терму «низька»:

$$\mu(p)=\begin{cases}1-20p, & якщо \quad 0\le p<0{,}05;\\ 1, & якщо \quad p=0;\\ 0, & якщо \quad p\ge 0{,}05.\end{cases}$$

Функція приналежності дельти до терму «середня»:

$$\mu(p)=\begin{cases}3{,}33-33{,}33p, & якщо \quad 0{,}07\le p<0{,}1;\\ 1, & якщо \quad 0{,}05<p<0{,}07;\\ 0, & якщо \quad p\ge 0{,}1, \quad p\le 0{,}05.\end{cases}$$

Функція приналежності до терму «висока»:

$$\mu(p)=\begin{cases}33{,}33p-2{,}33, & якщо \quad 0{,}07\le p<0{,}1;\\ -5p+2{,}5, & якщо \quad 0{,}3<p\le 0{,}5;\\ 1, & якщо \quad 0{,}1<p\le 0{,}3;\\ 0, & якщо \quad p\ge 0{,}5,\ p\le 0{,}07.\end{cases}$$

Функція приналежності до терму «дуже висока»:

$$\mu(p)=\begin{cases}5p-1{,}5, & якщо \quad 0{,}3<p<0{,}5;\\ 1, & якщо \quad 0{,}5\le p;\\ 0, & якщо \quad p\le 0{,}3.\end{cases}$$

5. Ризик збитків (високий, середній, низький). В якості показника даного ризику запропоновано використовувати оцінку ризику банкрутства, яка може бути розрахована за допомогою матричного методу прогнозування банкрутства Нєдосєкіна, заснованого на апараті нечітких множин. Метод Нєдосєкіна складається з наступних етапів: визначення лінгвістичних змінних і нечітких множин, побудова набору окремих показників, що впливають на оцінку ризику банкрутства, визначення рівня значущості кожного показника, класифікація ступеня ризику, класифікація значень показників, оцінка рівня показників, класифікація рівня показників, оцінка ступеня ризику, лінгвістичне розпізнавання [2].

Етап 5. Логічний висновок. На цьому етапі необхідно знайти рівні «відсікання» для передумов кожного із правил з використанням операції перетину у формі min, а також «усічені» функції приналежності для виходів правил.

Етап 6. Композиція виходів. Проводиться об'єднання знайдених усічених функцій з використанням операції max, що приводить до отримання кінцевої нечіткої підмножини для вихідної змінної.

Етап 7. Приведення до чіткості (дефазифікація). Приведення до чіткості здійснюється центроїдним методом [3].

4. Побудова інтелектуальної системи підтримки прийняття рішень (ІСППР)

Для практичної реалізації вищезазначених методів оцінки кредитоспроможності підприємств і порівняння їх ефективності та результатів роботи реалізовано ІСППР, що працює за алгоритмом, представленим на блок-схемі (рис. 1).

В якості вхідних змінних ІСППР використовуються фінансові показники юридичної особи, які можуть бути отримані з щорічної фінансової звітності, а саме: оборотні активи, поточні зобов`язання, товарно-матеріальні запаси, чистий дохід від реалізації продукції, собівартість реалізованої продукції, прибуток від операційної діяльності, чистий прибуток, необоротні активи, власний капітал, довгострокова заборгованість, короткострокова заборгованість, сума активів, прибуток до сплати відсотків і податків, фінансові витрати, амортизація, загальні зобов`язання, обсяг продажів, дебіторська заборгованість, кредиторська заборгованість, кредитна історія, нерозподілений прибуток, ринкова вартість власного капіталу, вартість позикового капіталу.

Після введення цих показників відбувається попередня обробка даних, що включає розрахунок фінансових факторів, що впливають на кредитоспроможність підприємства.

Наступним етапом роботи ІСППР є оцінка ризику банкрутства за моделлю Альтмана та нечітким матричним методом Нєдосєкіна. Після оцінки банкрутства підприємства проводиться аналіз кредитоспроможності юридичної особи за методом кредитного скорингу та за моделлю нечіткого логічного висновку Мамдані. На виході системи користувач отримує кредитний рейтинг позичальника, на основі якого він може прийняти рішення про надання йому кредиту або про відмову.

Для аналізу результатів було обрано 40 підприємств, 20 з яких є офіційно визнаними банкрутами і оббіг їх акцій було анульовано на ринку цінних паперів. Інші 20 підприємств офіційно вважаються працездатними на сьогоднішній день.

Проаналізувавши першу групу підприємств «банкрутів» за допомогою моделі Альтмана за два роки до дефолту, було отримано наступні результати (табл.3). Таким чином, модель Альтмана в середньому показує правильний результат з точністю 0,65 для підприємств, які є потенційними банкрутами.

Рис. 1. Блок-схема розробленої ІСППР

Табл. 3. **Результати прогнозу ризику банкрутства за моделлю Альтмана «підприємства - банкрути»**

Група	Кількість компаній	Прогноз ризику банкрутства		
		Високий	Середній	Низький
За два роки до банкрутства	20	60% (12)	5% (1)	35% (7)
За рік до банкрутства	20	70% (14)	0% (0)	30% (6)
Середнє за два роки	20	65%	2.5%	32.5%

Для другої групи підприємств, які вважаються працездатними, були отримані наступні результати (табл. 4). Рівень точності прогнозування ризику працездатності протягом досліджуваного періоду для підприємств не банкрутів складає 0,925.

Табл. 4. **Результати прогнозу ризику банкрутства за моделлю Альтмана «підприємства – не банкрути»**

Група	Кількість компаній	Прогноз ризику банкрутства		
		Високий	Середній	Низький
За 2009 рік	20	5% (1)	5% (1)	90% (18)
За 2010 рік	20	0% (0)	5% (1)	95% (19)
Середнє за два роки	20	2.5%	5%	92.5%

Також був проведений аналіз фінансового стану цих самих підприємств за допомогою нечітко-множинного методу Нєдосєкіна. Результати аналізу підприємств-банкрутів приведені у таблиці 5. Таким чином, підхід Нєдосєкіна дає правильний прогноз для підприємств, які є потенційними банкрутами з середньою точність 0,95.

Табл. 5. **Результати прогнозу ризику банкрутства за моделлю Нєдосєкіна «підприємства – банкрути»**

Група	**Кількість компаній**	**Прогноз ризику банкрутства**	
		Банкрути	**Не банкрути**
За два роки до банкрутства	20	95% (19)	5% (1)
За рік до банкрутства	20	95% (19)	5% (1)
Середнє за два роки	20	95%	5%

Для другої групи були отримані наступні результати, приведені у таблиці 6. Прогнозна точність підходу для підприємств не банкрутів складає 0,825.

Табл. 6. **Результати прогнозу ризику банкрутства за моделлю Нєдосєкіна «підприємства – не банкрути»**

Група	**Кількість компаній**	**Прогноз ризику банкрутства**	
		Банкрути	**Не банкрути**
За 2009 рік	20	10% (2)	90% (18)
За 2010 рік	20	25% (5)	75% (15)
Середнє за два роки	20	17.5%	82.5%

Також був проведений аналіз результатів оцінки кредитного ризику за моделлю нечіткого логічного висновку Мамдані. Результати аналізу підприємств не банкрутів представлені у таблиці 7, підприємств банкрутів – у таблиці 8, загальний аналіз підприємств – у таблиці 9. При цьому інтерпретація рейтингу наступна: рейтингу класу А відповідає низький рівень кредитного ризику, рейтингам класів Б і В – середній, а рейтингам класів Г і Д – високий.

Табл. 7. **Результати аналізу кредитоспроможності за моделлю Мамдані «підприємства – не банкрути»**

Назва класу	**А**	**Б**	**В**	**Г**	**Д**
Кількість підприємств, що належать до класу	8	4	6	2	0

Табл. 8. **Результати аналізу кредитоспроможності за моделлю Мамдані «підприємства – банкрути»**

Назва класу	**А**	**Б**	**В**	**Г**	**Д**
Кількість підприємств, що належать до класу	0	0	5	7	8

Табл. 9. **Результати аналізу кредитоспроможності підприємств за моделлю Мамдані**

Назва класу	**А**	**Б**	**В**	**Г**	**Д**
Кількість підприємств, що належать до класу	8	4	11	9	8

5. Висновки

Невід'ємною частиною оцінки кредитоспроможності є оцінка банкрутства позичальника. Інтеграція методів оцінки кредитного ризику та ризику банкрутства дозволяють покращити прогнозну якість. Використання моделей кредитного скорингу дозволяє уникнути дискримінації позичальників. Моделі нечіткого логічного висновку можуть виражати людські експертні знання і досвід, а також працювати з нечітко сформульованою інформацією про позичальників. У подальшому планується застосування штучних нейронних мереж для аналізу кредитного ризику.

Список посилань

1. Samreen A. Design and Development of Credit Scoring Model for the Commercial Banks in Pakistan: Forecasting Creditworthiness of Corporate Borrowers / A. Samreen, F. Batul Zaidi, A. Sarwar // International Journal of Business and Commerce. – 2013. – vol.2, № 5. – p. 1-26.
2. Зайченко Ю.П. Комплексный анализ риска банкротства корпораций в условиях неопределенности / Ю.П. Зайченко // International Journal 'Information Technologies Knowledge'. – 2012. – vol.6, № 2, – p. 103-125.
3. Зайченко Ю.П. Оцінка кредитних банківських ризиків з використанням нечіткої логіки / Ю.П. Зайченко // Системні дослідження та інформаційні технології. — 2010. — № 2. — С. 37–54.
4. Зайченко Ю.П. Нечіткі моделі і методи в інтелектуальних системах: учбовий посібник для студентів вищих навчальних закладів / Ю.П. Зайченко. – К.: Видавничий Дім «Слово», 2008. – 344 с.

УДК 004.056.53

БУЙБАРОВА М.Ф.,
ВИНОГРАДОВ Ю.М.,
ПРИЙМАК В.Ю.,

МЕТОД ЗАХИЩЕНОЇ РЕАЛІЗАЦІЇ ПЕРЕТВОРЕНЬ ФУР'Є НА ВІДДАЛЕНИХ РОЗПОДІЛЕНИХ КОМП'ЮТЕРНИХ СИСТЕМАХ

У статті пропонується метод захищеної реалізації Дискретного Перетворення Фур'є (ДПФ) та Швидкого Перетворення Фур'є (ШПФ) на віддалених процесорних засобах хмарних систем. Метод забезпечує захист від доступу до даних при їх передачі та під час їх обробки на віддалених комп'ютерних системах. Метод базується на використанні потокового шифрування і використання адитивного маскування відліків сигналу. Детально описані процедури шифрування даних і їх дешифрування після обробки. Наведено числові приклади шифрування та дешифрування даних для ДПФ та ШПФ. Теоретично та експериментально доведено ефективність запропонованого методу.

This paper proposes a method for protected implementation of Discrete Fourier Transform (DFT) and Fast Fourier Transform (FFT) at remote processors of cloud systems. Proposed method ensures protection of data against unauthorized access during network transmission and during processing on remote computer systems. The method is based on stream encryption and using of additional masking for signals. The proposed procedure for data encryption and decryption after processing are described in details. A numerical example for encryption and decryption for both DFT and FFT are given. The effectiveness of proposed method is proved theoretically and experimentally.

Ключові слова: Розподілені віддалені обчислення, хмарні обчислення, захищені обчислення, швидке перетворення Фур'є, обробка сигналів.

Вступ

Однією з фундаментальних операцій комп'ютерної обробки інформації є Дискретне Перетворення Фур'є (ДПФ). Ця операція лежить в основі технологій обробки сигналів, зокрема, зображень та звуків. ДПФ перетворює послідовність цифрових вимірів сигналу через певні проміжки часу в спектральне представлення сигналу, у вигляді набору амплітуд та фаз синусоїд, сума яких відтворює сигнал. Зрозуміло, що, чим більше синусоїд задіяні в представленні сигналу, тим вища якість представлення сигналу. З іншого боку, чим більше синусоїд використовується для відтворення сигналу, тим більший об'єм обчислень потребує реалізація ДПФ.

Для сучасного етапу розвитку інформаційних технологій характерним є динамічне розширення інтерфейсу між зовнішнім світом та засобами комп'ютерної обробки інформації. Домінуючу роль у цьому процесі відіграють засоби цифрової обробки сигналів, в основі яких лежить ДПФ. Це означає, що коло застосування ДПФ ширшає з кожним роком, як і вимоги до якості перетворення. Потужний імпульс розширенню використання ДПФ надає розповсюдження систем відео нагляду. Ці системи широко використовуються в технологічних процесах, системах регулювання, а також, є технологічною основою боротьби з проявами тероризму. Характерним для таких систем є те, що вони працюють в середовищі Інтернет у реальному часі, а це, в свою чергу, диктує жорсткі вимоги до швидкості реалізації ДПФ. Таким чином, важливою проблемою є пошук резервів для прискорення реалізації ДПФ.

Одним із таких резервів, який може бути використаний для швидшого обчислення ДПФ, є прогресивні хмарні технології. Вже на сьогоднішній день хмарні технології, які надають доступ користувачеві Інтернету до практично необмежених обчислювальних ресурсів, активно залучаються для вирішення широкого кола важливих для практики задач. З іншого боку, саме головні переваги хмарних технологій: обчислювальна потужність та загальнодоступність є причиною недоліків і проблем. Гостро постає питання стійкості віддаленої обчислювальної системи до втручання зловмисників та захищеності оброблюваної інформації.

Проведений аналіз показав, що для більшості практичних застосувань ДПФ, дані, які оброблюються, носять конфідентційний характер. Тому, аби залучати хмарні технології для ДПФ,

необхідно, в першу чергу, вирішити проблему захищеної реалізації ДПФ на непідконтрольних користувачеві обчислювальних потужностях.

Таким чином, наукова задача організації захищеної реалізації ДПФ на віддалених обчислювальних потужностях є важливою та актуальною для сучасного етапу розвитку інформаційних технологій.

Аналіз відомих технологій захищеної віддаленої обробки сигналів

Задачі обробки сигналів являються одними з найбільш масовими задачами комп'ютерних технологій. Значна частина їх виконується в реальному часі. Поява прогресивних технологій обробки інформації в хмарах, які надають для вирішення прикладних задач практично необмежені обчислювальні потужності, стимулювала інтенсивні дослідження в напрямку захищеної реалізації віддаленої обробки сигналів.

Головною проблемою, на вирішення якої направлені виконані до теперішнього часу дослідження, полягає в використанні спеціальних методів шифрування, які б дозволяли отримувати коректний результат шляхом дешифрування результатів віддаленої обробки зашифрованих даних. З наведеного слідує, що не існує універсальних методів шифрування даних перед їх віддаленою обробкою, які не залежать від операцій обробки сигналів. Це означає, що для кожного виду обробки сигналів слід окремо розробляти метод шифрування та дешифрування.

Зокрема, в роботі [1] запропоновано методи захищеної реалізації масових операцій обробки зображень – медіанної та середньоарифметичної фільтрації. В основі цих методів покладено інтервальне шифрування точок зображення перед передачею його для обробки в хмарних системах. Метод забезпечує виконання медіанної фільтрації на віддалених відкритих комп'ютерних системах, закриваючи при цьому доступ до справжнього зображення.

Проведений аналіз показав, що метод інтервального шифрування запропонований в роботі для задач захищеної організації фільтрації на віддалених комп'ютерних потужностях не може бути ефективно застосований для виконання операцій перетворення Фур'є. Специфіка цих операцій вимагає розробки спеціальних методів їх захищеної реалізації.

Важливість реалізації захищеного дискретного перетворення Фур'є та актуальність питань, які вирішуються за допомогою перетворення Фур'є зумовили інтенсивні дослідження цієї тематики.

Зокрема, в роботі [2] пропонується метод шифрування даних, над якими віддалено виконується ДПФ. При цьому, для шифрування даних використовуються алгоритми на основі модулярного експоненціювання. Це дає змогу гнучко регулювати рівень захищеності віддаленої реалізації ДПФ. Крім того, в розробці запропоновані ефективні механізми контролю правильності віддаленого ДПФ на непідконтрольних обчислювальних платформах. Разом з тим, використання в якості механізму шифрування модулярного експоненціювання помітно ускладнює вибір ключів та має наслідком значну обчислювальну складність реалізації ДПФ, яка в декілька разів перевищує складність простої реалізації ДПФ. Крім того, значну складність становить проблема генерації ключів в запропонованому методі, яка потребує також значних обчислювальних ресурсів. Якщо припустити, що ключі використовуються багаторазово, то це призводить до суттєвого зменшення рівня захищеності, через те, що відкриває зловмиснику значно ширші можливості для злому коду шифру. Використання ж одноразових ключів в запропонованій розробці має наслідком витрати значних за обсягом обчислювальних ресурсів, більших, ніж безпосереднє обчислення ДПФ.

Таким чином, основним недоліком відомого методу реалізації захищеного ДПФ на віддалених обчислювальних платформах, є значна обчислювальна складність реалізації ДПФ в хмарі, що не дозволяє виконувати ці обчислення в реальному часі. Крім того значною є складність генерації ключів, яка виконується безпосередньо користувачем.

Ціллю розробки є створення методу захищеної реалізації ДПФ та ШПФ на віддалених розподілених комп'ютерних системах, яка не потребує значних за обсягом обчислювальних ресурсів і дозволяє реалізувати обчислення в режимі реального часу.

Реалізація захищеного ДПФ та ШПФ

Вхідними даними для ДПФ є масив $x_0, x_2, \ldots x_{n-1}$ виміряних значень сигналу через фіксовані проміжки часу. В результаті перетворення отримується вихідний масив n комплексних чисел $y_0, y_2, .., y_{n-1}$, компоненти яких являють амплітуду та фазу синусоїд, сума яких відтворює заданий вхідний сигнал.

$$\forall k \in \{0,1,\ldots n\text{-}1\}:\ y_k = \sum_{l=0}^{n-1} x_l \cdot W_n^{lk}, \qquad (1)$$

де $W_n = e^{-j\cdot\frac{2\pi}{n}}$

Для реалізації захищеного обчислення формули (1) пропонується наступна організація дій користувача:

1. Генерація випадкових комплексних чисел $m_0, m_2, \ldots m_{n-1}$, що являють собою послідовність масок.

2. Формування маскованої послідовності $x'_0, x'_2, \ldots x'_{n-1}$ шляхом додавання послідовності масок до вхідної послідовності:

$$\forall l \in \{0,1,\ldots,n-1\}:\ x'_l = x_l + m_l$$

Отриману послідовність користувач відправляє на хмару, де над цією послідовністю виконується ДПФ, а результуюча послідовність y'_k відправляється назад користувачеві.

3. Формування інверсної послідовності $x''_0, x''_2, \ldots x''_{n-1}$ шляхом віднімання послідовності масок від вхідної послідовності:

$$\forall l \in \{0,1,\ldots,n-1\}:\ x''_l = x_l - m_l$$

Отриману послідовність користувач відправляє на хмару, де над цією послідовністю виконується ДПФ, а результуюча послідовність y''_k відправляється назад користувачеві.

4. Користувач формує вихідну послідовність ДПФ на власній обчислювальній платформі шляхом поелементного обчислення середнього арифметичного отриманих з хмари двох послідовностей y'_k та y''_k.

$$\forall k \in \{0,1,\ldots n\text{-}1\}:\quad y_k = \frac{y'_k + y''_k}{2} \qquad (2)$$

Конструктивність запропонованої організації захищеного обчислення ДПФ може бути доведено наступним чином.

Перша результуюча послідовність y'_k може бути представлена у вигляді:

$\forall k \in \{0,1,\ldots n\text{-}1\}$:

$$y'_k = \sum_{l=0}^{n-1}(x_l + m_l)\cdot W_n^{lk} = \sum_{l=0}^{n-1} x_l \cdot W_n^{lk} + \sum_{l=0}^{n-1} m_l \cdot W_n^{lk}$$

Друга результуюча послідовність y''_k може бути представлена у вигляді:

$\forall k \in \{0,1,\ldots n\text{-}1\}$:

$$y''_k = \sum_{l=0}^{n-1}(x_l - m_l)\cdot W_n^{lk} = \sum_{l=0}^{n-1} x_l \cdot W_n^{lk} - \sum_{l=0}^{n-1} m_l \cdot W_n^{lk}$$

Відповідно, користувач, формуючи середнє арифметичне двох отриманих послідовностей за формулою (2) отримує наступну послідовність:

$\forall k \in \{0,1,\ldots n\text{-}1\}$:

$$y_k = \frac{y'_k + y''_k}{2} = \frac{\sum_{l=0}^{n-1} x_l \cdot W_n^{lk} + \sum_{l=0}^{n-1} m_l \cdot W_n^{lk}}{2} +$$

$$+ \frac{\sum_{l=0}^{n-1} x_l \cdot W_n^{lk} - \sum_{l=0}^{n-1} m_l \cdot W_n^{lk}}{2} = \sum_{l=0}^{n-1} x_l \cdot W_n^{lk}$$

Варто зазначити, що запропонований метод може бути застосований також для реалізації захищеного Швидкого Перетворення Фур'є (ШПФ). Дійсно, у кожної з послідовностей y'_k та y''_k, доданок $\sum_{l=0}^{n-1} x_l \cdot W_n^{lk}$ може бути обчислений за допомогою ШПФ за наступним алгоритмом[3]:

Вхідна послідовність x_l розкладається на дві $\frac{n}{2}$- відлікові послідовності:

$$\chi_l^1 = x_{2l}, \qquad l = 0,1,\ldots,\frac{n}{2}-1$$

$$\chi_l^2 = x_{2l+1}, \qquad l = 0,1,\ldots,\frac{n}{2}-1$$

Тоді (1) можна переписати наступним чином:

$\forall k \in \{0,1,\ldots \frac{n}{2}-1\}$:

$$y_k = \sum_{l=0}^{\frac{n}{2}-1} \chi_l^1 \cdot W_n^{lk} + \sum_{l=0}^{\frac{n}{2}-1} \chi_l^2 \cdot W_n^{lk} = \qquad (3)$$

$$= \sum_{l=0}^{\frac{n}{2}-1} x_{2l} \cdot W_n^{2lk} + \sum_{l=0}^{\frac{n}{2}-1} x_{2l+1} \cdot W_n^{(2l+1)k}$$

Оскільки

$$W_n^2 = \left(e^{-j\cdot\left(\frac{2\pi}{n}\right)}\right)^2 = e^{-j\cdot\left(\frac{2\pi}{\frac{n}{2}}\right)} = W_{\frac{n}{2}}$$

То формулу (3) можна тотожно перетворити:

$\forall k \in \{0,1,\ldots \frac{n}{2}-1\}$:

$$y_k = \sum_{l=0}^{\frac{n}{2}-1} x_{2l} \cdot W_n^{2lk} + \sum_{l=0}^{\frac{n}{2}-1} x_{2l+1} \cdot W_n^{(2l+1)k} =$$

$$= \sum_{l=0}^{\frac{n}{2}-1} x_{2l} \cdot W_{\frac{n}{2}}^{lk} + W_n^k \cdot \sum_{l=0}^{\frac{n}{2}-1} x_{2l+1} \cdot W_{\frac{n}{2}}^{lk} =$$

$$= \sum_{l=0}^{\frac{n}{2}-1} \chi_l^1 \cdot W_{\frac{n}{2}}^{lk} + W_n^k \cdot \sum_{l=0}^{\frac{n}{2}-1} \chi_l^2 \cdot W_{\frac{n}{2}}^{lk}$$

Тобто,

$$\forall k \in \{0,1,\ldots \frac{n}{2}\text{-}1\}: \quad y_k = \gamma_k^1 + W_n^k \gamma_k^2 \qquad (4)$$

де γ_k^1 та γ_k^2 – ДПФ послідовностей χ_l^1 та χ_l^2 відповідно.

Формула (4) визначає y_k тільки для $k \in [0;\frac{n}{2}\text{-}1]$. Необхідно визначити y_k для $k \in (\frac{n}{2}\text{-}1; n\text{-}1]$. Враховуючи, що функції γ_k^1 та γ_k^2 періодичні з періодом $\frac{n}{2}$, а

$$W_n^{k+\frac{n}{2}} = W_n^k \cdot W_n^{\frac{n}{2}} = W_n^k \cdot e^{-j\cdot\left(\frac{2\pi}{n}\cdot\frac{n}{2}\right)} = -W_n^k,$$

то на проміжку $k \in (\frac{n}{2}\text{-}1; n\text{-}1]$ y_k визначається як:

$$\forall k \in \{0,1,\ldots \frac{n}{2}\text{-}1\}:$$

$$y_{k+\frac{n}{2}} = \gamma^1_{k+\frac{n}{2}} + W_n^{k+\frac{n}{2}} \gamma^2_{k+\frac{n}{2}} = \gamma_k^1 - W_n^k \gamma_k^2$$

Тобто:

$$y_k = \begin{cases} \gamma_k^1 + W_n^k \gamma_k^2, k \in [0;\frac{n}{2} - 1] \\ \gamma_k^1 - W_n^k \gamma_k^2, k \in (\frac{n}{2} - 1; n - 1] \end{cases}$$

В свою чергу, кожна з послідовностей γ_k^1 та γ_k^2 також може бути розкладена на дві $\frac{n}{4}$ -відлікові послідовності. Процес зменшення розміру ДПФ може рекурсивно продовжуватися до тих пір, поки не залишаться тільки 2-відлікові послідовності, тобто, остаточно, вхідну послідовність x_l можна розкласти на $\frac{n}{2}$ 2-відлікових послідовності.

Запропонована організація захищеного виконання ДПФ ілюструється наступним прикладом:

Нехай довжина вхідної послідовності складає 8 відліків, які наведені в другому стовпчику таблиці 1. В разі прямого виконання ДПФ за формулою (1), результуюча послідовність має вигляд наведений в третьому стовпчику таблиці 1. Згідно з першим пунктом запропонованого методу, користувач генерує 8 комплексних масок, які містяться в четвертому стовпчику таблиці 1.

Згідно з наступними кроками запропонованого методу, користувач формує масковану послідовність x'_l та інверсну масковану послідовність x''_l (які знаходяться відповідно у другому та третьому стовпчику таблиці 2).

