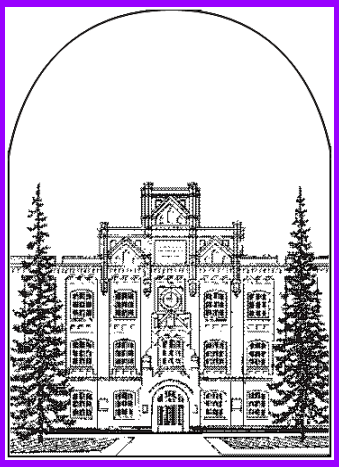
ВІСНИК

НАЦІОНАЛЬНОГО ТЕХНІЧНОГО УНІВЕРСИТЕТУ УКРАЇНИ
"Київський політехнічний інститут"

Інформатика, управління та обчислювальна техніка



60

2014

Міністерство освіти і науки України
Національний технічний університет України «КПІ»

ВІСНИК

НАЦІОНАЛЬНОГО ТЕХНІЧНОГО УНІВЕРСИТЕТА УКРАЇНИ «КПІ»

Інформатика, управління та обчислювальна техніка

Заснований у 1964 р.
Випуск 60

Київ “ВЕК+” 2014

УДК 004

Вісник НТУУ «КПІ». Інформатика, управління та обчислювальна техніка: Зб. наук. пр. – К.: Век+, – 2014. – № 60. – 131 с.

Рекомендований до друку Вченою радою факультету інформатики та обчислювальної техніки, протокол № 10 від 26.05.2014

Головний редактор: Луцький Г.М., д.т.н., проф.

Заст. гол. ред.: Стіренко С.Г., к.т.н., доц.,
Відповідальний секретар: Поспішний О.С.

Редакційна колегія: Павлов О.А., д.т.н., проф.,
Теленик С.Ф., д.т.н., проф.,
Бузовський О.В., д.т.н., проф.,
Симоненко В.П., д.т.н., проф.,
Жабін В.І., д.т.н., проф.,
Кулаков Ю.О., д.т.н., проф.,
Марковський О.П., к.т.н., доц.,
Стенін Н.А., д.т.н., проф.,
Томашевський В.М., д.т.н., проф.
Ляшевський С.Е., д.т.н., проф.

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.

ISSN 2310-3620

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 26.05.2014. 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”, 2014

ЗМІСТ

Павлов А.А., *Мисюра Е.Б.* ПДС-алгоритмы решения задач составления расписаний по критерию опережения/запаздывания на одном приборе…… 4

Амонс О.А. Синхронізація даних в розподілених інформаційних системах……... 20

Марковський О.П., Терещенко С.Ю., Федоречко О.І. Метод виправлення трьохкратних помилок передачі даних в двійкових симетричних каналах…… 33

Павлов А.А., Калашник В.В. Рекомендации по выбору зоны проведения активного эксперимента для одномерного полиномиального регрессионного анализа…….. 41

Марковський О.П., Іванов Д.Г., Великий М.М., Невдащенко М.В. Метод резервування та прискореного відновлення даних в системах їх віддаленого зберігання……..…….. 46

Стіренко С.Г. Метод перебалансування навантаження в кластерних компютерних системах в динамічному режимі…… 55

Марковський О.П., Губська А.О., Чечель В.Р., Федоречко О.І. Метод корекції двох помилок синхронізації в асинхронних лініях передачі цифрових даних…… 60

Роковой А.П., Габинет А.В. Способ прогнозирования дисперсии задержки при передаче голосовых данных…… 68

Бузовский О.В., Подрубайло А.А. Методы и алгоритмы объединения таблиц для распределенных хранилищ данных в оперативной памяти……... 73

Луцький Г.М., Долголенко О.М., Аксьоненко С.В., Сторожук В.О. Моделювання обмеженої реалізації архітектури потоку даних в структурі суперскалярного процесора…… 83

Короненко А.М. Дослідження динаміки навантаження мультисервісної мережі…….. 95

Зимчук И.В. Редукция моделей объектов управления для синтеза простых регуляторов…… 102

Гусев Е.И. Математическое моделирование распределённой кластерной системы использующей shared everything подход (ORACLE RAC)…….. 106

Сергієнко А.М., Сергієнко П.А. Реалізація функції квадратного кореня у ПЛІС…….. 114

Ролик А.И. Методика оперативного управління рівнем IT-послуг на основі сервісно-ресурсних моделей……. 119

Відомості про авторів…….. 130

Summary……. 131

УДК 519.854.2

ПАВЛОВ А.А.,
МИСЮРА Е.Б.

ПДС-АЛГОРИТМЫ РЕШЕНИЯ ЗАДАЧ СОСТАВЛЕНИЯ РАСПИСАНИЙ ПО КРИТЕРИЮ ОПЕРЕЖЕНИЯ/ЗАПАЗДЫВАНИЯ НА ОДНОМ ПРИБОРЕ

Предложены эффективные ПДС-алгоритмы решения NP-трудной задачи опережения/запаздывания для случаев, когда момент начала выполнения заданий фиксирован или находится в интервале времени $[t_1, t_k]$. Решение основано на ПДС-алгоритме решения задачи минимизации суммарного запаздывания заданий. Разработан эвристический алгоритм определения самого позднего момента начала выполнения заданий, при котором достигается минимальное значение функционала. Приведены примеры решения задач и результаты экспериментальных исследований.

The effective PDC-algorithms are proposed for the solution of NP-hard earliness/tardiness problem for the cases when the moment of the tasks execution start is fixed or is in the time interval $[t_1, t_k]$. The solution is based on the PDC-algorithm for the total tardiness minimization problem. A heuristic algorithm is developed to define the latest moment of the tasks execution start at which the minimal functional value is achieved. An examples of the problems solution and the results of the experiments are given.

Введение

Составлению расписаний относительно директивных сроков уделено значительное внимание в литературе. Одна из причин этого явления – возрастающая конкуренция на международных рынках: фирмы должны предложить большое разнообразие индивидуальных изделий, в то время как клиенты ожидают, что заказанные товары будут поставлены своевременно. Чтобы удовлетворить эти виды требований, были разработаны такие принципы и философии, как управление отклонением, одновременная разработка, производство «точно в срок» и т.д. Например, принцип «точно в срок» устанавливает, что необходимое количество товаров должна быть произведено и поставлено точно в необходимое время. Таким образом, можно ожидать, что задания будут выполнены вовремя, потому что запаздывающие поставки, так же, как и опережающие, рассматриваются как нежелательные [1, 2, 3, 4].

В то время как опережение заданий связано с затратами на складирование, запаздывание заданий связано с такими штрафами, как потеря доброжелательности клиента и потеря заказов. Поэтому задача минимизации опережения и запаздывания (О/З) очень важна в рыночной экономике.

Большинство исследований по задачам О/З связаны с моделями для одного прибора, хотя некоторые результаты для одного прибора были расширены на задачи для параллельных приборов. Некоторые из простейших результатов для задач О/З были получены для моделей, в которых все задания имеют общий директивный срок ($d_j = d$).

В данной статье мы рассматриваем задачу составления расписаний по критерию минимизации суммарного опережения и запаздывания при выполнении заданий одним прибором (О/З), которая может быть сформулирована таким образом.

Постановка задачи О/З

Множество из n независимых заданий $J = \{j_1, j_2, ..., j_n\}$ должно быть назначено без прерываний на один прибор, который может выполнять не более, чем одно задние одновременно. Прибор и задания предполагаются постоянно доступными с момента времени ноль, а простои прибора не допускаются. Задание j, где $j = 1, 2, ..., n$, требует времени выполнения l_j и должна в идеале быть завершена в свой директивный срок d_j. Для любого заданного расписания опережение и запаздывание задания j могут быть определены выражениями, соответственно:

$$E_j = \max(0, d_j - C_j) = (d_j - C_j)^+ \quad (1)$$

$$T_j = \max(0, C_j - d_j) = (C_j - d_j)^+, \quad (2)$$

где C_j – момент завершения задания j.

Цель состоит в том, чтобы найти расписание, минимизирующее суммарное опережение и запаздывание всех заданий:

$$\sum_{j=1}^{n}\left(E_j + T_j\right). \quad (3)$$

Задача относится к классу NP-трудных.

Рассматриваются следующие задачи:

ОЗ1. Момент начала выполнения заданий r фиксированный и равен нулю;

ОЗ2. Момент начала выполнения заданий находится в интервале времени $[t_1, t_k]$, $t_k > t_1 > 0$;

ОЗ3. Момент начала выполнения заданий не фиксирован.

Данные задачи включены в состав математического обеспечения четырехуровневой модели планирования, принятия решений и оперативного управления в сетевых системах с ограниченными ресурсами [5]: на третьем уровне при построении плана выполнения заданий и на четвертом уровне при оперативной коррекции построенного плана в случае его частичного невыполнения.

Задачи опережения и запаздывания рассматривались в работе Кэннета [6]. С использованием системы обозначений Лоулера и др. [7] статическая задача составления расписаний О/З может быть описана как $1||\Sigma(E_j + T_j)$. Бейкер и Скуддер [8] приводят обзор открытой литературы по задаче составления расписаний О/З, описывая разные модели О/З и разные штрафы за опережение и запаздывание заданий. Полезный обзор задачи также приведён в книге [9]. Когда директивные сроки задач определены, но различны, задачи О/З являются NP-полными [10]. Яно и Ким [11] разработали эвристику для частного случая взвешенных штрафов, в котором вес пропорционален длительностям заданий. Они используют процедуру попарной перестановки, чтобы улучшить начальные эвристические решения.

Оу и Мортон [12] исследуют функцию на основе приоритета и разновидность фильтрованного лучевого поиска для задач О/З, делая вывод, что качество решения значительно ухудшается, если затраты на опережение заданий игнорируются в пользу рассмотрения только стоимости запаздывания. Планировщик Дэвиса и Кейнта [13] может быть встроен в существующее эвристическое решение для улучшения качества расписания О/З при низких вычислительных затратах. Шварц и Мухопадьяй [14] разработали подход группировки (кластеризации) для определения оптимального момента запуска заданий на примере задачи О/З при предположении, что последовательность заданий известна априорно. Подход группировки осуществляет оптимальный выбор времени для 500 заданий за несколько секунд вычислительного времени. Сридхаран и Жоу [15] используют эвристику диспетчирования для задач О/З, позволяющую вставку допустимого резерва прибора.

Алгоритм, описанный в статье [16], основан на табу-поиске. Алгоритмы для определения оптимальной последовательности относительно слабо исследованы. Шварцем [17] был предложен алгоритм ветвей и границ, основанный на сортировке смежных заданий с разными весами штрафов. Лии и Чой [18] предложили генетический алгоритм, а Яно и Ким [11] изучили некоторые доминантные свойства последовательностей заданий со штрафами, пропорциональными времени обработки.

Наиболее значительные результаты для расписаний с запрещенными и разрешенными нарушениями директивных сроков представлены в статьях Бэнка и Вэрнена [19], Гордона [20], Валенте, Альвеса [21], Фельдмана и Бискапа [1], а также Тунг-ай Тсая [22].

Хугевен и Ван де Вельд [23] предложили алгоритм, основанный на методе ветвей и границ. По причине того, что алгоритм имеет экспоненциальный рост времени вычисления, он может использоваться только для задач с количеством заданий, не превосходящим 20. Сложность решения задачи больших размерностей обусловила необходимость на практике почти всегда использовать приближенные алгоритмы.

В настоящей статье представлены эффективные ПДС-алгоритмы решения исследуемых задач ОЗ1 и ОЗ2.

ПДС-алгоритмы [24] – это алгоритмы с полиномиальной и экспоненциальной составляющими, при этом в некоторых случаях возможна декомпозиция на подзадачи меньшей размерности, каждая из которых решается независимо. Например, в задаче минимизации суммарного взвешенного момента окончания выполнения заданий [24, гл. 2] осуществляется декомпозиция на множества максимального приоритета, в каждом из которых оптимизация осуществляется независимо.

В отличие от известных алгоритмов, ПДС-алгоритмы относят решаемую индивидуальную задачу к подклассу полиномиально разрешимых в процессе анализа ее решения. С этой целью для каждой труднорешаемой задачи строго выводятся логико-аналитические условия, проверяемые в процессе ее решения, в случае выполнения которых данная произвольная индивидуальная задача точно решается полиномиальным алгоритмом. Логико-аналитические условия и полиномиальный алгоритм их проверки составляют полиномиальную составляющую ПДС-алгоритма.

Полиномиальный алгоритм, проверяющий признак оптимальности допустимого решения,

может быть: а) частью экспоненциальной составляющей либо полиномиальной аппроксимации экспоненциальной составляющей ПДС-алгоритма [24, гл. 1, 2]; б) совпадать полностью с полиномиальной аппроксимацией экспоненциальной составляющей ПДС-алгоритма [24, гл. 3]; в) отдельным полиномиальным алгоритмом, не совпадающим с экспоненциальной составляющей либо полиномиальной аппроксимацией экспоненциальной составляющей ПДС-алгоритма (алгоритмы, приведенные в настоящей статье).

Декомпозиционная составляющая [24, гл. 2] реализуется только тогда, когда теоретически строго полиномиальным алгоритмом можно декомпозировать задачу на подзадачи меньшей размерности.

Таким образом, необходимым и достаточным условием построения ПДС-алгоритмов является нахождение теоретически обоснованных признаков оптимальности допустимого решения. Из этого определения следует, что ПДС-алгоритмы бывают двух типов. ПДС-алгоритмы первого типа включают полиномиальную составляющую и экспоненциальную составляющую, которая представлена точным экспоненциальным алгоритмом, построенным как на теоретически обоснованных направленных перестановках, так и на эффективных отсечениях. В результате решения [25, 24, гл. 1] задачи либо реализуется полиномиальная составляющая алгоритма, либо после выполнения всех перестановок и отсечений получаем решение, оптимальность которого строго обоснована.

В ПДС-алгоритмах второго типа экспоненциальная составляющая представлена ее полиномиальной аппроксимацией, которая может быть либо приближенным полиномиальным алгоритмом, если получена оценка отклонения решения от оптимального, либо эвристическим алгоритмом, если такой оценки нет. В результате решения получаем: а) оптимальное значение функционала, если в процессе выполнения алгоритма реализованы признаки оптимальности получаемых решений; б) приближенное решение с оценкой отклонения от оптимального; в) эвристическое решение, если такая оценка не может быть получена.

Алгоритмы решения исследуемых в статье задач относятся к ПДС-алгоритмам второго типа.

С целью повышения практической эффективности ПДС-алгоритмов в целом введем понятие эвристического признака оптимальности.

Определение 1. Эвристическим признаком оптимальности допустимого расписания называются логико-аналитические условия, выполнение которых в допустимом решении статистически значимо приводит к оптимальному решению.

Эти свойства допустимого решения качественно обосновываются и подтверждаются численными экспериментами. В силу того, что точное доказательство этих свойств отсутствует, проверка этого эвристического признака оптимальности выполняется экспоненциальной составляющей ПДС-алгоритма или ее полиномиальной аппроксимацией.

Введение эвристического признака оптимальности допустимого решения четко определяет пути дальнейших теоретических исследований – точное доказательство этого признака либо, скорее всего, его частного случая, позволит включить этот признак вместе с соответствующим полиномиальным алгоритмом в полиномиальную составляющую ПДС-алгоритма, что резко повышает его практическую эффективность.

Приведем краткий план решения поставленных задач.

Задача ОЗ1. Задача О/З с фиксированным моментом начала выполнения заданий.

1.1. Полиномиальная составляющая ПДС-алгоритма решения задачи.

1.1.1. Формулировка признака оптимальности допустимого решения.

1.1.2. Полиномиальный алгоритм, проверяющий признак оптимальности. Алгоритм основан на ПДС-алгоритме решения задачи минимизации суммарного запаздывания (МСЗ) [24, гл. 1], включает в себя как его полиномиальную составляющую, так и его экспоненциальную составляющую с наложением дополнительного ограничения на число выполняемых итераций.

1.2. Полиномиальная аппроксимация экспоненциальной составляющей ПДС-алгоритма решения задачи ОЗ1. Эвристический алгоритм с введением качественно обоснованных эвристических признаков оптимальности.

1.3. Пример решения задачи.

Задача ОЗ2. Момент начала выполнения заданий находится в интервале времени $[t_1, t_k]$.

2.1. Полиномиальная составляющая ПДС-алгоритма решения задачи сводится к полиномиальной составляющей ПДС-алгоритма решения Задачи ОЗ1.

2.2. Полиномиальная аппроксимация экспоненциальной составляющей ПДС-алгоритма решения задачи ОЗ2.

2.2.1. Эвристический алгоритм решения задачи ОЗ2, который базируется на алгоритме решения задачи ОЗ3.

2.2.2. Эвристический алгоритм решения задачи ОЗ3.

Алгоритмы решения задач ОЗ2 и ОЗ3 также базируются на основе положений построения экспоненциальной составляющей ПДС-алгоритма решения задачи МСЗ [24, гл. 1].

Задача ОЗ1. Фиксированный момент начала выполнения заданий (r = 0)

1.1 Полиномиальная составляющая ПДС-алгоритма решения задачи

1.1.1 Формулировка признака оптимальности допустимого решения

Признаком оптимальности решения Задачи ОЗ1 является отсутствие опережающих заданий в оптимальном расписании, полученном в результате решения задачи минимизации суммарного запаздывания (МСЗ).

Обоснование. Расписание, минимизирующее только суммарное запаздывание, представляет нижнюю границу минимального суммарного запаздывания, которое может быть достигнуто. Если в таком расписании суммарное опережение равно нулю, то такое расписание оптимально по критерию опережения/запаздывания (3).

1.1.2. Полиномиальный алгоритм, проверяющий признак оптимальности

Алгоритм основан на ПДС-алгоритме решения задачи МСЗ [24, гл. 1]. Полиномиальная составляющая ПДС-алгоритма решения задачи О/З реализуется полиномиальной составляющей ПДС-алгоритма решения задачи МСЗ либо экспоненциальной составляющей ПДС-алгоритма задачи МСЗ с наперед заданным ограничением по числу выполняемых итераций. Если в этом случае алгоритм реализуется на всем множестве заданий, т.е. если количество операций до останова алгоритма меньше введенного полиномиального ограничения, то полученное решение также является реализацией полиномиальной составляющей алгоритма.

Рассмотрим более подробно условия реализации полиномиальной составляющей ПДС-алгоритма решения задачи МСЗ.

Введем необходимые определения [24].

Определение 2. Последовательностью $\sigma^{\text{уп}}$ (сигма упорядоченная) называется последовательность заданий множества J, $j=\overline{1,n}$, в которой задания упорядочены по неубыванию длительностей l_j, т. е. $\forall j, i, j<i : l_j < l_i$, а при $l_j = l_i$, $d_j \le d_i$.

Определение 3. Процедурой свободной перестановки называется процедура перестановки задания $j_{[k]}$ на позицию q $(k<q)$ такую, что $d_{j_{[k]}} \ge C_{j_{[q]}}$, $d_{j_{[k]}} < C_{j_{[q+1]}}$, если хотя бы для одного задания на интервале $\overline{k+1,q}$ выполняется:

$$d_{j_{[i]}} < C_{j_{[i]}}, \quad i=\overline{k+1,q}.$$

Определение 4. Последовательность заданий, полученную в результате выполнения всех свободных перестановок в последовательности $\sigma^{\text{уп}}$, назовем $\sigma^{\text{сп}}$.

В следующих утверждениях 1, 2, 6 [24, гл. 1] сформулированы признаки оптимальности, соответственно, последовательности $\sigma^{\text{уп}}$, $\sigma^{\text{сп}}$, σ^{k} (текущей последовательности, полученной в результате выполнения очередной итерации алгоритма). В утверждениях 3, 4 сформулированы условия сокращения перестановок в случае наличия заданий с отрицательными директивными сроками. В утверждении 5 сформулировано условие декомпозиции последовательности $\sigma^{\text{сп}}$.

Утверждение 1. Пусть в последовательности $\sigma^{\text{уп}}$ $j_{[g]}$ – запаздывающее задание. Уменьшение значения функционала при перемещении $j_{[g]}$ на более ранние позиции в результате перестановок и встраиваний возможно только при выполнении хотя бы одного из следующих условий:

1. $\exists\, j_{[i]}, p \le i \le g \mid r_{j_{[i]}} > 0,\ d_{j_{[i]}} > d_{j_{[g]}}$. На интервале встраивания задания $j_{[g]}$ есть задания $j_{[i]}$ с резервами времени, для которых $d_{j_{[i]}} > d_{j_{[g]}}$. p – позиция, на которой запаздывание по заданию $j_{[g]}$ минимально (или равно нулю).

2. $\exists\, j_{[q]},\ q<g,\ d_{j_{[q]}} > C_{j_{[g]}}$. В последовательности $\sigma^{\text{уп}}$ на позиции q, предшествующей позиции g, есть задание с резервом времени, перестановка которого после задания $j_{[g]}$ уменьшает опоздание по заданию $j_{[g]}$. Задание $j_{[q]}$ остается незапаздывающим.