Табл. 1.Цифрова діаграма прямого виконання ДПФ

l	x_l	y_l	m_l
0	5.2	18.90	-2.3 + j·1.1
1	3.7	12.14 + j·15.11	7.1 - j·2.4
2	-6.6	-0.40 + j·5.09	4.8 + j·1.6
3	4.1	6.05 - j·14.68	-3.2 + j·1.9
4	-3.9	-12.89	6.4 + j·4.3
5	1.7	6.05 + j·14.68	3.3 - j·2.5
6	8.3	-0.40 - j·5.10	-5.9 + j·7.2
7	6.4	12.14 - j·15.11	-1.5 - j·0.8

Табл. 2. Цифрова діаграма виконання захищеного ДПФ

l	x'_l	x''_l	y'_l	y''_l	$\frac{y'_l + y''_l}{2}$
1	2.9 + j·1.1	7.5 - j·1.1	27.6 + j·10.39	10.2 - j·10.39	18.9
2	10.8 - j·2.4	-3.4 + j·2.4	3.70 - j·2.11	20.57 + j·32.33	12.14 + j·15.11
3	-1.8 + j·1.6	-11.4 - j·1.6	-1.19 - j·13.4	0.39 + j·23.59	-0.40 + j·5.09
4	0.9 + j·1.9	7.3 - j·1.9	1.05 - j·6.83	11.06 - j·22.54	6.05 - j·14.68
5	2.5 + j·4.3	-10.3 - j·4.3	-15.6 + j·18.1	-10.19 - j·18.1	-12.89
6	5.0 - j·2.5	-1.6 + j·2.5	-14.10 + j·4.11	26.22 + j·25.26	6.05 + j·14.68
7	2.4 + j·7.2	14.2 - j·7.2	10.79 + j·6.6	-11.6 - j·16.80	-0.40 - j·5.10
8	4.9 - j·0.8	7.9 + j·0.8	10.94 - j·7.96	13.33 - j·22.25	12.14 - j·15.11

Далі відбувається виконання ДПФ над отриманими послідовностями на віддалених комп'ютерних системах. Звідти користувач отримує вихідні замасковані послідовності y'_l та y''_l, значення яких наведено у четвертому та п'ятому стовпчиках таблиці 2. Для одержання вихідної послідовності y_l користувач виконує поелементне знаходження середнього арифметичного отриманих послідовностей. Результат цієї операції відображено в останньому стовпчику таблиці 2. Порівнявши останній стовпчик таблиці 2, де знаходяться значення вихідної послідовності, обраховані за допомогою запропонованого методу захищеної реалізації ДПФ та третій стовпчик таблиці 1, де знаходяться значення вихідної послідовності, обраховані прямим ДПФ, видно, що ці значення співпадають.

Для цього ж прикладу можна простежити схему реалізації захищеного ШПФ. Аналогічно з реалізацією ДПФ, користувач створює дві послідовності x'_l та x''_l, шляхом додавання та віднімання послідовності масок m_l до вхідної послідовності x_l. Далі, відповідно до запропонованого методу, відбувається обчислення ШПФ для послідовностей. Схему отримання послідовностей y'_l та y''_l показано на рисунку 1 та на рисунку 2 відповідно.

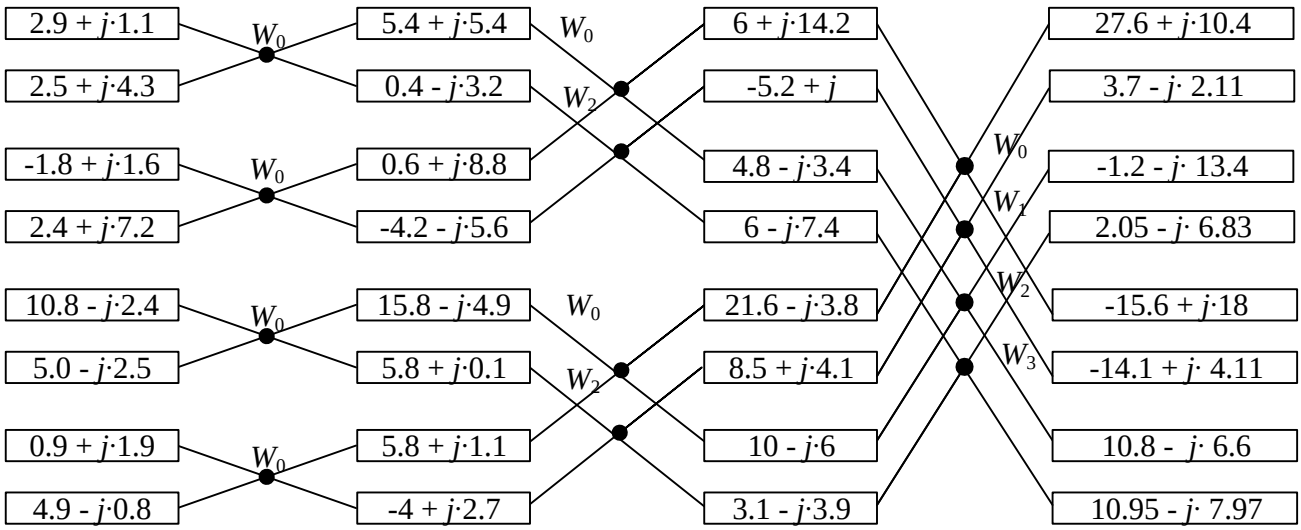


Рис. 1. Схема виконання ШПФ маскованої послідовності

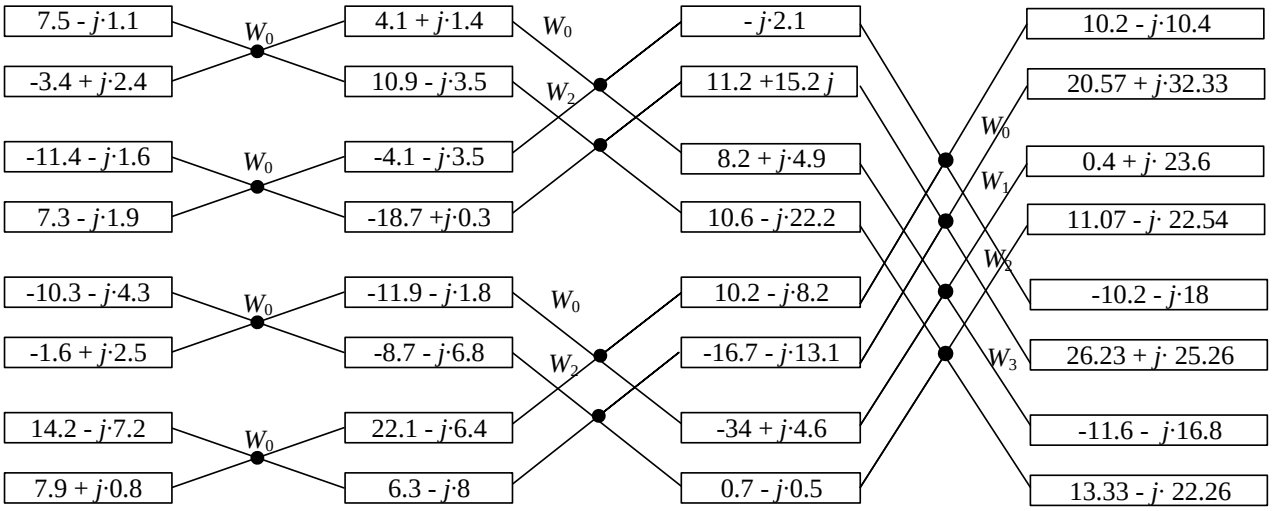


Рис. 2. Схема виконання ШПФ маскованої інверсної послідовності

Оцінка ефективності

Ефективність запропонованого методу захищеної реалізації перетворень Фур'є на віддалених обчислювальних потужностях оцінюється:

- зменшенням обчислювальної складності операцій, що виконуються безпосередньо користувачем на його обчислювальній платформі.
- рівнем захищеності вихідних даних та результатів перетворення Фур'є;

При виконанні ДПФ повністю на обчислювальній платформі користувача, кількість операцій множення, як слідує з формули (1) становить $4 \cdot n^2$, а кількість операцій додавання – $2 \cdot n^2$. Враховуючи, що згідно з [3] тривалість операцій процесорного множення в η раз більша за тривалість операції процесорного додавання, то час T_k виконання ДПФ користувачем може бути представлено у вигляді формули:

$$T_k = \left(4(\eta \cdot n)^2 + 2n^2\right) \cdot \tau_\partial = 4\left(\eta^2 + \frac{1}{2}\right) n^2 \cdot \tau_\partial \approx$$

$$\approx 4\eta^2 \cdot n^2 \cdot \tau_\partial,$$

де τ_∂ – час виконання процесорного додавання.

При реалізації запропонованого методу користувач виконує операції додавання випадкової послідовності (*n* операцій додавання), віднімання випадкової послідовності (*n* операцій додавання), а також обчислення середнього арифметичного одержаних результатів (*n* операцій додавання і *n* операцій зсуву). Оскільки операція зсуву виконується значно швидше операції додавання, то можна вважати, що час виконання користувачем ДПФ $T_k^{'}$ за запропонованим варіантом оцінюється наступним чином:

$$T_k^{'} = 3n \cdot \tau_\partial$$

Таким чином застосування запропонованого методу дозволяє зменшити час виконання ДПФ користувачем в *q* раз, причому чисельне значення *q* визначається наступною формулою:

$$q = \frac{T_k}{T_k^{'}} = \frac{4}{3}\eta^2 \cdot n \qquad (5)$$

Згідно з даними [4] для сучасних процесорів Intel значення η ≈ 10, розмірність – *n* перетворення Фур'є для типових систем обробки зображень становить 10^2. Таким чином, згідно (5), використання запропонованого методу забезпечує прискорення обробки зображень користувачем приблизно на 4 порядки. проведені експериментальні дослідження показали, що реальне збільшення швидкодії перетворення Фур'є становить близько $5 \cdot 10^3$. Це свідчить про високу ефективність застосування запропонованого методу захищеної реалізації перетворень Фур'є на віддалених комп'ютерних системах на основі хмарних технологій.

Основна ціль запропонованого методу полягає в тому, щоб практично унеможливити доступ до вихідних даних та результатів перетворення Фур'є, що виконується на віддалених обчислювальних потужностях.

Несанкціонований доступ до даних користувача може бути реалізовано як на стадії їх передачі по мережі Інтернет, так і безпосередньо на віддалених комп'ютерних системах, що виконують обчислення, пов'язані з перетворенням Фур'є.

Рівень захищеності даних овіється об'ємом обчислювальних ресурсів, потрібних зловмиснику для незаконного доступу до даних.

При застосуванні запропонованого методу для одиночної операції ДПФ або ШПФ вимога забезпечення захищеності операндів та результатів не забезпечується, оскільки зловмисних, перехопивши послідовності x_0', x_1',…,x_{n-1}' та x_0'', x_1'',…,x_{n-1}'' достатньо просто відновить значення вхідної послідовності x_0, x_1,…,x_{n-1} для перетворення Фур'є шляхом поелементного обчислення середнього арифметичного двох перехоплених послідовностей.

Проте аналіз практичних застосувань операції ДПФ та ШПФ для обробки зображень та звукових сигналів [3] показує, що реально виконується потік вказаних операцій. Тобто практично завжди виконується послідовність з h операцій ДПФ або ШПФ. При виконанні вказаного потоку запропонований метод передбачає:

- використання для кожної з h операцій ДПФ або ШПФ окремих послідовностей масок ;
- генерацію користувачем секретних, не співпадаючих між собою послідовності $v_1,v_2,\ldots,v_h$ та $w_1,w_2,\ldots,w_h$, $\forall l \in \{1,2,\ldots,h\}$: $v_l \in \{1,2,\ldots,h\}$, $w_l \in \{1,2,\ldots,h\}$ відправки на віддалені обчислювальні потужності вхідних даних для кожної з h операцій ДПФ або ШПФ. Вказані послідовності визначають порядок отримання результатів користувачем.
- обчислення згідно формули (2) результату l-ї операцій ДПФ або ШПФ виконується користувачем після отримання відповідних результатів від віддалених обчислювальних потужностей.

Відповідно, зловмиснику, для того, щоб відновити вхідні дані потоку операцій ДПФ або ШПФ співставити пару вхідних послідовностей з потоку $2 \cdot h$ посилок даних користувача. Очевидно, що для цього потрібно перебрати, в середньому, $2 \cdot h^2$ пар, для кожної з яких необхідно виконати дві операції ДПФ або ШПФ, що потребує $4 \cdot n^2$ операцій процесорного множення для ДПФ і $2 \cdot n \cdot log_2 n$ операцій процесорного множення для ШПФ.

Таким чином, для отримання незаконного доступу до даних одного перетворення Фур'є, що виконується на віддалених комп'ютерних потужностях, потрібний об'єм ресурсів оцінюється часом виконання $8 \cdot h^2 \cdot n^2$ операцій процесорного множення. Враховуючи, що в реальних системах [3], значення $n \approx 10^3$, а $h \approx 10^4$, то об'єм ресурсів для отримання незаконного доступу оцінюється часом виконання 10^{15} операцій процесорного множення, що для більшості застосувань робить злам запропонованого механізму захисту практично недоцільним.

Висновки

У результаті виконаних досліджень, досягнута поставлена ціль – розроблено оригінальний метод захищеної реалізації потоку операцій ДПФ та ШПФ на віддалених обчислювальних потужностях в рамках хмарних технологій.

Запропонований метод по суті реалізує потокове шифрування даних, що відправляються користувачем на віддалені комп'ютерні системи, де безпосередньо виконуються ДПФ та ШПФ, а також, дешифрування отриманих результатів. Детально розроблено методику застосування запропонованого методу для ДПФ та ШПФ, теоретично доведено конструктивність розроблених процедур.

Головною перевагою запропонованого методу є простота операцій шифрування та дешифрування даних, що забезпечує низький рівень витрат обчислювального часу на виконання операцій, пов'язаних з захистом інформації.

Теоретично доведено, що запропонований метод захищеної реалізації потоку операцій ДПФ та ШПФ забезпечує достатній для задач практики рівень захищеності даних та результатів перетворення Фур'є, що виконуються на віддалених обчислювальних потужностях та передаються по відкритим каналам Інтернет.

Теоретично та експериментально доведено, що використання запропонованого методу забезпечує підвищення швидкості обробки зображень та звукових сигналів на 3-4 порядки.

Запропонований метод може бути ефективно використаний для потокової обробки зображень в реальному часі, зокрема, в системах обробки супутникових зображень, а також в інтелектуальних системах відеоспостереження.

Список посилань

1. Марковський О.П. Захищена реалізації захисту зображень в GRID-системах / О.П. Марковський, М.В. Невдащенко, А.М. Білашевська // Вісник НТУУ «КПІ». Інформатика, управління та обчислювальна техніка – К.: Век+, – 2014. – № 61. – 180 с.
2. Bianchi T. On the Implementation of the Discrete Fourier Transform in the Encrypted Domain/ T. Bianchi, A. Piva, M. Barni // IEEE Transactions on Information Forensics and Security. – 2009. – Vol.4. – P. 86-97.
3. Смит С. Цифровая обработка сигналов. Практическое руководство для инженеров и научных работников / Смит Стивен; пер. с англ. А.Ю.Линовича, С.В.Витязева, И.С.Гусинского. – М.:Додэка XXI, 2012. — 720 с.
4. Брэй Б. Микропроцессоры Intel. Архитектура, программирование и интерфейсы. Шестое издание / Б. Брэй; пер. с анг. А.В.Жукова.- Санк-Петербург: БХВ-Петербург, 2005.-1328 с.

УДК 683.519

ГУСЕВ Е.И.

МОДЕЛИРОВАНИЕ СПОСОБОВ ОРГАНИЗАЦИИ ДОСТУПА К РАСПРЕДЕЛЁННЫМ СТРАНИЦАМ ПАМЯТИ В СИСТЕМАХ ОБЛАЧНЫХ ВЫЧИСЛЕНИЙ ОСНОВАННЫХ НА SHARED EVERYTHING АРХИТЕКТУРЕ

В работе предложена модель для исследования продолжительности конфликтов за распределённые ресурсы и страницы, применяемых в Oracle Real Application Clusters (Oracle RAC). Целью исследования является определение области модернизации существующих способов доступа и нахождение механизма оценки эффективности существующих и предлагаемых способов.

The subject of the article is research model of the of distributed resources and pages conflicts duration which used in the Oracle Real Application Clusters (Oracle RAC). The aim of the paper is to determine the area of modernization of the existing access methods and finding the way for effectiveness evaluation of existing and proposed assess methods.

Ключові слова: Oracle RAC, распределённая страница, организация доступа, общий ресурс;

1.Введение

Использование традиционных способов блокирования в системах облачных вычислений приводит к значительному увеличению времени обработки запроса и как следствие времени блокирования. Такая ситуация приводит к невозможности масштабирования облачной СУРБД при использовании классических способов блокирования общего ресурса без существенного увеличения скорости межузлового обмена [1]. Реализованная в Oracle RAC shared everything архитектура использует наиболее перспективный на сегодня способ обработки общих страниц памяти в облачных СУРДБ при обслуживании OLTP трафика. Для выявления проблем подхода shared everything предлагается использование модели, учитывающую специфику обработки страницы. Для моделирования shared everything архитектуры предлагается использовать 3 компонентную модель, состоящую из модели задержек [2], вероятностной модели и модели конфликтов. Структурно модель задержек реализует особенности аппаратной реализации, вероятностная модель (модель трафиков) – особенности функционирования, т.е. вероятностей прохождения тех или иных сценариев обработки в зависимости от трафика. А модель конфликтов описывает структуру конфликтов и позволяет оценить продолжительности конфликтов в зависимости от реализации и функционирования системы. Исходя из того, что система должна функционировать с различными трафиками и того, что возможности изменения аппаратной реализации системы ограничены – наиболее перспективным, с точки зрения модернизации, является изменение процедуры доступа к странице, которая полностью находится в фокусе модели конфликтов. Для решения этой задачи исследуются основные задержки, вносимые процедурой доступа исходя из модели конфликтов. Поскольку есть возможность снять временные задержки с реально функционирующего Oracle Real Application Clusters, целесообразно оценить адекватность предложенной модели, сравнив её с реально функционирующим Oracle RAC.

2.Модель конфликтов. Двухуровневая структура

Ключевой особенностью процедуры обработки общего ресурса в существующей реализации shared everything является наличие 2х конфликтов – это конфликт за ресурсы и конфликт за страницы содержащие ресурсы. Такой подход позволяет минимизировать количество пакетов пересылаемых при обмене ресурсами, но приводит к эскалации конфликта за ресурсы на уровень страниц. Во многом подход обусловлен наследием необлачных (традиционных) СУРДБ, где кэширование широко использовалось и выигрыш от него был очевиден. В контексте распределённой СУРДБ возникает конфликт за страницы, поскольку несколько узлов, а не один, как в традиционных системах, могут обрабатывать страницу. Кроме того актуальна задача обеспечения когерентности рас-

пределённых страниц. На рис.1 показана классическая процедура доступа к странице, используемая в Oracle RAC [3], и увеличение задержек (толстой сплошной линией) вследствие конфликта при обработке страницы на узле 4.

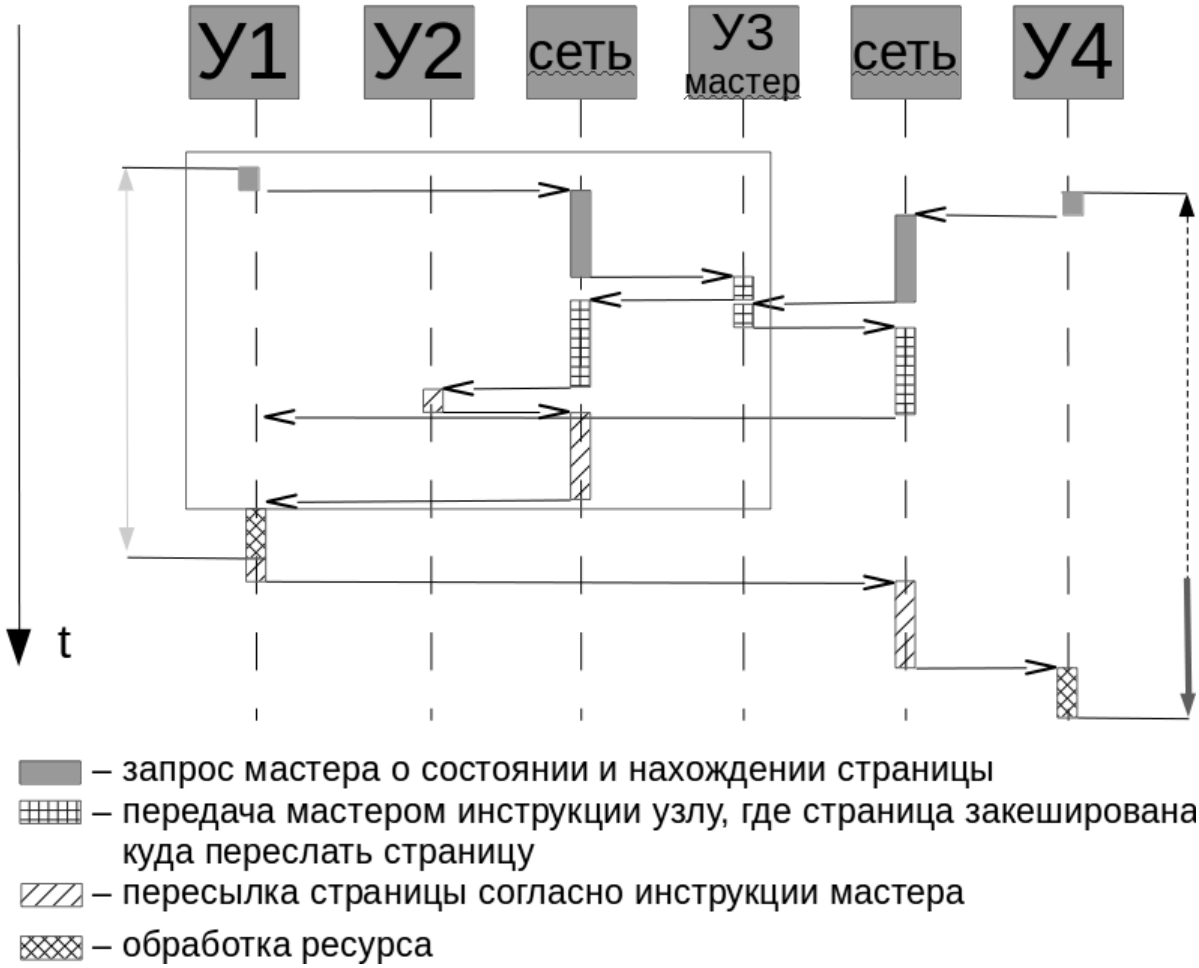


Рис.1 Процедура доступа к странице в Oracle RAC

Моделируя входящий поток доступа к страницам стационарным пуассоновским потоком, получаем отражённое в таблице 1 распределение ряда времён обработки (верхняя строка) и соответствующих им вероятностей (нижняя строка) в зависимости от количества конфликтов:

Таблица 1. Времена обработки и вероятности

$t_{send}/2$	$2t_{send}/2$	...	$nt_{send}/2$
$(\nu_{pl} t_{wpl}) e^{-\nu_{pl} t_{wpl}}$	$\frac{(\nu_{pl} t_{wpl})^2}{2!} e^{-\nu_{pl} t_{wpl}}$	...	$\frac{(\nu_{pl} t_{wpl})^n}{n!} e^{-\nu_{pl} t_{wpl}}$

Время t_{wpl} – это время в течение которого проводится наблюдение. t_{send} – время пересылки страницы между узлами. n – количество конфликтов за время наблюдения, ν_{pl} – интенсивность конфликтов за страницы для исследуемого ресурса (таблицы или индекса). Заметим, что для различных ресурсов интенсивности конфликта за страницы различные, что будет нами в дальнейшем учитываться через индексирование всех ресурсов входящих в транзакцию. Не сложно вычислить среднее время дополнительного ожидания как сумму ряда:

$$t_{p.queue} = \sum_{n=1}^{\infty} \frac{t_{send}}{2} \frac{(\nu_{pl} t_{wtl})^n}{n!} e^{-\nu_{pl} t_{wpl}} = \frac{\nu_{pl} t_{wtl} t_{send}}{2} \quad (1)$$

Нас в первую очередь интересует время в течение которого страницей ресурса может пользоваться одна заявка – т. е. время блокирования страницы. Это сумма t_{send} – времени пересылки и $t_{p.queue}$ – времени ожидания очереди к странице. Получаем уравнение и решаем его:

$$t_{wpl}=t_{send}+\frac{v_{pl}t_{wpl}t_{send}}{2} \quad => \quad t_{wpl}=\frac{2t_{send}}{2-v_{pl}t_{send}} \quad (2)$$

Поскольку время доступа к странице (t_{acc}) состоит из двух времён пересылки служебного пакета (t_{net}), времени пересылки самой страницы (t_{send}) и времени ожидания очереди к странице ($t_{p.queue}$) [2], подставив в формулу доступа $t_{p.queue}$, получаем для традиционного shared everything способа доступа общее время доступа к странице:

$$t_{acc}=2t_{net}+t_{send}+\frac{v_{pl}t_{send}^2}{2-v_{pl}t_{send}}=2t_{net}+\frac{2t_{send}}{2-v_{pl}t_{send}} \quad (3)$$

Рассматривая модель конфликтов за ресурс необходимо отметить, что блокировка за ресурс удерживается до конца транзакции, в отличии от блокировки на страницу, удерживаемую до конца обработки страницы. Это значительно увеличивает время ожидания блокировки за ресурс и как следствие суммарные задержки вследствие конфликтов за ресурс. Расписав ожидаемую длительность транзакции получаем формулу:

$$t_{wtl}=n\,t_{acc}+\sum_{i=1}^{n}t_{(tl.i)}\frac{v_{l.i}}{v_i}+\sum_{i=1}^{n}v_{(l.i)}\left(t_{acc}+t_{tl.i}+t_{queue}\right) \quad (4)$$

где t_{acc} – среднее время пересылок для получения страницы, $t_{tl.i}$ – среднее время ожидания очереди к i-тому ресурсу, t_{tl} - среднее время пересылок для получения ресурса, n – количество ресурсов в транзакции, v_{pl} – интенсивность обращений к i-му ресурсу, а t_{queue} - среднее время ожидания разрешения конфликта. Распишем время ожидания разрешения конфликта:

$$t_{queue}=t_{wtl}\sum_{i=1}^{n}v_i-\sum_{i=1}^{n}\left(\left(1-e^{\left(-v_{(l.i)}t_{wtl}\right)}\right)\left(i\cdot t_{acc}+\sum_{j=1}^{i}t_{tl.j}\right)\right) \quad (5)$$

3. Экспериментальная проверка модели сравнение с реально функционирующим Oracle RAC.

Для проверки адекватности предложенной модели реальному объекту была развёрнута система Oracle RAC состоящая из 4 узлов. Все узлы физически были развёрнуты на одном физическом сервере на платформе VMware, позволяющей эмулировать общий диск. Для обеспечения соотношения времени обработки ко времени пересылки (должно быть много меньше) будем использовать утилиту wondershaper, снизив скорость передачи между узлами до 600 KB/s командой "wondershaper vmnet1 600 600". Пятой виртуальной машиной, также развёрнутой на платформе VMware, будет генерироваться трафик с фиксированной интенсивностью в 32 потока. В процессе эксперимента мы изменяя интенсивность генератора измеряем общее время выполнения 1000000 транзакций, порциями по 10000, для уменьшения влияния времени пересылки между клиентом и сервером. Порции по 10000 сгруппированны в PL/SQL хранимых процедурах. В TPC-C трафике соотношение весов транзакций зафиксируем: New-Order – 41%, Payment – 44%, Order-Status – 5%, Delivery – 5%, Stock-Level – 5%.

На графике (Рис.2) показаны значения возвращаемые моделью и собранные в результате эксперимента: т.е. измеренное время обработки на Oracle RAC. Для удобства отображения приводятся нормированные величины: по оси абсцисс используется интенсивность умноженная на время одной пересылки, а по оси ординат время разделённое на время одной пересылки.

На основе формул 4 и 5, которые составляют систему уравнений, построен график (Рис. 2) зависимости длительности обработки транзакции от интенсивности. В эксперименте использовался элементарный трафик[4], трафик проводок[4] и синтетический тест TPC-C [5] используемый для оценки производительности СУРДБ. Соответствующие интенсивности конфликтов для каждого ресурса в каждом трафике вычисляются на основе модели трафиков (вероятностной модели) предложенной в [4].

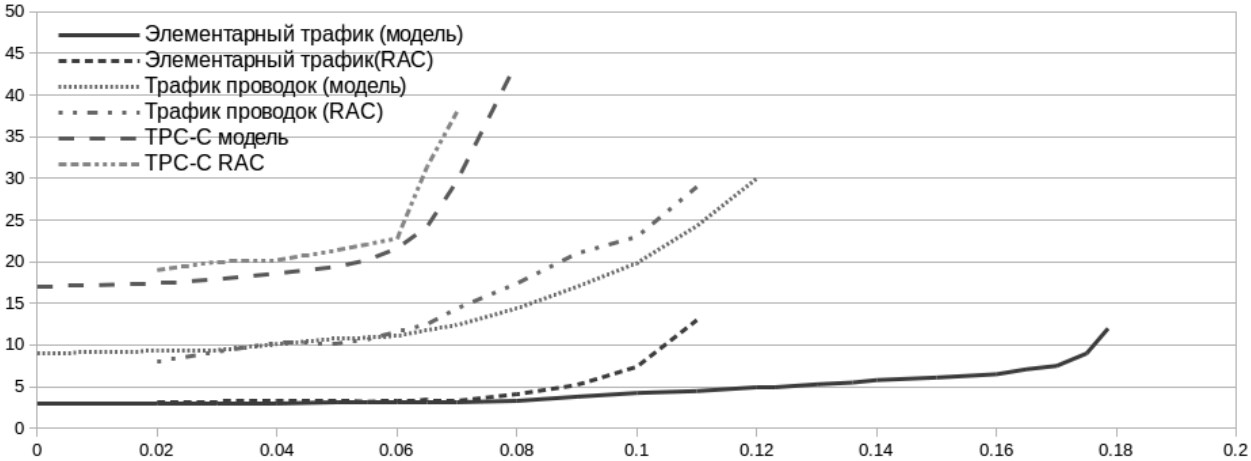


Рис.2 Моделируемое и фактическое время обработки транзакции

4. Оценка адекватности модели

Необходимо отметить, что вносимая погрешность носит не случайный, а систематический характер, поскольку при моделировании мы пренебрегали временем обработки данных по сравнению с временем пересылки, а также задержками при конфликтах за внутренние ресурсы. Под внутренними ресурсами в данной работе понимается очередь к центральному процессору, а также очереди к внутренним конструкциям Oracle, обеспечивающим целостность данных и корректность работы системы. Учитывая систематический, а не случайный характер вносимой погрешности, а также то, что прогнозирующая формула получена аналитическим путём, а не на основе обработки экспериментальной информации, критерии Фишера или Стьюдента, неприменимы. Для оценки адекватности предлагается оценить среднюю относительную погрешность. Однако, как видно из графика, переход к состоянию перегрузки (т.е. резкий рост времени обработки при увеличении интенсивности запросов) в реально-функционирующей системе происходит всегда раньше предсказанного значения. Это обуславливается в первую очередь увеличением времени обработки конфликтов за общие ресурсы (которое моделью игнорируется) при увеличении интенсивности.

В случае трафика проводок и TPC-C трафика реальная интенсивность перехода в перегрузку (коллапс) отличается от предсказанной моделью не более чем на 12%, а в случае элементарного трафика перегрузка (коллапс) возникает не от того, что время обработки транзакции превысило порог перехода в коллапс по ресурсам, а от того, что конфликт за горячие страницы (а в элементарном трафике на транзакцию приходится обработка 1 страницы, а с учётом данных UNDO – 2x страниц) возникает раньше. В первую очередь, такая ситуация является следствием эскалации конфликта за ресурсы на уровень страниц. Рассчитанная по формуле (3) при условии $t_{wpl}<2t_{send}$ интенсивность перехода в коллапс, с учётом нормирования как по оси y, так и по оси y составит 0.118, что также не превышает 12%. В таблице 2 приведены средние значения относительной погрешности в зависимости от выбранного диапазона значений нормированной интенсивности.

5. Выводы

Предложенная математическая модель позволяет с достаточной степенью достоверности определять временные задержки в реальной облачной системе использующей shared everything подход.

Таблица 2. Средняя погрешность в зависимости от трафика и диапазона

Трафик	Диапазон	Средняя относительная погрешность
элементарный	0-0.11	17.3%
	0-0.10	12.9%
	0-0.09	9.9%
Проводок	0-0.11	5.5%
	0-0.10	4.5%
	0-0.09	3.6%
TPC-C	0-0.07	11.9%
	0-0.065	10.4%
	0-0.06	8.3%

Модель позволила выявить такие недостатки организации доступа как эскалация конфликта за ресурсы на уровень страниц. На примере элементарного трафика продемонстрировано, что конфликт за страницы может существенно ограничивать возможности по пропускной способности cloud computing системы. Кроме того, на основе таких метрик как интенсивность перехода в коллапс (граничная интенсивность), среднее время доступа к странице и среднее время обработки транзакции, модель позволяет сравнивать различные способы доступа выявляя ограничения существующих и оценивая эффект от предлагаемых модернизаций. Модель может также применяться и в случае использования традиционных способов снижения вероятности конфликта за страницы (таких как партиционирование [6]), так и способов специфических для облачных систем [7]. Всё это позволяет сделать вывод о целесообразности применения модели при исследовании различных процедур доступа к распределённой странице.

Список литературы

1. Гусев Е.И. Исследование области применения неблокирующего алгоритма фиксации распределённых транзакций – 21 Вісник НТУУ "КПІ".Сер. Інформатика, управління та обчислювальна техніка. – 2012. Випуск 57. – С.76-80
2. Гусев Е.И. Математическое моделирование распределённой кластерной системы использующей shared everything подход (Oracle RAC). - Вісник НТУУ "КПІ".Сер. Інформатика, управління та обчислювальна техніка. - 2014. Випуск 60. - с.106 – 113
3. Markus Michalewicz, Burt Clouse, John McHugh Oracle Real Application Clusters (RAC). – Oracle White Paper . June 2013
4. Гусев Е.И. Моделирование трафиков и оценка скорости распределённого доступа в системах облачных вычислений с общим ресурсом на примере Oracle RAC -- Вісник НТУУ "КПІ".Сер. Інформатика, управління та обчислювальна техніка.- 2015. Випуск 62.
5. TPC BENCHMARK C Standard Specification Revision 5.11, February 2010, Transaction Processing Performance Council (TPC), www.tpc.org , © 2010 Transaction Processing Performance Council
6. Oracle Partitioning with Oracle Database 12c. Efficient Data Management and Performance Acceleration for every System -- ORACLE WHITE PAPER, SEPTEMBER 2014
7. Гусев Е.И. Оптимизация доступа к распределённым страницам памяти в cloud computing системах основанных на shared everything архитектуре используя метод разгрузки очередей. - Проблеми інформатизації та управління, Том 4, № 52 (2015)

УДК 004.9

МУРГА М.О.

ПРОСТІР ЗІ ЗМІННИМ БАЗИСОМ ПРИНАЛЕЖНОСТЕЙ ОЗНАКАМ ДЛЯ ПРОГНОЗУВАННЯ ПОКАЗНИКІВ ФІНАНСОВО-ЕКОНОМІЧНИХ ПРОЦЕСІВ

В роботі розробляється та досліджується структура простору приналежностей ознакам фінансово-економічних процесів, базис якого змінюється з часом, для прогнозування їх показників.

A structure of financial-economic processes' features' memberships space with changing during the time basis for their indicators forecasting is developed and investigated in the paper.

Ключові слова: аналіз фінансово-економічних процесів, нечітка логіка, прогнозування.

Вступ

На даний момент, громадяни України мають можливість вільно торгувати на іноземних та міжнародних фінансових ринках (наприклад, FOREX), що зумовлює потребу в прогнозуванні значень цін їх інструментів Окрім того, практично всі сфери господарства України для планування своєї діяльності потребують прогнозів фінансово-економічних показників. Це призводить до необхідності розробки ефективних методів прогнозування станів фінансово-економічних процесів.Розглядаючи процес прогнозування, як оцінку майбутніх станів процесів на основі їх передісторії, у науковій літературі, значна увага приділена застосуванню лінійних регресійних моделей та нейронних мереж для його проведення.Важливі наукові результати застосування лінійних регресійних моделей до прогнозування фінансово-економічних процесів можна знайти в джерелах [1; 7; 12]. Основною ціллю цих досліджень є побудова ефективних лінійних моделей прогнозування нестаціонарних процесів, які стикаються з трьома проблемами застосування методу найменших квадратів для відновлення коефіцієнтів моделей, що сформульовані в роботі [2] у вигляді вимог до моделей процесів: некоррельованість входів, некоррельованість входів із шумами процесів та шуми повинні бути гаусівськими з нульовим математичним сподіванням. Зазвичай, у реальних моделях фінансово-економічних процесів повністю всі вимоги виконуються не часто. Ці проблеми вирішуються: використанням методів попередньої обробки даних, введенням додаткових складових критеріїв якості, методами підбору структури моделей процесів, аналізом складу їх шумів.

На жаль, більшість процесів є суттєво нелінійними. Окрім того, лінійні моделі описують процес «в середньому», часто ігноруючи коливання з великими амплітудами. Попередня обробка даних – це додатковий елемент системи прогнозування, який збільшує її складність. З цих причин, в роботах [3-11] можна знайти дослідження, напрямлені на застосування нейронних мереж для прогнозування фінансово-економічних показників, яким не притаманні вищевказані обмеження.

До недоліків обох підходів до моделювання фінансово-економічних процесів можна віднести: необхідність перебудови моделей при появі нових факторів, які впливають на процеси; відсутність можливості динамічного видалення з моделей складових, які перестали бути актуальними; хоча нечіткі нейронні мережі і розглядають стан процесу, як перехідних між декількома еталонними, але виконують просте прогнозування динаміки його приналежностей цим станам, не розглядаючи закономірності у їх значеннях.

Для подолання цих недоліків в роботі пропонується підхід до прогнозування фінансово-економічних процесів, який полягає у їх розгляді в просторі приналежності станам, базис якого постійно змінюється, так званим, генератором простору, який неперервно аналізує дані про процеси, які розглядаються.

Простір зі змінним базисом приналежностей ознакам

У порівнянні з технічними процесами, фінансово-економічні характеризуються значно більшою нестаціонарністю динаміки. В той час, коли в більшості технічних систем таку нестаціонарність можна віднести до шумових впливів оточуючого середовища та описати випад-

ковою компонентою моделі процесу з деяким законом розподілу, нестаціонарність фінансово-економічних систем зумовлюється безперервною появою нових факторів, які впливають на них. Ці фактори можуть бути описані як кількісно, так і якісно. З цієї причини доцільно проводити аналіз фінансово-економічних систем, не тільки в просторі станів, а й у просторі приналежностей зі змінним набором базису, схематичне зображення якого можна побачити на Рис. 1.

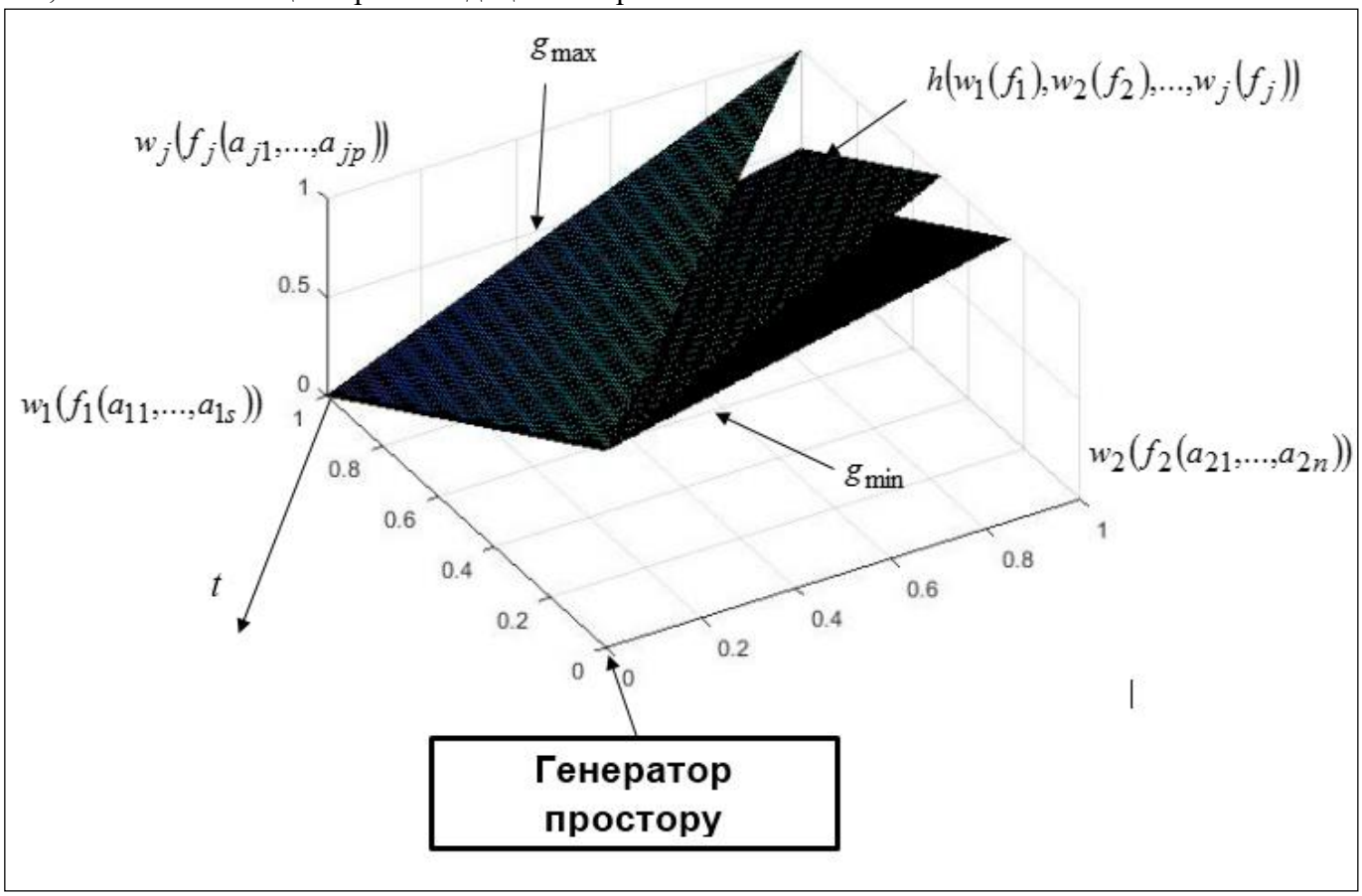


Рис. 1. Схема простору приналежностей зі змінним набором базису

На Рис. 1 використані наступні позначення: w_i – значення належності процесу стану, який описується показником $f_i(a_{i1}, ..., a_{iu})$, що може мати як кількісну так і якісну природу. $h(w_1(f_1), w_2(f_2), ..., w_j(f_j))$ – залежність між значеннями приналежностей показників, яка встановлена на підставі аналізу досліджуваних процесів.

$g_{\min}$ та $g_{\max}$ – верхня та нижні границі більш загальної залежності $g(t, w_1, ..., w_j, f_1, ..., f_j)$, яка обмежує коливання приналежностей w_i показників процесів, між якими встановлена залежність $h(w_1(f_1), w_2(f_2), ..., w_j(f_j), t)$.

Набор $(f_1, ..., f_j)$ не є постійним та неперервно змінюється генератором простору на основі аналізу даних процесів та даних зі зовнішніх джерел.

Слід зазначити, що механізм розрахунку w_i теж повинен змінюватися з часом і визначатися залежністю (1):

$$w_{ti}(f_i) = d_i(t, w_{0i}(f_i), ..., w_{(t-1)i}(f_i), g), \quad (1)$$

де t – поточний момент часу, $w_{0i}(f_i)$ – функція належності показнику процесу після його додавання до базису простору генератором, $w_{\tau i}(f_i)$ – механізм розрахунку значення приналежності показнику процесу f_i в момент часу τ.

Таким чином, до функцій генератору простору можна віднести наступні:

1. Знаходження показників f_i;
2. Визначення w_{i0};
3. Визначення d_i.

Для виконання цих функцій, на даний момент, єдиним шляхом визначення цих елементів уявляється підтримка словників методів.

До елементів словнику визначення f_i можна віднести:

1. Методи класифікації та кластеризації;
2. Методи встановлення взаємозв'язків між показниками процесів: корреляційний (беручи до уваги, що показники можуть бути не обов'язково визначені чисельно) та дисперсійний аналізи.

До елементів словнику визначення w_{io} можна віднести:

1. Гаусівську, трикутну та трапецивидну функції приналежності;
2. S та Z-подібні;
3. Різні метрики відстаней, які можна застосовувати в алгебраїчному просторі.

До елементів словнику визначення d_i можна віднести:

1. Жорстко задана модель в момент додавання показника до базису.
2. На підставі оцінки часу знаходження показника в базисі (можна вважати частковим випадком першого елемента).
3. Схема адаптації генератором.

Задача прогнозування показників процесу, чи процесів, таким чином зводиться до відтворення поверхні $h(w_1(f_1), w_2(f_2), ..., w_j(f_j), t)$ на основі наявних даних, отриманих в попередні проміжки часу та $g(t, w_1, ..., w_j, f_1, ..., f_j)$ - межі, в яких допускаються відхилення реальних значень приналежностей від h. Часте перетинання границь g може бути сигналом генератору про необхідність пошуку нового показника, що впливає на процеси або адаптації базису простору.

Для відновлення h можуть бути використані ті ж методи, які використовуються в просторі станів процесів.

При додаванні елементів до базису простору доцільно використовувати каскадні моделі – це відкидає необхідність повторного відновлення вже визначених залежностей разом з урахуванням нових. Окрім того, при видаленні елементів з базису усувається елемент лише одного з каскадів і немає необхідності перенастройки всієї їх системи.

Для виконання подальших досліджень необхідно отримати ствердні відповіді на наступні питання:

1. Прогнозування у просторі приналежностей не погіршує якість прогнозу у порівнянні із класичними методами прогнозування.
2. Динамічна зміна набору f_i в базисі має сенс.
3. Аналіз $g(t, w_1, ..., w_j, f_1, ..., f_j)$ дозволяє підвищити якість прогнозу.

Ці відповіді даються в наступних розділах роботи.

Прогнозування в просторі приналежностей

Використання розглянутого в попередньому розділі підходу до прогнозування доцільно, коли у ймовірнісному сенсі даний підхід не буде гіршим за традиційне прогнозування у просторі станів з використанням подальшої згортки результатів застосування методів.

Визначення 1. Нехай є n моделей прогнозованого процесу: $f_1, ..., f_n$, яким відповідно зіставлені ступені $w_1, ..., w_n$ приналежності поточного стану процесу класу станів, які описуються даними моделями. Тоді під традиційним підходом до прогнозування буде матися на увазі згортка даних моделей: $f = F(f_1, ..., f_n, w_1, ..., w_n)$.

Приклад 1. $f^1 = \sum_{i=1}^{n} w_i f_i, \sum_{i=1}^{n} w_1 = 1$.

Приклад 2. $f^2 = f_{\arg\max_i w_i}$.

Твердження 1. Ймовірність точної оцінки при прогнозуванні в просторі приналежностей не менше цієї ймовірності за використання традиційного підходу до прогнозування.

Доведення:

◄ Для виконання доведення, необхідно ввести додаткові позначення.

Очевидно, що в прогнозований момент часу t, процес буде не гарантовано належати класам станів, які описуються моделями f_i зі значеннями приналежностей w_i, а з деякими ймовірностями $P(w_i)$.

Нехай w_i' – прогнозовані у просторі приналежностей значення приналежностей процесу класам станів, які описуються моделями f_i.

Процес прогнозування у просторі приналежностей – це $\max P(w_i'), \forall i = \overrightarrow{1, ..., n}$. Таким чином,

$$P(w_i) \le P(w_i'). \qquad (2)$$

Слід відмітити, що, внаслідок можливості існування неврахованих факторів, які впливають на прогнозований процес (головне припущення підходу до прогнозування, який розгля-

дається в роботі), $\sum_{i=1}^{n} P(w_i') \le 1$. Таким чином, формула (2) може бути фізично реалізованою та є несуперечливою.

Використовуючи (2), можна побудувати наступний ланцюжок логічного виводу:

$$P\{f(t)\} = P\{f(t) = F(f_1,..., f_n, w_1,...,w_n)\} = \\ = \prod_{i=1}^{n} P(w_i) \le \prod_{i=1}^{n} P(w_i') = \\ = P\{f'(t) = F(f_1,..., f_n, w_1',...,w'_n)\} = P\{f'(t)\}. \quad (3)$$

В просторі приналежностей існують оцінки не тільки для моделей поточного процесу, а і для потенційно зв'язаних.

Нехай для приналежностей $w_1',...,w_n'$ встановлені залежності зі значеннями приналежностей $z_1,...,z_m$ – моделям інших аналізованих процесів $q_1,...,q_m$, тобто $w_i'' = u_i(z_1,...,z_m, q_1,...,q_m), \forall i = \overrightarrow{1,...,n}$.

Очевидно, що $P(w_i'') = P(u_i) \cdot \prod_{j=1}^{m} P(z_j)$.

Нехай $w_i''' = \arg\max\{P(w_i'), P(w_i'')\}$, тобто

$$P(w_i''') \ge P(w_i'). \quad (4)$$

Тоді має місце наступний ланцюжок міркувань:

$$P\{f'(t)\} = P\{f'(t) = F(f_1,..., f_n, w_1',...,w'_n)\} = \\ = \prod_{i=1}^{n} P(w_i') \le \prod_{i=1}^{n} P(w_i''') = \\ = P\{f'''(t) = F(f_1,..., f_n, w_1''',...,w'''_n)\} = P\{f'''(t)\}. \quad (5)$$

Окрім того, залежності можуть існувати і між w_i', а отже, і між w_i'''.

Нехай виділено дві підмножини індексів: I_1 та I_2 множини індексів оцінок станів w_i''': $I = \{1,...,n\}; I_1 \cup I_2 = I; I_1 \cap I_2 = \varnothing$ таких, що $P(w_i''') \ge p : i \in I_1$ та $P(w_i''') < p : i \in I_2$. Де p - деякий поріг значущості оцінки.

Нехай існують залежності $w_i''' = k_i(w_1''',..., w_{i-1}''', w_{i+1}''',..., w_n''')$ з імовірностями $P\{k_i\}$.

Нехай $\overline{k}_i(w_1''',..., w_j''',...)_{j \in I_1} = k_i(..., w_j''',...)_{j=\{1,...,n\}/i}$ з імовірностями $P\{\overline{k}_i\}$.

Тоді для $i \in I_2$:

$\overline{w}_i = \arg\max\{P(w_i'''), P\{\overline{k}_i\} \cdot \prod_{i \in I_1} P(w_i''')\}$, а отже:

$$P(w_i''') \le P(\overline{w}_i). \quad (6)$$

Нехай $\overline{w}_i = w_i'''$ для $i \in I_2$, тоді, враховуючи (6), справедливим є наступний ланцюжок логічного виводу:

$$P\{f'''(t)\} = P\{f'''(t) = F(f_1,..., f_n, w_1''',...,w'''_n)\} = \\ = \prod_{i=1}^{n} P(w_i''') \le \prod_{i=1}^{n} P(\overline{w}_i) = \\ = P\{\overline{f}(t) = F(f_1,..., f_n, \overline{w}_1,...,\overline{w}_n)\} = P\{\overline{f}(t)\}. \quad (7)$$

Таким чином, $P\{f(t)\} \le P\{\overline{f}(t)\}$, що і треба було довести.►

У необхідності прогнозування в просторі приналежностей можна переконатися і експериментально. Для проведення експерименту, який підтверджує це було взято денні ціни закриття тайм-фреймів за період з 1.06.2016 по 22.06.2016 валютних пар EUR/USD та GBP/USD на FOREX.

Часові ряди валютних пар були приведені до масштабу $[0;1]$, після чого були отримані прирости цін між тайм-фреймами.

З отриманих двох часових рядів була створена вибірка даних наступної структури: кожна реалізація відповідала поточному часу і для неї було взято поточні значення приростів для обох часових рядів та по два попередні.

Для отриманої вибірки даних було застосовано процедуру кластеризації методом різницевого групування для отримання оптимальної кількості кластерів – було виділено п'ять.

Базис простору приналежностей складався з оцінки приналежностей поточної реалізації утвореної вибірки кожному з кластерів. Для отримання значень приналежностей, обчислювалася величина обернена до кореню квадратного з Евклідової відстані між поточною реалізацією вибірки та центрами кластерів. Отримані приналежності ділилися на їх суму для забезпечення єдиного масштабу та їх суми рівній одиниці (як буде видно далі, це дозволяє краще побачити окремі результати експерименту).