3. $\exists\, j_{[q]},\ q<g,\ C_{j_{[g]}} - l_{j_{[g]}} < d_{j_{[q]}} < C_{j_{[g]}}$. Существует незапаздывающее задание $j_{[q]}$, директивный срок которого больше момента начала выполнения задания $j_{[g]}$, но меньше момента окончания задания $j_{[g]}$. При этом выполняется:

$$\min\left(C_{j_{[g]}} - d_{j_{[g]}}, l_{j_{[q]}}\right) - \left(C_{j_{[g]}} - d_{j_{[q]}}\right) > 0.$$

Следовательно, перестановка задания $j_{[q]}$ после задания $j_{[g]}$ приведет к уменьшению значения функционала за счет использования резерва задания $j_{[q]}$.

4. $\forall i \mid p \le i \le g \mid r_{j_{[i]}} \le 0$, но $\exists\, j_{[k]}$, $k<p \mid d_{j_{[k]}} > d_{j_{[g]}}$, $r_{j_{[k]}} > 0$, $d_{j_{[k]}} > C_{j_{[p-1]}}$. На интервале перестановки задания $j_{[g]}$ резервы отсутствуют, но существует задание $j_{[k]}$, $k<p$, директивный срок которого больше $d_{j_{[g]}}$ и резерв больше нуля.

Утверждение 2. Для последовательности $\sigma^{\text{сп}}$ не существует перестановок и встраиваний, приводящих к улучшению целевой функции, если выполняется хотя бы одно из условий:

а) $d_{j_{[i]}} - C_{j_{[i]}} \ge 0,\ \forall i = \overline{1,n}$;

б) $d_{j_{[i]}} - C_{j_{[i]}} \le 0,\ \forall i = \overline{1,l}$, $d_{j_{[i]}} - C_{j_{[i]}} \ge 0,\ \forall i = \overline{l+1,n}$;

в) $d_{j_{[i]}} - C_{j_{[i]}} \le 0,\ \forall i = \overline{1,n}$;

г) для каждого задания $j_{[k]}$, принадлежащего множеству заданий с резервами (R), и каждого задания $j_{[g]}$, принадлежащего множеству запаздывающих заданий (Y), выполняется: $d_{j_{[k]}} \le d_{j_{[g]}}$, $l_{j_{[k]}} \le l_{j_{[g]}}\ \forall k \in \overline{1,n}, g \in Y$.

Утверждение 3 [26]. В последовательности $\sigma^{\text{сп}}$ для заданий с отрицательными директивными сроками справедливо: при $i<j : l_i \le l_j$.

Утверждение 4 [26]. Если на интервале встраивания задания $j_{[g]}$ с отрицательным значением директивного срока есть другое задание $j_{[k]}$ с отрицательным значением директивного срока, то при выполнении процедуры встраивания задание $j_{[g]}$ займет позицию $k+1$.

Утверждение 5 [26]. Если в последовательности $\sigma^{\text{сп}}$ первые позиции $i = [1, k]$ занимают задания, для которых $r_{j_{[i]}} \le 0$, то последовательность $\sigma^{\text{сп}}$ декомпозируется на две подпоследовательности: σ^1 включает задания от $j_{[1]}$ до $j_{[k]}$, σ^2 – от $j_{[k+1]}$ до $j_{[n]}$. Подпоследовательность σ^1 оптимальна, и задания, принадлежащие этой подпоследовательности, в перестановках не участвуют. Оптимизация выполняется только для заданий подпоследовательности σ^2.

Утверждение 6. Пусть в последовательности σ^k $j_{[g]}$ – запаздывающее задание. Уменьшение значения функционала при перемещении $j_{[g]}$ на более ранние позиции возможно только при выполнении хотя бы одного из следующих условий:

1. $\exists\, j_{[i]}$, $p \le i \le g \mid r_{j_{[i]}} > 0$, $d_{j_{[i]}} > d_{j_{[g]}}$. На интервале встраивания задания $j_{[g]}$ есть задания $j_{[i]}$, для которых $d_{j_{[i]}} > d_{j_{[g]}}$, $d_{j_{[i]}} > C_{j_{[i]}}$, p – позиция, на которой запаздывание по заданию $j_{[g]}$ минимально (или равно нулю).

2. $\exists\, j_{[q]}$, $q<g$, $d_{j_{[q]}} > C_{j_{[g]}}$.

3. $\exists\, j_{[l]}$, $l<g$, $d_{j_{[l]}} > C_{j_{[g]}} - l_{j_{[g]}}$;

$\min\left(l_{j_{[l]}}, C_{j_{[g]}} - d_{j_{[g]}}\right) > C_{j_{[g]}} - d_{j_{[l]}}$.

4. $\forall i \mid p \le i \le g \mid r_{j_{[i]}} \le 0$, но $\exists\, j_{[k]}$, $k<p \mid d_{j_{[k]}} > d_{j_{[g]}}$, $d_{j_{[k]}} > C_{j_{[k]}}$, $d_{j_{[k]}} > C_{j_{[p-1]}}$.

5. $\forall i \mid i = \overline{1, g-1}$, $r_{j_{[i]}} \le 0$, но $\exists\, j^{*}_{[m]}$ ($j^{**}_{[m]}$), $m<g$.

Характеристика полиномиальной составляющей ПДС-алгоритма решения задачи МСЗ. Справедливы следующие утверждения [24].

Утверждение 7. Пусть в последовательности $\sigma^{\text{уп}}$ $j_{[g]}$ – первое запаздывающее задание, и выполняется: $max\ r(\sigma^{\text{уп}}) \le l_{j_{[g]}}$. В этом случае оптимальное значение функционала достигается за полиномиальное время с трудоемкостью, определяемой функцией O(n *log* n).

Утверждение 8. Пусть $j_{[g]}$ – первое запаздывающее задание в последовательности $\sigma^{\text{сп}}$. Если в этой последовательности на интервале $\overline{1, p-1}$, где p – позиция встраивания задания $j_{[g]}$, резервы отсутствуют, задание $j_{[g]}$ на позиции p остается запаздывающим, на интервале $\overline{p, g-1}$ для каждой пары заданий $j_{[s]}, j_{[t]}$ выполняется $l_{j_{[s]}} \le l_{j_{[t]}}, d_{j_{[s]}} \le d_{j_{[t]}}$, и $\forall\, j_{[r]}$, $r = \overline{g+1, n}$, выполняется $d_{j_{[r]}} \le C_{j_{[r]}}$, $d_{j_{[g]}} \ge d_{j_{[r]}}$, то задача решается за полиномиальное время с трудоемкостью, не превышающей $O(n^2)$.

Утверждение 9. Пусть $j_{[g]}$ – первое запаздывающее задание в последовательности $\sigma^{\text{сп}}$. Если на интервале $\overline{1, g-1}$ все задания упорядочены по неубыванию значений длительности выполнения; $max\ r_i \le l_{j_{[g]}}$, $i = \overline{1, g-1}$ и $\forall\, j_{[s]}$, $s = \overline{g+1, n}$, $d_{j_{[s]}} \le C_{j_{[s]}}$, то задача решается за полиномиальное время с трудоемкостью $O(n^2)$.

Утверждение 10. Пусть при выполнении каждой итерации оптимизации при встраивании очередного запаздывающего задания $j_{[g]}$ (конкурирующего или порожденного) выполняется: а) фактическая позиция встраивания p^H, опре-

деляемая в процессе выполнения алгоритма, больше позиции задания, встроенного последним на предыдущих шагах алгоритма; б) на интервале $\overline{1, p^H - 1}$ нет заданий $j_{[k]}$, для которых выполняются условия: $d_{j_{[k]}} > d_{j_{[g]}}$, $d_{j_{[k]}} > C_{j_{[k]}}$, $d_{j_{[k]}} > C_{j_{[p^H-1]}}$; в) на интервале $\overline{1, p^H - 1}$ для заданий, помеченных «*» («**»), условие $C_{j^{\mathrm{i}}_{[p]}} - d_{j^*_{[m]}} < \sum_{i=m}^{p^{\mathrm{н}}} \max\left(0, C_{j_{[i]}} - d_{j_{[i]}}\right)$, позволяющее сместить эти задания на более поздние позиции, не выполняется. Тогда задача решается за полиномиальное время с трудоемкостью $O(n^3)$.

Утверждение 11. Если условия утверждений 1 или 6 (для всех конкурирующих заданий) не выполняются, или выполняются условия хотя бы одного из утверждений 2 или 7–10, то задача решается посредством полиномиальной составляющей ПДС-алгоритма с трудоемкостью $O(n^3)$.

Если в расписании, полученном полиномиальной составляющей ПДС-алгоритма решения задачи О/З, нет опережающих заданий, это расписание оптимально по критерию (3).

Характеристика экспоненциальной составляющей ПДС-алгоритма решения задачи МСЗ. Алгоритм относится к ПДС-алгоритмам первого типа. Приведем *общую структуру ПДС-алгоритма* [24]. Решение рассматриваемой задачи осуществляется в 2 этапа: предварительного и оптимизационного.

На *предварительном этапе* реализуется:

а) построение последовательности $\sigma^{\text{уп}}$;

б) выполнение свободных перестановок (получена последовательность $\sigma^{\text{сп}}$);

в) формирование множества конкурирующих заданий, т. е. запаздывающих заданий в последовательности $\sigma^{\text{сп}}$, которым в этой последовательности предшествуют задания с резервами времени и бóльшим директивным сроком. Если таких заданий в последовательности $\sigma^{\text{сп}}$ нет, то последовательность $\sigma^{\text{сп}}$ оптимальна.

Этап оптимизации: на этом этапе последовательность $\sigma^{\text{сп}}$ декомпозируется на подпоследовательности меньшего размера. На каждой итерации строится оптимальное расписание для последовательности σ^k для заданий $j_{[i]}$, $i = \overline{1, g_k}$, где $j_{[g_k]}$ – очередное конкурирующее задание. На первой итерации рассматривается подпоследовательность $\sigma^1 \subset \sigma^{\text{сп}}$, включающая задания, которые занимают в последовательности $\sigma^{\text{сп}}$ позиции $\overline{1, g_1}$, где g_1 – позиция первого конкурирующего запаздывающего задания. На второй итерации полученная оптимальная подпоследовательность дополняется заданиями $j_{[l]}$, которые в последовательности $\sigma^{\text{сп}}$ занимают позиции $l = \overline{g_1 + 1, g_2}$, и строится оптимальное расписание для подпоследовательности на интервале $\overline{1, g_2}$. На каждой итерации оптимизация осуществляется за счет использования резервов времени незапаздывающих заданий (а также резервов, полученных в результате перестановок на более поздние позиции, заданий, ранее использовавших существующие резервы – заданий, помеченных «*» («**»)) очередным конкурирующим заданием и порожденными в результате его перестановок новыми запаздывающими заданиями. Оптимизация выполняется посредством ряда перестановок и встраиваний, направленных на улучшение значения функционала на рассматриваемом подмножестве заданий. Исключаются только те перестановки, которые заведомо ухудшают значение функционала. Таким образом, строится оптимальное расписание для всего множества заданий.

На каждой текущей итерации проверяются условия, сформулированные в утверждениях 1.18–1.31 [24, гл. 1], которые позволяют уменьшить число необходимых направленных перестановок для получения оптимального решения:

- уменьшить интервалы встраивания;
- сократить бесперспективные процедуры встраивания и перестановок;
- сформулировать условия декомпозиции, условия исключения заданий из множества конкурирующих и признаки оптимальности полученных решений.

Наиболее трудоемкий перебор различных вариантов использования резервов конкурирующими заданиями возникает, если для всех конкурирующих заданий выполняется

$$\forall\, l_{j_{[i]}} \le l_{j_{[g]}} \quad d_{j_{[i]}} \ge d_{j_{[g]}}, \qquad (4)$$

причем длительности выполнения отличаются на незначительную величину, а резервы таковы, что итерации оптимизации выполняются для каждого конкурирующего задания. При этом для задания $j^*_{[m]}$ ($j^{**}_{[m]}$) в свою очередь ищется другое задание, помеченное «*» («**»), которое может быть встроено после него, что приведет к уменьшению значения функционала. В этом случае осуществляется перебор заданий, поме-

ченных «*» («**») с последующей оптимизацией подпоследовательности, что может быть связано с экспоненциальной трудоемкостью.

Эффективность экспоненциальной составляющей алгоритма в целом определяется следующими факторами:

1. На этапе оптимизации последовательность $\sigma^{СП}$ декомпозируется на подпоследовательности меньшего размера. Оптимизация осуществляется на подпоследовательности, ограниченной позицией встраивания очередного конкурирующего задания и позицией, занимаемой этим заданием в последовательности $\sigma^{СП}$. В эту подпоследовательность в процессе решения могут быть включены только задания, образующие резервы на интервале встраивания рассматриваемого конкурирующего задания. Таким образом, реализуется декомпозиция задачи на подзадачи меньшего размера.

2. В процессе решения задачи проверяются условия утверждений 1.24, 1.28, 1.29 [24], позволяющие часть заданий исключить из множества конкурирующих. В утверждениях 1.28, 1.29 [24] сформулированы признаки оптимальности полученных решений, что существенно сокращает число выполняемых итераций оптимизации, а следовательно, и время решения задачи.

3. Экспериментальные исследования в целом показали, что при невыполнении условия (4) при решении индивидуальных задач существует лишь незначительный перебор. И только для специальных случаев параметров задачи можно выйти на полный перебор вариантов для отдельных подпоследовательностей.

1.2. Полиномиальная аппроксимация экспоненциальной составляющей ПДС-алгоритма для Задачи О31 (алгоритм А1)

Если не реализовалась полиномиальная составляющая ПДС-алгоритма, задача О31 решается полиномиальной аппроксимацией экспоненциальной составляющей ПДС-алгоритма, основанной на положениях построения экспоненциальной составляющей ПДС-алгоритма решения задачи МСЗ.

Алгоритм А1 состоит из двух блоков:

Блок 1 – построение расписания $\sigma_{опт}$ ПДС-алгоритмом решения задачи МСЗ [24, гл. 1]; если в полученном расписании суммарное опережение равно нулю, то это расписание оптимально по критерию (3).

Блок 2 – оптимизация расписания $\sigma_{опт}$ по критерию (3). Блок 2 состоит из однотипных итераций, на каждой итерации проверяется возможность уменьшения значения функционала за счет вставки на позицию, занимаемую опережающим заданием, заданий большей длительности.

Находим в текущем расписании очередное опережающее задание $j_{[p]}$, которое занимает позицию p. На интервале $[p + 1, n]$ определяем множество заданий большей длительности:

$$N = \{i \in \overline{p+1, n} : l_{j_{[i]}} > l_{j_{[p]}}\}$$

Для заданий $j_{[k]}$ из множества N находим значение функционала на интервале переноса в результате выполнения процедуры перестановки на позицию p:

$$\left| d_{j_{[k]}} - \left(C_{j_{[k]}} - \sum_{i=p}^{k-1} l_{j_{[i]}} \right) \right| + \sum_{i=p}^{k-1} \left| \left(d_{j_{[i]}} - C_{j_{[i]}} \right) - l_{j_{[k]}} \right| + \sum_{i=k+1}^{n} \left| d_{j_{[i]}} - C_{j_{[i]}} \right| \quad (5)$$

Для встраивания выбираем задание $j_{[k]}$, при перестановке которого на позицию p значение функционала в соответствии с выражением (5) уменьшается на максимальную величину.

Переносим задание $j_{[k]}$ на позицию p. В полученном расписании ищем следующее опережающее задание на интервале $[p + 1, n]$, находим новое множество заданий, претендующих на перестановку, и выполняем аналогичную процедуру переноса. Такие процедуры выполняются, пока не будут рассмотрены все опережающие задания.

В полученном расписании переупорядочиваем задания на интервалах опережающих и запаздывающих заданий в соответствии с Леммами 1 и 2 [12], если это приводит к уменьшению значения функционала.

Если при реализации алгоритма не выполнилась ни одна из перестановок (см. эвристический признак оптимальности 1, приведенный ниже), то полученное расписание оптимально по критерию (3). Проверяем эвристический признак оптимальности 2, если он выполняется, то полученное расписание также оптимально по критерию (3). Конец алгоритма.

Обоснование алгоритма А1. Известны следующие свойства задачи О/З.

***Лемма* 1** [12]. Если в последовательности, упорядоченной по неубыванию длительностей заданий, нет опережающих заданий, то такая последовательность оптимальна по критерию (3).

***Лемма* 2** [12]. Если в последовательности, упорядоченной по невозрастанию длительностей

заданий, нет запаздывающих заданий, то такая последовательность оптимальна по критерию (3).

Алгоритм основан на следующей эвристике.

Эвристика 1. Поиск решения при фиксированном моменте начала выполнения заданий осуществляется в окрестности, образованной множеством последовательностей, построенных в результате выполнения уменьшающих значение функционала (3) направленных перестановок в последовательности, оптимальной по критерию МСЗ, на позиции, занимаемые опережающими заданиями, заданий большей длительности. Перестановки на позиции, занимаемые заданиями без опережения, приведет к увеличению запаздывания, а перестановка заданий меньшей длительности, чем опережающие, в соответствии с Леммой 1, заведомо приведут к ухудшению значения функционала.

Эвристический признак оптимальности 1. Если в расписании, оптимальном по критерию МСЗ, не существует уменьшающих значение функционала (3) перестановок на более ранние позиции, занимаемые опережающими заданиями, заданий большей длительности, то такое расписание оптимально по критерию (3).

Обоснование. Если в результате выполнения алгоритма А1 не реализовалась ни одна из перестановок, уменьшающих значение функционала в соответствии с выражением (5), то расписание, оптимальное по критерию МСЗ, является оптимальным по критерию (3).

Эвристический признак оптимальности 2. Если в расписании, оптимальном по критерию МСЗ, позиции от 1 до k занимают запаздывающие задания, а позиции от $k+1$ до n – опережающие задания, упорядоченные по невозрастанию длительностей, то такое расписание оптимально по критерию (3).

Обоснование. Перестановка запаздывающих заданий на более поздние позиции приведет к увеличению запаздывания по запаздывающим заданиям и опережения по опережающим заданиям, т.к. в результате перестановки они сместятся на более ранние позиции. Аналогично, перестановка опережающих заданий на более рание позиции приведет к увеличению опережения по этим заданиям и запаздывания по запаздывающим заданиям, т.к. в результате перестановки они сместятся на более поздние позиции.

Трудоемкость алгоритма А1 определяется трудоемкостью алгоритма решения задачи МСЗ, равной $O(n^3)$.

1.3. Пример решения задачи алгоритмом А1

Таблица 1 – Исходные данные для примера 1

j	l_j	d_j	j	l_j	d_j
1	65	180	5	116	500
2	81	334	6	118	400
3	92	303	7	132	650
4	96	724	8	133	691

Таблица 2 – Оптимальное расписание по критерию МСЗ при $r = 0$

j	l_j	d_j	C_j	T_j
1	65	180	65	
2	81	334	146	
3	92	303	238	
6	118	400	356	
5	116	500	472	
7	132	650	604	
4	96	724	700	
8	133	691	833	142
			Функционал:	**142**

В этом расписании для каждого опережающего задания находим множество заданий большей длительности, претендующих на перестановку, и из них выбираем задание, при перестановке которого на позицию опережающего задания значение функционала в соответствии с выражением (5) уменьшается на максимальную величину.

Таблица 3 – Оптимальное расписание по критерию О/З для примера 1

j	l_j	d_j	C_j	$\lvert C_j - d_j \rvert$
6	118	400	118	282
1	65	180	183	3
3	92	303	275	28
2	81	334	356	22
5	116	500	472	28
7	132	650	604	46
8	133	691	737	46
4	96	724	833	109
			Функционал:	**564**

Это расписание совпадает с оптимальным расписанием, полученным точным алгоритмом.

Задача О32. Момент начала выполнения заданий находится в интервале времени $[t_1, t_k]$

2.1. Полиномиальная составляющая ПДС-алгоритма решения задачи

Аналогично Задаче О31, *признаком оптимальности* решения Задачи О32 является отсутствие опережающих заданий в оптимальном расписании, полученном ПДС-алгоритмом решения задачи МСЗ [24, гл. 1] для момента начала выполнения заданий, равного t_1.

Обоснование. Действительно, момент начала выполнения заданий должен находиться в отрезке $[t_1, t_k]$, и если для момента t_1 в расписании

нет опережающих заданий, то дальнейшее увеличение момента начала выполнения заданий приведет к ухудшению значения функционала.

Таким образом, *полиномиальная составляющая ПДС-алгоритма решения задачи* сводится к полиномиальной составляющей ПДС-алгоритма решения Задачи ОЗ1.

2.2. Полиномиальная аппроксимация экспоненциальной составляющей ПДС-алгоритма решения задачи ОЗ2

Если расписание, полученное ПДС-алгоритмом решения задачи МСЗ при $r = t_1$, не удовлетворяет признаку оптимальности, выполняется *полиномиальная аппроксимация экспоненциальной составляющей ПДС-алгоритма*, в состав которой входят алгоритмы А1, А2 и А3. *Эвристический алгоритм А3* определения фактического момента начала выполнения заданий в заданном интервале $r_ф$ базируется на *эвристическом алгоритме А2* решения *Задачи ОЗ3* (со свободным моментом начала выполнения заданий).