На Рис. 2 та Рис. 3 можна побачити зміну значень належності реалізацій вибірки двом обраним кластерам в залежності від номеру тайм-фрейму.

Рис. 2. Графік залежності значення приналежності реалізації вібірки кластеру в залежності від номеру тайм-фрейму

Рис. 3. Графік залежності значення приналежності реалізації вібірки кластеру в залежності від номеру тайм-фрейму

Перший кластер було обрано серед інших через чіткі закономірності, які можна побачити на Рис. 2, а другий (Рис. 3) – через те, що на ньому можна побачити динаміку подібну до інших кластерів.

З Рис. 2 видно існування «пар» зубців в динаміці значень приналежностей і цей факт недоцільно нехтувати при виконанні прогнозу. Окрім того, в динаміці показника видно певний тренд.

З Рис. 3 видно, що показник має тренд та амплітуди його коливань менші ніж у показника з Рис. 2.

Якщо ж для всіх п’яти кластерів побудувати лінійні тренди приналежностей ним, то можна побачити картину, представлену на Рис. 4.

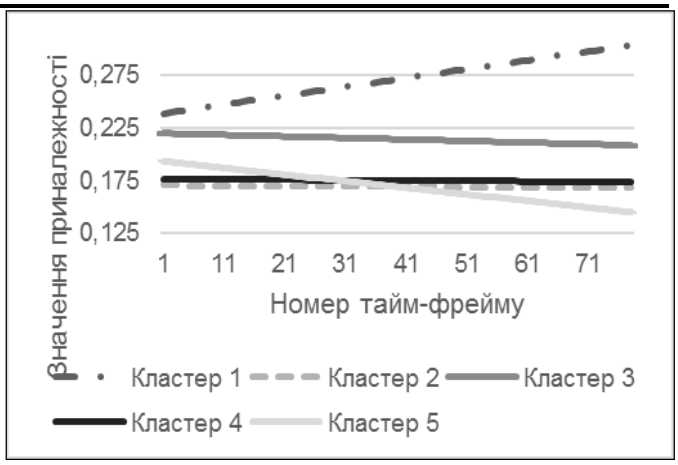


Рис. 4. Графіки лінійних трендів динаміки значень приналежностей реалізацій вибірки кластерам

Як видно на Рис. 4, в динаміці приналежностей прослідковуються значні тренди, які, по суті, нехтуються нечіткими системами та згортками методів.

Ще слід зауважити, що, оскільки нечіткі системи та згортки методів орієнтуються на значення належностей у поточний момент часу, вони слідують за коливаннями, які можна побачити на Рис. 2 та Рис. 3. На цих рисунках видно, що при великому викиді зазвичай йде повернення значення показника до його тренду з такою ж або більшою амплітудою. І, в той час, коли при прогнозуванні наступних значень, перевага буде надана одній з моделей, необхідно було б надати її іншим. Якщо моделі дають близькі результати – цей ефект буде непомітно, але, якщо протилежні, то, напевне, найбільш раціональною стратегією буде дотримання трендів значень приналежностей та використання звичайного простору станів буде недостатньо.

Якщо ж відкинути вісь часу та обрати три кластери (було обрано: перший, другий, п’ятий) і побудувати діаграму значень приналежностей ним у тривимірному просторі (див. Рис. 5), то можна побачити чітку закономірність.

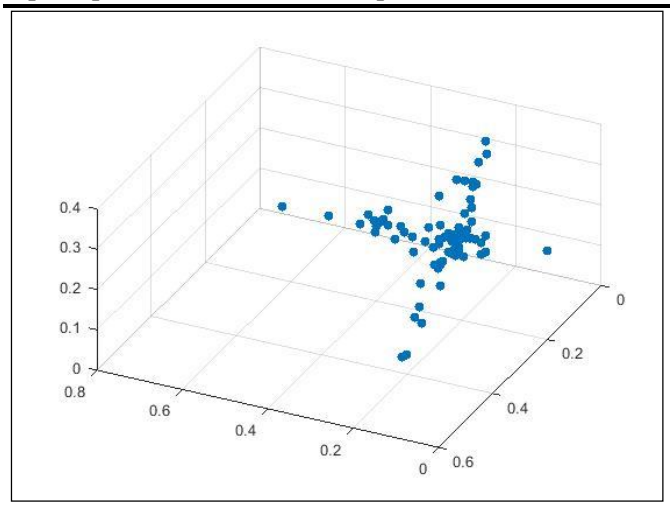


Рис. 5. Значення приналежностей кластерам 1,2 та 5 для реалізацій вибірки

З Рис. 5 видно, що значення приналежностей для реалізацій вибірки не розміщуються хаотично на симплексі, а утворюють деяку поверхню, відновлення якої повинно стабілізувати процес прогнозування.

Динамічна перебудова базису простору приналежностей

Те, що врахування додаткових факторів підвищує якість прогнозу є очевидним фактом. Для прикладу, слід відзначити, що при проведенні дослідів виключно з парою EUR/USD, графіку приналежності подібного випадку до приведеного на Рис. 2 не спостерігалось.

Але зазвичай цей факт ігнорується жорстким заданням показників, що входять до складу моделі з подальшим віднесенням всього іншого до шумової складової, яка в ідеальному випадку повинна бути нормально розподіленою. Також робляться спроби – подібні до [9] – працювати з кожною складовою спектра шуму окремо з подальшою агрегацією результатів.

Дійсно, статистичні характеристики реальних процесів можуть бути виявлені, але такі структурні особливості, які можна побачити на Рис. 2 – наврядче.

Особливістю фінансово-економічних процесів є вплив на них інших процесів, які можуть активувтися час від часу (наростання кризи в одній з країн), або бути одноразовими подіями (вихід новин), які можуть навіть не мати числову форму.

Нехтувати ці процеси та події не можливо, оскільки вони суттєво впливають на динаміку прогнозованого процесу, а щоразу перенастроювати модель при їх додаванні – не доцільно.

З цієї причини, оскільки структура простору приналежності дозволяє це робити просто, пропонується динамічно додавати показники (їх функції належності) до базису, не перенастроюючи модель цілком, а використовуючи каскадування ([4]) моделей.

Обгортка процесу

Границі значень $g(t, w_1,..., w_j, f_1,..., f_j)$ визначають «обгортку» G процесів в просторі приналежностей. Така обгортка може бути побудована, як це показує візуальний аналіз Рис. 5.

Коректування виходу значень за межі G може згладити дійсно шумові викиди процесів, а також бути сигналом про появу нового показника, який почав впливати на процеси, що прогнозуються.

Якщо виходів за межі G почало відбуватися дуже багато, то доцільно створити нову вісь приналежностей в просторі приналежностей, в котрій значення 0 зіставити значенням невідомого показника, які не виходять за межі G, а 1 – виходять та почати будувати новий каскад з її використанням.

В той же час, новий впливаючий показник можна виявити, перевіряючи внутрішні та зовнішні дані за допомогою дисперсійного аналізу. Для цього його значення діляться на дві множини, що співпадають у часі зі значеннями 0 та 1 попередньо введеної вісі простору приналежностей на основі обгортки, та, в разі встановлення впливу, вводиться нова вісь значень приналежностей показнику, що аналізувався.

Таким чином, за допомогою G можна визначити один з механізмів генератора простору приналежностей.

Напрямки подальших досліджень

Для виявлення граничних можливостей розглянутого підходу до прогнозування та розробки інструментарію його застосування у подальшому необхідно провести такі дослідження:

1. Оцінка доцільності застосування традиційних моделей прогнозування в просторі приналежностей.

2. Виділення ефективних типів поверхонь для апроксимації залежностей між значеннями в просторі приналежностей.

3. Виділення ефективних елементів словників генератору простору.

4. Аналіз ефективності каскадування моделей в просторі приналежностей та пошук альтернативних шляхів модифікації моделей простору приналежностей.

Висновки

Робота присвячена актуальній темі прогнозування показників фінансово-економічних процесів. В ході її виконання були отримані наступні результати:

1. Розроблено, обґрунтовано та досліджено підхід до прогнозування показників фінансово-економічних процесів, який використовує його аналіз в просторі приналежностей зі змінним базисом.

2. Визначено, що в просторі приналежностей існують закономірності, які не враховуються традиційними методами прогнозування.

3. Розроблено механізм додавання нових складових до базису простору приналежностей.

Список посилань

1. Бідюк П.І., Романенко В.Д., Тимощук О.Л. Аналіз часових рядів (навчальний посібник) – К.: Політехніка, 2010. – 317 с.
2. Браммер К., Зиффлинг Г. Фильтр Калмана-Бьюси: детерменированное наблюдение и стохастическая фильтрация – пер. с нем. – М.: Наука. Главная редакция физико-математической литературы, 1982. – 200 с.
3. Згуровский М.З., Зайченко Ю.П. Модели и методы принятия решений в нечётких условиях. – К.: Наукова Думка, 2011. – 279 с.
4. Згуровский М.З., Зайченко Ю.П. Основы вычислительного интеллекта. – К.: НВП "Видавництво "Наукова думка" НАН України", 2013. – 408 с.
5. Зайченко Ю.П. Нечёткие модели и методы в интеллектуальных системах. – К.: «Издательский Дом «Слово», 2008. – 344 с.
6. Зайченко Ю.П. Основи проектування інтелектуальних систем. Навчальний посібник. – К.: Видавничий Дім «Слово», 2004. – 352 с.
7. Зайченко Ю.П. Линейное оценивание параметров моделей финансовых временных рядов/ Ю.П. Зайченко, Н.А. Мурга// Applicable Information Models. – Sofia, Bulgaria:ITHEA, - 2011. – Vol. 22. – P.76-82.
8. Мурга М.О. Модифікація методу навчання радіальної базисної мережі та її порівняння з нечіткою нейронною мережею Мамдані та ННМАННКП/ М.О. Мурга// Вісник НТУУ «КПІ». Інформатика, управління та обчислювальна техніка: Зб. наук. пр. – К.: Век+, - 2010. – № 53. – С. 123-133.
9. Мурга Н.А. Нечёткий логический вывод Херста/ Н.А. Мурга// Вісник Черкаського державного технологічного університету – 2010. – № 4. – С. 36-41.
10. Мурга Н. Нейросетевая архитектура на частичных обучениях/Н. Мурга//New Trends in Classification and Data Mining. – Sofia, Bulgaria: ITHEA, – 2010. – Vol.16 – P.170-184.
11. Мурга Н.А. Применение нечётких нейронных сетей к задаче анализа поведений котировок валют/Н.А. Мурга // Вісник НТУУ «КПІ». Інформатика, управління та обчислювальна техніка: Зб. наук. пр. – К.: Век+, – 2010. – № 52. – С. 71-78.
12. Снитюк В.Є. Прогнозування. Моделі. Методи. Алгоритми: Навчальний посібник. – К.: “Маклаут”, 2008. – 364 с.

УДК 004.724.4(045)

ДИБРОВА М. А.,
КОГАН А. В.,
КАПОРИН Р. М.

СПОСОБЫ ОРГАНИЗАЦИИ МНОГОПУТЕВОЙ МАРШРУТИЗАЦИИ

В работе предложены модифицированный алгоритм Беллмана- Форда и потоковый алгоритм многопутевой маршрутизации, которые позволяют улучшить пропускную способности сети и обеспечить ее безопасность. Проведен сравнительный анализ данных алгоритмов и определена область их применения. Модифицированный алгоритм Беллмана-Форда целесообразно использовать для внутридоменной маршрутизации при относительно большом количество узлов, Модифицированный волновой алгоритма целесообразно использовать при небольшом количестве узлов, например, при междоменной маршрутизации.

In this paper we propose a modified algorithm Bellmana- Ford and streaming multipath routing algorithm, which can improve network capacity and ensure its safety. A comparative analysis of these algorithms and identify areas of their application. Modified Bellman-Ford algorithm should be used for intra-domain routing in a relatively large number of nodes, the modified wave algorithm is useful when a small number of nodes, such as inter-domain routing.

Ключевые слова: многопутевая маршрутизация, волновой алгоритм, алгоритм Беллмана-Форда

Введение

Расширение области применения компьютерных сетей повышает требования к качеству (*QoS*) передачи информации, таких как передача голоса по *IP* (*VoIP*), потоковое видео и др. Многопутевая маршрутизация один из перспективных способов решения данной задачи. Большинство известных протоколов маршрутизации ориентированы на формирования одного наилучшего пути для передачи данных.

Основным условием организации безопасной многопутевой маршрутизации является наличие достаточного количества путей необходимых для передачи данных и возможного обхода скомпрометированных или вышедших из строя вершин. Для этого в сети, необходимо сформировать множество непересекающихся путей [1, 2, 3], что позволит уменьшить вероятность прослушивания и ускорит процесс передачи за счет использования нескольких путей. Использование многопутевой маршрутизации предусматривает ряд преимуществ, а именно:

- Балансирование нагрузки
- Контроль полосы пропускания
- Улучшение показателей *QoS*, таких как задержка, и т.д.

Решение поставленной задачи

Известные способы формирования множества путей базируются на алгоритмах поиска в ширину, глубину или использования комбинированного способа. Наиболее распространённый алгоритм поиска в ширину - это волновой алгоритм [4], который позволяет сформировать кратчайший путь на графе. Формирование множества путей, используя данный алгоритм, влечет за собой значительное увеличение вычислительной сложности которая близка к $O(N^2)$.

В работе [5] описан подход теории игр (*FDG - forwarding dilemma game*) для того, чтобы передать пакеты множественной рассылкой в беспроводных специальных сетях. У этого подхода есть два преимущества: во-первых, узлы используют *FDG*, чтобы вычислить вероятность передачи адаптивно, во-вторых, в отличие от иерархических или кластеризирующихся методов, предложенная модификация не вызывает дополнительные издержки маршрутизации. Этот способ может быть использован к большинству классов протоколов маршрутизации, у которых есть множественная рассылка.

В работе [6] предложен модифицированный метод «ветвей и границ» [7], позволяющий формировать множество непересекающихся путей между двумя узлами сети, позволяющий исключить операций направленного перебора, характе-

рного для комбинаторных алгоритмов формирования путей. С помощью данного алгоритма строиться дерево путей из начальной вершины до конечной. Основным недостатком данного алгоритма, как и всех других алгоритмов волновых алгоритмов и алгоритмов поиска в глубину или ширину, является достаточно большая область поиска решений.

В связи с этим, в рамках данной работы был предложен модифицированный волновой алгоритм адаптирован под многопутевую маршрутизацию (рис. 1). Особенностью данного модифицированного алгоритма является возможность сформировать все возможные непересекающиеся пути.

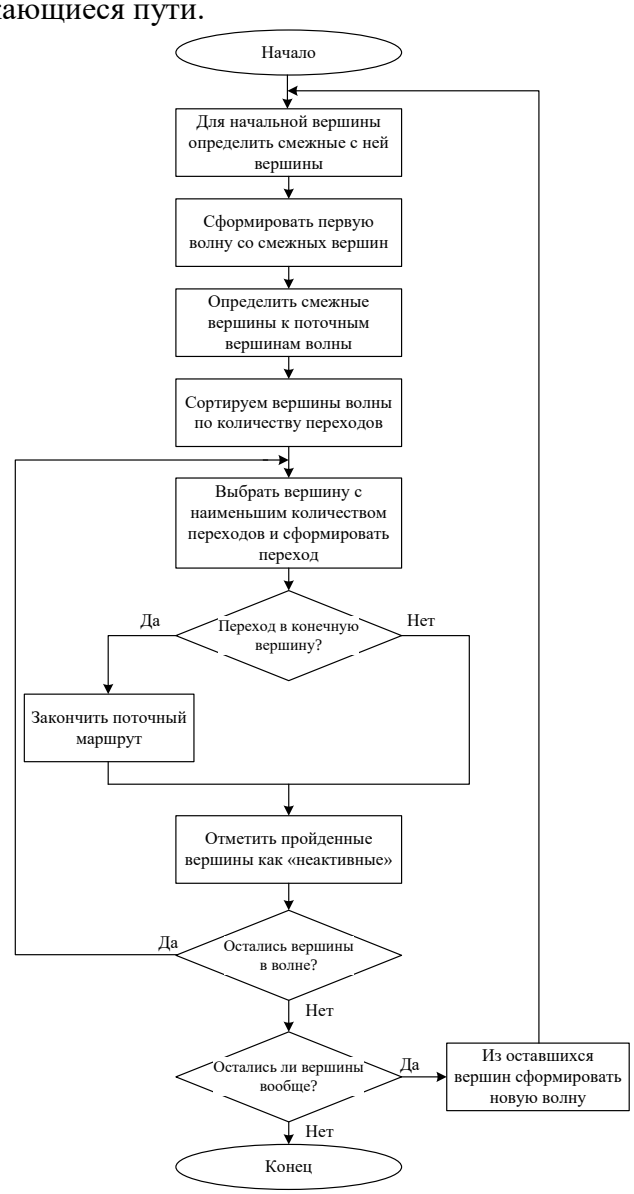


Рис. 1 – Блок схема модифицированного волнового алгоритма

Рассмотрим пример модифицированного волнового алгоритма формирования множества путей между начальной вершиной *V1* и конечной вершиной *V16* (рис.2). Основной задачей предложенного алгоритма является формирования максимально возможного множества $K_1 \cup K_2 \cup \cdots \cup K_t = K$ непересекающихся безопасных путей $|K| \to \max$ между двумя вершинами. Основным критерием формирования множества путей это отсутствие общих узлов кроме начальной и конечной вершины

$$K_1 \cap K_2 \cap \cdots \cap K_t = \varnothing.$$

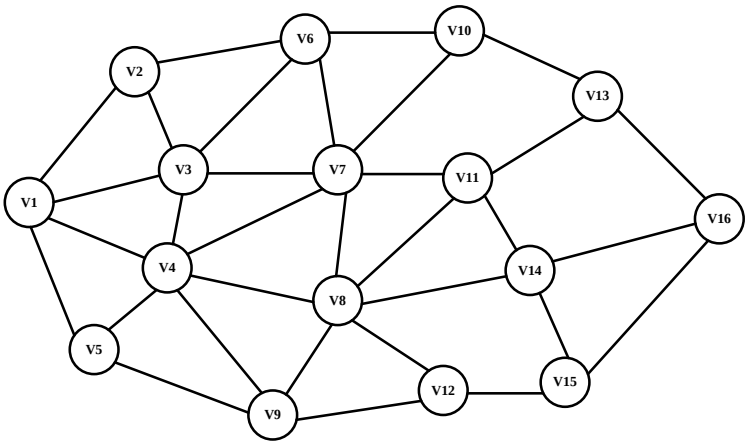


Рис. 2. Граф сети

На начальном этапе строится дерево из начальной вершины и смежных с ней вершин. Например, для графа, представленного на рис. 2, для начальной вершины *V1* смежными являются вершины *V2, V3, V4, V5*. Таким образом на каждом шаге формируются своеобразные волны (рис. 3)

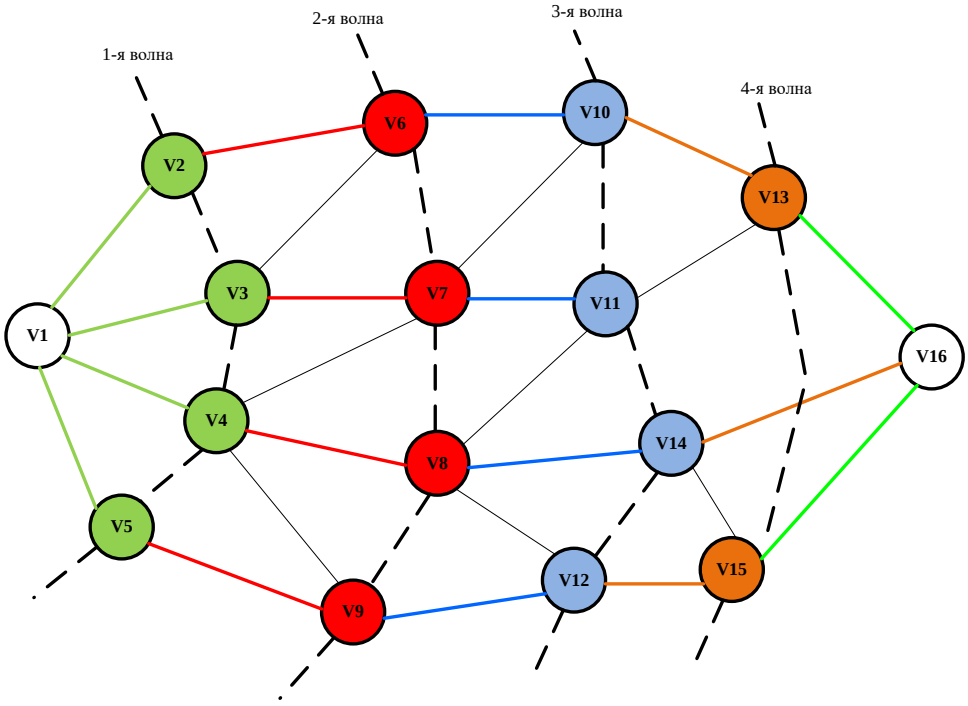


Рис. 3 Формирование «волн»

Основной метрикой данного алгоритма является стоимость C_{ij}, где индексы ij это начальная и конечная вершины. Условие минимальной стоимости сформированого пути определяется как:

$$\sum_{i}^{m}\sum_{j}^{m} c_{ij}x_{ij} \to \min.$$

где X_{ij} - переход с вершины i в вершину j. Представим переходы и их стоимости в виде матрицы

$$C = \begin{pmatrix} - & c_{ij} & c_{ij} & \infty & \infty & \infty \\ c_{ij} & - & c_{ij} & c_{ij} & \infty & \infty \\ c_{ij} & c_{ij} & - & c_{ij} & c_{ij} & \infty \\ \infty & c_{ij} & c_{ij} & - & c_{ij} & c_{ij} \\ \infty & \infty & c_{ij} & c_{ij} & - & c_{ij} \\ \infty & \infty & \infty & c_{ij} & c_{ij} & - \end{pmatrix}$$

В результате сформировано три непересекающихся пути:

P1={V1, V2, V6, V10, V13, V16} – путь сформирован;

P2={V1, V3, V7, V11} – путь не сформирован;

P3={V1, V4, V8, V14, V16} – путь сформирован;

P4={V1, V5, V9, V12, V15, V16} – путь сформирован.

В рамках данной работы предложен комбинированный способ формирования множества путей, а именно модифицированный алгоритм Беллмана-Форда. Такой способ гарантирует поиск всего множества непересекающихся путей с наименьшей стоимостью. Блок схема алгоритма работы представлена на рис. 4.

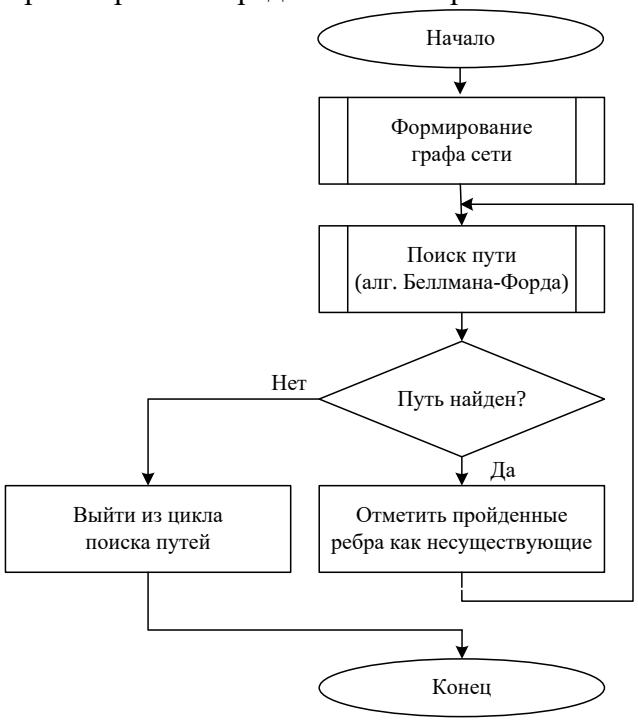


Рис. 4. Блок схема модифицированного алгоритма Беллмана-Форда

Для достоверных результатов была разработана моделирующая программа (рис.5) с помощью которой были проведены эксперименты, на графах различной размерности. Время фор-

мирования множества непересекающихся путей используя предложенные модифицированные алгоритмы представлена на рис.6. Программа разработана на языке программирования *Ruby* с использованием графического интерфейса *Shoes 3.2*.