2.2.1. Эвристический алгоритм решения Задачи ОЗ3 (алгоритм А2: определение самого позднего момента начала выполнения заданий, при котором достигается минимальное значение функционала О/З)

Обозначим: $R_j = (d_j - C_j)^+$ – резерв задания j; $R_{\min} = \min_j R_j$; N_R – число заданий с резервами; N_T – число запаздывающих заданий. N_T включает задания с нулевым резервом.

Алгоритм А2 состоит из трех блоков:

1. Построение расписания σ_0 ПДС-алгоритмом решения задачи МСЗ [24, гл. 1] для момента начала выполнения заданий $r = 0$.

2. Построение расписания, оптимального по критерию МСЗ, для $r_i > 0$.

3. Оптимизация расписания по алгоритму А1.

Блок 1

1.1. Построение расписания σ_0, оптимального по критерию МСЗ для $r_0 = 0$. Если в расписании σ_0 нет опережающих заданий, оно оптимально по критерию О/З, $r_{opt} = 0$, конец алгоритма. Иначе переход на шаг 1.2.

1.2. Определение в расписании σ_0 N_R, N_T и $R_{\min}$.

1.3. Если $N_R > N_T$ (или $N_R - N_T > 1$, если число заданий нечётное), переход на 1.4, иначе на блок 3.

1.4. $r_{i+1} = r_i + R_{\min}$; $i = i + 1$. Переход на блок 2.

Блок 2 – коррекция расписания σ_0 по запаздывающим заданиям по алгоритму задачи МСЗ [24, гл. 1]. Если в полученном расписании σ_i нет опережающих заданий, оно оптимально по критерию О/З, $r_{opt} = r_i$, конец алгоритма. Иначе, если в полученном расписании σ_i $N_R > N_T$ (или $N_R - N_T > 1$, если число заданий нечётное), переход на шаг 1.4, иначе на блок 3.

Блок 3 – оптимизация расписания σ_i при полученном оптимальном значении момента $r_i = r_{opt}$ алгоритмом А1. Проверка эвристических признаков оптимальности алгоритма А1. Если в полученном расписании по критерию О/З $N_R = N_T$ и $R_{\min} = k > 0$, то момент начала выполнения заданий можно увеличить на k единиц, т.е. имеем множество оптимальных моментов запуска заданий с тем же значением функционала: $M = \{r_{opt}, r_{opt} + 1, \ldots, r_{opt} + k\}$. Конец алгоритма.

Обоснование алгоритма А2. Справедливы следующие утверждения.

Утверждение 12. Если в расписании σ $N_R > N_T$, то при увеличении момента начала выполнения заданий на величину $R_{\min}$ значение функционала О/З уменьшается на $R_{\min}(N_R - N_T)$.

Доказательство. Действительно, в этом случае уменьшение суммарного опережения равно $R_{\min}N_R$, а увеличение суммарного запаздывания равно $R_{\min}N_T$. Следовательно, уменьшение значения функционала равно $R_{\min}(N_R - N_T)$.

Утверждение 13. Пусть в оптимальном расписании σ, полученном ПДС-алгоритмом решения задачи МСЗ, $N_R > N_T$. Если при увеличении момента начала выполнения заданий на величину $R_{\min}$ и последующей оптимизации по критерию МСЗ в полученном расписании нет опережающих заданий, то такое расписание оптимально по критерию О/З для рассматриваемого момента начала выполнения заданий.

Утверждение 14. Уменьшение значения функционала при увеличении момента начала выполнения заданий происходит, пока выполняется $N_R > N_T$. При $N_T > N_R$ значение функционала увеличивается.

Доказательство основано на утверждении 12.

Утверждение 15. Пусть в расписании, оптимальном по критерию (3), $R_{\min} = k$ и $N_R = N_T$. Момент начала выполнения заданий можно увеличить на k единиц без изменения значения функционала.

Доказательство. Утверждение справедливо, т.к. в этом случае не меняются значения N_R и N_T.

Утверждение 16. Для каждого заданного момента начала выполнения r_i оптимальное расписание по критерию (3) строится на основе оптимального расписания по критерию МСЗ для рассматриваемого момента начала выполнения заданий.

Алгоритм А2 основан на следующей эвристике.

Эвристика 2. Поиск оптимального момента начала выполнения заданий и оптимального

расписания по критерию О/З осуществляется следующим образом. Для момента момента начала выполнения заданий $r_0 = 0$ строим оптимальное расписание по критерию МСЗ (σ_0) (п. 1.1 алгоритма). Если в расписании σ_0 $N_R \le N_T$, то оптимальный момент момента начала выполнения заданий $r_{\text{opt}} = 0$. И если в σ_0 нет опережающих заданий, то оно оптимально по критерию О/З. Если же $N_R > N_T$, момент начала выполнения заданий увеличим на $R_{\min}$ и полученное расписание оптимизируем по критерию МСЗ (получаем расписание σ_{new}) (п. 1.3–1.4 и блок 2 алгоритма, утверждения 12, 13 и 16). Такие процедуры увеличения момента начала выполнения заданий на $R_{\min}$ с последующей оптимизацией по критерию МСЗ выполняются до тех пор, пока $N_R > N_T$ (п. 1.3–1.4 алгоритма, утверждение 14). Если в расписании σ_{new} $N_R \le N_T$ для чётного числа заданий или $N_R - N_T \le 1$ для нечётного числа заданий, то полученный момент начала выполнения заданий является оптимальным, и ему соответствует оптимальное расписание по критерию О/З ($N_R < N_T$ может быть только в том случае, если при последнем увеличении момента начала выполнения минимальное опережение, равное $R_{\min}$, было у нескольких заданий). Расписание σ_{new} оптимизируется по критерию (3) алгоритмом А1, проверяются эвристические признаки оптимальности алгоритма А1. Если в полученном расписании по критерию О/З $N_R = N_T$ и $R_{\min} = k > 0$, то момент начала выполнения заданий можно увеличить на k единиц (блок 3 алгоритма, утверждение 15).

Таким образом, *эвристическим признаком оптимальности* самого позднего момента начала выполнения заданий r_{opt}, при котором достигается минимальное значение функционала (3), является выполнение условия: в расписании, оптимальном по критерию МСЗ, полученном в результате выполнения эвристики 2, при $r = 0$ $N_R \le N_T$, либо при $r > 0$ $N_R \le N_T$, если число заданий чётное, или $N_R - N_T \le 1$, если число заданий нечётное (см. пример 4 ниже).

Примечание. Если при $N_R = N_T$ в расписании σ увеличение момента начала выполнения заданий на $R_{\min}$ приводит к расписанию, в котором также $N_R = N_T$, то момент начала выполнения заданий увеливается на $R_{\min}$ (см. пример 3).

Трудоемкость алгоритма А2 определяется трудоемкостью алгоритма задачи МСЗ, равной $O(n^3)$.

Для оценки эффективности разработанного алгоритма А2 (т.к. не были найдены результаты решения рассматриваемых задач для сравнения у других авторов), были проведены следующие иследования. Для различных моментов начала выполнения заданий с заданным шагом строились оптимальные по критерию (3) расписания точным алгоритмом. Оптимальные расписания и самые поздние моменты начала выполнения заданий, полученные алгоритмом А2, совпали с результатами, полученными точными алгоритмами.

Ниже приведены примеры решения задачи ОЗЗ разработанным алгоритмом А2, таблицы, содержащие оптимальное расписание и самые поздние моменты начала выполнения заданий, полученные точным алгоритмом, и графики, отражающие зависимости функционала О/З от моментов начала выполнения заданий.

Примеры решения задачи алгоритмом А2

Таблица 4 – Исходные данные для примера 2

j	l_j	d_j	j	l_j	d_j
1	50	500	7	100	900
2	79	534	8	102	668
3	80	700	9	121	528
4	83	604	10	130	605
5	88	987	11	147	537
6	96	951	12	150	1100

Таблица 5 – Оптимальное расписание по критерию МСЗ при $r = 0$

j	l_j	d_j	C_j	T_j	E_j
1	50	500	50		450
2	79	534	129		405
3	80	700	209		491
9	121	528	330		198
11	147	537	477		60
4	83	604	560		44
8	102	668	662		6
10	130	605	792	187	
7	100	900	892		8
5	88	987	980		7
6	96	951	1076	125	
12	150	1100	1226	126	
			Функционал:	**438**	

В этом расписании $N_R = 9$, $N_T = 3$, $R_{\min} = 6$. Принимаем момент начала выполнения заданий $r = 6$.

Таблица 6 – Оптимальное расписание по критерию МСЗ при $r = 6$

j	l_j	d_j	C_j	T_j	E_j
1	50	500	56		444
2	79	534	135		399
3	80	700	215		485
9	121	528	336		192
11	147	537	483		54
4	83	604	566		38
8	102	668	668	0	
10	130	605	798	193	
7	100	900	898		2
5	88	987	986		1
6	96	951	1082	131	
12	150	1100	1232	132	
			Функционал:	**456**	

В этом расписании $N_R = 8$, $N_T = 4$, $R_{\min} = 1$. Момент начала выполнения заданий $r = 6+1 = 7$.

Таблица 7 – Оптимальное расписание по критерию МСЗ при $r = 7$

j	l_j	d_j	C_j	T_j	E_j
1	50	500	57		443
2	79	534	136		398
3	80	700	216		484
9	121	528	337		191
11	147	537	484		53
4	83	604	567		37
8	102	668	669	1	
10	130	605	799	194	
7	100	900	899		1
5	88	987	987	0	
6	96	951	1083	132	
12	150	1100	1233	133	
			Функционал:	**460**	

В этом расписании $N_R = 7$, $N_T = 5$, $R_{\min} = 1$. Момент начала выполнения заданий $r = 7+1 = 8$.

Таблица 8 – Оптимальное расписание по критерию МСЗ при $r = 8$

j	l_j	d_j	C_j	T_j	E_j
1	50	500	58		442
2	79	534	137		397
3	80	700	217		483
9	121	528	338		190
11	147	537	485		52
4	83	604	568		36
8	102	668	670	2	
10	130	605	800	195	
7	100	900	900	0	
5	88	987	988	1	
6	96	951	1084	133	
12	150	1100	1234	134	
			Функционал:	**465**	

В этом расписании $N_R = N_T = 6$. Дальнейшее увеличение момента начала выполнения заданий приведет к увеличению N_T. Выбираем $r = 8$. Расписание при $r = 8$ оптимизируем по алгоритму А1 и получаем оптимальное расписание по критерию О/З (табл. 9):

Таблица 9 – Оптимальное расписание по критерию О/З для примера 2

j	l_j	d_j	C_j	$\lvert C_j - d_j \rvert$
11	147	537	155	382
10	130	605	285	320
9	121	528	406	122
1	50	500	456	44
2	79	534	535	1
4	83	604	618	14
3	80	700	698	2
8	102	668	800	132
7	100	900	900	0
5	88	987	988	1
6	96	951	1084	133
12	150	1100	1234	134
			Функционал:	**1285**

Для примера 2 были получены такие расписания (табл. 10):

Таблица 10 – Оптимальные расписания по критерию О/З для примера 2

r	σ_{opt}	f_{opt}	r	σ_{opt}	f_{opt}
0	11,10,9,1,2,4, 3,8,7,6,5,12	1307	10	11,10,9,1,2,4, 3,8,7,6,5,12	1289
1		1305	20		1329
2		1303	30		1369
3		1301	40		1409
4		1297	50		1449
5		1293	60	11,9,2,1,10,4, 3,8,6,5,7,12	1492
6		1289	70		1506
7		**1285**	80	11,9,2,1,4,8,3, 10,6,5,7,12	1527
8		**1285**	90		1535
9		1287	100	11,9,2,1,4,8,3, 7,6,5,10,12	1541

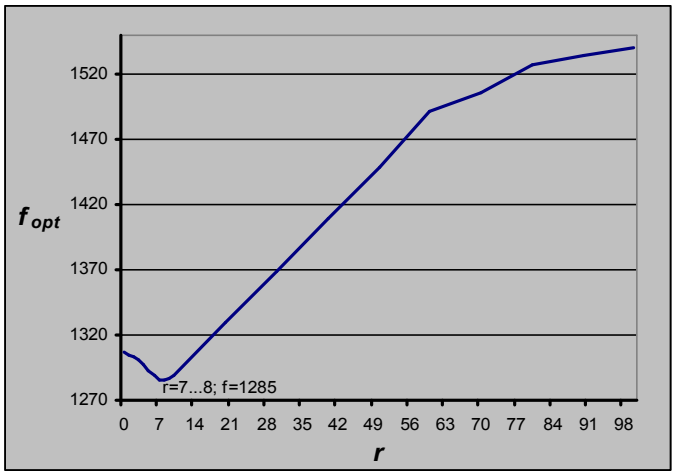


Рис. 1. Зависимость функционала О/З от r для примера 2

Таблица 11 – Исходные данные для примера 3

j	l_j	d_j	j	l_j	d_j
1	50	500	6	100	900
2	79	534	7	102	668
3	83	604	8	121	528
4	88	987	9	130	605
5	96	951	10	147	537

Таблица 12 – Оптимальное расписание по критерию МСЗ при $r = 0$

j	l_j	d_j	C_j	T_j	E_j
1	50	500	50		450
2	79	534	129		405
3	83	604	212		392
8	121	528	333		195
10	147	537	480		57
9	130	605	610	5	
7	102	668	712	44	
6	100	900	812		88
5	96	951	908		43
4	88	987	996	9	
			Функционал:	**58**	

Имеем $N_R = 7$, $N_T = 3$, $R_{\min} = 43$. Принимаем момент начала выполнения заданий $r = 43$.

Таблица 13 – Оптимальное расписание по критерию МСЗ при $r = 43$

j	l_j	d_j	C_j	T_j	E_j
1	50	500	93		407
2	79	534	172		362
3	83	604	255		349
8	121	528	376		152
10	147	537	523		14
9	130	605	653	48	
7	102	668	755	87	
6	100	900	855		45
5	96	951	951	0	
4	88	987	1039	52	
			Функционал:	**187**	

$N_R = 6$, $N_T = 4$, $R_{\min} = 14$. Момент начала выполнения заданий $r = 43 + 14 = 57$.

Таблица 14 – Оптимальное расписание по критерию МСЗ при $r = 57$

j	l_j	d_j	C_j	T_j	E_j
1	50	500	107		393
2	79	534	186		348
3	83	604	269		335
8	121	528	390		138
10	147	537	537	0	
9	130	605	667	62	
7	102	668	769	101	
6	100	900	869		31
5	96	951	965	14	
4	88	987	1053	66	
			Функционал:	**243**	

В этом расписании $N_R = N_T = 5$, $R_{\min} = 31$. Момент начала выполнения заданий $r = 57+31 = 88$.

Таблица 15 – Оптимальное расписание по критерию МСЗ при $r = 88$

j	l_j	d_j	C_j	T_j	E_j
1	50	500	138		362
2	79	534	217		317
8	121	528	338		190
10	147	537	485		52
3	83	604	568		36
7	102	668	670	2	
9	130	605	800	195	
6	100	900	900	0	
4	88	987	988	1	
5	96	951	1084	133	
			Функционал:	**331**	

В этом расписании $N_R = N_T = 5$. Дальнейшее увеличение момента начала выполнения заданий приведет к увеличению N_T. Выбираем $r = 88$. Расписание при $r = 88$ оптимизируем по алгоритму А1 и получаем оптимальное расписание по критерию О/З (табл. 16):

Таблица 16 – Оптимальное расписание по критерию О/З для примера 3

j	l_j	d_j	C_j	$\lvert C_j - d_j \rvert$
10	147	537	235	302
8	121	528	356	172
2	79	534	435	99
1	50	500	485	15
3	83	604	568	36
7	102	668	670	2
9	130	605	800	195
6	100	900	900	0
4	88	987	988	1
5	96	951	1084	133
		Функционал:		**955**

В этом расписании $N_R = N_T = 5$, минимальный резерв $R_{\min}$ – у задания 1 – равен 15. Это дает возможность увеличить момент начала выполнения заданий на 15 (до 103) без изменения значения функционала.

Таблица 17 – Оптимальные расписания по критерию О/З для примера 3

r	σ_{opt}	f_{opt}	r	σ_{opt}	f_{opt}
0	10,9,8,1,2,3,7,6,5,4	1097	**103**	**10,8,2,1,3,7,9,6,4,5**	**955**
70	10,8,2,1,9,3,7,6,5,4	982	104	10,8,2,1,3,7,9,6,4,5	957
80	10,8,2,1,9,3,7,6,5,4	966	105	10,8,2,1,3,7,9,6,4,5	959
85	10,8,2,1,9,3,7,6,4,5	962	110	10,8,2,1,3,7,9,6,4,5	969
87	10,8,2,1,3,7,9,6,4,5	957	120	10,8,1,2,3,7,9,6,4,5	984
88	**10,8,2,1,3,7,9,6,4,5**	**955**	130	10,8,1,2,3,7,9,6,4,5	996
89	**10,8,2,1,3,7,9,6,4,5**	**955**	140	10,8,1,2,3,7,9,5,4,6	1020
90	**10,8,2,1,3,7,9,6,4,5**	**955**	150	10,8,1,2,3,7,9,5,4,6	1054
100	**10,8,2,1,3,7,9,6,4,5**	**955**			

Зависимость функционала О/З от r для примера 3 приведена на рис. 2.

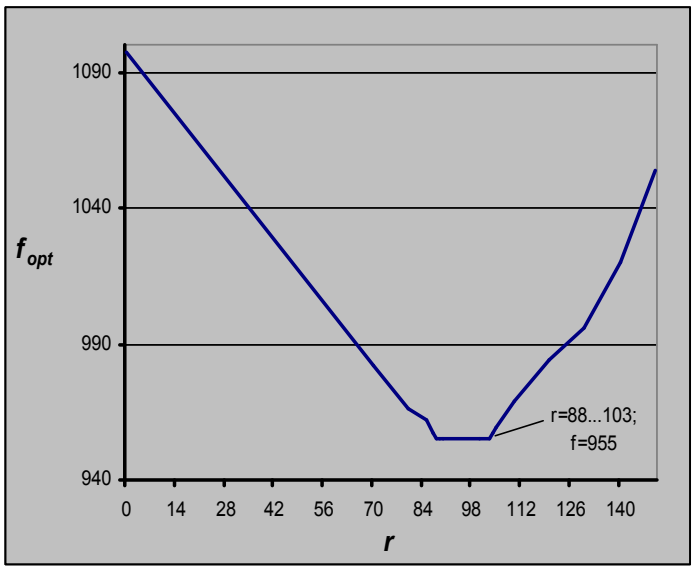


Рис. 2. Зависимость функционала О/З от r для примера 3

Таблица 18 – Исходные данные для примера 4

j	l_j	d_j	j	l_j	d_j
1	65	180	6	118	400
2	81	334	7	132	650
3	92	303	8	133	500
4	96	724	9	150	691
5	116	500			

Таблица 19 – Оптимальное расписание по критерию МСЗ при $r = 0$

j	l_j	d_j	C_j	T_j	E_j
1	65	180	65		115
2	81	334	146		188
3	92	303	238		65
6	118	400	356		44
5	116	500	472		28
8	133	500	605	105	
4	96	724	701		23
7	132	650	833	183	
9	150	691	983	292	
			Функционал:	**580**	

Имеем $N_R = 6$, $N_T = 3$, $R_{\min} = 23$. Принимаем момент начала выполнения заданий $r = 23$.

Таблица 20 – Оптимальное расписание по критерию МСЗ при $r = 23$

j	l_j	d_j	C_j	T_j	E_j
1	65	180	88		92
2	81	334	169		165
3	92	303	261		42
6	118	400	379		21
5	116	500	495		5
8	133	500	628	128	
4	96	724	724	0	
7	132	650	856	206	
9	150	691	1006	315	
			Функционал:	**649**	

В этом расписании $N_R = 5$, $N_T = 4$.

Таблица 21 – Оптимальное расписание по критерию О/З при $r = 23$

j	l_j	d_j	C_j	$\lvert C_j - d_j \rvert$
3	92	303	115	188
1	65	180	180	0
2	81	334	261	73
6	118	400	379	21
5	116	500	495	5
8	133	500	628	128
4	96	724	724	0
7	132	650	856	206
9	150	691	1006	315
			Функционал:	**936**

Анализ статистических исследований. Были проведены многочисленные исследования на размерностях до 20 заданий. Оптимальные расписания и самые поздние моменты начала выполнения заданий, полученные алгоритмом А2, совпали с результатами, полученными точными алгоритмами.