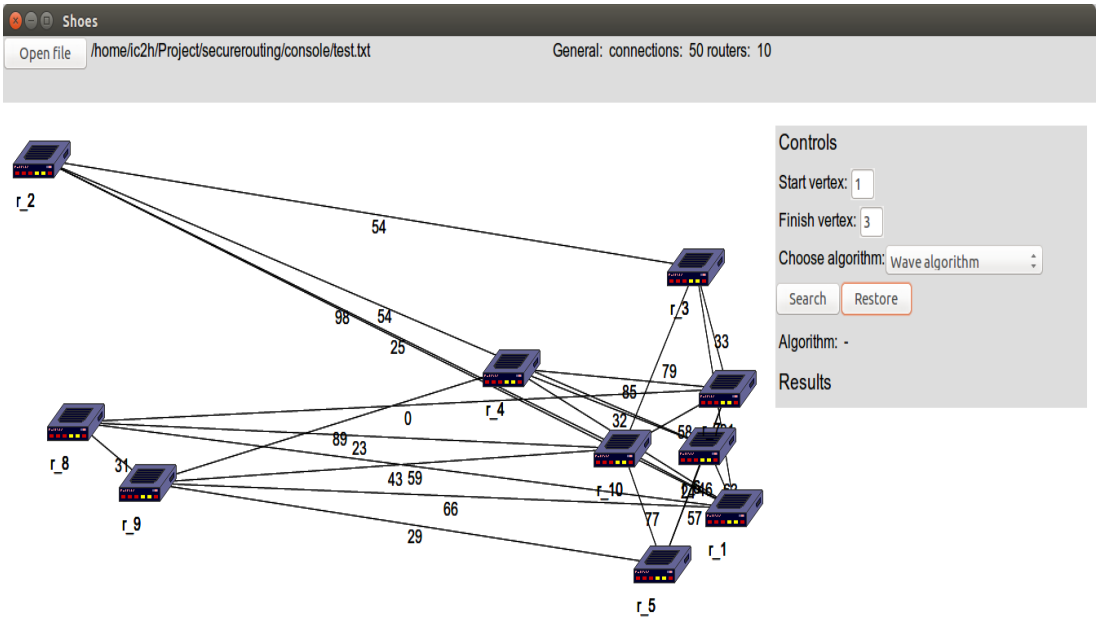


Рис. 5. Интерфейс моделирующей программы

Табл. 1. Время формирования путей

Алгоритмы	Количество вершин				
	10	20	50	100	200
	Время, за которое было сформировано пути, (с)				
Беллмана – Форда (мод.)	0,000106	0,000477	0,001369	0,005309	0,013547
Волновой (мод.)	0,000140	0,000517	0,002590	0,013465	0,047211

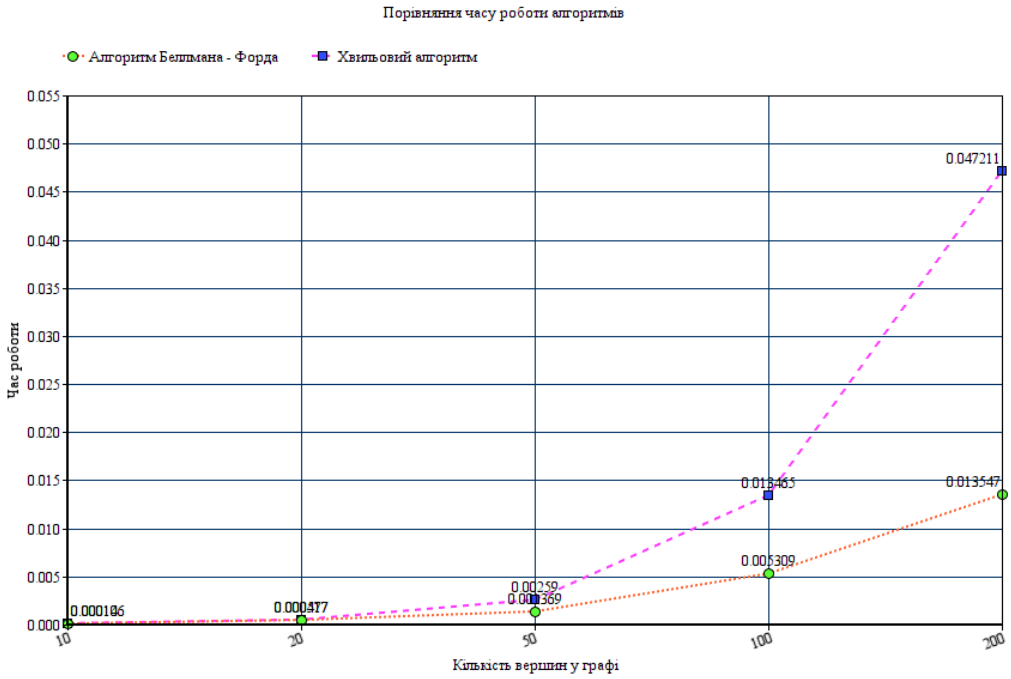


Рис. 6. Анализ работы алгоритмов

По результатам эксперимента можно сделать вывод, что оба способа адаптивны к многопутевой маршрутизации и формируют множество непересекающихся путей. Модифицированный алгоритм Беллмана-Форда подходит для внутридоменной маршрутизации, где достаточно большое количество узлов. Модифицированный волновой алгоритм лучше применять при междоменной маршрутизации, так как он показал достаточно хороший результат при небольшом количестве узлов.

Выводы

В данной работе были рассмотрены способы формирования множества путей между узлом-источником и узлом назначения. В результате проведенного анализа было определенно, что большинство способов не подходят для многопутевой маршрутизации. Основным критерием при организации многопутевой маршрутизации является безопасность, достичь которой удается за счет формирования множества непересекающихся путей, которые позволят повысить уровень безопасности в сети. В рамках данной работы были предложены и обоснованы два модифицированных способа формирования множества непересекающихся путей и определены их области применения.

Список литературы

1. Кулаков Ю. А. Формирование множества непересекающихся путей между граничными маршрутизаторами сети MPLS / Кулаков Ю. А., Диброва М. А., Коган А. В. // Electronics and Communications. Електроніка та зв'язок. Електроніка та зв'язок. – 2016. – Том 21. – №1(90), – С.50-55.
2. Deepak Ganesan. Highly-Resilient, Energy-Efficient Multipath Routing in Wireless Sensor Networks / Deepak Ganesan, Ramesh Govindan, Scott Shenker, Deborah Estrin // Mobile Computing and Communications Review, 2014, Volume 1, Number 2. – P.1-13.
3. Aboli Arun Anasane. A Survey on Various Multipath Routing Protocols in Wireless Sensor Networks / Aboli Arun Anasane, Prof. Rachana Anil Satao// Proceedings of International Conference on Communication, Computing and Virtualization (ICCCV) 2016. - Volume 79, 2016. – P. 610-615
4. Keijo Ruohonen. Graph theory. – 2013. – 114p.
5. Mohammad Naserian. Game theoretic approach in routing protocol for wireless ad hoc networks / Mohammad Naserian, Kemal Tepe // Ad Hoc Networks. - № 7, 2009. – P. 569–578.
6. Кулаков Ю. А. Способ организации многопутевой маршрутизации с помощью модифицированного метода ветвей и границ / Кулаков Ю. А., Коган А. В., Морозовский Т. О. // Вісник НТУУ «КПІ». Інформатика, управління та обчислювальна техніка: збірник наукових праць. – К.: Століття +, 2015. – №62. – С.27-31.
7. Ю. Л. Костюк. Эффективная реализация алгоритма решения задачи коммивояжёра методом ветвей и границ / Ю. Л. Костюк // Вычислительные методы в дискретной математике. – 2013, №2(20). С. 78-90.

УДК 004.2

ЖАБИН В.И.,
ЖАБИНА В.В.,
СКОРИЧЕНКО А.В.

РЕАЛИЗАЦИЯ НЕАВТОНОМНЫХ ВЫЧИСЛЕНИЙ В ИЗБЫТОЧНЫХ СИСТЕМАХ СЧИСЛЕНИЯ НА ПЛИС

Показана возможность реализации неавтономных вычислений с плавающей запятой в вычислительных системах с непосредственными связями между вычислительными модулями. Предложен метод вычисления полиномов с поразрядной обработкой операндов со старших разрядов в симметричных избыточных системах счисления. Использование указанного метода позволяет расширить диапазон представления чисел и сократить необходимый ресурс ПЛИС благодаря малому числу связей между модулями.

Possibility of realization of on-line floating point computations in systems with direct connections between computation units is shown. Method of polynomial calculation with the use of digit-by-digit operand processing from left-hand digits in symmetrical redundant numeric systems is proposed. Implementation of the above-mentioned method enables to extend the range of numbers representation as well as to save FPGA resources due to small number of connections between units.

Ключевые слова: вычисление полиномов, неавтономный режим, поразрядная обработка, ресурсы ПЛИС.

Введение

Особенности работы вычислительных систем в контуре управления объектами и процессами во многих случаях требуют обеспечения высокой скорости реализации алгоритмов с мелкозернистой структурой [1, 2]. Такую структуру имеют, например, алгоритмы траекторных задач, когда необходимо решать системы уравнений, выполнять полиномиальную интерполяцию функций, преобразование координат в многомерном пространстве и т.д. Как известно, для ускорения вычислений применяются параллельные вычислительные системы, обеспечивающие совмещение обработки данных в параллельных ветвях алгоритмов на различных уровнях параллелизма. Для мелкозернистых алгоритмов эффективным может быть преобразование информации на низком уровне параллелизма, например, на уровне обработки машинных слов, то есть выполнения отдельных операций. По сравнению с более высоким уровнем параллелизма это предполагает увеличение числа параллельных ветвей.

Однако, скорость обработки информации зависит не только от времени выполнения операций в параллельных ветвях, но и от затрат времени на обмен информацией между ветвями, то есть между вычислительными модулями (ВМ) параллельной системы. При пересылке информации между ВМ на уровне отдельных слов нецелесообразно использовать процедуры обмена информацией через общую память со сложными процедурами доступа. Уменьшить затраты времени на обмен данными позволяет использование потоковых систем с непосредственными связями (ПСНС) между ВМ [3-6].

В ПСНС выходы одних ВМ подключаются к входам других ВМ в соответствии с графом потока данных (ГПД). ВМ работают в неавтономном (on-line) режиме. В процессе вычислений данные пересылаются непосредственно от одних ВМ к другим, преобразуясь на каждом шаге в соответствии с операциями, заданными вершинами ГПД. В таком случае отсутствуют сложные процедуры пересылки данных между ВМ, то есть уменьшаются затраты времени на обмен данными между ними.

Достижения в области интегральной технологии позволяют создавать параллельные системы CSoC (Configurable System on Chip – конфигурируемые системы на кристалле), которые реализуются на ПЛИС с использованием интерактивных средств разработки. В общем случае микросхемы содержат вычислительные ядра, память и программируемую логику, что дает потенциальную возможность оптимально адаптировать аппаратные средства к конкретному применению.

ПСНС могут быть реализованы на одной или нескольких микросхемах. Это определяется как размерностью задач, так и сложностью вычислительных средств, что зависит от реализуемых ими алгоритмов выполнения операций. Важной проблемой является сокращение аппаратных ресурсов ПЛИС, необходимых для реализации систем [5-7].

Использование нескольких микросхем создает дополнительные проблемы. Недостатком такой реализации является то, что микросхемы могут не располагать требуемым количеством выводов для обеспечения связей между частями системы. При этом часть ресурсов микросхем может оставаться незадействованной.

Учитывая важность проблемы недостатка выводов микросхем, компанией Virtual Machine Works предложена технология VirtualWire (виртуальные соединения) для построения систем на нескольких микросхемах [7]. Идея, заложенная в основе технологии, заключается в использовании незадействованного оборудования для реализации специальных цепей, обеспечивающих поочередное подключение к выводам микросхемы информации из разных источников внутри микросхемы. Для этого внутри каждой ПЛИС должен быть реализован автомат управления потоками данных.

Данная технология, хотя и может решить проблему нехватки выводов микросхем, создает большие временные задержки продвижения потоков данных, что противоречит самой идее потоковой модели вычислений.

Одним из подходов к решению проблемы уменьшения количества связей между ВМ является использование квазипараллельной арифметики, позволяющей совмещать процессы поразрядного ввода в ВМ операндов и поразрядного формирования результатов [8-10]. Поразрядный обмен данными со старших разрядов позволяет не только уменьшить число связей между ВМ, но и обеспечить выполнение зависимых по данным операций в режиме совмещения. Формализованные методики перехода от ГПД к структуре ПСНС известны, например, [11].

Методы неавтономных вычислений лучше исследованы для чисел с фиксированной запятой. Известны алгоритмы выполнения арифметических операций с плавающей запятой (например, [12]), однако в этой области требуются дальнейшие исследования.

Представление чисел с фиксированной запятой приводит к ряду недостатков:

– существенные ограничения на диапазон представления чисел;

– накопление погрешности при выполнении последовательности операций в случае фиксированной разрядности операндов;

– необходимость масштабирования операндов для обеспечения необходимого соотношения их величин в определенном месте цепочки операций (например, при сложении разряды слагаемых должны иметь одинаковый вес);

– необходимость синхронизации поступления разрядов операндов на входы ВМ из разных источников введением в схему элементов задержки.

Ниже на примере вычисления полиномов показана возможность устранения указанных недостатков за счет обработки данных в неавтономном режиме с плавающей запятой.

Организация аппаратных средств вычислителя полиномов

Для вычисления полиномов воспользуемся методом Горнера первого порядка. Например, полином 5-го порядка можно записать в виде

$$F = ((((a_5 x + a_4)x + a_3)x + a_2)x + a_1)x + a_0 .$$

Система представляет собой цепочку ВМ, каждый из которых выполняет промежуточную операцию $Z = XY + A$ (рис. 1).

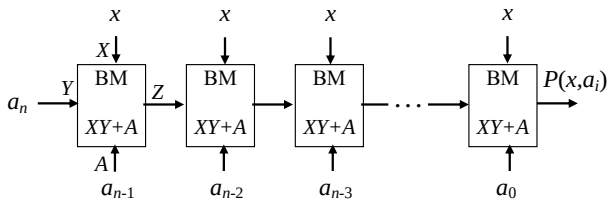


Рис. 1. Система для вычисления полиномов $P(x, a_i)$ по схеме Горнера первого порядка

ВМ позволяют совмещать выполнение зависимых операций на уровне обработки разрядов слов следующим образом.

На каждом шаге вычислений в ВМ вводится по одному разряду операндов и формируется один разряд результата. При этом разряд промежуточного результата, полученный на i-м шаге в одном ВМ при выполнении j-й операции, может быть использован на $(i+1)$-м шаге в другом ВМ при выполнении $(j+1)$-й операции. При таком режиме вычислений выполнение следующей операции будет начинаться не после завершения выполнения предыдущей операции, а сразу же после получения первого разряда результата этой операции. Режим работы таких ВМ называют неавтономным, так

как для выполнения последовательности операций необходимо несколько ВМ, которые синхронно обмениваются информацией в процессе работы.

Рассматриваемый режим выполнения операций со старших разрядов возможен в избыточных системах счисления, например, в двоичной системе с основанием $k=2$ и цифрами $g=\{-1,0,1\}$.

Хотя на входах и выходах ВМ числа представлены последовательным кодом, такие устройства по внутренней организации ближе к параллельным устройствам. В связи с этим они получили название квазипараллельные [3, 4]. С использованием квазипараллельных ВМ при выполнении последовательности зависимых по данным операций реализуется параллелизм на уровне обработки разрядов операндов.

Как известно [7], в ПЛИС можно сконфигурировать группу программируемых логических блоков для работы в качестве ВМ в виде программных или микропрограммных ядер. С помощью специальных средств в ядрах можно сформировать требуемый набор команд и микроархитектуру ядра. Этот факт является очень важным для построения ВМ со специальными алгоритмами выполнения операций, в том числе, с использованием избыточной системы счисления.

Обобщенная структура ВМ для выполнения операций с плавающей запятой (рис. 2) содержит блок обработки порядков (БОП), блок обработки мантисс (БОМ), буферы мантисс типа FIFO (Буф M_X, Буф M_Y и Буф M_A) и регистры порядков (Рег P_X, Рег P_Y и Рег P_A).

Сигналы инициализации и управления ВМ условно не показаны, чтобы не затенять рисунок.

Мантиссы представляются в симметричном избыточном коде с цифрами {-1,0,1}. Для пересылки цифры мантиссы между ВМ необходимо иметь два проводника. Цифры -1, 0, 1 кодируются соответственно парами разрядов 10, 00, 01. С порядками чисел выполняются элементарные операции, поэтому они могут быть представлены в канонической двоичной системе счисления. Для уменьшения связей между ВМ порядки могут пересылаться последовательным кодом по одному проводнику.

Поступление разрядов порядков и мантисс в ВМ тактируется внешними синхросигналами. Темп поступления разных операндов может быть различным. Наличие информации в буферах и регистрах показывают специальные признаки.

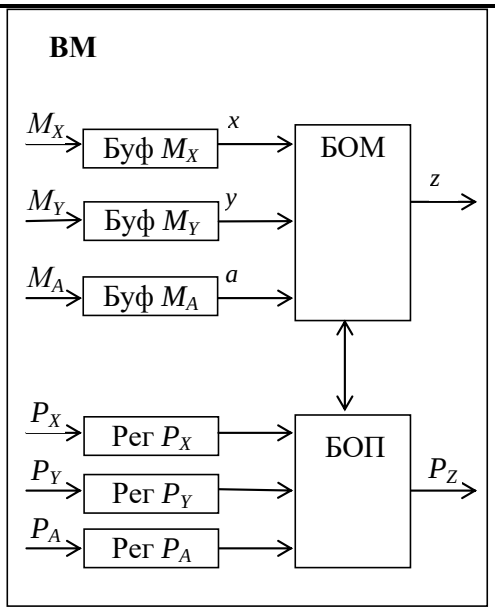


Рис. 2. Вычислительный модуль

Выдача данных из ВМ сопровождается синхросигналами, которые формируются в этом ВМ.

Алгоритмы обработки мантисс и порядков

Форматы представления чисел с плавающей запятой в избыточных системах счисления имеют свои особенности, которые описаны в [12]. Будем считать, что мантиссы являются нормализованными дробными числами и представлены в форме:

$$X=\sum_{i=1}^{n} x_i 2^{-i},\ Y=\sum_{i=1}^{n} y_i 2^{-i},\ A=\sum_{i=1}^{n} a_i 2^{-i}, \quad (1)$$

где $x_i, y_i, a_i \in \{-1,0,1\}$ – цифры операндов, n – разрядность операндов.

Операнды, содержащие только i разрядов справа от запятой, обозначим соответственно через X_i, Y_i, A_i. Например, $A_i = a_1 a_2 a_3 0...0$.

Результат Z может формироваться поразрядно в системе с такими же цифрами с запаздыванием на некоторое число шагов.

Потребуем, чтобы погрешность результата была знакопеременной и по абсолютной величине не превышала половины веса n-го разряда после запятой. Это требование будет выполняться, если на i-м шаге цифру z_i результата выбирать таким образом, чтобы имело место соотношение

$$Z_i - 2^{-i-1} \le 2^{-p}(X_i Y_i + A_i) < Z_i + 2^{-i-1}, \quad (2)$$

где p – число шагов запаздывания формирования цифр результата.

Используя методику в [8] и формулы (1), (2), можно получить алгоритм вычисления Z в неавтономном режиме. В рассматриваемом

случае $p=3$. Следовательно, для получения n разрядов результата после запятой необходимо выполнить $n+3$ шагов вычисления.

Алгоритм обработки мантисс имеет следующий вид.

1. X_0, Y_0, R_0 присвоить значение 0.
2. Для $i=\overline{1,n+3}$ выполнять пункты 3-7.
3. $H_i = 2R_{i-1} + 2^{-3}X_{i-1}y_i + 2^{-3}Y_{i-1}x_i + 2^{-3}a_i + + 2^{-3-i}x_i y_i$.
4. $X_i = X_{i-1} + x_i 2^{-i}$.
5. $Y_i = Y_{i-1} + y_i 2^{-i}$.
6. $z_i = \begin{cases} -1, \text{ если } H_i < -2^{-1}; \\ 0, \text{ если } -2^{-1} \le H_i < 2^{-1}; \\ 1, \text{ если } 2^{-1} \le H_i. \end{cases}$
7. $R_i = H_i - z_i$.

Здесь H_i и R_i – вспомогательные переменные.

Порядки являются целыми числами. Они поступают в ВМ вместе с первым разрядом соответствующего операнда и обрабатываются совместно с мантиссами.

Обработка порядков производится следующим образом.

1. Принять порядки P_X, P_Y и P_A, установить вспомогательные переменные $q:=0,\ j:=0,\ s:=0$.
2. Получить предварительный порядок результата P_Z, выполнив преобразования: $P_W := P_X + P_Y$; если $P_W \ge P_A$, то $q:=1$; $\Delta P := \max(P_W, P_A) - \min(P_W, P_A)$; $P_Z := \max(P_W, P_A) + p$.
3. Если в Буф M_X, Буф M_Y и Буф M_A есть очередные цифры мантисс операндов x, y, a (см. рис.1), то перейти к п. 4, иначе выполнить повторно п. 3.
4. Если $\Delta P = 0$, то принять в БОМ цифры мантисс x, y, a и перейти к п. 7.
5. Если $q=0$, то принять в БОМ цифры x, y и 0 вместо a, иначе принять a и нули вместо x, y.
6. $\Delta P := \Delta P - 1$.
7. Сформировать цифру результата z.
8. Если $s \ne 0$, то перейти к п. 11.
9. Если $z=0$, то $P_Z := P_Z - 1$ и перейти к п. 3.
10. $s:=1$ и выдать из БОП окончательный порядок результата P_Z.
11. Выдать из БОМ очередную цифру результата z и $j := j+1$.
12. Если $j \ne n$, то перейти к п. 3, иначе конец операции.

В алгоритме цифры операндов представлены без индексов, поскольку порядковые номера циклов вычисления и цифр в коде операндов не совпадают.

Порядок результата выдается из ВМ вместе с первым ненулевым разрядом мантиссы результата. За счет этого обеспечивается нормализация мантиссы результата.

Оценка временных характеристик и аппаратурных затрат

Следует заметить, что операции в БОП и БОМ практически совмещаются во времени. В связи с этим для оценки времени вычислений достаточно рассмотреть обработку мантисс чисел.

Процесс выполнения операций иллюстрируется диаграммой на рис. 3.

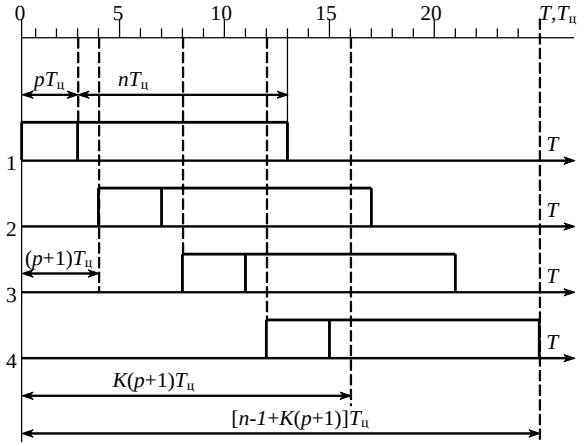


Рис. 3. Временная диаграмма выполнения последовательности зависимых по данным операцій

Операции выполняются в неавтономном режиме с помощью цепочки ВМ, которые позволяют совмещать выполнение зависимых операций на уровне обработки разрядов операндов.

Из диаграммы на рис. 3 следует, что время вычислений, необходимое для получения окончательного результата, составляет

$$T = \left[n - 1 + \sum_{j=1}^{K} \left(p_j + 1 \right) \right] t_ц, \qquad (3)$$

где n – разрядность операндов; K – число операций в цепочке; p_j – задержка формирования разрядов результата j-й операции; $t_Ц$ – длительность цикла формирования одного разряда результата в ВМ.

В рассматриваемом случае для каждой операции задержка составляет $p = 3$. Например, для вычисления полинома 5-ой степени при 64-разрядных операндах понадобится выполнить 83 цикла. Длительность одного цикла определяется в основном вычислением переменной H_i в приведенном выше алгоритме обработки мантисс (см. п. 3). Указанная длительность определяется суммированием пяти слагаемых с помощью дерева комбинационных сумматоров и составляет единицы наносекунд при реализации на ПЛИС.

Первый разряд окончательного результата формируется через $K(p+1)$ циклов после начала вычислений. В дальнейшем в каждом такте формируется очередная цифра результата. Заметим, что в системах автоматического управления во многих случаях управляющее воздействие может начинать формировать при поступлении первого старшего разряда результата, а затем уточняться при поступлении каждого следующего разряда.

При использовании для выполнения последовательности зависимых по данным операций параллельных ВМ, работающих в канонической двоичной системе счисления, время выполнения цепочки операций будет определяться формулой

$$T = T_o + \sum_{i=1}^{K} t_i \,, \qquad (4)$$

где t_i – время выполнения i-й операции; T_0 – дополнительные затраты времени, например, на обмен данными между ВМ (возможно через общую память), прием операндов в устройство по очереди или последовательным кодом для уменьшения требуемого числа выводов микросхем и т.д.

Для количественной сравнительной оценки времени вычислений по формулам (3) и (4) требуется более детальное рассмотрение конкретной реализации ВМ разного типа.

Для сравнительной оценки затрат ресурсов ПЛИС при реализации устройств с параллельной и последовательной передачей данных проведено моделирование двух вариантов реализации БОМ на базе микросхемы EP2C35F672C6 семейства Cyclone II фирмы Altera в среде проектирования Quartus II Version 9.1 Build 304. Результаты моделирования в соответствии с отчетами о компиляции приведены в табл. 1.

Табл. 1. Используемые ресурсы ПЛИС

Ресурс ПЛИС	**Блок обработки мантисс**	
	Параллельный	**Квазипараллельный**
Логические элементы	502/33216 (1,5%)	695/33216 (2%)
Регистры	0/33216 (0%)	271/33216 (0,8%)
Блоки умножения	32/70 (46%)	0/70 (0%)
Выводы	256/475 (54%)	10/475 (2%)

В табл. 1 через разделитель показан соответственно ресурс для построения вычислителей и общий ресурс ПЛИС определенного вида.

Для параллельного блока рассматривались только аппаратные средства выполнения операции $XY + A$ без учета средств выравнивания порядков операндов и нормализации результата. Умножитель 64×64 реализован на базе встроенных в ПЛИС быстродействующих блоков умножения.

На основании табл. 1 можно сделать вывод, что затраты ресурсов ПЛИС больше для устройства параллельного типа. Учитывая, что при данной организации вычислений практически половина выводов ПЛИС используется только для одного каскада вычислителя полиномов, применение режима параллельного ввода операндов вряд ли целесообразно. Более реальным является режим предварительного ввода в определенном порядке 64-разрядных операндов с последующей их обработкой в параллельном устройстве. Время вычисления полиномов будет во многом определяться режимом ввода операндов.

Выводы

В работе исследована возможность повышения эффективности систем с непосредственными связями за счет реализации неавтономных вычислений с плавающей запятой при поразрядной передаче данных между модулями системы.

Показано, что по сравнению с вычислениями в формате фиксированной запятой выполнение операций с плавающей запятой позволяет расширить диапазон представления чисел, повысить точность получения результатов и ускорить вычисления за счет параллельной обработки мантисс и порядков в асинхронном режиме.

По сравнению с системами, в которых передача данных между ВМ производится параллельным кодом, для реализации систем с последовательной передачей данных на базе квазипараллельных ВМ требуется меньший ресурс ПЛИС. Экономятся как элементы внутреннего ресурса (функциональные и коммуникационные модули, встроенные устройства), так и выводы ПЛИС. Бдагодаря этому увеличивается вероятность размещения системы на одной микросхеме или уменьшения их числа. Реализация системы на одной микросхеме обеспечивает повышение ее надежности, уменьшение энергопотребления и габаритов. Обмен данными между ВМ внутри микросхемы выполняется быстрее, чем между компонентами системы, реализованными на разных микросхемах. Это дает потенциальную возможность повысить частоту тактирования, что, в свою очередь, ускоряет обработку информации.

Таким образом, полученные результаты подтверждают эффективность применения неавтономных методов поразрядной обработки информации в вычислительных системах на кристалле.