Таблица 22 – Оптимальные расписания по критерию О/З для примера 4

r	σ_{opt}	f_{opt}	r	σ_{opt}	f_{opt}
0	6,1,3,2,8,5,7,4,9	939	13	3,1,2,6,8,5,4,7,9	953
1	6,1,3,2,8,5,7,4,9	942	14	3,1,2,6,8,5,4,7,9	952
2	6,1,3,2,8,5,7,4,9	945	15	3,1,2,6,8,5,4,7,9	951
3	6,1,3,2,8,5,7,4,9	948	20	3,1,2,6,5,8,4,7,9	945
4	6,1,3,2,8,5,7,4,9	951	22	3,1,2,6,5,8,4,7,9	939
5	6,1,3,2,8,5,7,4,9	954	**23**	**3,1,2,6,5,8,4,7,9**	**936**
6	6,1,3,2,8,5,7,4,9	955	24	3,1,2,6,5,8,4,7,9	937
7	6,1,3,2,8,5,7,4,9	956	25	3,1,2,6,5,8,4,7,9	938
8	6,1,3,2,8,5,7,4,9	957	30	3,1,2,6,5,8,4,7,9	947
9	6,1,3,2,8,5,7,4,9	958	35	2,1,3,6,5,8,4,7,9	962
10	3,1,2,6,8,5,4,7,9	958	40	1,3,2,6,5,8,4,7,9	970
11	3,1,2,6,8,5,4,7,9	955	45	1,3,2,6,5,7,4,8;9	977
12	3,1,2,6,8,5,4,7,9	954			

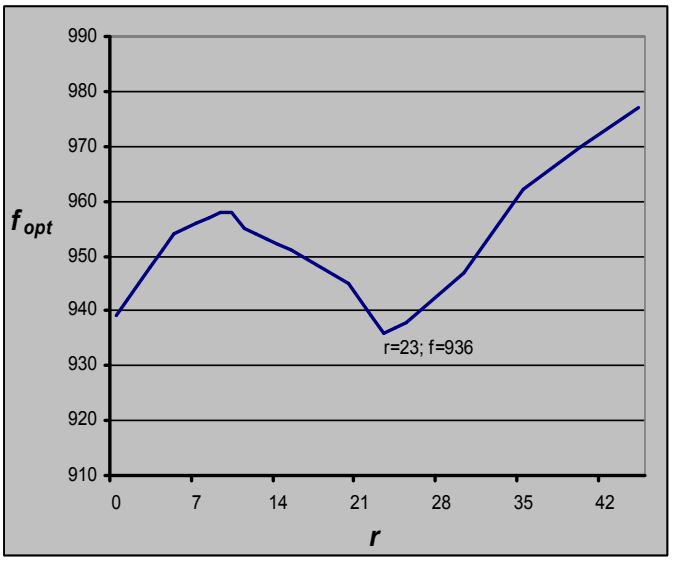


Рис. 3. Зависимость функционала О/З от r для примера 4

Кривые на графиках, отражающих зависимости функционала О/З от r, условно можно разбить на участки трех типов (см., напр., рис. 3):

1. При увеличении момента начала выполнения заданий значение функционала уменьшается за счет уменьшения опережения заданий с резервами, т.к. их число больше числа запаздывающих заданий (утверждение 14, участок [10, 23] на рис. 3).

2. При увеличении момента начала выполнения заданий значение функционала в отдельных случаях может возрастать, т.к. резервы уменьшаются, и задания большой длительности могут их не использовать (участок [0, 10] на рис.3).

3. После получения самого позднего момента начала выполнения заданий, при котором достигается оптимальное значение функционала О/З, наблюдается монотонный рост суммарного опе-

режения и запаздывания (участок [23, 45] на рис. 3).

2.2.2. Эвристический алгоритм решения Задачи О32 (алгоритм А3: определение фактического момента начала выполнения заданий в заданном интервале)

1. Построение расписания σ_0, оптимального по критерию МСЗ для $r_0 = t_1$.

2. Находим множество оптимальных значений момента запуска заданий M, при которых достигается минимальное значение функционала О/З, алгоритмом А2. Если $M \bigcap [t_1,\ t_k] \neq \varnothing$, то множество фактических моментов запуска заданий $M_ф = M \bigcap [t_1,\ t_k]$, переход на шаг 3. Иначе, если $\forall r \in M,\ r < t_1$, то фактический момент запуска заданий $r_ф = t_1$, переход на шаг 3. Иначе если $\forall r \in M,\ r > t_k$, то для построения оптимального расписания по критерию О/З выполняем *модифицированный алгоритм* А2, конец алгоритма.

3. Для момента $r_ф$ или любого из моментов $r \in M_ф$ алгоритмом А1 строим расписание по критерию О/З. Проверяем эвристические признаки оптимальности алгоритма А1.

Модифицированный алгоритм А2:

Блок 1

1.1. Определение в расписании σ_0 N_R, N_T и $R_{\min}$.

1.2. Если $N_R > N_T$ (или $N_R - N_T > 1$, если число заданий нечётное), переход на 1.3, иначе на блок 3.

1.3. $r_{i+1} = \min(r_i + R_{\min},\ t_k)$, $i = i + 1$, переход на блок 2.

Блок 2 – коррекция расписания σ_0 по запаздывающим заданиям по алгоритму задачи МСЗ [24, гл. 1]. Переход на шаг 1.2.

Блок 3 – для всех $r_i \in [t_1,\ t_k]$, а также моментов t_1 и t_k, алгоритмом А1 построить расписания по критерию О/З. Проверить эвристические признаки оптимальности алгоритма А1. Выбрать момент r_i^*, при котором достигается наименьшее значение функционала. Если в этом расписании σ_i^* $N_R = N_T$, то находим задание с наименьшим резервом $R^*_{\min}$. Если $R^*_{\min} = k > 0$, то с таким же значением функционала будут все расписания с моментом начала выполнения заданий $r \in [r_i^*, \min(r_i^* + k,\ t_k)]$ (утверждение 15). Конец алгоритма.

Обоснование алгоритма А3. Уменьшение значения функционала происходит за счет двух факторов: за счет использования резервов опережающих заданий заданиями большей длительности (алгоритм А1) и за счет увеличения момента начала выполнения заданий, при котором уменьшается суммарное опережение.

Утверждение 17. После достижения оптимального момента начала выполнения заданий при его дальнейшем увеличении значение функционала монотонно возрастает.

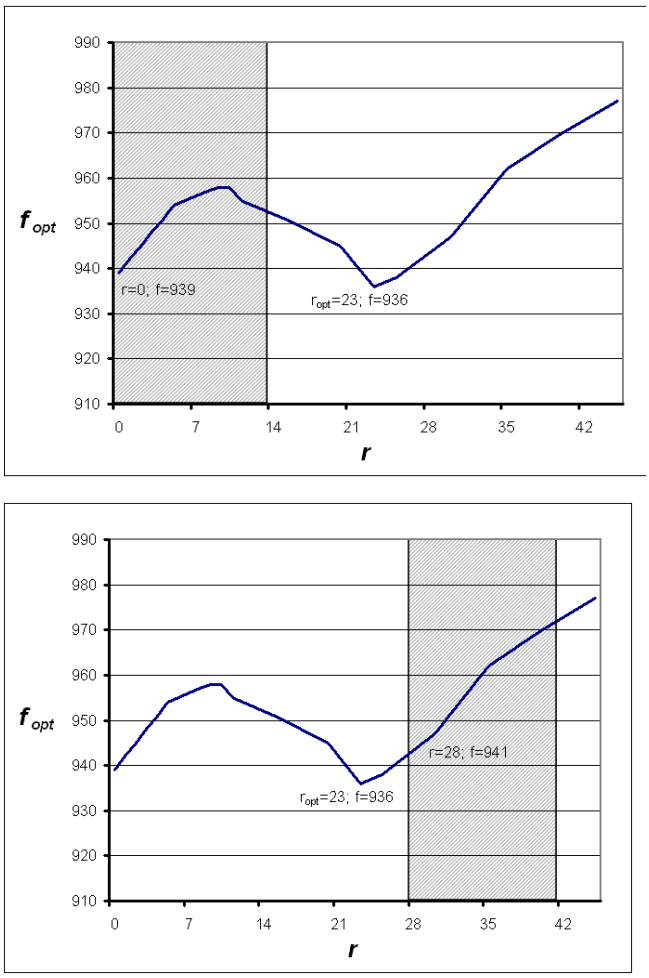


Рис. 5. Зависимость функционала О/З от r для примера 6 (задача О32)

Доказательство. В этом случае идет ухудшение по обоим факторам: менее эффективно используются резервы опережающих заданий, а число запаздывающих заданий становится больше числа опережающих (утверждение 14).■

Если $M \bigcap [t_1,\ t_k]$ не пусто, то это пересечение и является множеством фактических моментов запуска заданий $M_ф$. Если множество M находится до t_1 (см. пример 5 ниже), то фактический момент запуска заданий равен t_1 (утверждение 17, см. п. 3 анализа статистических исследований). Если же множество M находится после t_k (см. пример 6 ниже), то выполняется модифицированный алгоритм А2, поскольку возможно как увеличение, так и уменьшение значения функционала в зависимости от момента начала выполнения заданий (п. 2 анализа статистических исследований). Это связано с тем, что в таких расписаниях существуют большие резервы (опережения), которые используют задания большой длительности, а при увеличении мо-

мента начала выполнения заданий опережения уменьшаются, и их уже не могут использовать задания, ранее их использовавшие.

Примеры решения задачи ОЗ2

Для решения задачи ОЗ2 рассмотрим пример 4 для задачи ОЗ3 (исходные данные и решение см. в таблицах 18–22 и рис. 3).

Пример 5. Задан интервал времени $[t_1, t_k]$ = [28, 42] (на рис. 4 интервал выделен штриховкой). Алгоритмом А2 получен оптимальный момент начала выполнения заданий $r_{opt} = 23 < 28$. Поэтому фактический момент запуска заданий $r_ф = 28$, т.к. при нем достигается минимальное значение функционала О/З.

Пример 6. Задан интервал времени $[t_1, t_k]$ = [0, 14] (на рис. 5 интервал выделен штриховкой). Алгоритмом А2 получен оптимальный момент начала выполнения заданий $r_{opt} = 23 > 14$. Поэтому выполняем модифицированный алгоритм А2, при этом проверяются моменты начала выполнения 0 и 14, выбираем $r_ф = 0$, т.к. при нем достигается минимальное значение функционала О/З.

Выводы

Разработаны эффективные ПДС-алгоритмы решения задач минимизации суммарного опережения и запаздывания на одном приборе при фиксированном моменте начала выполнения заданий или когда момент начала выполнения заданий находится в интервале времени $[t_1, t_k]$. Разработан эвристический алгоритм определения самого позднего момента начала выполнения заданий, при котором достигается минимальное значение функционала (3).

Эффективность разработанных алгоритмов подтверждается статистическими исследованиями с большим количеством примеров. Оптимальные расписания и самые поздние моменты начала выполнения заданий, полученные разработанными и точным алгоритмом, совпадают, что позволяет сделать вывод о необходимости дальнейших теоретических и статистических исследований

Список литературы

1. Feldmann M., Biskup D. Single–machine scheduling for minimizing earliness and tardiness penalties by meta–heuristic approaches // Computers & Industrial Engineering.– 2003.– №44.– P.307–323
2. Tötüncüoglu R.A. Sequencing with earliness and tardiness penalties.– Department of Industrial Engineering, Bilkent University, 1999.– 14 p.
3. Valente J. M. S., Alves R. A. F. S. Improved heuristics for the early/tardy scheduling problem with no idle time // Computers & Operations Research.– 2005.– №32.– P.557–569
4. Jin S., Mason S.J. Minimizing earliness and tardiness costs on a single machine with uncommon job due dates // Department of Industrial Engineering, Bell Engineering Center, University of Arkansas Fayetteville, 2004.– 23 p.
5. Павлов А.А., Мисюра Е.Б., Лисецкий Т.Н., Сперкач М.О., Халус Е.А. Четырехуровневая модель планирования, принятия решений и оперативного управления в сетевых системах с ограниченными ресурсами // Вісник НТУУ “КПІ”. Серія «Інформатика, управління та обчислювальна техніка». – К.: “ВЕК+”, 2013. – №58 –с. 11–23.
6. Kanett J.J. Minimizing the average deviation of job completion times about a common due date // Nav Res Logist // Naval Research Logistics. – 1981. – №28. – P. 643–651.
7. Lawler E.L. A fully polynomial approximation scheme for the total tardiness problem [Текст] / E.L. Lawler // Operations Research Letters. – 1982. – №1. – P.207–208
8. Baker K.R. Sequencing with earliness and tardiness penalties: a review [Текст] / K.R. Baker, G.D. Scudder // Operations Research. – 1990. – Vol.1. – №38. – P.22–36.
9. T’kindt V., Billaut J.–C. Multicriteria scheduling: theory, models and algorithms: Springer, Berlin; Heidelberg, New York. – 2002.
10. Garey M.R. One–processor scheduling with symmetric earliness and tardiness penalties [Текст] / M.R. Garey, R.E. Tarjan, G.T. Wilfong // Mathematics of Operations Research. – 1988. – №13. – P.330–348.
11. Yano C.A. Algorithms for a class of single–machine weighted tardiness and earliness problems [Текст] / C.A. Yano, Y.D. Kim // European Journal of Operations Research. – 1991. – Vol.52. – P.167–178.
12. Ow P.S. The single machine early/tardy problem [Текст] / P.S. Ow, T.E. Morton // Management Science. – 1989. – Vol.2. – №35. – P.177–191.
13. Davis J.S. Single–machine scheduling with early and tardy completion costs [Текст] / J.S. Davis, J.J. Kanet // Naval Research Logistics. – 1993. – №40.– P. 85–101.
14. Szwarc W. Optimal timing scheduling in earliness–tardiness single machine sequencing [Текст] / W. Szwarc, S.K. Mukhopadhyay // Naval Research Logistics. – 1995. – №42. – P.1109–1114.

15. Sridharan V. A decision theory based scheduling procedure for single–machine weighted earliness and tardiness problem [Текст] // V. Sridharan, Z. Zhou // European Journal of Operations Research. – 1996. – Vol.94. – P.292–301.
16. Wan G., Yen B.P.C. Tabu search for single–machine scheduling with distinct due windows and weighted earliness/tardiness penalties // European Journal of Operations Research. – №142. – 2002. – P.271–281
17. Szwarc W. Adjacent ordering in single–machine scheduling with earliness and tardiness penalties // Naval Research Logistics. – 1993. – №40. – P.229–243
18. Lee C.Y., Choi J.Y. A generic algorithm for job sequencing problem with distinct due dates and general early–tardy penalty weights // Computers & Operational Research. – 1995. – №22. – P.857–869.
19. Bank J., Werner F. Heuristic algorithm for unrelated parallel machine scheduling with a common due date, release dates, and linear earliness and tardiness penalties // Math Comput Model. – 2001. – №33. – P.363–383.
20. Gordon V., Proth J.P., Chu C. A survey of the state–of–art of common due date assignment and scheduling research // European Journal of Operations Research. – 2002. – №139. – P.1–25
21. Valente J.M.S., Alves R.A.F.S. Filtered and recovering beam search algorithms for the early/tardy scheduling problem with no idle time // Computers & Industrial Engineering. – 2005. – №48(2) . – P.363–375
22. Tung–I Tsai. A genetic algorithm for solving the single machine earliness/tardiness problem with distinct due dates and ready times // International Journal of Advanced Manufacturing Technologies. – 2007. – №32. – P.994–1000
23. Hoogeveen J.A., Van de Velde L.S. A branch–and–bound algorithm for single–machine earliness–tardiness scheduling with idle time // INFORMS Journal of Computing. – 1996. – №8. – P.402–412.
24. Згуровский М.З., Павлов А.А. Принятие решений в сетевых системах с ограниченными ресурсами: Монография.– К.: Наукова думка, – 2010.– 573 с.
25. Павлов О.А., Місюра О.Б., Шевченко К.Ю. Побудова ПДС-алгоритму розв’язання задачі мінімізації сумарного зваженого запізнення виконання робіт на одному приладі // Вісник НТУУ “КПІ”. Серія «Інформатика, управління та обчислювальна техніка». – К.: “ВЕК+”, 2012. – №56.– С. 58–70.
26. Павлов А.А., Мисюра Е.Б., Костик Д.Ю. Минимизация суммарного запаздывания при наличии заданий с отрицательными значениями директивных сроков // Вісник НТУУ “КПІ”. Серія «Інформатика, управління та обчислювальна техніка». – К.: “ВЕК+”, 2011. – №53.– С.3–5.

УДК 004

АМОНС О.А.

СИНХРОНІЗАЦІЯ ДАНИХ В РОЗПОДІЛЕНИХ ІНФОРМАЦІЙНИХ СИСТЕМАХ

В статті запропоновано механізм синхронізації даних у розподілених системах з урахуванням типів роботи з таблицями та інтеграцією елементів версійності даних. Дослідження присвячено вирішенню завдання підтримки актуальності даних на окремих вузлах розподіленої системи. Проаналізовано особливості роботи розподіленої системи, в рамках якої кожен окремий вузол має локальну копію даних, з якими він працює, та періодично синхронізує данні з центральним сервером. Запропоновано загальний підхід для вирішення проблеми забезпечення актуальності та несуперечливості стану даних кожного користувача системи засобами ведення контролю версій даних та вирішення конфліктів, що виникають в процесі проведення синхронізації. Розглянуто моделі, що використовуються для реалізації підходу.

In the article proposes a data synchronization mechanism in distributed systems based on the types of work with tables and the integration of data versioning elements. Investigation focused on solving tasks support the relevance of the data on the individual nodes of a distributed system. Analyzed the distributed system features in which each individual node has a local copy of the data with which it works and periodically synchronize data with a central server. A general approach has been proposed to solving the problem of ensuring relevance and coherence state of the information for each user of the system by means of doing versioning control of data, recording the changed data and resolving conflicts that arise in the process of synchronization. The models used for the approach implementation have been considered.

Вступ

В процесі функціонування будь-якої системи центральну роль відіграють дані якими вона оперує. Більшість існуючих програм використовують бази даних в якості сховища для довготривалого зберігання інформації користувача. Більшій частині систем, що орієнтовані для використання на ПК притаманна архітектура де БД винесена на окремий сервер. Однак такий підхід доцільно використовувати лише у випадку наявності стабільного підключення до мережі інтернет.

Мобільні пристрої поширені у всьому сучасному бізнес-середовищі, а це означає обов'язковість роботи з таблицями баз даних, які підключаються лише час від часу. Це спричинено тим, що в процесі їх використання не одноразово виникають ситуації коли з'єднання з мережею інтернет зникає, або взагалі здійснюється лише час від часу. Тому в процесі проектування та розробки системи необхідно передбачити можливість її використання в автономному режимі. З точки зору системи баз даних проблема полягає в ефективності оновлення сховища даних на пристрої, котрі підключаються не часто. [1]

В наш час інформаційна інфраструктура систем приймає все більш розподілений характер. Все більшої децентралізації вимагають рівні управлінських рішень та контроль і керування інформаційними ресурсами [2 – 7]. Структура сучасних організацій являє собою розгалужену систему, до складу якої входять десятки територіально розподілених підрозділів, які функціонують незалежно один від одного, однак використовують одні й ті ж дані. Часто якість функціонування такої складної системи багато в чому залежить від застосування новітніх технологій.

Враховуючи подібні обмеження перед розробниками програмного забезпечення постала проблема організації розповсюдження інформації в системах, які розгорнуто на розподілених клієнтських вузлах. Кожен клієнт повинен мати змогу оперувати актуальними даними. Це означає, що замість зберігання інформації в єдиному сховищі потрібно розповсюдити їх копії по кінцевих вузлах та налагодити механізм синхронізації змін, які будуть вноситись в процесі роботи клієнтських систем.

Питання синхронізації даних на різних вузлах розподіленої система має поширений, однак мало формалізований характер. Тому його огляд та дослідження має важливе значення для пошуку ефективних способів приведення локальних даних в актуальний несуперечливий стан.

Актуальність та доцільність даного дослідження полягає в необхідності створення підсистеми, яку можна було б інтегрувати до складу розподіленої системи для забезпечення син-

хронізації даних між її вузлами. Завдяки використанню такого інструменту з'явиться можливість підтримки даних зв'язаних вузлів розподілених систем в актуальному стані.

Загальний огляд області дослідження та огляд існуючих рішень

Останнім часом інформаційні технології стрімко розвиваються: з'являються нові пристрої, технології, принципи проектування систем та нові платформи; спостерігається переміщення від стаціонарних комп'ютерів до мобільних пристроїв. Кількість провайдерів та якість підключення до мережі інтернет значно покращились за останні роки, однак далеко не завжди користувачам доцільно тримати підключення активним. Це означає, що до проектування та реалізації розподілених програмних систем ставляться нові вимоги: вони повинні давати можливість користувачам взаємодіяти з актуальними даними і при цьому вміти працювати в режимі, коли підключення до інтернету з'являється лише час від часу. Для мобільних телефонів, планшетів, ноутбуків та інших представників портативних пристроїв завжди було актуальним питання часу їх автономної роботи, причому він залежить не лише від параметрів апаратної складової, але й від програмної частини, що використовується користувачем. Все це створило проблему синхронізації даних між розподіленими клієнтськими системами, яка мінімальним чином використовувала б мережевий трафік та обчислювальні ресурси пристрою.

Проблема синхронізації даних (і саме поняття синхронізації даних) досить широка та не чітка, тому для початку оглянемо визначення, які з нею пов'язані.