Список литературы

1. Байков В.Д. Решение траекторных задач в микропроцессорных системах ЧПУ / В.Д.Байков, С.Н.Вашкевич. – Л.: Машиностроение, 1986, 105 с.
2. Жабин В.И. Архитектура вычислительных систем реального времени / В.И.Жабин. – К.: ВЕК+, 2003. – 176 с.
3. Жабин В.И. Эффективность потоковых вычислений в системах с непосредственными связями, реализованных на ПЛИС / В.И.Жабин, В.В.Жабина, М.А. Безгинский // Вісник НТУУ "КПІ". Інформатика, управління та обчислювальна техніка: Зб. наук. праць. – К.: ВЕК+. – 2012. – №55. – С. 149-156.
4. Жабин В.И. Выполнение последовательностей зависимых операций в режиме совмещения / В.И.Жабин. // Вісник НТУУ «КПІ». Інформатика, управління та обчислювальна техніка: Зб. Наук. Пр. – К.: Век+. – 2007. – №46. – С. 226-233.
5. Палагин А.В. Реконфигурируемые вычислительные системы: Основы и приложения / А.В.Палагин, В.Н.Опанасенко. – К.: Просвіта, 2006, 280 с.
6. Каляев И.А. Архитектура семейства реконфигурируемых вычислительных систем на основе ПЛИС / И.А. Каляев, И.И. Левин, Е.А. Семерников // Искусственный интеллект. – 2008. – № 3. – С. 663-674.
7. Максфилд К. Проектирование на ПЛИС. Архитектура, средства и методы / К.Максфилд. – М.: Издательский дом «Додэка-XXI», 2007, 408 с.
8. Жабин В.И. Некоторые машинные методы вычисления рациональных функций многих аргументов / В.И. Жабин, В.И.Корнейчук, В.П.Тарасенко // Автоматика и телемеханика. – 1977. – №12. – С. 145-154.
9. Жабин В.И. Исследование методов построения вычислительных устройств на основе FPGA фирмы XILINX / В.И.Жабин, Н.А.Ковалев // Технология и конструирование в электронной аппаратуре. – 2002. – №1. – С. 35-39.
10. Самофалов К.Г. Основы теории многоуровневых конвейерных вычислительных систем / К.Г.Самофалов, Г.М.Луцкий. – М.: Радио и связь, 1989. – 272 с.
11. Жабин В.И. Построение быстродействующих специализированных вычислителей для реализации многоместных выражений / В.И.Жабин, В.И.Корнейчук, В.П.Тарасенко // Автоматика и вычислительная техника. – 1981. – №6. – с. 18-22.
12. Жабин В.И. Повышение эффективности параллельных вычислений в потоковых системах / В.И.Жабин, В.В.Жабина // Вісник НТУУ "КПІ". Інформатика, управління та обчислювальна техніка: Зб. наук. праць. – К.: ВЕК+. – 2013. - № 59. – С. 122-128.

УДК 683.519

ГАНЖА І. М.,
ШЕМСЕДИНОВ Т. Г.,
АЛЄНІН О. І.,
СТІРЕНКО С. Г.

МОДИФІКАЦІЯ ТА ЗАСТОСУВАННЯ АЛГОРИТМУ МАРКУВАННЯ ЗВ'ЯЗНИХ КОМПОНЕНТ НА БІНАРНОМУ ЗОБРАЖЕННІ

В статті розглянуто двухпрохідний порядковий алгоритм маркування зв’язних компонент, приведено спосіб реалізації алгоритму та запропоновано використання динамічного масиву для збереження класів еквівалентності. Алгоритм маркування зв’язних компонент використовується для виділення зв’язної компоненти, що використовується для подальшого морфологічного аналізу компоненти.

In the article considered the two-pass consistent connected components labeling algorithm, a way to implement the algorithm and the use of a dynamic array to store the equivalence classes. Connected components labeling algorithm used to select a connected component, which used for further morphological component analysis.

Ключові слова: зв'язані компоненти, морфологічний аналіз, цифрова обробка зображень, розпізнавання зображень.

1. Вступ

Двухпрохідний порядковий алгоритм маркування зв'язних компонент або класичний алгоритм маркування зв'язних компонент на бінарному зображенні [2] виявляє зв'язні компоненти (області) переднього плану та надає виявленій компоненті чисельне значення або маркер. В результаті роботи алгоритму, з початкового зображення, в якому всі пікселі компонент мають значення 1, формується маркіроване зображення, в якому пікселі компоненти мають значення маркера компоненти. На рис. 1а наведено початкове бінарне зображення, в якому всі чотири компоненти мають однакові значення пікселів. На рис. 1б зображено результат алгоритму маркування, кожна компонента отримала свій маркер, всі пікселі компоненти мають значення маркера, внаслідок чого компоненти зображені різними значеннями інтенсивності.

Маркування зв'язних компонент необхідне для подальшого морфологічного аналізу виявлених компонент. Морфологічний аналіз формує множину ознак компоненти, дані ознаки використовуються в класифікації або розпізнаванні об’єкта. Також алгоритм може використовуватись для маркування зв’язних компонент фону, маючи марковані компоненти переднього плану та фону можливо скласти граф зв’язності компонент[1].

а. Початкове зображення *б. Маркіроване зображення*

Рис. 1. Приклад роботи алгоритму маркування зв’язних компонент

2. Опис алгоритму

Класичний алгоритм виконується за два проходи. На першому проході формуються класи еквівалентності та надаються тимчасові маркери зв'язним компонентам або їх частинам. На другому проході тимчасові маркери заміняються маркерами відповідно до кореня структури об'єднання-пошуку класу еквівалентності.

Клас еквівалентності являє собою множину тимчасових маркерів окремої зв'язної компоненти. Клас еквівалентності в класичному алгоритмі маркування представлено в структурі об'єднання-пошуку, яка зберігає тимчасові мітки в деревоподібній структурі, з'єднує дерева між собою та дозволяє швидко знаходити корінь дерева [1]. На рис. 2 наведено приклад структури об'єднання-пошуку у вигляді таблиці та графів дерев.

0	1	2	3	4	5	6	7	8
0	2	6	6	8	6	0	8	0

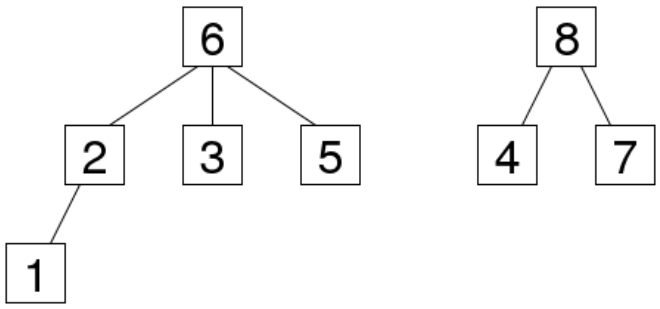


Рис. 2. Структура об'єднання-пошуку у вигляді таблиці та графу

Таблиця на рис. 2 в першому рядку містить значення маркерів від 1 до кількості маркерів, значення 0 не враховується адже відповідає значенню фону зображення. Другий рядок містить значення маркера-батька для маркера з першого рядка, значення 0 означає, що маркер являється коренем дерева. На рис. 2 зображено два графи дерев, що відповідають двом структурам об'єднання-пошуку або класам еквівалентності. Два дерева об'єднують дві множини маркерів – {6, 5, 3, 2, 1} та {8, 7, 4}. Корені дерев – маркери 6 та 8, тому вони мають значення 0 в другому рядку таблиці. Маркер 1 має маркер-батька 2, тому в другому рядку має значення 2, маркери 2, 3, 5 мають маркера-батька 6, тому значення в другому рядку в усіх 6, маркери 4, 7 мають маркера-батька 8, тому в другому рядку міститься 8. Оскільки в першому рядку значення являють нумерацію, то дану структуру можливо представити у вигляді одномірного масиву, в якому індекси – це дані першого рядка, а значення масиву – це дані другого рядка.

При необхідності об'єднати два маркери в структуру, або об'єднати дві структури, якщо їхні корені не співпадають, виконується операція об'єднання. Для виконання операції об'єднання двох дерев досить в таблиці в другий рядок коре-ня дерева, що додається замінити 0 на значення кореня дерева, в яке виконується додавання. Якщо необхідно знайти батька для маркера, щоб визначити до якого класу еквівалентності він належить, виконується операція пошуку. Для пошуку кореня в даній структурі виконується перебирання значень батьків маркерів до того моменту доки не буде знайдено 0.

На першому проході алгоритм надає тимчасовий маркер ще не поміченій області та намагається розповсюдити маркер правим та нижнім сусідам компоненти. Якщо виникає ситуація, що два не однакових маркера намагаються розповсюдитись на один і той же піксель то обирається маркер з меншим значенням. Кожен маркер заноситься в структуру об'єднання-пошуку та при наявності двох різних маркерів в одній компоненті маркери об'єднуються за допомогою описаної вище операції об'єднання. Після першого проходу класи еквівалентності сформовані у вигляді структур об'єднання-пошуку та мають унікальний маркер або ідентифікатор, що відповідає кореню дерева структури. На другому проході тимчасові маркери замінюються ідентифікаторами класів еквівалентності, застосовуючи операцію пошуку кореня дерева, яка описана вище.

На рис. 3 показано результат роботи першого та другого проходів алгоритму та наведено структуру об'єднання-пошуку, яка має класи еквівалентності {2, 1} та {4, 3}.

1	1	0	0	0	0	0	0	0	0
1	1	0	0	2	2	2	2	2	0
1	1	0	0	2	2	2	2	2	0
1	1	1	1	1	1	0	2	2	2
1	1	1	1	1	1	0	2	2	2
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	3	3	0
0	0	0	0	0	0	0	3	3	0
0	4	4	4	4	4	4	3	3	0
0	4	4	4	4	4	4	3	3	0

а. Результат першого проходу

1	1	0	0	0	0	0	0	0	0
1	1	0	0	1	1	1	1	1	0
1	1	0	0	1	1	1	1	1	0
1	1	1	1	1	1	0	1	1	1
1	1	1	1	1	1	0	1	1	1
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	2	2	0
0	0	0	0	0	0	0	2	2	0
0	2	2	2	2	2	2	2	2	0
0	2	2	2	2	2	2	2	2	0

б. Результат другого проходу

0	1	2	3	4
0	2	0	4	0

в. Структура об'єднання-пошуку після першого проходу

Рис. 3. Результати роботи двох проходів та структура об'єднання-пошуку

3. Реалізація алгоритму

Для збереження класів еквівалентності або структур об'єднання-пошуку використано динамічний масив, що дозволяє створювати масив відповідно до кількості тимчасових маркерів.

Перший прохід реалізовано двома послідовними циклами, в яких оброблюються пікселі зі значенням 1 вхідного зображення. Відносно поточного пікселя перевіряться наявність сусідів зі значенням 1 зліва та зверху, з задачею ініціювати маркер MC. Якщо є сусід зліва з маркером ML, то MC := ML, якщо сусід зверху також існує і його маркер MT менше MC, то MC:=MT. Якщо сусідів не виявлено то в MC записується значення наступного маркера по порядку. Маркер MC записується в піксель результуючої матриці.

Якщо маркер MC не дорівнює значенню зліва або зверху то виконується об'єднання маркерів в структурі об'єднання-пошуку.

На другому проході також виконується прохід двома циклами, в яких оброблюються пікселі зі значенням більше 0. Піксель зберігає значення тимчасового маркера, по якому за допомогою операції пошуку знаходиться корінь структури об'єднання-пошуку. Для уникнення збою в порядку нумерації маркерів створюється окремий масив, в якому значенню кореня структури присвоюється порядковий номер. Порядковий номер, що відповідає кореню структури записується в пікселі зв'язної компоненти та є її остаточним маркером.

4. Порівняння класичного та рекурсивного алгоритмів

Порівняння класичного та рекурсивного алгоритмів проводиться на основі сформованого вручну зображення в форматі PGM розміром 800x800 пікселів. Щоб отримати множину зображень різного розміру, початкове зображення розміром 800x800 зменшується на 50 пікселів по ширині та висоті. В результаті маємо 16 однакових по змісту тестових зображень розміром від 50x50 до 800x800 пікселів з кроком 50 пікселів.

Для кожного типу алгоритму проводиться по 100 експериментів та визначається середній час роботи алгоритму. Час роботи алгоритмів показано на рис. 4.

На рис. 4 видно, що рекурсивний алгоритм працює швидше класичного, адже класичний алгоритм робить два проходи з обробко пікселів, рекурсивний алгоритм робить всього один прохід. При розмірі зображення 500x500 пікселів рекурсивний алгоритм видає помилку StackOverflow , що означає що програма вичерпала кількість викликів рекурсивних методів, тому на графіках значення часу 0. Оскільки кількість пікселів в компонентах збільшилась, а на кожен піксель при обробці необхідно викликати рекурсивну функцію, значить збільшується кількість рекурсивних викликів функцій. Налаштування JVM при експерименті стандартне – heap size 700 Мб.

Класичний алгоритм працює при розмірі зображення 500x500 і більше, адже відсутній рекурсивний виклик функцій, маркери назначаються пікселям на основі значень сусідів та структури об’єднання-пошуку. При збільшенні розміру зображення збільшується час роботи алгоритму.

Класичний алгоритм можливо застосовувати для великих зображень на відміну від рекурсивного. Зі збільшенням якості зображень, які формують сучасні фотокамери, потреба в обробці зображень великого розміру стає актуальною.

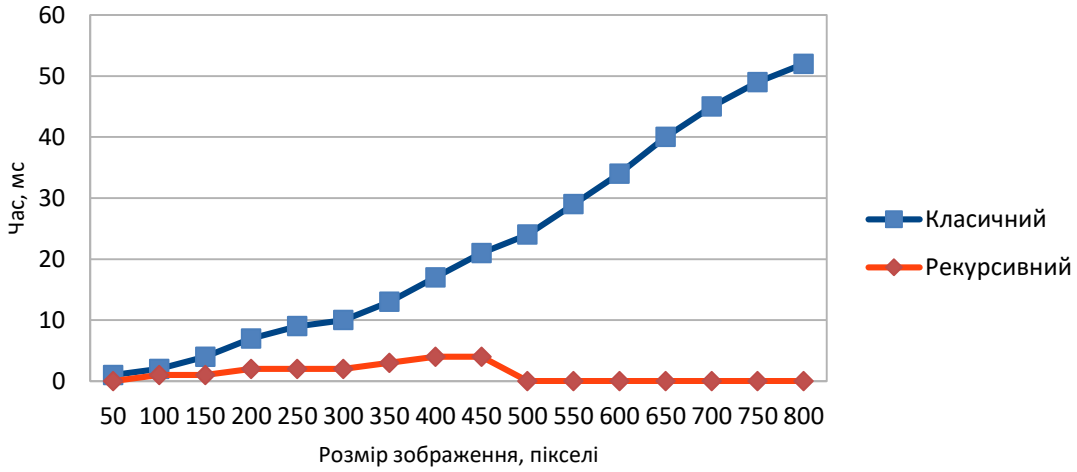


Рис. 4. Час роботи класичного та рекурсивного алгоритмів

5. Застосування алгоритму

Алгоритм маркування зв'язних компонент виконується перед розрахунком характеристик зв'язних компонент. Виділення характеристик компоненти на зображенні називається морфологічним аналізом. Алгоритм маркування зв'язних компонент виділяє компоненту, надає їй унікальний ідентифікатор, що дозволяє аналізувати певну компоненту на зображенні. Модифікація алгоритму нумерує компоненти по порядку, що спрощує реалізацію морфологічного аналізу для випадку коли більше однієї компоненти.

До морфологічних характеристик області зображення відносять:

- площа;
- центр ваги;
- центральний момент другого порядку відносно рядків, стовбців;
- змішаний центральний момент другого порядку;
- момент другого порядку відносно головної та другорядної осі.

Позначимо множину пікселів, що входять до зв'язної компоненти буквою R. Площею A компоненти на бінарному зображенні є кількість пікселів множини R[1]:

$$A = \sum_{(r,c)\in R} 1 \qquad (1.1)$$

Центр ваги є центральною точкою компоненти. Координати центра ваги розраховуються як середні значення координат пікселів області R[1]:

$$\bar{r} = \frac{1}{A} \sum_{(r,c)\in R} r \qquad (1.2)$$

$$\bar{c} = \frac{1}{A} \sum_{(r,c)\in R} c \qquad (1.3)$$

Центральний момент другого порядку показує відхилення координат пікселів множини R від координат центра ваги $\bar{r}$ (1.2), $\bar{c}$ (1.3) по рядкам або стовбцям, нормоване площею A(1.1) компоненти[1]:

$$\mu_{rr} = \frac{1}{A} \sum_{(r,c)\in R} (r - \bar{r})^2 \qquad (1.4)$$

$$\mu_{cc} = \frac{1}{A} \sum_{(r,c)\in R} (c - \bar{c})^2 \qquad (1.5)$$

Змішаний центральний момент показує відхилення по рядкам і стовбцям одночасно:

$$\mu_{rc} = \frac{1}{A} \sum_{(r,c)\in R} (r - \bar{r})(c - \bar{c}) \qquad (1.6)$$

Для компоненти можна виділити головну та другорядну осі, які проходять через центр ваги. Головна вісь об'єкту на бінарному зображенні може бути знайдена як вісь, відносно якої пікселі об'єкта мають найменший момент другого порядку. Другорядна вісь перпендикулярна головній. Головна та другорядна осі аналогічні великій та малій осям еліпса. Вісь характеризується кутом відносно вертикальної осі декартових координат. Розрахунок кута головної осі наступний[1]:

1. $\mu_{rc} = 0, \mu_{rr} > \mu_{cc}$

$$\alpha = 0^{\circ}$$

2. $\mu_{rc} = 0, \mu_{rr} \leq \mu_{cc}$

$$\alpha = 90^{\circ}$$

3. $\mu_{rc} \neq 0, \mu_{rr} > \mu_{cc}$

$$\alpha = tg^{-1}\left\{\frac{\left[\mu_{cc} + \mu_{rr} + [(\mu_{cc} - \mu_{rr})^2 + 4\mu_{rc}^2]^{1/2}\right]^{1/2}}{-2\mu_{rc}}\right\}$$

4. $\mu_{rc} \neq 0, \mu_{rr} \leq \mu_{cc}$

$$\alpha = tg^{-1}\left\{\frac{-2\mu_{rc}}{\mu_{rr} - \mu_{cc} + [(\mu_{rr} - \mu_{cc})^2 + 4\mu_{rc}^2]^{1/2}}\right\}$$

Кут другорядної осі $\beta = \alpha + 90^{\circ}$.

Знайшовши кут осей, розраховуємо момент другого порядку відносно головної осі за формулою[1]:

$$\mu_{\bar{r},\bar{c},a} = \frac{1}{A}\sum_{(r,c)\in R}\left((r - \bar{r})cos\alpha + (c - \bar{c})sin\alpha\right)^2 \qquad (1.7)$$

Формула для розрахунку моменту другого порядку відносно другорядної осі аналогічна.

6. Приклад застосування

Моменти другого порядку є морфологічними характеристиками та можуть застосовуватись для розпізнавання об’єктів на зображенні.

Автором статті реалізована програма розрахунку моментів другого порядку та розпізнавання об’єкта на основі готових моментів другого порядку еталонів. Об’єктом розпізнавання обрано цифри. Використано моменти другого порядку μ_{rr}(1.4), μ_{cc} (1.5), μ_{rc}(1.6), $\mu_{\bar{r},\bar{c},a}$(1.7) головної та другорядної осей. Наприклад момент μ_{cc} для цифри 1 має менше значення ніж μ_{cc} для цифри 2, моменти $\mu_{\bar{r},\bar{c},a}$ відносно головної осі матимуть різні значення для цифр 1 та 2.

На першому етапі розраховано моменти другого порядку для бінарних зображень розміром 50x50 з цифрами від 0 до 9. Ці еталонні дані збережено в файлі в структурі JSON. Наприклад для цифри 1 сформовано наступний JSON об’єкт: {"muRCaL":29.378721237182617,"muRC":24.959863662719727,"muRR":201.96498107910156,"muCC":21.200931549072266,"muRCaS":193.7871856689453,"value":1}. На другому етапі малюється цифра такого ж розміру як еталонні цифри, знаходяться моменти другого порядку цифри, знаходиться різниця моментів другого порядку з кожним еталоном, еталон з найменшою різницею є шуканим класом вхідного об’єкту.

Експерименти показали, що розпізнавання цифр 1, 2, 3, 4, 5, 7, 8 працює вірно, розпізнавання цифр 6 може дати результат 9, розпізнавання цифри 9 може дати результат 6. Таким чином даний метод не дає 100% правильний результат, адже моментів другого порядку недостатньо для повного розпізнавання цифр.

Перевагою даного методу є те, що для класифікації використовуються характеристики еталонів, самі еталони не завантажуються в пам'ять та не обробляються при розпізнаванні.

7. Висновок

В статті детально розглянуто двухпрохідний порядковий алгоритм маркування зв’язних компонент, запропоновано модифікацію алгоритму, яка полягає в упорядкуванні маркерів, та приведено опис реалізації алгоритму з використанням динамічних масивів для зберігання структур об'єднання-пошуку. Порівняння часу роботи класичного та рекурсивного алгоритму показало, що класичний алгоритм програє в швидкості рекурсивному, однак рекурсивний алгоритм при розмірі зображення 500x500 пікселів та більше призводить до переповнення стеку задач. Класичний алгоритм дозволяє обробляти зображення великих розмірів , при цьому збільшується час обробки.

Наведено приклад застосування алгоритму маркування зв’язних компонент – морфологічний аналіз маркірованих компонент та розпізнавання цифр на основі даних морфологічного аналізу. При розпізнаванні цифр за допомогою моментів другого порядку, більшість цифр розпізнаються правильно, цифри, схожі по структурі, можуть взаємо замінятись.

Список посилань

1. Л. Шапиро, Дж. Стокман, Компьютерное зрение, М.: Бином. Лаборатория знаний, 2006.
2. A. Rosenfeld, P. Pfaltz, “Sequential Operations in Digital Picture Processing”, Journal of the Association for Computing Machinery, Vol. 12., 1966.
3. Інтернет ресурс. Image moment. https://en.wikipedia.org/wiki/Image_moment

УДК 004.052.42

МАРКОВСЬКИЙ О.П.,
ЗАХАРІУДАКІС ЛЕФТЕРИС,
ФЕДОТОВ М.Ф.

МЕТОД СТРОГОЇ ІДЕНТИФІКАЦІЇ ВІДДАЛЕНИХ АБОНЕНТІВ НА ОСНОВІ СТАНДАРТИЗОВАНИХ ШИФРОБЛОКІВ ТА ХЕШ-ПЕРЕТВОРЕНЬ

В статті пропонується метод підвищення ефективності ідентифікації абонентів багатокористувацьких систем за рахунок зниження ризику несанкціонованого доступу до її ресурсів та підвищення продуктивності ідентифікації.

З теоретичної точки зору найефективніші схеми ідентифікації мають за основу концепцію нульових знань. Відомі схеми, що реалізують цю концепцію використовують функціональні перетворення, основані на операціях модулярної арифметики, які реалізуються повільно. Для схем ідентифікації, що базуються на концепції нульових знань запропоновано новий метод, оснований на використанні стандартизованих шифроблоків та хеш-перетворень. Доведено, що використання запропонованої технології дозволяє прискорити ідентифікацію на 2-3 порядки.

This paper proposes a method for increasing of efficiency of of the multi-user system abonent identifications by impairing the risk of illegal access to system resources and by increasing of identification productivity.

From the theoretical point of view the most effectiveness schemes of abonent identification are based on conception of zero-knowledge. But known schemes which realized this conception utilized the function transformation based on modular arithmetic which calculate slowly. For zero-knowledge based identification performance increase the new method base on utilization of standard block ciphers and hash transformation has been proposed. It has been shown that utilization of proposed identification technology allowed to increase the identification rate by 2-3 orders.

Ключові слова: схеми ідентифікації, інтерактивна ідентифікація, ідентифікація на основі концепції нульових знань, шифроблоки, хеш-алгоритми.

Вступ

Інтегровані системи зберігання та обробки даних з колективним доступом відіграють зростаючу роль в глобальних процесах інформаційної інтеграції. Така інтеграція дозволяє підвищити ефективність комп'ютерної обробки даних за рахунок надання доступу до інформаційних, програмних та обчислювальних ресурсів широкому колу користувачів.

Розвиток інтегрованих систем обробки інформації значною мірою залежить від ефективності реалізації в них функцій захисту інформації та розподілення прав доступу. Ключова роль у вирішенні цієї проблеми належить засобам ідентифікації абонентів. Розширення використання багатокористувацьких систем пов'язане зі зростанням ризиків несанкціонованого доступу до їх інформаційних та обчислювальних ресурсів. Це зумовлено, з однієї сторони, ростом технічних можливостей для реалізації несанкціонованого доступу, а з іншого – збільшенням потенційних вигод з цього. Факторами, що підвищують ризик несанкціонованого доступу до ресурсів інтегрованих систем обробки інформації є розширення використання потенційно відкритих для стороннього впливу бездротових систем передачі даних комп'ютерних мереж, підвищення продуктивності обчислювальних засобів, що використовуються для несанкціонованого доступу, збільшення кількості віддалений користувачів, і, відповідно зменшення обчислювальних ресурсів, які використовуються для захисту від несанкціонованого доступу.

За цих умов необхідністю є адекватне вдосконалення всього арсеналу засобів, які виключають можливість несанкціонованого доступу до ресурсів інтегрованих систем, в тому числі методів та засобів ідентифікації їх віддалених абонентів.

Таким чином, задача підвищення ефективності ідентифікації абонентів багатокористувацьких систем є важливою і актуальною для сучасного етапу розвитку комп'ютерних та мережевих технологій.

Аналіз відомих технологій ідентифікації віддалених абонентів

Базовими критеріями ефективності будь-якої системи захисту є рівень захищеності, що досяга-

ється при її використанні та об'єм ресурсів, що застосовується для реалізації функцій захисту. Відповідно, в процесі розробки будь-якої системи захисту даних, в тому числі і засобів ідентифікації віддалених абонентів, потрібно досягати компромісу між рівнем захищеності та продуктивністю реалізації функцій захисту.

Складність проблеми визначається неможливістю побудови адекватної формальної моделі дій сторони, що намагається реалізувати незаконний доступ до ресурсів системи. В першому наближенні процедура ідентифікації має задовольняти таким вимогам [1]:

1) Організація зберігання ідентифікуючої інформації має бути такою, щоб одна її частина зберігалася у абонента, а друга – в системі і кожна з цих частин не була б самодостатньою для доступу до ресурсів системи.