Синхронізація даних – процес ліквідації відмінностей між двома копіями даних. Передбачається що раніше ці копії були однаковими, а після цього одна з них, або обидві були змінені незалежно одна від одної. [8]

В [9, 10] синхронізацію визначають як процес генерування та відтворення кількох копій даних, що розміщені на одному чи кількох сайтах, в [2] під синхронізацією даних СУБД розуміють приведення баз даних, що функціонують в розподіленому середовищі, в актуальний стан за рахунок виявлення змінених даних, передачі та збереження цих змін в базі даних отримувача.

Для вирішення цієї задачі часто використовують засоби реплікації баз даних, які вбудовано в обрану СУБД (таких як MS SQL, Oracle та ін.). Це дає потужний засіб для поширення змінених даних на всіх кінцевих точках. Однак головним недоліком такого способу є використання великої кількості обчислювальних та мережевих ресурсів.

Крім того використання вбудованих засобів синхронізації СУБД виключають можливість розробки гетерогенних систем, що є достатньою причиною для пошуку альтернативних способів вирішення подібної проблеми, оскільки спектр пристроїв та платформ на сьогоднішній день надзвичайно широкий.

Застосування технології тиражування даних [4, 6, 7] може виявитись важливим етапом в успішній роботі розподіленої системи. Застосування цієї технології забезпечує гарантовану доставку, своєчасну та цілісну передачу даних. Синхронізація даних передбачає їх дублювання в різних вузлах. При цьому робота системи виконується з локальною базою даних, оскільки вони розміщуються завжди на тому вузлі, на якому обробляються і всі транзакції в системі виконуються локально.

Система синхронізації даних дозволить проектувати та розробляти системи, котрим необхідно підтримувати актуальний стан даних при відсутності стабільного підключення до мережі інтернет. При використанні системи з'явиться інструмент, що дозволить проводити сеанси синхронізації даних оптимальним чином та з використанням не великої кількості обчислювальних ресурсів.

В процесі синхронізації даних неодноразово можуть виникати проблемні ситуації, коли зміни в клієнтському сховищі даних конфліктують зі змінами на центральному сервері. Система синхронізації даних дозволить розв'язувати подібні ситуації та мінімізувати вірогідність їх виникнення.

Задачею системи синхронізації даних є автоматичне виявлення змінених даних та розповсюдження цих змін по всіх вузлах розподіленої системи. Передбачається, що даний спосіб синхронізації буде доречним для використання на мобільних пристроях, тому для запланованої системи потрібно обрати максимально швидкі та енергоекономічні методи синхронізації і пошуку змінених даних в базі даних клієнта.

Синхронізацію можна класифікувати по різному. Для проектування та подальшої

розробки системи синхронізації варто чітко визначитись з основними властивостями та шляхами її проведення. В рамках даного дослідження було виділено наступну структуру класифікації [7, 9 – 15]:

1. За напрямком:

Однонаправлена, або одностороння синхронізація – дані змінюються тільки в одній з БД, а в іншій лише зберігаються і не підлягають змінам.

Мультинаправлена, або багатостороння синхронізація – дані можуть змінюватись та вводитись у всіх БД.

2. За часом проведення синхронізації:

Синхронізація реального часу, або синхронна синхронізація – тип синхронізації за якої дані повинні бути синхронізовані негайно після змін.

Відкладена, або асинхронна синхронізація – тип синхронізації при якій процес синхронізації запускається після певної події. Тобто зміни в одній з копій даних поширюється на інші після певного проміжку часу, а не в тій же транзакції.

3. За способом передачі інформації під час синхронізації:

Пряма синхронізація – синхронізація виконується за допомогою програми-клієнта, котра має стабільне підключення до сервера БД.

Непряма, або недетермінована синхронізація – синхронізація виконується за допомогою програми-клієнта, що не має стабільного підключення до сервера БД.

4. За способом аналізу інформації, що синхронізується:

Синхронізація за станом – алгоритм синхронізації працює за принципом порівняння записів однієї таблиці з записами іншої і на основі цієї інформації приймається рішення про проведення синхронізації.

Синхронізація змін, або дельта синхронізація – алгоритм синхронізації виконує перенесення лише змін інформації БД, які зберігаються в журналі виконаних транзакцій.

Ведення версій та історії даних. При розробці баз даних зазвичай необхідно забезпечити підтримку обліку версій та збереження історії об'єктів. Наприклад, у працівника підприємства може змінитися посада, у посади в свою чергу може змінитись оклад – в багатовимірному моделюванні це називається Slowly changing dimensions (далі SCD) – виміри, що рідко змінюються. Тобто виміри, не ключові атрибути яких мають тенденцію з часом змінюватись. Всього виділяють 6 основних типів (методів) SCD, що відображають як історія змін може бути відображена в моделі [16]. Кожен з методів передбачає введення додаткових полів та таблиць до записів проблемної області для відслідковування їх історії зміни. Проведемо короткий огляд кожного типу.

Тип 0 полягає в тому, що дані після першого збереження в таблицю більше ніколи не змінюються. Цей метод практично майже ніким не використовується оскільки він не підтримує ведення версій. Він потрібен лише як нульова точка відліку для методології SCD.

Тип 1 – це звичайний перезапис старих даних новими. В чистому вигляді цей метод теж не містить ведення обліку версій і використовується лише там де історія фактично не потрібна. Тим не менше в деяких СУБД для цього типу можна додати обмежену підтримку версійності засобами самої СУБД (наприклад Flashback query в Oracle), або відслідковуванням змін через тригери.

Тип 2 полягає в створенні для кожної версії окремий запис в таблиці з додаванням поля-ключового атрибута даної версії, наприклад: номер версії, дата зміни чи дата початку та закінчення періоду існування версії.

Тип 3 полягає у тому, що в самому записі знаходяться додаткові поля для минулих значень атрибута. При отриманні нових даних старі дані перезаписуються поточними значеннями.

Тип 4 передбачає збереження історії змін в окремій таблиці: основна таблиця завжди перезаписується поточними даними з перенесенням старих даних в іншу таблицю. Зазвичай цей тип використовують для обліку змін або створення архівних таблиць. Підтипом або гібридом цього варіанта (з типом 2) слід вважати секціонування по признаку поточної версії з дозволеним переміщенням рядків, але це вже за границею моделювання та скоріше відноситься до адміністрування.

Тип 6 (1+2+3) – так званий гібридний тип був придуманий Ральфом Кімболлом (Ralph Kimball) як комбінація методів, що описано вище, та призначений для ситуацій, які вони не враховують або для більшої зручності роботи з даними. Він полягає у внесенні додаткової збитковості: береться за основу тип 2, додається сурогатний атрибут для альтернативного обзору версій (тип 3) та перезаписуються одна, або всі попередні версії (тип 1). В цілому будь-яка

комбінація основних типів SCD відноситься до гібридного типу, тому як їх недоліки так і переваги залежать від конкретної реалізації.

Методи обліку змінених даних. Для розробників однією з проблем являється відслідковування того, які дані було змінено в базі. Ще більш серйозною задачею являється створення простого рішення, яке не наносило б серйозного удару по продуктивності робочих навантажень та було б не складним в створенні, реалізації та керуванні.

Мобільні пристрої поширені у всьому сучасному бізнес-середовищі, а це означає обов'язковість роботи з таблицями баз даних, які підключаються лише час від часу. З точки зору системи баз даних проблема полягає в ефективності оновлення сховища даних на пристрої, котрі не мають стабільного підключення до інтернету. Наявність локальної копії даних вирішує дану проблему та надає користувачу можливість використовувати систему в режимі оф лайн (тобто без наявності активного підключення до інтернету) так, ніби він взаємодіє з центральним сховищем даних. Однак в цьому випадку виникає нова проблема – оновлення локального сховища даних із внесенням змін, що були виконані на центральному сервері з моменту останньої синхронізації, вирішенням конфліктів, що виникли у зв'язку з невідповідністю локальних і серверних даних, та з передачею власних змін серверу для поширення на інших вузлах системи. Для всіх перерахованих пунктів ключовою являється задача пошуку замінених даних як в локальній базі даних, так і на серверній стороні. Передача всіх записів бази даних від одного вузла до іншого в процесі синхронізації – не найкраще рішення. Воно тягне за собою фінансові затрати та затрати обчислювальних ресурсів, причому це стосується всіх учасників обміну інформацією: клієнтського пристрою, каналу зв'язку та серверної частини (або ж пристрою іншого клієнта, в залежності від обраної архітектури системи та принципів проведення синхронізації). Наприклад, мобільні пристрої мають обмежений час автономної роботи (мається на увазі роботу без під'єднаного зарядного пристрою), тому це вимагає від розробників максимально ефективно оптимізувати свої програмні продукти, оскільки будь-яке виконання зайвих обчислень пришвидшує зменшення заряду акумулятора пристрою. Тобто передаючи весь обсяг інформації бази даних (так званий моментальний знімок даних – snapshot) відбувається неефективне використання ресурсів. По каналу зв'язку будуть передаватись великі обсяги даних, навіть якщо в них буде змінено лише одне поле, а отже мережевий трафік та плата за нього значно зростуть. А у разі використання мобільного, чи не стабільного інтернет з'єднання з'являється висока імовірність того, що сеанс синхронізації завершиться невдачею. Щодо серверної частини то тут теж присутні обмеження. Використовуючи хмарні обчислення в своїх рішеннях необхідно орендувати в провайдера головним чином процесорний час, обсяг сховища бази даних та оплачувати виконання транзакцій до нього. Кожне рішення, що не оптимальним чином використовує вказані ресурси тягне за собою неоправдані фінансові витрати. Для вирішення повідних проблем більш доцільним є передача лише тих записів, що зазнали змін на вузлі-джерелі з часу останнього сеансу синхронізації. Враховуючи це необхідно включити до системи синхронізації даних механізм пошуку змін, який буде виділяти лише змінені дані з-поміж всіх інших записів бази даних.

Розглянемо існуючі методи для виявлення змінених даних. В рамках даної роботи було умовно їх розділено на журнальні методи та методи пошуку змін за запитом. Перша група методів передбачає збереження дій, що виконуються з записами бази даних до журналу та їх відтворення на інших вузлах синхронізації. При використанні другої групи – пошук змінених даних відбувається безпосередньо під час проведення сеансу синхронізації, або перед ним.

У випадку застосування методу ведення журналу змінених даних система реєструє в журналі інформацію про зміну даних в таблицях користувача, відслідковуючи як сам факт виконання операцій по зміні даних так і конкретні зміни. Як показано на рис. 1 зміни в таблиці користувача записуються у відповідних таблицях змін. Ці таблиці забезпечують журнальне представлення виконаних змін, що розподілені в часі.

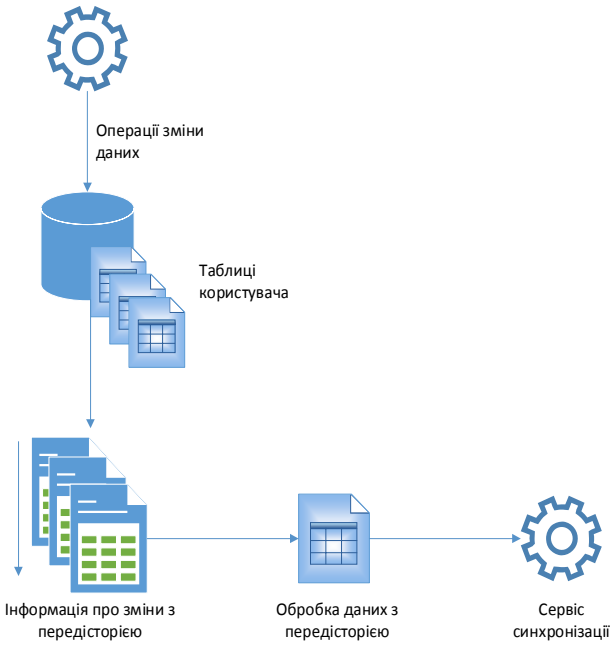


Рис. 1. Використання журналу змінених даних

Рис. 2 Використання журналу змін

Також можливий варіант, коли в журнал вносяться не змінені дані, а сам факт про зміну рядків таблиці (рис. 2). Це дозволяє програмним системам визначати змінені рядки, отримуючи інформацію про зміни безпосередньо з таблиць користувача. Тобто в журналі міститься інформація про те, які рядки таблиць змінені без додаткової інформації про те, які поля було змінено. Процес вибірки змінених рядків відбувається безпосередньо перед проведенням процесу синхронізації. При цьому відбувається проходження по записах журналу та вибір відповідних рядків з таблиць бази даних.

В цьому сценарії на сервіс синхронізації передаються всі рядки таблиці, що змінено з моменту проведення останньої синхронізації, причому передаються поточні дані рядків цілком. Оскільки, як вже було сказано, для проведення синхронізації використовується синхронний механізм для відслідковування змін, система може виконати двосторонню синхронізацію і визначити можливі конфлікти. Однак слід зауважити, що використовуючи даний підхід слід мати на увазі, що у разі виникнення конфлікту буде вкрай складно реалізувати його розв'язання шляхом об'єднання даних.

В [17] розглянуто синхронізацію журналу транзакцій – найбільш простий в реалізації метод та один з найефективніших і найпоширеніших. В цьому випадку до журналу записуються транзакції бази даних, що були виконані впродовж певного часу. При реалізації процесу синхронізації двох та більше БД таким способом на кожній з них повторюються всі дії, що виконані на іншій БД, витягуючи їх по черзі з журналу. Звідси випливає одна з основних переваг синхронізації журналу транзакцій – немає необхідності в постійному з'єднанні з віддаленою базою даних під час передачі змін – частина журналу являється автономною одиницею і її можна передавати будь-якими каналами зв'язку в будь-який час, в тому числі навіть на зовнішньому флеш-накопичувачі. Тобто журнал володіє властивістю переносимості, яку доволі важко реалізувати у разі використання інших методів.

Для реалізації цього метода необхідно ретельно враховувати які записи журналу ще не передавались. Ця проблема стає складнішою при наявності кількох баз даних, що синхронізуються – потрібно запам'ятовувати який останній запис було передано в кожну з БД.

Недоліком використання журналу транзакцій являється те, що журнал займає доволі багато місця. При достатньо частій зміні інформації в таблицях бази даних і достатньо великих проміжках часу між сеансами синхронізації журнал може сягати великих розмірів і навіть перевищувати за розміром самі дані, що зберігаються в базі даних. Зрозуміло, що після успішного застосування журналу до іншої БД журнал можна урізати. Однак тим не менше журнал за-

ймає доволі багато місця – в кращому випадку дублює нові та змінені записи.

Журнальна синхронізація реалізується відносно просто, однак колізії (конфлікти) при цьому неминучі, оскільки процес, що проводить синхронізацію не має одночасного доступу до кількох баз даних, які приймають участь в процесі.

На противагу журнальним методам пошуку змінених даних можна виконувати пошук шляхом порівняння даних двох таблиць, що беруть участь у синхронізації. Одним із методів порівняння блоків даних є визначення значення хеш-функції. Хеш-функція – функція, що перетворює вхідні дані будь-якого (як правило, великого) розміру в дані фіксованого розміру. Відповідно хешування – процес перетворення вхідного масиву даних довільної довжини у вихідний бітовий рядок фіксованої довжини. При цьому, якщо хеш-функції повертають різні значення для двох вхідних масивів, то вони гарантовано різні. Якщо ж хеш-значення двох вхідних масивів співпадають, то вхідні рядки скоріше за все однакові. [18]

У разі застосування цього методу для проведення синхронізації пошук змін в базі даних відбувається для кожної таблиці окремо. Найменшим елементом вважається запис в таблиці. Спочатку обчислюється хеш-сума для всієї таблиці в цілому і якщо це значення на обох вузлах синхронізації виявиться різним, то далі виконується розбиття таблиці на інтервали, для кожного з яких також проводиться визначення та порівняння результатів хеш-функції. Для інтервалів з різними результатами проводиться подальше подрібнення і для кожного проміжку процес повторюється до тих пір, доки не будуть виявлені змінені рядки таблиці.

Для застосування пошуку змін за допомогою хешування не потрібно вносити зміни до існуючої структури бази даних чи додавати логіку до процесу збереження даних в системі. Також не потрібно проводити ведення журналу змін. Однак разом з цим значно підвищуються вимоги до обчислювальних ресурсів на всіх вузлах синхронізації, а у разі проведення синхронізації даних, де змінено невелику кількість записів все одно буде необхідно досліджувати всі записи таблиці, що вкрай неефективно (особливо у випадку великої кількості рядків таблиці).

Постановка проблем дослідження

В реаліях сучасних інформаційних систем є багато задач на реалізацію рішень взаємодії окремих вузлів розподіленої системи. Окремої уваги заслуговують ті з них, що орієнтовані на мобільні пристрої та роботу за умов з нестабільним підключенням до мережі інтернет. В такому випадку раціональним підходом вважається використання локальної бази даних на пристрої користувача та використання копії даних розподіленої системи (чи копії частини даних), яка зберігається в цій локальній БД. Для реалізації подібного сценарію необхідно спроектувати та розробити механізм виявлення та розповсюдження даних, що змінені на одному з вузлів системі між всіма іншими вузлами. Для цього потрібно знайти вирішення ряду проблем, що виникають впродовж проведення синхронізації даних. Однією з таких проблем є вибір моделі розподіленої системи, характер взаємодії клієнтів та характеристики процесу синхронізації. Враховуючи незалежність роботи локальних СУБД на різних вузлах необхідно забезпечити унікальність первинних ключів записів таблиць. Вкрай недопустимо щоб в різних БД були створені рядки з однаковими ідентифікаторами. Це може призвести до некоректного проведення процесу синхронізації та непередбачуваних наслідків. Також в рамках дослідження необхідно вирішити специфічні проблеми, що безпосередньо стосуються процесу синхронізації даних та притаманні обраній стратегії проведення синхронізації. Ще одним важливим аспектом проблеми є питання пошуку даних, які змінено на окремо взятому вузлі системи з часу проведення останнього сеансу синхронізації, та забезпечення безпеки доступу до даних в БД. І насамкінець лишається найбільш обширна та складна проблема – розробка алгоритму проведення процесу синхронізації. Це питання в більшій мірі визначає наскільки ефективно будуть використовуватись рішення, які було прийнято раніше.

Запропоноване рішення

Враховуючи вимоги до системи синхронізації та відповідно до наведеної класифікації типів проведення синхронізації вирішено, що процес синхронізації в рамках роботи підсистеми матиме наступні характеристики:

– за напрямком – мультинаправлена: кожна з віддалених БД виступатиме в ролі як джерела так і цілі синхронізації;

– за часом проведення – асинхронна: початком для проведення сеансу синхронізації слугуватиме запит клієнтської системи;

– за способом передачі інформації – недетермінована: проводитиметься час від часу за запитом клієнтської системи.

– за способом аналізу інформації – синхронізація змін: на кожному вузлі проводитиметься ведення версійності та реєстрації змінених даних.

В рамках системи синхронізації даних реалізовано двонаправлену синхронізацію між центральним сервером та кожним з клієнтів. Проводиться журналізація всіх змінених даних. Фізичне видалення та зміна записів замінюється логічним – тому конфлікти стають розв'язуваними.

Вирішення проблеми забезпечення унікальності ідентифікаторів. Необхідно зазначити, що кожен вузол системи, що бере учать в процесі синхронізації сам по собі являється системою бази даних. Інакше кажучи, на кожному вузлі є власна локальна СУБД та програмне забезпечення, що з нею пов'язано.

Протягом проведення процесу синхронізації важливо щоб кожен запис мав унікальний ідентифікатор. Тому з метою забезпечення унікальності первинних ключів записів таблиць було вирішено відмовитись від ідеї автоматичного їх генерування засобами СУБД. Дане рішення викликане імовірністю того, що в двох базах даних можуть з'явитись запаси з однаковими ключами (особливо у разі використання автоінкрементованих ключів). Для виходу з ситуації можна використовувати зіставні ключі, однак це додасть збитковості записів та ускладнить операції з даними в системі.

Враховуючи все перераховане вище було вирішено використовувати тип GUID в я кості типу первинних ключів. Генерація нових ключів має відбуватись в самій системі. Це унеможливить імовірність створення двох записів однієї таблиці з однаковими ідентифікаторами та спростить механізм проведення синхронізації та розв'язання конфліктних ситуацій.

Вирішення проблем двосторонньої синхронізації. Якщо реалізовувати схему з двосторонньою синхронізацією, тобто коли вузли являються і джерелом і приймачем інформації, що синхронізується, виникає ряд специфічних проблем. Розглянемо випадок лише для двох учасників (рис. 3). При цьому будемо будем мати на увазі, що один з них являється клієнтом (окремо-взятим вузлом системи), а інший – центральним сервером синхронізації, з яким зв'язуються всі клієнти системи.

При подібній схемі реалізації синхронізації даних кожен з вузлів-учасників процесу виступає в якості джерела та приймача даних. В приведеній схемі можуть виникнути проблеми, що описано далі.

По-перше, кожен з вузлів отримує журнал дій, що виконано на іншій стороні.

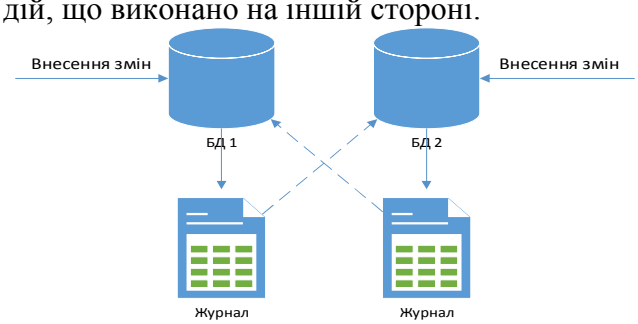


Рис. 3. Схема двосторонньої синхронізації

Кожен вузол приймає запити на модифікацію даних як від локальних користувачів, які працюють з базою даних, так і від віддаленої сторони синхронізації. При відтворенні цього журналу вузлом-приймачем виконуються всі записані дії, які було виконано на вузлі-джерелі. Якщо цей вузол-приймач виступає в ролі також і вузла-джерела, то дані, що виконано в процесі відтворення журналу, також потраплять в журнал вузла-приймача і, відповідно, при наступному сеансі зв'язку їх буде передано відправнику знову. Тобто може виникнути процес неконтрольованого лавиноподібного процесу накопичення інформації, яка синхронізується, коли вузли передають один одному інформацію про один і той же запис.

Для запобігання подібної поведінки можна зробити наступне: при прийомі та виконанні журналу, що отримано від іншої сторони, – не вести власний журнал щоб виконані зміни були виконані лише над локальними даними і не потрапили в журнал, що призначено для відправки вузлу, який щойно прислав свій журнал. Якщо для реалізації журнальної синхронізації використовується механізм тригерів, то потрібно блокувати їх виконання під час виконання журналу іншої сторони, або написати таким чином, щоб вони розпізнавали момент виконання журналу іншої сторони і не писали б в такому випадку власний журнал.

Якщо немає блокування ведення журналу інформації, отриманої від партнера синхронізації, виникає ситуація, яку представлено на рис. 4, тобто інформація, що синхронізується ходить від одного вузла до іншого, що звичайно ж недопустимо.

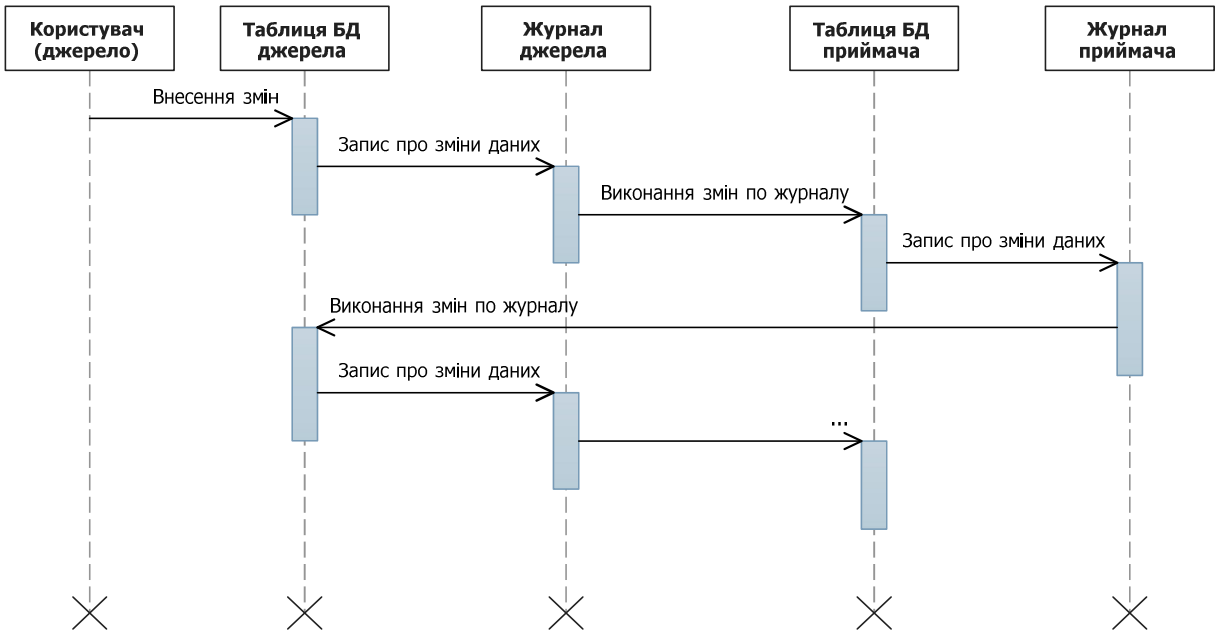


Рис. 4. Проблема блокування під час проведення двосторонньої синхронізації

По-друге, при реалізації двосторонньої синхронізації неодмінно виникає проблема конфлікту доступу до одних і тих же даних на обох сторонах учасників процесу синхронізації.

Цей конфлікт виникає у тому випадку, якщо на двох вузлах виконуються зміни одних і тих же даних. Припустимо, на вузлі 1 створили запис, а потім на вузлі 2 її видалили. Однак одночасно з цим видаленням на вузлі 1 було виконано операцію по зміні цього запису. Звичайно ж виникає конфлікт, який потребує вирішення – які зміни рахувати вірними. Основне питання, на яке потрібно дати відповідь при виникненні подібного роду конфліктів, – які дані все ж вірні і хто з вузлів-учасників правий. Відмітимо, що далеко не завжди можливо прийняти автоматизоване рішення про перевагу змін для якогось запису і необхідне рішення користувача.

Для розв'язання цього конфлікту використаємо наступні принципи:

– більш вірними вважаються записи у того з вузлів, який створив запис і хто вважається його власником. Для реалізації цього правила необхідно щоб в кожній з таблиць процесу синхронізації було поле, що вказує на автора запису;

– вводиться додаткове поле до журналу – "час виконання зміни". При виконанні журналу час зміни запису порівнюється з часом локальної зміни цього запису і більш вірною вважається інформація, яка отримана пізніше. Слід зазначити, що це далеко не найкращий спосіб вирішення конфліктів, оскільки може створювати конфлікт зміни одного і того ж запису, що відбувся в різний час;

– одним з найнадійніших способів – це механізм відслідковування за виникненням конфліктних ситуацій, а у випадку виникнення таких ситуацій – вимагати рішення користувача, не вирішуючи яка із сторін права, однак даючи оцінку кожній з таких ситуацій.

Вирішення проблеми контролю доступу до даних та мінімізації обсягу трафіку. В рамках роботи системи було вирішено виділити чотири типи сутностей даних, в залежності від яких обмежується доступ до них. Обмеження залежать від користувача, що намагається отримати доступ до даних:

– службові дані: доступ для зміни та перегляду інформації належить лише адміністратору;

– особисті дані: створювати записи такого типу можуть всі користувачі, однак переглядати та редагувати – лише автори записів;

– публічні дані: право на створення та перегляд інформації належить всім користувачам системи, однак редагувати записи може лише автор;

– загальнодоступні дані: створення, перегляд та редагування даних можуть здійснювати всі користувачі системи.

Ці всі обмеження наочно зображено в табл.1.

Табл. 1 Можливість виконання дій над даними системи в залежності від типу записів таблиці та ролі користувача

Тип користувача	**Дії**	**Тип запису**			
		Службові дані	**Особисті дані**	**Публічні дані**	**Загальнодоступні дані**
Адміністратор	Створення	✓			
	Перегляд	✓			
	Редагування	✓			
	Видалення	✓			
Автор запису	Створення		✓	✓	✓
	Перегляд		✓	✓	✓
	Редагування		✓	✓	✓
	Видалення		✓	✓	✓
Інші користувачі	Створення	×			
	Перегляд	✓	×	✓	✓
	Редагування	×	×	×	✓
	Видалення	×	×	×	✓

Введемо певні пояснення щодо таблиці: роль адміністратора може бути суміщена з роллю автора запису, або роллю іншого користувача у відношенні до даних, які не являються службовими.

В контексті бази даних типи об’єктів та обмеження поширюються на таблиці БД. Це означає, що, наприклад, якщо в певній таблиці зберігаються об’єкти типу службових даних то доступ до неї матимуть лише користувачі з правами адміністраторів.

Право на редагування записів таблиць зі службовими даними мають лише адміністратори. При цьому не має значення чи являється адміністратор автором запису (користувачів з правами адміністратора може бути декілька і не має значення хто з них додав запис). Всі інші користувачі системи мають права лише на перегляд даних такого типу. Прикладом можуть бути таблиці-довідники.

Доступ до особистих даних: право на перегляд та редагування можуть мати лише автори цих записів. Для всіх інших користувачів дані такого типу повністю скриті.

Суть публічних даних така ж як і в особистих даних, однак користувачі, що не являються авторами можуть переглядати подібні записи.

Загальнодоступні дані повністю відкриті для перегляду та редагування всіма користувачами. Очевидно, що при цьому авторство не має ніякого значення.

Розглянемо більш детально ролі користувачів та типи даних у контексті синхронізації баз даних. Оскільки вже визначено основні характеристики системи синхронізації і вказано, що використовується сервер в якості центрального вузла для проведення синхронізації то всі подальші описи виконано з урахуванням саме прийнятої архітектури.

Синхронізація службових даних. Службові дані доступні для перегляду всім користувачам системи, а для редагування – лише користувачам з правами адміністратора. Тому в процесі синхронізації службові дані доступні для читання для всіх вузлів системи, тобто розповсюджуються для всіх користувачів. У разі редагування такого запису клієнтською системою при спробі зберегти зміни на сервері здійснюється перевірка прав користувача: якщо користувач являється адміністратором, то зміни приймаються і в майбутньому будуть розповсюджені по всім іншим вузлам. У тому разі, якщо користувач не має прав адміністратора – зміни не приймаються на сервері, а на клієнті повертаються до попереднього стану.

Важливо відмітити сценарій, коли запис службових даних редагується двома адміністраторами одночасно. В такому разі система синхронізації намагається поєднати зміни. Якщо у обох адміністраторів змінені поля не пересікаються – відбувається поєднання і обидві зміни стають актуальними. Якщо ж змінені поля пе-

ресікаються то актуальними залишаються зміни, що виконані раніше, а корекції другого користувача відкидаються і оновлюються на клієнті.

Синхронізація особистих даних. Особисті дані доступні лише автору запису, тому в даному випадку ніяких особливих проблем не виникає. В процесі проведення синхронізації таблиць з особистими даними беруть участь лише ті дані, автор яких проводить синхронізацію. Тобто записи з таких таблиць копіюються лише їх авторам і не відображаються в таблицях інших користувачів. У випадку спроби зберегти на сервері перевіряється авторство кожного запису таблиці. Якщо користувач намагається зберегти запис особистих даних, автором якого являється інший користувач – збереження не відбувається. Якщо ж запис зберігає сам автор то відбувається збереження змінених даних.

Синхронізація публічних даних. Ситуація з публічними даними дуже схожа з особистими даними. Відмінність лиш полягає у тому, що публічні дані доступні всім користувачам, тобто вони копіюються на всі вузли системи. У разі редагування такого запису клієнтською системою при спробі зберегти зміни на сервері здійснюється перевірка авторства запису: якщо користувач являється автором, то зміни приймаються і в майбутньому будуть розповсюджені по всім іншим вузлам. У тому разі, якщо користувач не являється автором – зміни не приймаються на сервері, а на клієнті повертаються до попереднього стану.

Синхронізація загальнодоступних даних. Найцікавіший і водночас найскладніший випадок складають загальнодоступні дані – вони доступні для читання та редагування всім користувачам системи. Тобто в процесі синхронізації вони поширюються по всіх вузлах системи. У разі редагування такого запису клієнтською системою при спробі зберегти зміни на сервері необхідно передбачити логіку обробки різних ситуацій. У разі якщо клієнт відредагував загальнодоступний запис, який був перед цим актуальним то його зміни приймаються та в майбутньому поширюються по всіх інших вузлах.

Основні концепції алгоритму проведення синхронізації. Після аналізу вимог до системи та проблем, що виникають в процесі синхронізації було обрано алгоритм для проведення сеансів синхронізації. Перед його описом слід зазначити кілька важливих моментів.

Ідентифікатори версій генеруються на клієнті, а зберігаються в таблиці на сервері. Це дозволить вести облік версій та відслідковувати хронологію внесення змін. У кожного запису таблиць БД на сервері додано поле в яке слід зберігати відповідний номер.

Для зберігання інформації про змінені поля при кожному збереженні даних на сервері вирішено використовувати *.xml-файли. Назва кожного файлу буде співпадати з ідентифікатором версії даних, в рамках якої збережено зміни. Таким чином можна буде легко знайти необхідні зміни для проведення процесів розв’язання конфліктів та формування пакету змінених даних для відправки клієнту. Використання файлів дозволить зменшити об’єм пам’яті на збереження подібної інформації та дозволить швидше знаходити необхідну інформацію, оскільки одразу відомо абсолютний шлях до файлу в файловій системі серверу.

Розглянемо основні принципи алгоритму проведення синхронізації даних клієнта по кроках:

1. В певний момент часу роботи системи клієнта до підсистеми синхронізації надходить запит на початок проведення сеансу синхронізації даних. Якщо пристрій користувача в цей момент має підключений доступ до мережі інтернет та є можливість зв’язатись з центральним сервером синхронізації то продовжуємо цей процес, якщо ні – повертається помилка про неможливість проведення синхронізації в даних момент.
2. Генеруємо новий ідентифікатор версії стану локальних даних клієнта.
3. Далі на клієнті відбувається формування пакету змінених даних. Якщо з моменту проведення останнього сеансу синхронізації жодні зміни не були внесені – список змінених рядків буде порожнім.
4. Після того як список змінених даних сформований він відправляється на центральний сервер разом з ідентифікатором версії даних останньої синхронізації та ідентифікатором поточної версії, який щойно згенеровано. Якщо синхронізація ще жодного разу не проводилась то версія останньої синхронізації буде порожньою.
5. Отримавши пакет сервіс синхронізації перевіряє чи є в ньому змінені рядки: якщо немає то клієнт не вносив змін в локальну БД, тому переходимо до кроку 10, якщо є – виконуємо процес далі.

6. Якщо версія останньої синхронізації виявилась актуальною (останньою версією на сервері) – отже клієнт працював з поточним станом даних серверу і його зміни точно не конфліктують зі змінами на сервері (Розв'язання конфліктів не потребується тому переходимо до кроку 8).

7. У тому разі якщо на попередньому кроці виявилось, що з моменту коли клієнт синхронізувавсь востаннє були внесені зміни до центрального сховища то перед збереженням його змін потрібно виконати дослідження на наявність конфліктів та розв'язати їх.

Вирішено виконувати розв'язання конфліктних ситуацій методом об'єднання даних (merge). Проводити його будемо наступним чином: з таблиці ідентифікаторів версій вибираємо всі записи, що було здійснено з часу останньої синхронізації клієнта. За цими ідентифікаторами зчитуємо *.xml-файли в яких зберігається інформація про змінені поля рядків. Якщо зміни пересікаються – на основі прийнятих правил вирішуємо якій зі змін дати перевагу.

8. Зберігаємо ідентифікатор нової версії, який було згенеровано раніше користувачем (тепер вона стає актуальною) та ідентифікатор користувача, що вносить зміни, до таблиці версій.

9. Коли конфлікти в даних розв'язано можна переходити безпосередньо до збереження змін в базу даних. Виконувати збереження слід з урахуванням ролі користувача (адміністратор, автор запису, не автор запису) та типу даних, які зберігаються (службові дані, особисті дані, публічні дані, загальнодоступні дані). До кожного збереженого запису у відповідну колонку додається ідентифікатор поточної версії. Паралельно зі збереженням записів в базу даних необхідно створити *.xml-файл, назва якого співпадає з ідентифікатором актуальної версії, та записувати в нього інформацію про змінені поля записів.

10. Після збереження змін користувача необхідно сформувати пакет змінених даних, що відбулися з часу останньої синхронізації клієнта. Перевіряємо версію даних коли востаннє синхронізувався клієнт. Якщо вона порожня, (або ж такої версії не знайдено в таблиці ідентифікаторів версій) то з цим клієнтом ще жодного разу не проводилась синхронізація і вибираємо всі записи з БД, які доступні даному клієнту. В інакшому випадку використовуємо ті ж *.xml-файли, що були вибрані на кроці 7 та формуємо пакет змін для користувача з урахуванням його ролі та типу даних таблиць.

11. Сформований пакет повертаємо на пристрій користувача, а зміни, що містяться в пакеті зберігаємо до його локальної бази даних.

12. Оскільки синхронізація відбулась успішно можна очистити журнал змін на пристрої користувача (вони вже успішно збережені на центральному сервері).

13. Зберігаємо ідентифікатор поточної версії даних. Тепер він буде використовуватись у наступному запиті на синхронізацію.

14. Сеанс синхронізації завершено. Локальні дані користувача в актуальному стані.

Виконуючи синхронізацію за описаним принципом можна досягти розповсюдження інформації та змін в ній між всіма вузлами розподіленої системи. Як видно з алгоритму центральний сервер синхронізації приймає роль пасивного вузла, оскільки кожен з клієнтів посилає запит на проведення процесу оновлення даних на центральний сервіс.

Для реалізації виконання сервером активної ролі можна модифікувати описане рішення, додавши розсилання Push-повідомлень, котрі будуть розсилатись всім розподіленим користувачам та інформувати їх про оновлення даних сервера. Після отримання такого повідомлення система користувача може провести оновлення власних даних. Тобто час між проведенням окремих сеансів синхронізації значно зменшиться, що підвищить достовірність стану локальних даних.

Для більш оптимального використання файлового сховища можна передбачити механізм архівації старих *.xml-файлів. Вірогідність їх використання з часом зменшується, а при необхідності файл легко може бути отриманий з архіву.

Насамкінець потрібно дати важливе зауваження щодо забезпечення надійності збереження даних центральної бази даних та файлів зі зміненими полями: враховуючи важливість цих даних потрібно передбачити їх дублювання та зберігання в декількох місцях. Сучасні хмарні платформи забезпечують достатній рівень надійності зберігання даних, тому вірним рішенням буде використання таких платформ для розгортання серверної частини системи.

Висновки

В результаті проведеного дослідження запропоновано новий підхід щодо синхронізації даних розподіленої системи на різних її вузлах. Такий підхід перш за все дозволяє реалізовувати механізми забезпечення актуальності даних на різних вузлах системи та поширювати зміни між ними. Основною особливістю даного методу являється розбиття даних системи на типи та здійснення контролю доступу до них в залежності від ролі користувачів. Таке рішення водночас забезпечує безпеку доступу та зменшує обсяг трафіку даних, що передаються по мережі в процесі синхронізації.

Під час дослідження цільової області було проведено детальний аналіз проблем, які притаманні синхронізації та виконано огляд існуючих методів їх розв'язання. Модифікації цих методів, що запропоновані в дослідженні, надають більш ефективний спосіб боротьби з проблемами, а запропонований метод синхронізації в залежності від типу даних взагалі позбавляє від можливості виникнення деяких з них.

Запропонований алгоритм проведення сеансу синхронізації дозволяє виявляти змінені дані на клієнті та на сервері з моменту проведення останньої синхронізації, зберігати ці зміни на обох вузлах та вирішувати конфлікти, які виникають в процесі збереження. Рішення базується на передачі замінених даних серверу та їх поширенню між всіма іншими вузлами.

Запропонована архітектура системи дозволяє здійснювати запити на проведення синхронізації даних з клієнтів на сервер та отримання всіх змін, що внесені до центральної бази даних з часу виконання останнього сеансу синхронізації, при використанні мінімальної кількості обчислювальних ресурсів. Особливістю розробки є можливість використання підходу на різних апаратних платформах. Описані ідеї можуть бути застосовані для розв'язання й інших наукових завдань.