2) Ідентифікаційна посилка абонента при кожному сеансі підключення до системи має змінюватися.

3) Об'єм секретної інформації, що зберігається в системі і використовується для ідентифікації має бути якомога меншим з тим, щоб надати можливість збереження такої інформації в спеціальній захищеній на апаратному рівні пам'яті.

Всі сучасні протоколи ідентифікації абонентів розділяють на два класи: з використанням паролів, що перевіряються системою шляхом порівняння ("слабка" ідентифікація) та на основі теоретичної концепції "нульових знань" ("строга" ідентифікація) [2].

В основі більшості протоколів ідентифікації віддалених абонентів лежить теоретична концепція "нульових знань". Сутність цієї концепції полягає в тому, що для доведення своєї автентичності абонент має неявним чином виявити знання певної інформації, якою система не володіє, але може перевірити її наявність у абонента.

При цьому в системі не зберігається ніякої секретної інформації, яка дозволяє відновити ідентифікаційні дані абонента, що пояснює походження назви концепції "нульових знань". Важливим є те, що при кожному зверненні до системи абонентом генерується нова ідентифікуюча інформація.

Таким чином, концепція нульових знань в теоретичному плані найбільш повною мірою відповідає сформульованим вище вимогам щодо системи ідентифікації абонентів. В сучасних умовах і у найближчій перспективі потрібний для практики рівень надійності ідентифікації може бути забезпечений лише методами, що спираються на концепцію "нульових знань".

Концепція "нульових знань" передбачає використання незворотних математичних перетворень. Це означає, що існує алгоритм перетворення в прямому напрямку, але принципово неможливим є аналітичне віднаходження алгоритму зворотного перетворення. В більшості існуючих схем ідентифікації на основі концепції "нульових знань" для реалізації такого перетворення використовуються аналітично нерозв'язувані задачі теорії чисел, зокрема відома задача дискретного логарифмування.

Найбільш відомими з схем ідентифікації цього класу є FFSIS (Feige Fiat Shamir Identification Scheme) [2], методи Шнора (Schnorr) та Гіллоу-Квіскватера (Guillou- Quisquater) [3]. Базовими обчислювальними операціями для FFSIS є $A^2 \cdot B \bmod m$, а для методів Шнора і Гіллоу-Квіскватера – $A^e \cdot B^v \bmod m$.

З викладеного слідує, що базовою обчислювальною операцією більшості схем строгої ідентифікації віддалених абонентів є модулярне експоненціювання над числами, довжина яких значно перевищує розрядність процесору. В умовах стійкої тенденції до зростання розрядності чисел, обчислювальна складність реалізації вказаного типу операцій збільшується експоненційно, випереджаючи темпи зростання продуктивності комп'ютерних систем.

Опубліковано [4] результати спроб використання при реалізації концепції "нульових знань" в якості незворотного булевих функціональних перетворень. Таке рішення дозволяє на порядки прискорити процедуру ідентифікації віддалених абонентів. Недоліками такого рішення є суттєве обмеження на кількість сеансів ідентифікації, складність процедури генерації незворотного та багатозначного булевого перетворення, значний об'єм пам'яті для зберігання в системі таблиць перетворень для кожного з абонентів. Суттєвим моментом є те, що не проведені об'ємні та всебічні дослідження рівня захищеності. Тому цей метод не дійшов до рівня практичного застосування в діючих протоколах ідентифікації.

Таким чином, основний недолік існуючих методів реалізації теоретично строгої ідентифікації на основі концепції "нульових знань" полягає в значній обчислювальній складності їх базових обчислювальних процедур. Ясно, що швидкість ідентифікації обмежується значною обчислювальною складністю цих операцій і, відповідно, виникає необхідність в її зменшенні.

Ціллю дослідження є зменшення обчислювальної складності процесу ідентифікації при забезпеченні високого рівня захищеності.

Метод ідентифікації на основі стандартизованих криптографічних примітивів

Для досягнення поставленої мети пропонується метод строгої ідентифікації віддалених абонентів, що реалізує теоретичну концепцію "нульових знань" з використнням стандартизованих шифроблоків та хеш-перетворень.

Стандартизований шифроблок (ШБ) являє сертифікований органами державної влади, алгоритм симетричного, тобто з використанням однакового ключа, шифрування-дешифрування блоку даних фіксованої довжини. До теперішнього часу найбільшого поширення набув шифроблок Rijndael [5], створений в рамках міжнародного проекту AES для заміни алгоритму шифрування першого покоління – DES. В ряді країн та окремих міжнародних корпораціях стандартизовано інші шифроблоки – фіналісти AES: MARS, SERPENT, RC-6 [6].

В структурному плані шифроблок в режимі шифрування можна розглядати як функціональний перетворювач φ, на входи якого подаються m-розрядний блок D даних та k–розрядний ключ K, а на виході формується m–розрядний блок C зашифрованих даних: $C = \varphi(D,K)$. Відповідно, в режимі дешифрування шифроблок реалізує зворотне перетворення φ^{-1}: на його входи подаються $\boldsymbol{m}$-розрядний блок C зашифрованих даних та k–розрядний ключ K, а на виході відновлюється m-розрядний блок даних $D=\varphi^{-1}(C,K)$. Для шифроблоку Rijndael регламентовано шифрування блоків, довжиною 128, 192 або 256 розрядів. В пропонованому методі використовується режим шифрування блоків довжиною 256 бітів (m = 256).

Стандартизований хеш-перетворювач (H) – сертифікований відповідними органами алгоритм незворотного перетворення інформаційного блоку довільної довжини в код хеш-сигнатури фіксованої розрядності h. Найбільш відомими є хеш-перетворювачі SHA-1 та RIPEMD-160 [1], що формують 160-бітову хеш-сигнатуру (h=160). Найважливішою якістю хеш-перетворювачів є їх незворотність – тобто практична неможливість віднаходження інформаційного блоку, хеш-сигнатура якого дорівнює заданій.

Для ефективної реалізації концепції нульових знань стосовно ідентифікації віддалених абонентів пропонується використання шифроблоку, довжина m блоку даних якого суттєво більша за розрядність h хеш-сигнатури. Зокрема, для практичного застосування методу пропонується використання шифроблоку з розрядністю блоку даних 256 (m=256) та хеш-перетворювачів з розрядністю хеш-сигнатури 160 (h=160). Різниця розрядностей блоку даних ШБ та хеш-сигнатури надалі позначається як $d=m-h$.

Головна перевага використання стандартизованих шифроблоків та хеш-перетворювачів полягає в тому, що їх криптостійкість надійно перевірена спеціальним тестуванням та в процесі практичного застосування.

Пропонований метод регламентує процедури ініціалізації та сеансової ідентифікації при кожному зверненні абоненту до системи.

Процедура ініціалізації включає два етапи:

1) Абонент випадковим чином формує d–розрядний двійковий код z, а також m–розрядний код R ідентифікатора абонента. Коди z та R в закритому режимі передаються в систему.

2) Абонент довільним чином обирає h-розрядний код q_n, після чого послідовно, з використанням стандартизованого хеш-перетворення H, обчислює n-1 h-розрядних кодів $q_{n-1}, q_{n-2},\ldots,q_2, q_1$ так, що кожен наступний з цих кодів являє собою хеш-сигнатуру попереднього, тобто, для кожного $i\in\{1,2,\ldots,n-1\}$: $q_i = H(q_{i+1})$. Сформовані таким чином коди зберігаються лише у абонента.

Процедура –то сеансу ідентифікації абонента полягає у виконанні наступної послідовності дій:

1) Абонент формує m-розрядний код сеансового ключа K_i як конкатенацію –розрядного коду q_i та d-озрядного коду z: $K_i = q_i|z$.

2) Абонент формується m-розрядний код U_i як результат зворотного шифрування коду R ключем K_i : $U_i = \varphi^{-1}(R,K_i)$. Пара кодів q_i та U_i утворюють (h+m)-бітовий i-тий сеансовий пароль: $P_i = < q_i, U_i >$.

3) Сеансовий пароль P_i передається системі.

4) Система розділяє отриманий сеансовий пароль на складові q_i та U_i.

5) Перевіряється правильність отриманого коду q_i шляхом перевірки того, що результат хеш-перетворення q_i збігається з кодом q_{i-1} попереднього сеансу ідентифікації, тобто перевіряється умова: $q_{i-1} = H(q_i)$. Якщо ця умова не виконується, то сеанс ідентифікації переривається.

6) Система конкатенацією отриманого коду q_i та секретного коду z формує сеансовий ключ K_i : $K_i = q_i|z$.

7) З використання сеансового ключа система виконує пряме шифрування m-розрядного коду U_i : $Y=\varphi(U_i, K_i)$.

8) Якщо отриманий в результаті шифрування код Y дорівнює коду доступу R абонента, тобто

$Y=R$, то ідентифікація абонента вважається успішною.

Таким чином, розроблений метод забезпечує ідентифікації віддалених абонентів з використанням паролей, що змінюються при кожному сеансі. При цьому система не здатна генерувати пароль, а може лише перевіряти його коректність. Перевірка виконується не через порівняння, а шляхом використанням незворотних перетворень. Відповідно, запропонований метод реалізує ідентифікацію, що цілком вкладається в рамки теоретичної концепції "нульових знань".

Оцінка ефективності

Порівняльна оцінка ефективності запропонованого методу ідентифікації віддалених абонентів може бути виконана за наступними критеріями:

- рівень захищеності від спроб отримати незаконний доступ до ресурсів системи як з боку сторонньої зловмисника, так і з боку осіб, що незаконно мають доступ до пам'яті системи;
- обчислювальна складність процедури ідентифікації системою;
- можливості апаратної реалізації ідентифікації системою, яка дозволяє збільшити рівень захищеності та прискорити процедуру ідентифікації;
-використання потенційно відкритих ліній обміну даними: чим менше використовуються такі лінії в процесі ідентифікації, тим менші можливостей незаконного доступу до ресурсів системи.

Запропонований метод реалізує концепцію строгої ідентифікації віддалених абонентів з використаннясм стандартизованих криптографічних примітивів.

Очевидно, що при кожному зверненні до системи абонент використовує інший пароль.

Сеансовий пароль i-го сеансу ідентифікації складається з двох складових q_i та U_i. В разі перехоплення послідовності паролей $P_1, P_2,\ldots,P_i$ стороннім зловмисником, його ціллю може бути відтвореня паролю P_{i+1} для наступного, $(i+1)$-го сеансу ідентифікації. Цей пароль складається з двох компонентів: h-розрядного коду q_{i+1} та m-розрядного коду U_{i+1}. Код q_{i+1} пов'язаний з кодом q_i попереднього сеансового пароля через співвідношення: $q_i = H(q_{i+1})$, тобто, для реконструкції q_{i+1} потрібно виконати зворотнє хеш-перетворення. Для стандартизованих хеш-перетворень єдиним методом реалізації зворотного перетворення є перебір. Іншими словами, для вирішення задачі реконструкції коду q_{i+1} по значенню попередньої компоненти q_i методом перебору, в середньому, потрібно виконати 2^{h-1} прорахунків хеш-перетвореня. Зокрема, для стандартизованого хеш-алгоритму SHA-1 потрібно перебрати, в середньому, 2^{159} варіантів.

Для реконструкції коду U_{i+1} сторонньому зловмиснику потрібно підібрати такий код, щоб виконувалася умова: $U_{i+1} = \varphi^{-1}(R,K_{i+1})$, при невідомих значеннях m-розрядних кодів R та K_{i+1}. Технологічно, підбір доцільно виконувати по відомому значенню U_i, з використаннім рівняння $\varphi(U_i, K_i) = \varphi(U_{i+1}, K_{i+1})$. Якщо до цього сторонній зловмисник теоретично зможе реконструювати код q_{i+1}, то йому потрібно підібрати m-розрядний код U_{i+1} для кожного з можливих значень d-розрядного коду z – закритої компоненти ключа K. Таким чином, об'єм перебору становить $2^{m+d} = 2^{2\cdot m-h}$. При використанні в якості шифроблоку алгоритму Rijndael, а в якості хеш-перетворення SHA-1, m=256 і h=160, тобто обєм перебору для реконструкції коду U_{i+1} може буди оцінений як 2^{352} реалізацій процедури шифрування за алгоритмом Rijndael, що виходить за рамки сучасних технічних можливостей.

Для імітації звернення абонента самою системою або реалізації незаконного доступу за наявності в розпорядженні зловмисника секретних кодів R та z, що зберігаються в системі, потрібно виконати реконструкцію коду q_{i+1} по значенню попереднього коду q_i. Як зазначалося вище, використання стандартизованого хеш-перетворює унеможливлює застосування для цього інших методів крім перебору. При цьому, в середньому, потрібно виконати 2^{h-1} прорахунків хеш-перетвореня, що для стандартизованого хеш-алгоритму SHA-1 становить 2^{159} реалізацій алгоритму і виходить за рамки практичної доцільності.

В перспективі, рівень захищеності запропонованого методу строгої ідентифікації, що реалізує концепцію "нульових знань" може бути підвищений за рахунок простого збільшення розрядності шифроблоку та хеш-сигнатури.

Найбільш важлива перевага запропонованого методу реалізації ідентифікації на основі концепції "нульових знань" в порівнянні з існуючими методами (схеми Fiat-Shamir, Schnorr) полягає в тому, що він забезпечує значне зменшення обчислювальної складності процедури ідентифікації. Більшість існуючих методів ідентифікації віддалених користувачів, що реалізують концепцію "нульових знань" мають за основу незворотні перетворення теорії чисел, базовою операцією яких є модулярне експоненціювання над числами великої розрядності – 2048 або 4096. За оцінками [4] обчислювальна складність операції експонен-

ціювання для розрядності 2048 на три порядки перевищує обчислювальну складність шифрування з використанням стандартизованого шиброблоку. Це означає, що обчислювальна складність запропонованого методу реалізації строгої ідентифікації на три порядки менша в порівнянні з відомими методами, що реалізують концепцію "нульових знань". На практиці це означає, що запропонований метод дозволяє на три порядки скоротити час ідентифікації абонента, що критично важливо для ефективності систем колективного доступу.

Значимою для практики перевагою запропонованого методу в порівнянні з відомими є також те, що шифроблоки та хеш-перетворення побудовані на логічних операціях, які просто та ефективно можуть бути реалізовані апаратними засобами, в тому числі ПЛІС [7]. Апаратна реалізація операцій модулярного експоненціювання над числами великої розрядності, що лежать в основі відомих методів строгої ідентифікації, суттєво більш складна і менш ефективна.

Орієнтація розробленого методу на використання стандартизованих шифроблоків та хеш-перетворень відкриває широкі можливості його реалізації з використанням мікросхем криптопроцесорів, що випускаються промисловістю провідних країн світу. Такі криптопроцесори на апаратному рівні реалізують стандартизовані шифроблоки та хеш-перетворення.

На відміну від відомих методів, для одного сеансу ідентифікації використовується лише одна пересилка пароля через потенційно відкриті лінії передачі даних.

Висновки

В результаті проведених досліджень запропоновано новий метод строгої, тобто такої, що має за основу концепцію "нульових знань", ідентифікації віддалених абонентів. Відмінність методу полягає в використанні стандартизований криптографічних примітивів: шифроблоків та хеш-перетворювачів.

Використання добре досліджених криптографічних примітивів, що пройшли перевіркою практичним застосуванням, забезпечує високий рівень надійності захисту від спроб порушення ідентифікації.

З іншого боку, обчислювальна складність шифроблоків та хеш-перетворювачів на три порядки менша за складність операції модулярного експоненціювання аналогічної розрядності. Це означає, що запропонований метод потребує на порядки менше операцій для своєї реалізації. Експериментальні дослідження показали, що реально швидкість програмної реалізації запропонованого методу строгої ідентифікації на 2-3 порядки вище в порівнянні з відомими методами FFSIS та Гіллоу-Квіскватера.

Список посилань

1. Schneier B. Applied Cryptography. Protocols. Algorithms and Source codes in C. -Ed.John Wiley, 1996 - 758 pp.
2. Feige U. Zero knowledge proofs of identity / Feige U.,Fiat A.,Shamir A.// Journal of Cryptology.- v.1.- №.2.- 1988, pp.77-94.
3. Pourand G. A realistic security analysis of identification schemes based on combinatorial problems // European Transactions on Telecommunications.- V.8,-№.5.- 1997,- pp.471-480.
4. Markovskyy O. Fast subscriber identification based on the zero knowledge principle for multimedia content distribution/ O. Markovskyy, N. Bardis, N. Doukas // International Journal of Multimedia Intelligence and Security 2010 - Vol. 1,- pp.78-82.
5. Зенин О.С. Стандарт криптографической защиты AES./ Зенин О.С., Иванов М.А. - М.:Из-во Кудиц-Образ.- 2002.- 174 с.
6. Elbirt A. An FPGA Implementation and Performance Evaluation of the AES Block Cipher Candidate Algorithm Finalists // Proceedings of The Third Advanced Encryption Standard Candidate Conference. NIST, Gaithersburg, MD,- 2000.- pp 13-27.
7. Марковский А.П. Аппаратная реализация алгоритма Rijndael на FPGA-структурах./ А.П. Марковский, В.А. Стефанская, С.В. Грищенко // Вісник Національного технічного університету України "КПІ" Інформатика, управління та обчислювальна техніка- № 38.- 2002,- С.24-33.

УДК 004.75

ТЕЛЕНИК С.Ф.,
ТРОЦЬКИЙ С.О.

ВИКОРИСТАННЯ КОМПОНЕНТНОГО ПІДХОДУ ДО ПРОЕКТУВАННЯ ІНТЕЛЕКТУАЛЬНИХ АДАПТИВНИХ СЕРВІСІВ

В останні роки розробка на основі компонентів стала визнаним підходом. Розглянуто метод, що дозволяє спроектувати сервіс, вихід з ладу якого, призводить до автоматичного перерозподілу навантаження в системі. Компонентом ІАС-сервісу є бібліотека (файл з розширенням dll), яка написана для .NET Framework та має головний клас, що реалізує деякий API. Компоненти зв'язуються на етапі виконання між собою за допомогою декларацій в конфігураційному файлі. Завантажувач компонент автоматично ініціює компоненти згідно конфігурації.

In recent years component-based development has become an established approach. The method, which allows to design services in a such way, that failure of any leads to an automatic load redistribution is considered in the article. Components of IAS service are the libraries (files with extension dll), written for the .NET Framework, and which have the main class that implements specific API. The components are linked to each other at runtime based on configuration file. Loader of the components automatically initiates the components according to the configuration.

Ключові слова: компоненто-базований підхід, сервіс-орієнтована архітектура, модуль, проектування

Вступ

В умовах розвитку сервіс-орієнтованого підходу фінансові установи переходять до єдиної методології управління ризиками. При цьому розрахунки ризиків, потенційних втрат та інших важливих для прийняття рішень показників у сучасних фінансових установах базуються на документі Базель III. Цей документ Базельського комітету з банківської нагляду містить відповідні методичні рекомендації в сфері банківського регулювання. Усталений підхід до проектування інформаційних систем (ІС) банків передбачає, що всі розрахунки з одного напряму діяльності (підсистеми, функції, бізнес-процесу) здійснюються в одній системі у зоні відповідальності одного департаменту, а їх результати потім використовуються всіма іншими підсистемами, функціями, бізнес-процесами) банку у зоні відповідальності різних департаментів.

З точки зору побудови ІС банків на єдиній інфраструктурі інформаційних технологій (ІТ-інфраструктурі) при реалізації такого підходу виникає низка проблем. За складністю і терміновістю виділяються проблеми проектування (створення, вибору, модифікації) сервісів та інтеграції як такої системи з іншими системами, наприклад системою HFT (high frequency trading), так і її компонентів між собою. Ці складність і терміновість викликані високим рівнем вимог до ІС банків загалом і важливістю з огляду на потреби ефективної реалізації сервісного підходу в системах такого класу. Насамперед потрібно особливу увагу приділити забезпеченню високих вимог до таких параметрів системи:

1) Максимальний час відповіді з врахуванням часу передачі даних (при їх великих об'ємах);
2) Відмовостійкість, легкість підтримки і масштабованість;
3) Можливість розгортання системи в різних географічних зонах та ін.

Загальна процедура під час виходу з ладу банківської системи (її підсистеми) полягає у надсиланні до команди її підтримки відповідного повідомлення (за допомогою програми автоматичного фіксування виходу системи з ладу) і реакції на нього команди підтримки (в разі необхідності перехід на деякий час на запасну систему (підсистему) – DR (disaster recovery). Недоліком процедури є те, що вона не сприяє задоволенню високих вимог до параметрів ІС.

Постановка задачі

Інтелектуальні адаптивні сервіси (ІАС) – це наділені інтелектуальними можливостями сервіси, здатні перерозподілити функціональність і

навантаження між вузлами (сервісами) ІТ-середовища (на сьогодні це хмари), в якому вони базуються. Саме сервіси цього типу можна розглядати як основу надання ІС банків нових якостей. З іншого боку, їх можна розглядати як архітектурний елемент компоненто-базованого підходу до проектування, реалізації і експлуатації ІС банків, основу нової гнучкої і ефективної методології створення, експлуатації і модернізації ІС. Крім того, це ще й динамічний сектор ринку ІТ. Ефективне виконання сервісів ІАС забезпечують системи, що мають архітектуру, зображену на рисунку 1.

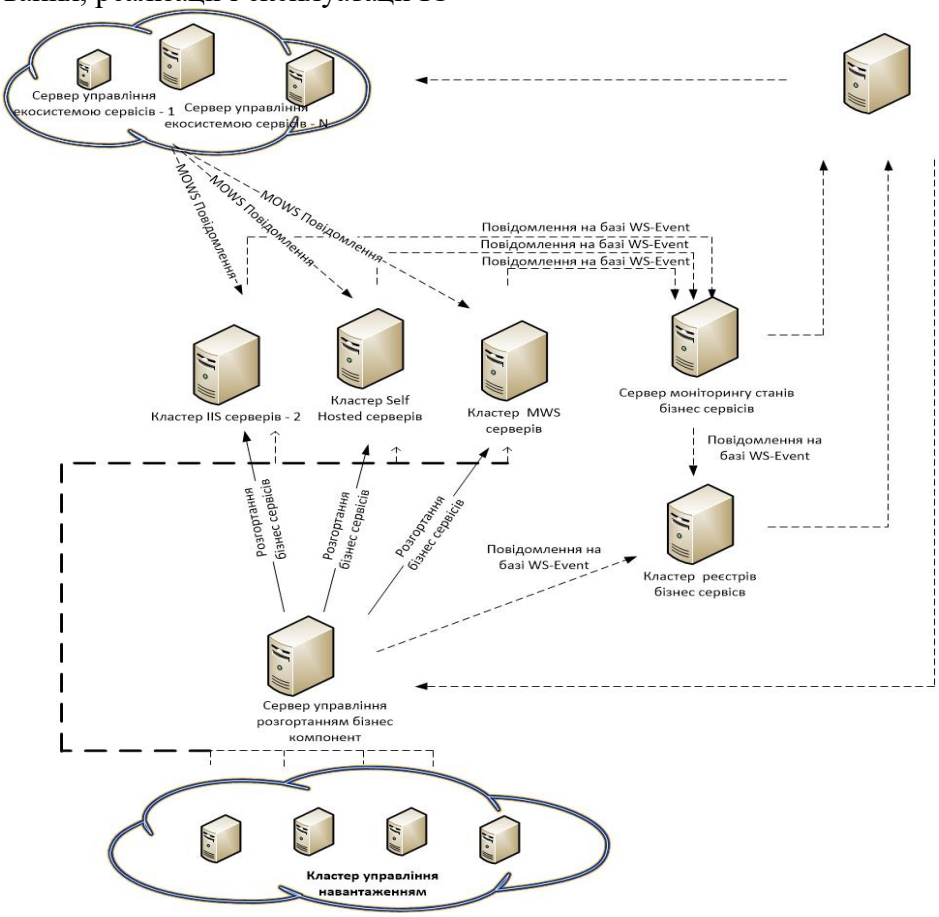


Рис. 1 Загальна архітектура системи виконання ІАС сервісів

Важливими для забезпечення високого рівня вимог до ІС банків особливостями цих сервісів є автоматичне детектування збою сервісів, відповідальних за бізнес-процеси, перерозподіл бізнес-процесів між іншими сервісами, перезапуск згідно конфігурації модуля компонента, який відповідає за виконання бізнес-процесу, можливість виконання додатків, що були створенні для .NET Framework в режимі сервісу SOA згідно BPML документу, навіть якщо ці додатки не бути спочатку створені для виконання в режимі сервісу.

Якщо бізнес-процес має певний внутрішній стан і виникає проблема під час його виконання, модуль компонента, який відповідає за виконання бізнес-процесу, ініціює його перезапуск згідно конфігурації, навіть за використання ланцюга бізнес-процесів.

Компонентом сервісу ІАС є написана для .NET Framework бібліотека (файл з розширенням dll), яка має головний клас, що реалізує деякий API. Компоненти зв'язуються між собою на етапі виконання за допомогою декларацій в конфігураційному файлі. Завантажувач компонент автоматично ініціює їх згідно конфігурації. Модуль компонента – це логічна програмна абстракція, шо виконує специфічну для компонента функцію. Кожен компонент має декілька (від 1 до *K*) модулів в залежності від конфігурації. Різні компоненти можуть використовувати той же модуль. Кластер управління навантаженням може містити декілька (від 1 до *N*) серверів, в залежності від максимальної дозволенної затримки згідно специфікації.

Огляд існуючих рішень

Проблема інтеграції розрізненої інформації сервісів з метою її подальшого опрацювання та прийняття рішень на її основі постала разом із

появою сховищ даних, ще у 80-х роках минулого століття. Передумовою її виникнення було зростання інтересу до розподілених баз даних та їх масове впровадження у бізнесові структури [1]. У цій же праці визначені методи роботи з джерелами з різними структурами даних:

1. Пошук інформаційних джерел на основі метаданих.
2. Відбір джерел з використанням онтологічних визначень.
3. Інтеграція на рівні сховищ даних.

Оскільки кожен IAC сервіс має свій набір метаданих, то найкращим вибором буде метод пошуку сервісів на основі метаданих. Оскільки для специфікації метаданих сервісу використовують різні типи метаданих, то тип метаданих, визначений для описання сервісу, є частиною його моделі вимог і може не збігатись з типами метаданих інших сервісів [1].