Список посилань

1. Отслеживание изменений в корпоративной базе данных SQL Server [Електронний ресурс]. Режим доступу: http://www.cyberguru.ru/database/sqlserver/tracking-changes.html.
2. Дейт К.Дж. Введение в системы баз данных, 6-е издание: Пер. с англ. / Дейт К.Дж. – К.: Издательский дом "Вильямс", 2000. – 848 с.
3. Третяк В.Ф. Тиражирование данных в системе управления базами данных / В.Ф. Третяк, Д.Ю. Голубничий, Ю.В. Челенко // Управління розвитком – Х.: ХНЕУ, 2005. – № 3. – С. 94-95.
4. Третяк В.Ф. Актуальність побудови grid систем / В.Ф. Третяк, В.М. Приходько, Д.Ю. Голубничий // Перша науково-технічна конференція Харківського університету Повітряних Сил. – Х.: ХУ ПС, 2005. – С. 240-241.
5. Третяк В.Ф. Технология репликации в распределенных системах управления базами данных / В.Ф. Третяк // Інформаційно-керуючі системи на залізничному транспорті. – Х.: ХАЗДТ, 2004. – № 1. – С. 7-10.
6. Третяк В.Ф. Проблеми розвитку програмного забезпечення середовища розподілених обчислень GRID / В.Ф. Третяк, Д.Ю. Голубничий, І.О. Золотарьова // Матеріали третьої наукової конференції Харківського університету Повітряних Сил мені Івана Кожедуба. – Х.: ХУПС, 2007. – С. 7.
7. Голубятников И.В., Матвеев Ю.В., Сергеев И.В. Создание программного комплекса репликации СУБД-независимых данных / И.В. Голубятников, Ю.В. Матвеев, И.В. Сергеев // Системный анализ, информатика и оптимизация. Сборник научных трудов. Под ред. д.т.н., проф. Букреев В.З. – М.: РЗИТЛП, 2001. – С. 106-116.
8. Синхронизация [Електронний ресурс]. Режим доступу: http://ru.wikipedia.org/wiki/Синхронизация_%28информатика%29.
9. Ciciani B. Analysis of Concurrency-Coherency Control Protocols for Distributed Transaction Processing Systems with Regional Locality / B. Ciciani, D.M. Dias, P.S. Yu // IEEE Transactions on Software Engineering. - October 1992. – Vol. 18, No. 10. – P. 899-914.
10. Конноли Т. Базы данных: проектирование, реализация и сопровождение. Теория и практика, 2-е изд.: Пер. с англ.: Уч. пос. / Т. Конноли. – М.: Издательский дом "Вильямс", 2000. – 1120 с.
11. Database Systems / B. Ciciani, D.M. Dias, P.S. Yu // IEEE Transactions on Knowledge and Data Engineering. – June 1990. – Vol. 2, No. 2. – P. 247-261.
12. Использование Oracle 8 / [Вильям Г. Пейдж и др.]. – К.; М.; СПб.: Изд. дом «Вильямс», 1998. – 752 с.
13. Мейер М. Теория реляционных баз данных / М. Мейер. – М.: Мир, 1987. – 608 с.

14. Саймон А.Р. Стратегические технологи баз данных: менеджмент на 2000 год. Пер. с англ. / А.Р. Саймон. Под ред. и с предисл. М.Р. Когаловского. – М.: Финансы и статистика, 1999. – 479 с.
15. В.Ф. Третьяк. Использование техрологии репликации в системе управления распределенными базами данных / В.Ф. Третьяк, И.Е. Кужель, В.М. Приходько // Збірник праць Храківського університету Повітряних Сил, – Х., 2010. – № 2(24).
16. Киреев В.А. Проектирование хранилища данных социальной информации. – Конференция Сибирского федералного университета. Математика и компютерные науки, 2011.
17. А.Д. Плутенко, А.А. Ситников. Проблемы имитационного моделирования процесса журнальной репликации данных в распределенной СУБД. Амурский государственный университет 2005.
18. A. Tridgell, P. Mackerras. The Rsync Algorithm. Technical Report TR-CS-96-05, Department of Computer Science, The Australian National University, Canberra, Australia, 1996.

УДК 004.056.5

МАРКОВСЬКИЙ О.П.,
ТЕРЕЩЕНКО С.Ю.,
ФЕДОРЕЧКО О.І.

МЕТОД ВИПРАВЛЕННЯ ТРЬОХКРАТНИХ ПОМИЛОК ПЕРЕДАЧІ ДАНИХ В ДВІЙКОВИХ СИМЕТРИЧНИХ КАНАЛАХ

В статті пропонується метод корекції трьохкратних помилок в двійкових симетричних каналах передачі даних. Метод базується на використанні логічних зважених контрольних сум. Детально описані процедури формування контрольного коду та корекції одно, двократних і трьохкратних помилок. Наведено числові приклади. Проведено теоретичне та експериментальне оцінювання часу корекції для запропонованого методу. Показано, що головною перевагою запропонованого методу є те, що час потрібний для корекції помилок не залежить від довжини блоку даних і те, що цей час суттєво менший в порівнянні з відомими методами виправлення багатократних помилок.

This paper investigates a simple and effective method for correction of triple errors that occur during data transmission in bimary symmetric channel is proposed The method is based on logical weighted checksum. The proposed procedure for control code forming and correction of single, double and triple data transmission errors are described in details. A numerical example for procedure are given. The theoretical and experimental estimation of error correction time for proposed method are presented. It has been shown that main advantage of proposed method is that errors correction time is not depend of data transmission block length and is significantly less in compare to known multiple errors correction methods.

Вступ

Вдосконалення технології передачі даних є одним з найважливіших чинників поглиблення глобального процесу інформаційної інтеграції. Разом з тим, традиційно, передача даних є одним з найбільш критичних, з точки зору надійності, процесів обробки інформації в комп'ютерних системах. Інтенсивність виникнення помилок при передачі даних на декілька порядків вища в порівнянні з обчисленнями на процесорних вузлах комп'ютерних систем. Тому однією з вузлових проблем розвитку комп'ютерних технологій є забезпечення високої надійності передачі даних в комп'ютерних системах, в тому числі, в лініях, теоретичною моделлю яких є двійковий симетричний канал.

В сучасних системах комп'ютерної обробки даних домінуючу роль в процесах обміну інформацією між компонентами відіграють саме послідовні інтерфейси, які забезпечують вищу швидкість та суттєво більшу дальність передачі даних між компонентами обчислювальних систем в порівнянні з паралельними лініями. Теоретичною моделлю більшості послідовних інтерфейсів компонентів комп'ютерних систем є двійковий симетричний канал.

В останнє десятиліття проблема забезпечення ефективного контролю та корекції помилок передачі даних стає ще більш важливою. З одного боку, це пов'язано, з динамічним розвитком розподілених комп'ютерних систем, для яких процеси обміну даними відіграють домінуючу роль. З іншого, дієвим чинником протягом останнього десятиліття значно зросли об'єми та швидкість передачі даних. Ці процеси супроводжуються рядом чинників, що негативно впливають на надійність передачі цифрових даних. Зокрема, зменшення часових інтервалів між сигналами мають наслідком зростання числа помилок, які викликані явищами міжсигнальної інтерференції. Зростання інтенсивності зовнішніх електромагнітних полів, зумовлене динамічним розширенням використання ефірних, бездротових технологій передачі має наслідком зростання помилок, викликаних зовнішніми завадами.

Важливим аспектом вдосконалення технології передачі даних між компонентами комп'ютерних систем є проблема ефективного виправлення виявлених помилок. Повторна передача блоку в тисячі біт при виявленні одного спотвореного біту стає всі менш ефективною з огляду на стійку тенденцію до зростання довжини блоку. Тому актуальним є пошук можливостей для підвищення ефективності процесу корекції помилок.

Наведені фактори диктують необхідність вдосконалення засобів контролю та корекції помилок, адекватного прогресу технології передачі інформації.

Разом з тим, триває процес розширення використання розподілених комп'ютерних систем в усіх сферах людської діяльності, включаючи і ті, що пов'язані з техногенним ризиком. Це потребує високої надійності всіх компонент обчислювальних систем, включаючи засоби передачі даних.

Наведене визначає необхідність підвищення ефективності засобів виявлення і виправлення помилок. Досягнутий в останні роки прогрес в галузі інтегральної технології відкриває нові можливості для вдосконалення засобів контролю та корекції помилок в лініях передачі даних.

Таким чином, проблема підвищення ефективності виявлення та виправлення помилок передачі даних в лініях, теоретичною моделлю яких є двійковий симетричний канал з огляду на особливості сучасного етапу розвитку інформаційних технологій є актуальною та важливою для практики.

Аналіз існуючих методів корекції помилок передачі даних

Більшість ліній обміну даними комп'ютерних систем можна розглядати як низькочастотні канали з імпульсно-кодовою модуляцією двійкових кодів. З точки зору характеру виникнення помилок, теоретичною моделлю таких ліній є двійковий симетричний канал.

Виходячи з теоретичної моделі двійкового симетричного каналу помилки передачі даних в таких лініях можна розглядати як незалежні одна від одної, а кількість помилок, що виникає при передачі блоку даних підпорядкована біноміальному закону розподілу [1].

Основними причинами виникнення помилок є зовнішні завади та міжсигнальна інтерференція [2]. Відповідно, виникаючі помилки мають характер бітових спотворень, причому кількість помилок, в цілому, підпорядкована біноміальному закону розподілу. Фактично це означає, що домінуючим типом помилок є спотворення одного біту. Значно рідше трапляються двократні помилки і ще рідше - трьохкратні. Однак, приймаючи до уваги зростання об'ємів даних, що нині становлять мільйони байтів, ймовірність появи трьохкратної помилки є цілком реальною.

Для забезпечення високої надійності передачі даних в лініях обміну даних комп'ютерних систем, теоретичною моделлю яких є двійковий симетричний канал до теперішнього часу розроблено широкий арсенал засобів [1-5].

Для широкого класу сучасних комп'ютерних систем, що працюють в реальному часі, ключовою характеристикою є час, потрібний для виконання операції корекції помилок. З іншого боку, значимість об'єму контрольної інформації в сучасних умовах динамічного зростання швидкості передачі падає [1].

Традиційними для корекції помилок передачі даних є дві технології

- технологія прямого виправлення помилок (*FEC-forward error correction*) за рахунок використання корегуючих кодів, які дозволяють прямо виправити обмежену кількість помилок передачі даних за рахунок контрольних кодів, що передаються разом з блоком;
- технологія автоматичного повтору передачі в разі виявлення помилок (*ARQ-Automatic Repeat Request*).

Перевагою *ARQ*-технології є менша кількість контрольних розрядів та простота виявлення помилки. Основним недоліком є суттєві затримки при повторній передачі блоку даних. З огляду на специфіку комп'ютерних систем, що працюють в реальному часі, вказаний недолік є доволі суттєвим. Причому, вважаючи на стійку тенденцію до збільшення розмірів блоків значимість вказаного недоліку зростає.

З корегуючи кодів, що орієнтовані на виправлення бітових спотворень найбільш відомими є коди Хемінга [3] та БЧХ [4], а також коди на основі зважених сум [5]. Коди Хемінга доволі просто корегують однократні помилки. Значно складніше виконується корегування двократної помилки. Фактично, для локалізації позицій в блоці спотворених бітів потрібно розв'язувати систему нелінійних рівнянь на полях Галуа. Технологічно розв'язання такої системи рівнянь виконується шляхом перебору, об'єм якого пропорційний n^2, де n - довжина блоку. Перевагою застосування кодів Хемінгу при корекції двократної помилки є те, що вони потребують мінімальної кількості контрольних розрядів - $2 \cdot \log_2 n$.

Більш досконалими з точки зору часу корекції пари помилок є коди БЧХ, подальшим розвитком яких є коди Ріда-Соломона. Як і коди Хемінга, в основі цих корегуючи кодів лежить використання арифметики полів Галуа. Для корекції двократних помилок ці коди використовують більше контрольних розрядів - $4 \cdot \log_2 n$. Це дає змогу зменшити об'єм перебору до n для локалізації позицій спотворених бітів.

Проте використання перебору за умови тенденції зростання довжини блоку до 1-2 Кбайтів робить використання кодів Хемінга і БЧХ для корекції помилок обміну даних в системах реального часу неефективним. Крім того, архітектура та система команд сучасних мікроконтролерів не пристосовані для виконання операцій в арифметиці полів Галуа.

Одним з найбільш простих і ефективних методів корекції однократних помилок передачі даних є зважені контрольні суми [4]. Якщо позначити біти блоку B, що передається, як $b_1,b_2,...,b_m$, то зважена контрольна сума S обчислюється як сума за модулем 2 добутків бітів блоку на їх ваговий коефіцієнт, в якості якого виступає їх порядковий номер (починаючи з одиниці) в блоці:

$$S = \bigoplus_{j=1}^{m} b_j \cdot j \qquad (1)$$

Оскільки розрядність номеру j дорівнює $\lceil \log_2 (m+1) \rceil$, то кількість η контрольних розрядів також становить дорівнює $\eta = \lceil \log_2 (m+1) \rceil$. Зважена контрольна сума обчислюється на передавачеві (позначається як S), а також на стороні приймача по реально прийнятому блоку $B'=\{b'_1, b'_2,...,b'_m\}$. Обчислена на боці приймача зважена контрольна сума починається як S'. На стороні приймача обчислюється також різниця Δ_0 зважених контрольних сум передавача та приймача, як сума за модулем 2 відповідних кодів:

$$\Delta_0 = S \oplus S' = \bigoplus_{j=1}^{m} b_j \cdot j \oplus \bigoplus_{j=1}^{m} b'_j \cdot j = \bigoplus_{j=1}^{m} (b_j \oplus b'_j) \cdot j \qquad (2)$$

Очевидно, що при виникненні однократної помилки передачі q-го біту, $q \in \{1,..,m\}$, $\Delta_0 = q$, тобто однозначно локалізує біт, що було передано з помилкою.

Проте існуючі варіанти зважених контрольних сум не дозволяють виправляти багатократні помилки.

Таким чином, існуючі корегуючи коди не забезпечують ефективної корекції трьохкратних помилок обміну даними в сучасних комп'ютерних системах управління об'єктами та процесами в реальному часі.

Ціллю роботи є створення методу корекції одно, двох та трьохкратних помилок передачі даних, який забезпечує високу швидкість локалізації спотворених бітів без використання перебору і орієнтований на використання простих логічних операцій, що можуть бути ефективно реалізовані апаратними засобами.

Метод корекції трьохкратних помилок

Якщо поставити за мету корегування більш, ніж одного біту, то потрібно визначити позиції спотворених при передачі бітів.

Зокрема, якщо поставити на меті гарантоване корегування 3-кратних помилок, то задача зводиться до визначення порядкових номерів q, p, g цих бітів. Для визначеності можна вважати, що $q < p < g$. Визначення чисельних значень q, p та g може бути виконано в результаті розв'язання системи рівнянь:

$$\begin{cases} F_1(q,p,g) = 0 \\ F_2(q,p,g) = 0 \\ F_3(q,p,g) = 0 \end{cases} \qquad (3)$$

де F_1, F_2 та F_3 - деякі функції.

У найпростішому випадку функції F_1, F_2, F_3 є лінійними. Тоді розв'язання системи (3) може бути виконане аналітично за час, що не залежить від довжини m блоку даних.

Очевидно, що для того, аби розв'язання системи (3) було успішним, кожен із невідомих компонентів q, p і g має входити хоча б до однієї із функцій F_1, F_2, F_3. Також, лінійні рівняння, що утворюють систему (3) мають бути лінійно-незалежними.

Для гарантованої корекції 3-кратної помилки пропонується наступний метод формування системи (3). В якості першого рівняння пропонується використати рівняння (2) – різницю зважених контрольних сум приймача і передавача. Наступні η рівнянь пропонується формувати таким чином. Номер j будь якого з бітів блоку може бути представлено у двійковій системі в наступному вигляді: $j = j_1 + 2 \cdot j_2 + 4 \cdot j_3 + 2^{\eta-1} \cdot j_\eta$.

Кожне i-те рівняння (i=1,..,η) формується як різниця суми за модулем 2 таких номерів одиничних бітів блоку передавача, у яких значення i-го двійкового розряду дорівнює одиниці і суми за модулем 2 таких само номерів одиничних бітів прийнятого блоку:

$$\Delta_i = \bigoplus_{j=1, j_i=1}^{m} b_j \cdot j \oplus \bigoplus_{j=1, j_i=1}^{m} b'_j \cdot j = \bigoplus_{j=1, j_i=1}^{m} (b_j \oplus b'_j) \cdot j \qquad (4)$$

Оскільки тільки q,p,g - позиції спотворених бітів, то $b_q \oplus b'_q = 1$, $b_p \oplus b'_p = 1$ та $b_g \oplus b'_g = 1$, то перше рівняння матиме вигляд:

$$q \oplus p \oplus g = \Delta_0 \qquad (5)$$

Інші η рівнянь матимуть вигляд:

$$\forall i = 1,..,\eta : \Delta_i = q_i \cdot q \oplus p_i \cdot p \oplus g_i \cdot g \qquad (6)$$

де q_i, p_i, g_i – значення i-го розряду відповідно кодів q, p та g.

З отриманої таким чином системи рівнянь виключаються лінійно залежні, в результаті чого отримується система з 3-х лінійних рівнянь, яка може бути розв'язана відомими способами.

Наприклад, коли m=15 і, відповідно, η=4, можна допустити, що в результаті виникнення 3-кратної помилки спотворені біти будуть з номерами g=3, p=7, q=15. Відповідно, значення двійкових розрядів q дорівнюють: q_1=1, q_2=1, q_3=1 та q_4 =1. Значення двійкових розрядів p дорівнюють: p_1=1, p_2=1, p_3=1 та p_4=0; значення двійкових розрядів g дорівнюють: g_1=1, g_2=1, g_3 =0 та g_4 =0 .

З урахуванням цього система з 5-ти рівнянь матиме такий вигляд:

$$\begin{cases} q \oplus p \oplus g = \Delta_0 = 11 \\ q \oplus p \oplus g = \Delta_1 = 11 \\ q \oplus p \oplus g = \Delta_2 = 11 \\ q \oplus p = \Delta_3 = 8 \\ q = \Delta_4 = 15 \end{cases} \qquad (7)$$

Три перших рівняння системи (7) є ідентичними, тобто два з них можуть бути вилучені так, що в результаті залишиться ортогональна система лінійних рівнянь, розв'язання якої однозначно визначає позиції спотворених при передачі бітів блоку даних.

$$\begin{cases} q \oplus p \oplus g = \Delta_0 = 11 \\ q \oplus p = \Delta_3 = 8 \\ q = \Delta_4 = 15 \end{cases} \qquad (8)$$

де q_i, p_i, g_i – значення i-го розряду відповідно кодів q, p та g.

Покажемо, що завжди існує можливість виділення з η+1 рівнянь, що формуються, описаним вище способом, трьох лінійно-незалежних, які однозначно визначають значення q, p і g.

Для локалізації 3-х помилок шляхом визначення їх позицій q, p і g в блоці, має існувати така пара розрядів l і r, що три вектори <1, 1, 1>, $< q_l, p_l, g_l >$, $< q_r, p_r, g_r >$ є лінійно незалежними. Вказана умова виконується, якщо всі компоненти векторів $< q_l, p_l, g_l >$ і $< q_r, p_r, g_r >$ не дорівнюють нулю, не дорівнюють одне одному, а також якщо їх сума за модулем 2 не дорівнює <1, 1, 1>.

Якщо вважати, що $q > p > g$, то обов'язково існує одна з двох ситуацій:

1) Існує розряд l такий, що q_l=1, p_l =1, g_l =0 і розряд $r<l$ такий, що q_r = 1, p_r = 0. Очевидно, що для трьох векторів <1, 1, 1>, < 1,1,0>,<1,0,x> (де x – невизначене значення) виконуються умови лінійної незалежності.

2) Існує розряд l такий, що q_l =1, p_l =0, g_l =0 і розряд $r<l$ такий, що p_r = 1, g_r = 0. Для трійки векторів <1, 1, 1>, < 1,0,0>, <x,1,0> , що утворюються в цій ситуації також повністю виконуються вказані вище умови лінійної незалежності.

Таким чином при формуванні рівнянь у вигляді (6) система (3) завжди матиме три лінійно-незалежні рівняння, які однозначно дозволяють визначити численні значення позицій q, p, g спотворених при передачі бітів блоку. Розроблена технологія корекції багатократних помилок передачі даних включає в себе визначення їх кратності та безпосередньо процедуру виправлення.

Якщо всі різниці контрольних кодів передавача та приймача дорівнюють нулю, тобто якщо $\forall i \in \{0...\eta\}:\Delta_i=0$, то вважається, що помилки відсутні.

Якщо всі різниці контрольних кодів приймають лише два значення, тобто $\forall i \in \{0...\eta\}:\Delta_i=\in\{0,q\}$, а $\Delta_0 = q$, причому нульові компоненти Δ_i відповідають нульовим розрядам двійкового коду q: якщо Δ_i =0, то q_i =0, якщо Δ_i = q,то q_i =1, то така ситуація класифікується як однократна помилка. Позиція спотвореного при цьому біту визначається значенням q. Відповідно, корекція такої помилки зводиться до інвертування q-го біту даних.

При виникненні двох помилок на позиціях p і q різниця $\Delta_0 = q \oplus p$. Всі інші різниці контрольних кодів приймача і передавача приймають одне з 4-х значень: $p \oplus q$, p, q та 0: $\forall i \in \{0...\eta\}:\Delta_i=\in\{ p \oplus q, p, q, 0\}$. При цьому обов'язково існуватиме різниця $\Delta_u \neq 0$; $\Delta_u \neq \Delta_0$, $u \in \{1,..,\eta\}$ така, що визначатиме позицію q в блоці однієї з помилок: $\Delta_u=q$, причому для всіх v: $u<v\leq\eta$: $\Delta_v \in \{\Delta_0,0\}$. Ідентифікація наявності саме 2-х помилок може бути виконана таким чином:

Визначається код $q=\Delta_u$ (де Δ_u- старший з ненульових кодів, що не дорівнює Δ_0). Для всіх i=1...η проводиться перевірка: u-тий розряд коду q має дорівнювати одиниці. Аналогічно u-тий розряд Δ_0 має дорівнювати одиниці. Якщо Δ_i , $\forall i \in \{0...\eta\}$ приймають три значення: Δ_0, Δ_u

та 0, то такої перевірки достатньо, щоб однозначно наявність двох помилок. Якщо Δ_i , $\forall i \in \{0...\eta\}$ приймають чотири значення: Δ_0, Δ_u, 0 та Δ_y то слід додатково перевірити, що y-ті розряди коду $q=\Delta_u$ дорівнюють нулю.