Формальні моделі, підходи і інструменти інтеграції програмних застосувань можна знайти у багатьох літературних джерелах, зокрема [2-4].

Але сьогодні постає потреба в технологіях компоненто-базованого проектування і експлуатації інформаційних систем в розвинених IT-середовищах на основі поєднання сервіс-орієнтованого, процесного і композитного підходів.

Загальний підхід до вирішення проблеми

Сервер управління розгортанням бізнес-компонентів формує сервіс, який складається з: WS-компонентів загального використання (які базуються на WS-стандартах); компонентів, що інкапсулюють бізнес-логіку процесу (побудованих за специфікацією BPEL4WS + BPML); компонентів, які реалізують програмні додатки та розширення специфікацій. Загалом сервіс складається з 16 компонентів, основне призначення яких пов'язане з реалізацією певної функціональності, тобто з виконанням програмних модулів підтримки бізнес-процесів. Але в умовах побудови сервісів на засадах SOA вони повинні мати низку компонентів, що відповідають за ефективне управління внутрішнім станом самого сервісу і його взаємодію з іншими сервісами. Сервіс складається з таких рівнів:

- Прикладний;
- Сервісний;
- Комунікаційний.

Загальна структура сервісу зображена на рисунку 2.

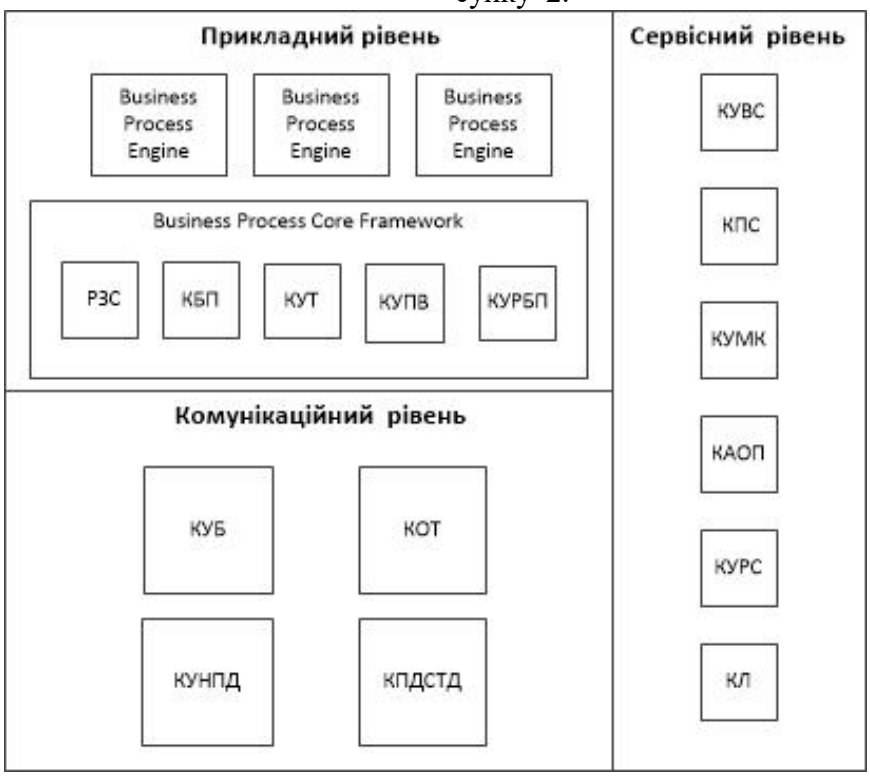


Рис. 2 Структура сервісу, розділеного на компоненти

Оскільки сервіс і його компонентна структура є найважливішими складовими компоненто-базованого підходу, розглянемо структура сервісу, розділеного на компоненти більш детально. Це дозволити не тільки описати призначення і можливості кожного компоненту, але й уточнити проблеми реалізації сервісів та шляхи їх вирішення, насамперед окреслити математи-

чні моделі й методи, здатні забезпечити заявлені можливості компонентів.

Розпочнемо з прикладного рівня, який відповідає за виконання програмних модулів бізнес-процесів, що описуються за допомогою BPEL4WS + BPML. Саме тут в програмних модулях реалізована певна функціональність, обумовлена потребами підтримки певних бізнес-процесів

Реєстр зв'язаних сервісів (РЗС) – компонент, який взаємодіє з КПС та на основі правил внутрішньої мови специфікації доменів – domain specific language (DSL), зв'язує себе та інші сервіси в кластері.

Кеш бізнес процесів (КБП) – компонент що реалізує буфер з внутрішнім станом бізнес-процесу на випадок перезавантаження процесу, його зупинки та старту. Завантажувач бізнес процесів виконується один раз під час запуску сервісу.

Компонент управління послідовностями викликів (КУПВ) – якщо бізнес-ланцюг складається з викликів від 1 до М бізнес-процесів, цей компонент зберігає інформацію про сервіс (бізнес-процес), до якого був зроблений виклик, щоб отримати дані, та координує повторні запити, якщо дані не надійшли.

Компонент управління транзакціями (КУТ) – компонент прикладного рівня, який застосовується, якщо бізнес процес або декілька бізнес процесів, що виконуються як єдине ціле потребують підтвердження атомічного виконання.

Компонент управління ресурсами бізнес-процесів (КУРБП) – компонент, який разом з КУРС забезпечує ефективне використання відповідних ресурсів.

Наступний розглянемо сервісний рівень, який відповідає за управління внутрішнім станом сервісу, забезпечує оптимізацію функціонування сервісу.

Компонент управління внутрішнім станом (КУВС) – компонент, який відповідає за збереження внутрішнього стану, його передачу в разі необхідності при виходу з ладу до диспетчера бізнес-сервісів, аналізу та збору внутрішніх характеристик та метрик.

Компонент пошуку сервісів (КПС) – компонент, який відповідає за пошук і визначення можливостей взаємодії з новими сервісами для підтримки певної функціональності.

Компонент аналізу та оптимізації продуктивності (КАОП) – компонент, який у взаємодії з компонентом КУВС відповідає за підтримку часу реакції сервісу на заданому рівні згідно SLA (service level agreement).

Компонент управління міжсервісним контекстом (КУМК) – компонент, який застосовується у разі необхідності передачі бізнес-процесу чи його частини (активності) зі збереженням стану та дані між сервісами. Схема передачі контексту зображена на рисунку 3. Літерою М позначаються повідомлення (messages), літерою К – контексти, що передаються між сервісами.

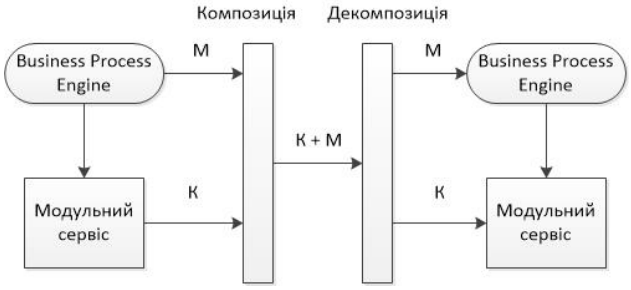


Рис. 3 Схема передачі контексту

Розширення функціональності будь-якого рівня зводиться до створення нових компонентів при одночасній декларації їх інтерфейсів. Завантажувач компонентів динамічно завантажує та ініціює компоненти, переналаштовує інші компоненти в залежності від інтерфейсу компонента та його конфігурації.

Компонент управління ресурсами сервісу (КУРС) – компонент, який разом з КУРБП забезпечує ефективне використання відповідних ресурсів..

Компонент логування (КЛ) – компонент сервісного рівня, який відповідає за зберігання та логування всіх подій сервісу для подальшого використання під час аналізу проблем чи малої продуктивності.

Комунікаційний рівень відповідає за формування точок доступу для інших сервісів на базі стандартів WS-Addressing, а також формування запитів до інших сервісів, забезпечуючи їх взаємодію з метою реалізації завдань, що вимагають використання можливостей сукупності сервісів.

Компонент управління надійністю передачі даних (КУНПД) – компонент, який відповідає за підтримку надійної комунікації між бізнес-процесом прикладного рівня та будь-яким іншим сервісом в разі відсутності мережі, проблем з компонентом, якщо це передбачено самим бізнес-процесом під час конфігурації.

Компонент управління безпекою (КУБ) – компонент комунікаційного рівня, який базується на стандартах WS-Security, WS-

Federation, WS-Security: Username Token Profile і забезпечує безпечну взаємодію з сервісами і між сервісами.

Компонент оптимізації трафіку (КОТ) – компонент, який в автоматичному режимі контролює швидкість передачі даних мережею, і за певним алгоритмом може змінювати параметри повідомлень, запитів.

Моделі і методи для реалізації підходу

Моделі планування схеми виконання сервісів. Для підтримки кожного бізнес-процесу чи певної їх системи визначається певна схема виконання програмних модулів підтримки бізнес-процесів. Інформація щодо кожного сервісу, компонента, модуля фіксується в системі (реєстр, інтерфейси та інші джерела) і використовується для створення схеми виконання програмних модулів підтримки бізнес-процесів. Зазначена схема будується на основі опису бізнес-процесу чи їх системи. Реалізовані ручний і автоматичний способи побудови зазначеної схеми. Перший реалізується адміністратором за допомогою зручного графічного інтерфейсу.

Другий базується на логічних моделях, які формалізують процес пошуку відповідно до заданого опису бізнес-процесу потрібних сервісів, компонентів, модулів і побудови структури їх викликів зі специфікацією передачі відповідних параметрів. Таким чином, створюється схема виконання програмних модулів підтримки бізнес-процесів. Відповідний компонент системи відстежує виконання викликів необхідних сервісів і відправляє оброблені дані у відповідності з ланцюжком дій. Це відбувається згідно побудованої схеми виконання програмних модулів підтримки бізнес-процесів.

Опис бізнес-процесу обробляється з урахуванням відомих системі сервісів, їх функціональності, поданих засобами логічного формалізму у вигляді системи аксіом і правил виведення. Сам процес побудови схеми виконання програмних модулів підтримки бізнес-процесів у такому випадку базується на певному методі логічного виведення і механізмі відтворення виводу у вигляді схеми виконання програмних модулів підтримки бізнес-процесів.

Логічний підхід видається прийнятним, оскільки дозволяє вийти на високий рівень вище вимог до інформаційних систем, позбутися недоліків традиційного алгоритмічного підходу, інтегрувати накопичений арсенал сервісів і навіть підключати компоненти, що не створювалися як сервіси. Для описання сервісів будемо використовувати логіку, запропоновану у праці [2]. У тій же праці наведений відповідний метод виведення.

Моделі оптимального використання ресурсів. Для пошуку і ефективного використання вільних ресурсів системи будемо використовувати моделі і методи, запропоновані у праці [5]. Там же наведені моделі управління навантаженням і розподілу ресурсів у разі їх дефіциту.

Особливості реалізації окремих компонентів

На найвищому рівні існують три інтерфейси, які повинні успадкувати компоненти (причому кожен компонент успадковує тільки один з цих трьох інтерфейсів):

IAapplicationLayer – інтерфейс, який представляє можливості доступу до функціональних ресурсів сервісів;

IServiceLayer – інтерфейс, який представляє можливості доступу до функцій управління сервісами;

ITransportLayer – інтерфейс, який представляє можливості доступу до комунікаційних ресурсів сервісів.

Також кожний компонент повинен успадкувати інтерфейс *IComponent* <*K*,*T*>, де *K* – вхідний інтерфейс, *T* – вихідний інтерфейс, а також базовий клас *Component*. Кожний із вхідних і вихідних інтерфейсів успадковує від інтерфейсу *IInterServiceCommunication* та базового класу *ServiceChannel*. Клас *ServiceChannel* успадковує від базового класу *BaseChannel*

Власне кажучи, сервіс перетворюють у структурний елемент компоненто-базованого підходу саме WS-компоненти загального використання, до яких належать:

- КУТ, реалізований на базі WS-Coordination, WS-Business Activity, WS-Atomic Transaction;
- КУПВ, реалізований на базі Web Service Choreography Interface;
- КУРБП, реалізований на базі WS-Resource, Web Services Resource Properties, WS-ResourceLifetime, WS-Service Group, WS-BaseFaults;
- КУБ, реалізований на базі WS-Security;
- КУМП, реалізований на базі WS-Context;
- КУРС, реалізований на базі WS-Resource, Web Services Resource Properties, WS-ResourceLifetime, WS-Service Group, WS-BaseFaults;

- КПС, реалізований на базі WS-Discovery.

До головних компонентів загального використання, які реалізують програмні додатки та методи їх виконання, належить компонент Business Process Engine. Він складається з таких модулів:

- верифікації даних на основі DSL-метадекларацій;
- перевірки BPML-документа, завантаженого до компоненту з метою виконання;
- виконання BPML-документа;
- перевірки та передачі стану BPML-документа під час або після виконання;
- завантаження та запуску додатків, створених для .NET Framework.

Модель сервісу, побудованого на базі компонентно-базованого підходу, базується на таких стандартах та специфікаціях:

1) специфікаціях бізнес-процесів:

a) специфікація Business Process Execution Language for Web Services (BPEL4WS) версії 2.0 [1];

b) специфікація XML Process Definition Language (XPDL) версії 2.0;

c) специфікація Business Process Management Language (BPML) версії 1.1 [6];

2) специфікації управління сервісом:

a) специфікація Management Of Web Services (MOWS) версії 1.1 [6];

3) специфікації управління ресурсами:

a) специфікація Web Services Resource Framework (WSRF) версії 1.2 [7]:

i) специфікація Web Services Resource (WS-Resource) версії 1.2 [8];

ii) специфікація Web Services Resource Properties (WS-ResourceProperties) версії 1.2 [9];

iii)специфікація Web Services Resource Lifetime (WS-ResourceLifetime) версії 1.2 [10];

iv)специфікація Web Services Service Group (WS-ServiceGroup) версії 1.2 [11];

v) специфікація Web Services Base Faults (WS-BaseFaults) версії 1.2 [12];

4) специфікації управління передачею повідомлень:

a) специфікація WS-Notification версії 1.3;

b) специфікація SOAP 1.2 [13];

5) специфікації управління безпечної передачі даних:

a) специфікація WS-Security версії 1.1 [14];

6) специфікація управління передачею та обміном міжсервісного контексту - WS-Context (WS-CTX) версії 1.0 [15].

На основі запропонованого підходу реалізовано програмний комплекс розгортання SOA-сервісів для системи компоненто-орієнтованого проектування Component Forest Suite. Його використання для створення і експлуатації інформаційних систем декількох установ підтвердило працездатність закладених рішень. При цьому вдалося до 20% поліпшити деякі параметри інформаційних систем, до яких звичайно висуваються високі вимоги, насамперед максимальний час відповіді і відмовостійкість. Це ж стосується важливої для банківських інформаційних систем легкості підтримки.

Висновки

Запропонований підхід до проектування, реалізації та підтримки сервісів, призначених для використання при створенні, реалізації і експлуатації ІС, побудованих на основі технологій компоненто-базованого проектування. На основі підходу розроблений програмний комплекс, який дозволяє будь-яку бібліотеку на базі .NET Framework розгорнути у вигляді SOA-сервісу з необхідною для реалізації ІС базовою функціональністю. Зазначений комплекс дозволяє здійснювати моніторинг, відстежувати внутрішній стан сервісу, динамічно перезавантажувати та переналаштовувати параметри сервісу та параметри бізнес-процесів, які виконуються за підтримки сервісу. Кожен сервіс може виконувати від одного до N бізнес-процесів в залежності від SLA та конфігурації.

Список посилань

1. Шаховська Н. Б. Сховища та простори даних – інформаційний фундамент систем прийняття рішень / Н. Б. Шаховська, Я. І. Виклюк // Електротехнічні та комп'ютерні системи. – № 08 (84). – 2012. – С. 93-95.
2. Теленик С.Ф. Логічний підхід до інтеграції програмних застосувань підтримки міждисциплінарних наукових досліджень / С.Ф. Теленик, О.А.Амонс, К.В.Єфремов, В.Т.Лиско // Наукові вісті НТУУ «КПІ». – №5 (91). – 2013. – С. 53-72.
3. Теленик С.Ф. Семантична інтеграція різнорідних інформаційних ресурсів / С.Ф. Теленик, О.А.Амонс, К.В.Єфремов, С.В.Жук // Вісник НТУУ «КПІ», Інформатика, управління та обчислювальна техніка. – №58. –2013. – С. 29 - 45.

4. Kikuchi S. Adaptive Integration of Distributed Semantic Web Data / S. Kikuchi, S. Sachdeva, S. Bhalla et al. // Berlin: Databases in Networked Information Systems, Volume 5999, «Springer Berlin Heidelberg», 2010. P. 174-193.
5. Telenyk Sergii. Models and methods of resource management for VPS hosting / Sergii Telenyk, Oleksandr Rolik, Maksym Bukasov, Dmytro Halusko // Technical Transactions series Automatic Control. Vol.4 – AC/2013. – P.41-52.
6. https://www.oasis-open.org/committees/download.php/20574/wsdm-mows-1.1-spec-os-01.pdf.
7. https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsrf#overview.
8. http://docs.oasis-open.org/wsrf/wsrf-ws_resource-1.2-spec-os.pdf.
9. http://docs.oasis-open.org/wsrf/wsrf-ws_resource_properties-1.2-spec-os.pdf.
10. http://docs.oasis-open.org/wsrf/wsrf-ws_resource_lifetime-1.2-spec-os.pdf.
11. http://docs.oasis-open.org/wsrf/wsrf-ws_service_group-1.2-spec-os.pdf.
12. http://docs.oasis-open.org/wsrf/wsrf-ws_base_faults-1.2-spec-os.pdf.
13. https://www.w3.org/TR/soap/.
14. https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss#overview.
15. http://docs.oasis-open.org/ws-caf/ws-context/v1.0/wsctx.pdf.

Відомості про авторів

Алєнін О. І.	заступник директора Центру суперкомп'ютерних обчислень НТУУ«КПІ ім. І. Сікорського»
Бабенко О. А.	студент кафедри системного проектування Навчально-наукового комплексу «ІПСА» НТУУ «КПІ ім. І. Сікорського»
Бронфман Є. Б.	студентка кафедри ОТ, ФІОТ НТУУ«КПІ ім. І. Сікорського»
Буйбарова М.Ф	студент кафедри ОТ, ФІОТ НТУУ«КПІ ім. І. Сікорського»
Винничук С. Д.	д.т.н., Інститут проблем моделювання в енергетиці ім. Г.Є. Пухова НАН України.
Виноградов Ю.М.	ст. викладач кафедри ОТ, ФІОТ НТУУ«КПІ ім. І. Сікорського»
Ганжа І. М.	студент кафедри ОТ, ФІОТ НТУУ«КПІ ім. І. Сікорського»
Гергало А.М.	аспірант Херсонського національного технічного університету
Гіоргізова-Гай В.Ш.	к.т.н., доцент кафедри системного проектування Навчально-наукового комплексу «ІПСА» НТУУ «КПІ ім. І. Сікорського»
Гломозда Д. К.	к.т.н., ст. викладач кафедри інформатики факультету інформатики Національного університету «Києво-Могилянська академія»
Гусєв Є.І.	асистент кафедри ОТ, ФІОТ НТУУ«КПІ ім. І. Сікорського»
Діброва М. А.	аспірант НТУУ«КПІ ім. І. Сікорського»
Долголенко А.О.	аспірант НТУУ«КПІ ім. І. Сікорського»
Жабін В.І.	професор, д.т.н., професор кафедри ОТ, ФІОТ НТУУ«КПІ ім. І. Сікорського»
Жабіна В.В.	к.т.н., доцент кафедри ПЗКС, ФІОТ НТУУ«КПІ ім. І. Сікорського»
Жаріков Е.В.	к.т.н., доцент кафедри АСОІУ, ФІОТ НТУУ«КПІ ім. І. Сікорського»
Заковоротний О. Ю.	к.т.н., доцент кафедри обчислювальна техніка та програмування, НТУ "Харьковский политехнический институт"
Захаріудакіс Лефтерис	ст. викладач навчального центру ім. Макарія Кіпр, Греція,
Капорін Р. М.	студент каф. АСОІУ, ФІОТ НТУУ«КПІ ім. І. Сікорського»
Клименко І. А.	доцент, к.т.н., доцент кафедри ОТ, ФІОТ НТУУ«КПІ ім. І. Сікорського»
Коган А. В.	к.т.н., асистент. кафедри АСОІУ, ФІОТ НТУУ«КПІ ім. І. Сікорського»
Костенко Ю. В.	студентка кафедри ОТ, ФІОТ НТУУ«КПІ ім. І. Сікорського»
Куц В.Ю.	к.т.н., ст. викладач кафедри ОТ, ФІОТ НТУУ«КПІ ім. І. Сікорського»
Максименко Є. В.	аспірант Інституту проблем моделювання в енергетиці ім. Г.Є. Пухова НАН України.
Марковський О.П.	к.т.н., доцент кафедри ОТ, ФІОТ НТУУ«КПІ ім. І. Сікорського»
Мурга М.О.	к.т.н., ст. викладач кафедри математичних методів системного аналізу ННК "ІПСА" НТУУ «КПІ ім. І. Сікорського»
Новотарський М. А	с.н.с., д.т.н., професор кафедри ОТ, ФІОТ НТУУ«КПІ ім. І. Сікорського»
Папуч Т. Ю.	аспірант ІПСА, НТУУ «КПІ ім. І. Сікорського»
Порєв В. М.	к.т.н., ст. викладач кафедри ОТ, ФІОТ НТУУ«КПІ ім. І. Сікорського»
Приймак В.Ю.	студент кафедри ОТ, ФІОТ НТУУ«КПІ ім. І. Сікорського»
Радченко К.О.	асистент кафедри ОТ, ФІОТ НТУУ«КПІ ім. І. Сікорського»
Рибак В. О.	завідуючий кафедрою програмного забезпечення Білоруської державної академії зв'язку
Ролік О.І.	д.т.н., професор кафедри АУТС НТУУ«КПІ ім. І. Сікорського», директор навчально-наукового центру інформаційно-комунікаційних технологій «Неткрекер -НТУУ "КПІ"»
Рудакова Г.В.	професор, д.т.н., зав.кафедри технічної кібернетики Херсонського національного технічного університету
Русанова О. В	доцент, к.т.н., доцент кафедри ОТ, ФІОТ НТУУ«КПІ ім. І. Сікорського»
Семенюк Ю.В.	студент кафедри ОТ, ФІОТ НТУУ«КПІ ім. І. Сікорського»
Сергієнко А. М.	д.т.н., с.н.с. кафедри ОТ, ФІОТ НТУУ«КПІ ім. І. Сікорського»
Сергієнко П. А.	студент кафедри СКС НТУУ «КПІ ім. І. Сікорського»
Скориченко О.В.	студент кафедри ОТ, ФІОТ НТУУ«КПІ ім. І. Сікорського»
Стіренко С. Г.	доцент, д.т.н., професор кафедри ОТ, ФІОТ НТУУ«КПІ ім. І. Сікорського»
Теленик С.Ф.	професор, д.т.н., завідувач. кафедри АУТС, ФІОТ НТУУ«КПІ ім. І. Сікорського»
Троцький С.О.	аспірант кафедри АУТС, ФІОТ НТУУ«КПІ ім. І. Сікорського»
Федотов М.Ф.	студент кафедри ОТ, ФІОТ НТУУ«КПІ ім. І. Сікорського»
Шевченко О.М.	ст. викладач кафедри іноземних мов НТУУ«КПІ ім. І. Сікорського»
Шокр Ахмад	аспірант Білоруського національного технічного університету
Шемсединов Т. Г.	асистент. кафедри ТК, ФІОТ НТУУ«КПІ ім. І. Сікорського»
Яцун В.О.	студент кафедри ОТ, ФІОТ НТУУ«КПІ ім. І. Сікорського»

SUMMARY

Babenko O., Hiorhizova-Hai V. Load balancing of applications in a private cloud environment…….. 4

Vynnychuk S., Maksymenko E. Repeated decimation for accelerating the factorization Fermat's method with uneven steps for the unknown……. 13

Kuts V. Structure of the computer for multiscale phase meter based on modular arithmetic……..… 25

Rusanova O. Scheduling method for multi-core clusters…… 31

Hlomozda D. Application of invariants method to analysis of colored Petri nets with deadlocks……. 38

Sergiyenko A., Klimenko I., Sergiyenko P. Reconfigurable many-core computer based on FPGA…….. 47

Kostenko Y., Markovskyy O., Rusanova O. Method for protected modular exponentiation on remote computer systems…… 51

Herhalo A., Rudakova A. The problems automation process of functional diagnostics digital devices power accounting……. 55

Markovskyy O., Semenuk J., Shevzhenko O., Radchenko K. Method for data transmission errors correction in singl-chip radio modems…….. 60

Rybak V., Chokr Ahmad Using of artificial intelligence technologies of solutions in the field of environmental management tasks…… 68

Zakovorotniy A. Expanding the geometric control theory by automating the analysis transformation in MATLAB package……... 76

Porev V. Algorithmic aspects computer-aided design curricula…….. 84

Telenyk S., Rolik O., Zharikov E. Virtual machine allocation management in the data center…….. 90

Novotarskyi M., Bronfman E. Simulation of fluid flow in a three-dimensional area with moving boundaries…… 100

Dolgolenko A., Yatsun V. Implementation of operational device of floating point adder/subtractor for superscalar processor kernel…… 106

Papuch T. Integration of bankruptcy risk evaluation methods and models adopted for estimating the creditworthiness of enterprises…… 117

Bujbarova M., Vynogradov Y., Priymak V. Method for protected fourier transforms on remote distributed computer systems……. 125

Gusev E. Modeling of access techniques to shared memory pages in cloud computing systems based on shared everything architecture……. 133

Murga N. Space with changing during the time basis of features' memberships for financial-economic processes' indicators forecasting…….. 138

Dibrova M., Kohan A., Kaporin R. Methods of organization multipath routing……... 145

Zhabin V., Zhabina V., Skorichenko A. Implementation of on-line computing in redundant number systems on FPGAs……... 150

Hanzha I., Shemsedinov T., Alienin O,. Stirenko S. Modification and implementation of classical algorithm for marking connected components in binary image……..…… 156

Markovskiy O., Zacharioudakis Eleftherios., Fedotov M. Method of stringent remote abonents identification based onstandard block cipher and hash transformation……... 161

Telenyk S., Trotskyi S. Usage of component based approach during intelligent adaptive services design phase…… 166

Відомості про авторів……..……...…….. 173

Summary……..……...……..... 174