Виправлення 2-кратної помилки виконується в такому порядку: позиція другого зі спотворених при передачі бітів блоку - q дорівнює коду Δ_u- старшому з ненульових кодів, що не дорівнює Δ_0. Позиція p першого пошкодженого при передачі біту визначається як сума за модулем два: $p = \Delta_0 \oplus \Delta_u$.

При виникненні трьох помилок передачі блоку в позиціях q,p та g, причому $q>p>g$, різниці контрольних кодів передавача та приймача можуть приймати від 4-х до 8-ми значень з множини: 0, $q\oplus p\oplus g$, $p\oplus g$, $q\oplus g$, $q\oplus p$, q, p, g, 0.

Як було показано вище, в силу нерівностей $q > p > g$, обов'язково реалізується одна з двох ситуацій:

- Існує розряд l такий, що $q_l=1$, $p_l=1$, $g_l=0$ і розряд $r<l$ такий, що $q_r = 1$, $p_r = 0$.
- Існує розряд l такий, що $q_l=1$, $p_l=0$, $g_l=0$ і розряд $r<l$ такий, що $p_r = 1$, $g_r = 0$.

Фактично, перша ситуація означає, що існує $\Delta_l = q \oplus p$, причому для всіх v більших від l значення різниці Δ_v дорівнює або Δ_0 (v-тий розряд всіх трьох чисел q,p і g дорівнює одиниці) або 0 (v-тий розряд всіх трьох чисел q,p і g дорівнює нулю) : $l<v\le\eta$: $\Delta_v \in \{\Delta_0,0\}$. Це дозволяє доволі просто виявити значення l. Чисельне значення g визначається як $g=\Delta_l \oplus \Delta_0$. Для різниць Δ_u що лежать між Δ_l та Δ_r: $r<u<l$ відповідні розряди q та p приймають однакові одиночні значення. Це означає, що Δ_u може приймати лише три значення: $\Delta_0 = q\oplus p\oplus g$ (u-ті розряди всіх трьох чисел q,p і g дорівнюють одиниці), $\Delta_l = q \oplus p$ (u-ті розряди чисел q і p дорівнюють одиниці, а u-тий розряд g дорівнює нулю), 0 (u-ті розряди всіх трьох чисел q,p і g дорівнюють нулю): $r<u< l$: $\Delta_u \in \{\Delta_0,\Delta_l,0\}$. Це дозволяє доволі просто виявити значення r. Як зазначалось вище, можливими є два значення Δ_r: $\Delta_r = q$ і $\Delta_r = q\oplus g$. Якщо r-тий розряд g дорівнює нулю, то $\Delta_r = q$ інакше $\Delta_r = q\oplus g$. В першому випадку значення позиції p визначається як сума $p= \Delta_l \oplus \Delta_r$. В другому випадку значення q визначається як $q=\Delta_0 \oplus \Delta_l \oplus\Delta_r$, значення позиції p визначається як сума $p= \Delta_0 \oplus \Delta_r$.

Друга ситуація полягає в тому, що існує $\Delta_l = q$, причому для всіх v більших від l значення різниці Δ_v дорівнює або Δ_0 (v-тий розряд всіх трьох чисел q,p і g дорівнює одиниці) або 0 (v-тий розряд всіх трьох чисел q,p і g дорівнює нулю) : $l<v\le\eta$: $\Delta_v \in \{\Delta_0,0\}$. Це означає, що значення l може бути виявлене доволі просто. Чисельне значення q визначається як $q=\Delta_l$. Для різниць Δ_u що лежать між Δ_l та Δ_r: $r<u<l$ відповідні розряди p та g приймають однакові значення. Це означає, що Δ_u може приймати лише чотири значення: $\Delta_0 = q\oplus p\oplus g$ (u-ті розряди всіх трьох чисел q,p і g дорівнюють одиниці), $\Delta_h = p \oplus g =\Delta_0 \oplus \Delta_l$ (u-ті розряди чисел p і g дорівнюють одиниці, а u-тий розряд q дорівнює нулю), $\Delta_h = q= \Delta_l$ (u-ті розряди чисел p і g дорівнюють нулю, а u-тий розряд q дорівнює одиниці), 0 (u-ті розряди всіх трьох чисел q,p і g дорівнюють нулю): $r<u< l$: $\Delta_u \in \{\Delta_0,\Delta_l,\Delta_0\oplus\Delta_l,0\}$. Це дозволяє доволі просто виявити значення r. Як зазначалось вище, можливими є два значення Δ_r: $\Delta_r = p$ і $\Delta_r = q\oplus p$. Якщо r-тий розряд q дорівнює нулю, то $\Delta_r = p$ інакше $\Delta_r = q\oplus p$. В першому випадку значення g визначається як: $g=\Delta_0 \oplus \Delta_l \oplus \Delta_r$. В другому випадку значення p визначається як $p=\Delta_r \oplus \Delta_l$, а значення g обчислюється як $g=\Delta_0 \oplus \Delta_r$.

Таким чином, запропонований спосіб корекції помилок передачі даних передбачає наступну послідовність виявлення позицій спотворених при передачі бітів блоку.

1) Номер j поточної різниці контрольних кодів приймача та передавача встановити в η: $j= \eta$.

2) Якщо $\Delta_j = \Delta_0$ або $\Delta_j =0$ зменшити на одиницю значення j: $j=j-1$ і повернутися на п.2.

3) Якщо j-тий розряд Δ_j дорівнює одиниці, то має місце друга ситуація і виконується перехід на п.8 алгоритму.

4) Позиція g першого з пошкоджених при передачі блоку бітів визначається як сума: $g=\Delta_l \oplus \Delta_0$. Фіксується значення l: $l=j$.

5) Виконується декремент j: $j=j-1$. Якщо $\Delta_j \in \{\Delta_0,\Delta_l,0\}$, то перехід на повторне виконання п.5 алгоритму.

6) Якщо $\Delta_j\neq\Delta_0$, $\Delta_j\neq \Delta_l$, $\Delta_j\neq0$, то аналізується j-тий розряд визначеного в п.4 коду g: в разі якщо j-тий розряд g дорівнює нулю, то перехід на п.7., інакше значення позиції q визначається як $q=\Delta_0 \oplus \Delta_l \oplus\Delta_j$, а значення позиції p визначається як сума $p= \Delta_0 \oplus \Delta_j$. Кінець.

7) Позиція q останнього з пошкоджених при передачі бітів дорівнює Δ_j : $q=\Delta_j$, а позиція тре-

тього пошкодженого біту обчислюється як сума: $p = \Delta_l \oplus \Delta_j$. Кінець.

8) Позиція q останнього зі спотворених при передачі блоку бітів визначається як сума: $q=\Delta_j$. Фіксується значення l: $l=j$.

9) Виконується декремент j: $j=j-1$. Якщо $\Delta_j \in \{\Delta_0, \Delta_l, \Delta_0 \oplus \Delta_l, 0\}$, то перехід на повторне виконання п.9 алгоритму.

10) Якщо $\Delta_j \neq \Delta_0$, $\Delta_j \neq \Delta_l$, $\Delta_j \neq \Delta_0 \oplus \Delta_l$, $\Delta_j \neq 0$, то аналізується j-тий розряд визначеного в п.8 коду q: якщо j-тий розряд q дорівнює нулю, то перехід на п.11., інакше значення позиції p визначається як $p=\Delta_j \oplus \Delta_l$, а значення позиції g обчислюється як сума $g=\Delta_0 \oplus \Delta_j$. Кінець.

11) Позиція p середнього з пошкоджених при передачі бітів дорівнює Δ_j : $p=\Delta_j$, а позиція g першого пошкодженого біту обчислюється як сума: $g=\Delta_0 \oplus \Delta_l \oplus \Delta_j$. Кінець.

Описаний алгоритм визначення позицій спотворених при передачі бітів блоку даних може бути ілюстровано наступним прикладом корекції 3-кратних помилок при передачі 15-бітового блоку B={1,0,1,0,0,0,0,1,0,1,1,1,0,0,0}, m=15, а η=4. Відповідно, на боці передавача контрольні коди обчислюються у наступному вигляді: $S_0 = 1 \oplus 3 \oplus 8 \oplus 10 \oplus 11 \oplus 12 = 7$; $S_1 = 1 \oplus 3 \oplus 11 = 9$; $S_2 = 3 \oplus 10 \oplus 11 = 2$; $S_3 = 12$; $S_4 = 8 \oplus 10 \oplus 11 \oplus 12 = 5$. Якщо припустити, що при передачі пошкоджено 8-й, 9-й та 14-й біти блоку (q = 14, p = 9 та g = 8), то на стороні приймача блок B′ даних матиме такий вигляд B′={1,0,1,0,0,0,0,**0**,**1**,1,1,1,0,**1**,0}. Відповідно, контрольні коди на боці приймача обчислюються у такому вигляді: $S_0' = 1 \oplus 3 \oplus 9 \oplus 10 \oplus 11 \oplus 12 \oplus 14 = 8$; $S_1' = 1 \oplus 3 \oplus 9 \oplus 11 = 0$; $S_2' = 3 \oplus 10 \oplus 11 \oplus 14 = 12$; $S_3' = 12 \oplus 14 = 2$; $S_4' = 9 \oplus 10 \oplus 11 \oplus 12 \oplus 14 = 10$. Обчислені приймачем різниці його контрольних кодів з прийнятими від передавача мають такий вигляд: $\Delta_0 = S_0 \oplus S_0' = 7 \oplus 8 = 15$; $\Delta_1 = S_1 \oplus S_1' = 9 \oplus 0 = 9$; $\Delta_2 = S_2 \oplus S_2' = 2 \oplus 12 = 14$; $\Delta_3 = S_3 \oplus S_3' = 12 \oplus 2 = 14$; $\Delta_4 = S_4 \oplus S_4' = 5 \oplus 10 = 15$;

Згідно з викладеним вище алгоритмом локалізації помилково переданих бітів блоку, номер j поточної різниці контрольних кодів приймача та передавача встановлюється в η=4: j=4. Оскільки $\Delta_4 = 15 = \Delta_0$, то значення j згідно в п.2 зменшується на одиницю: j=3. Відповідне значення $\Delta_3 = 14 \neq \Delta_0$. У відповідності до п.3 алгоритму аналізується 3-й біт $\Delta_3 = 14 = 1\mathbf{1}10_2$: він дорівнює одиниці. Це означає, що має місце друга ситуація - тобто $q=\Delta_3$ =14. У відповідності з п.8 фіксується значення $l=j$=3. Далі, згідно п.9 виконується декремент j : j= 3-1=2. Відповідне значення Δ_2= 14 порівнюється з чотирма кодами: Δ_0=15, Δ_3 = 14, та $\Delta_0 \oplus \Delta_3 = 15 \oplus 14 = 1$ та 0. Оскільки $\Delta_2 = \Delta_3$ =14, то п.9 алгоритму виконується повторно: декрементується значення j : j= 2-1=1. Відповідна різниця Δ_1= 9 не дорівнює ні одному з кодів: Δ_0=15, Δ_3 = 14, $\Delta_0 \oplus \Delta_3 = 15 \oplus 14 = 1$ та 0. Тому, у відповідності з п.10 алгоритму, аналізується значення першого розряду визначеного в п.8 числа q = 111**0**. Оскільки цей розряд дорівнює нулю, то реалізується перехід на п.11 алгоритму, у відповідності з яким значення p визначається як $p=\Delta_1$ = 9. Значення g обчислюється у вигляді суми: $g=\Delta_0 \oplus \Delta_3 \oplus \Delta_1 = 15 \oplus 14 \oplus 9 = 8$. Таким чином, для наведено прикладу визначено позиції спотворених при передачі трьох бітів: q=14, p=9, g=8. Виправлення вказаних бітів виконується шляхом їх інвертування в прийнятому блоці даних.

Аналіз ефективності

Основною перевагою запропонованого методу корекції 3-кратних помилок в порівнянні з відомими є прискорення та спрощення процедури корекції в порівнянні з відомими методами. З викладеної в попередньому розділі запропонованої методики корекції трьохкратних помилок слідує, що вона передбачає послідовний перебір різниць контрольних кодів передача та приймача, кількість яких не перевищує $\log_2 m$. При цьому середнє значення кількості різнить становить $0.75 \cdot \log_2 m$. На кожному кроці аналізу виконується одна операція порівняння та логічного додавання. Таким чином, оцінкою складності операції корекції є $O(0.75 \cdot \log_2 m)$. При цьому використовуються прості логічні операції, що можуть бути просто реалізовані відповідними апаратними засобами.

Відомі методи корекції багатократних помилок, зокрема лінійні та циклічні коди і їх модифікації передбачають для локалізації помилок послідовну процедуру аналізу всіх бітів блоку. Зокрема, модифікації коду Хемінга [2] для корекції трьохкратних помилок прямо передбачає перебір всіх можливих позицій трансформованих при передачі бітів. Очевидно, що об'єм такого перебору пропорційний m^3, причому, для кожного з можливих значень позицій спотворених бітів виконуються операції мно-

ження та ділення, тобто виконується об'єм обчислень, складність яких значно перевищує операції логічного додавання за модулем 2, що використовуються в запропонованому методі.

При використанні циклічних кодів для локалізації 3-картних помилок передачі даних потрібно виконувати обчислення синдрому помилки для кожного з m бітів блоку. Обчислення синдрому передбачає 3 операції множення η-розрядних чисел та 3 операції додавання за модулем 2. Вважаючи, що складність операції множення приблизно в η раз більша за складність операції додавання, коректно вважати, що складність обчислення синдрому при використанні циклічних кодів $O(3\cdot\eta)$. Таким чином, загальна складність операції локалізації 3-кратних помилок при використанні циклічних кодів може бути оцінена як $O(3\cdot\eta\cdot m)$ або $O(3\cdot m\cdot\log_2 m)$.

Виходячи з наведеного аналізу можна дійти до висновку, що запропонований метод корекції помилок, кратність яких не перевищує трьох забезпечує зменшення обчислювальної складності локалізації помилок в порівнянні з відомими методами не менше ніж у $3\cdot m$ раз. Вважаючи, що розмір блоків обміну між компонентами комп'ютерних систем становить 128-1024 байтів, виграш у часу корекції становить суттєву величину.

Досягнутий суттєвий виграш у часі виправлення помилок значною міною зумовлений тим, що запропонований метод використовує суттєво більшу кількість контрольної інформації. Кількість k контрольних розрядів, які передаються з блоком даних визначається формулою:

$$k = \eta\cdot(1+\eta) = \lceil\log_2 m\rceil\cdot(1+\lceil\log_2 m\rceil) \qquad (9)$$

Неважко показати, що кВ порівнянні з відомими методами корекції багатократних помилок, зокрема корегуючими кодами Хемінга та БЧХ, запропонований метод потребує більшої кількості контрольних розрядів. Проте у сучасних умовах зростання швидкості передачі даних між компонентами комп'ютерних систем, яка сягає десятки і сотні Мегабіт за секунду, значимість передачі додаткових контрольних бітів не є суттєвою в порівнянні зі значним виграшем у часі корекції помилок.

Важливий для практики чинник ефективності розробленого методу полягає у тому, що в його рамках може бути здійснена з доволі великою ймовірністю корекція помилок кратністю більше 3-х. При цьому, принципова можливість корекції h-кратної помилки, що виникає при передачі m-бітового блоку виникає лише за умови, що $h<\log_2 m$. Ймовірність можливості корекції багатократних помилок, кратністю більше за три зростає зі збільшенням довжини блоку. Зокрема, ймовірність ρ_4 того, що 4-кратна помилка не може бути виправлена з використанні запропонованого методу при $m>15$ визначається формулою:

$$\rho_4 = \frac{\log_2^2 m}{8\cdot m^3} \qquad (10)$$

Наприклад, ймовірність ρ_4 того, що 4-кратна помилка не може бути виправлена для $m=1024$ становить 10^{-9}: $\rho_4=10^{-9}$. Аналогічно, можна показати, що 5-ти кратна помилка для блоків такої довжини виправляється зі ймовірністю 0.9999.

Вважаючи на те, що відомі корегуючі коди мають жорстко обмежену кількість помилок, що можуть бути виправлені, запропонований метод, гарантуючи виправлення 3-кратних помилок, має значно більшу корегуючу здатність, дозволяючи виправляти значну частину помилок більшої кратності.

Проведені експериментальні дослідження показали, що використання запропонованого методу при довжині блоку в 1024 байтів дозволяє прискорити процес корекції трьохкратних помилок приблизно на чотири порядки в порівнянні з БЧХ-кодами. При цьому досліджувалась програмна модель: вважаючи на значно простіші операції, що використовуються в запропонованому методі, при апаратній реалізації, виграш у часі корекції буде суттєво більшим.

Зрозуміло, що таке суттєве підвищення швидкості корекції помилок обміну даних між компонентами комп'ютерних систем управління дозволяє відповідним чином скоротити максимальний час реакції цих систем на зміну стану об'єкту управління.

Висновки

В результаті проведених досліджень запропоновано простий метод корекції помилок передачі даних, який гарантовано виправляє всі помилки передачі даних, кратність яких не перевищує трьох та з великою ймовірністю корегує помилки більшої кратності за умови, що вона не перевищує $\log_2 m$. Метод дозволяє підвищити кратність та швидкість корекції помилок при обміні даними між компонентами

комп'ютерних систем по лініях з кодово-імпульсною модуляцією.

Локалізація позицій спотворених при передачі бітів блоку в запропонованому методі зводиться до розв'язання системи лінійних булевих рівнянь, що, разом з використанням простих логічних операцій зумовлює його ефективну апаратну реалізацію. Таким чином, метод, на відміну від аналогічних модифікацій коду Хемінга та БЧХ, для розв'язання системи рівнянь не використовує перебір, що забезпечує виграш в часу корекції, пропорційний довжині блоку даних. Суттєве прискорення реалізації операцій корекції помилок та підвищення корегуючої здатності в порівнянні з відомими методами досягнуто за рахунок збільшення кількості контрольних розрядів, що для сучасних високошвидкісних каналів обміну даними не є критичним.

Метод орієнтовано для використання в комп'ютеризованих системах управління, що працюють у реальному часі.

Список посилань

1. Klove T. Error Detecting Codes: General Theory and Their Application in Feedback Communication Systems / T. Klove, V. Korzhik.- Norwell, MA: Kluwer, 1995. – 433 p.
2. 2.Хемминг Р.В. Теория кодирования и теория информации / Р.В.Хемминг.; пер.с англ. С.И.Гельдфанда.- М.:Радио и связь, 1983.- 176 с.
3. Морелос-Сарагоса Р. Искусство помехоустойчивого кодирования. Методы, алгоритмы, применение / Р. Морелос-Сарагоса.- М.: Техносфера, 2005.- 319 с.
4. Федоречко О.І. Метод виправлення двократних помилок передачі даних в комп'ютерних системах управління реального часу. // Вісник Національного технічного університету України "КПІ" Інформатика, управління та обчислювальна техніка, – Київ: ВЕК+ – 2012 – № 56. с.155-159.
5. Самофалов К.Г. Обнаружение и исправление ошибок передачи данных с использованием взвешенных контрольных сумм / К.Г. Самофалов, А.П. Марковский, Мулки Яссин Ахмед Ал Бадайнех. // Проблеми інформатизації та управління. Збірник наукових праць.- 2008.- Випуск 3(14).-С.121-128.

УДК 62-50

ПАВЛОВ А.А.,
КАЛАШНИК В.В.

РЕКОМЕНДАЦИИ ПО ВЫБОРУ ЗОНЫ ПРОВЕДЕНИЯ АКТИВНОГО ЭКСПЕРИМЕНТА ДЛЯ ОДНОМЕРНОГО ПОЛИНОМИАЛЬНОГО РЕГРЕССИОННОГО АНАЛИЗА

В статье рассматривается выбор зоны проведения активного эксперимента так, чтобы уменьшить дисперсии оценок коэффициентов регрессионного полинома. Проводится сравнение различных промежутков, и приводятся таблицы с полученными дисперсиями оценок коэффициентов на них. Рассматривается возможность масштабирования уже полученных исходных данных.

The choice of the area of an active experiment execution is considered so that the dispersions of the estimates of regression polynomial's coefficients be decreased. The comparison of different gaps is fulfilled and the tables with obtained dispersions of the coefficients' estimates for them are given. A possibility of zooming already retrieved data is considered.

1. Вступление

Проблема построения нелинейной регрессии по данным с шумом является одной из наиболее востребованных на практике задач. Значительную роль играет соответствие построенной регрессионной модели реальной зависимости. В данной статье приводятся рекомендации, относительно зоны проведения активного эксперимента, над результатами которого будет выполняться регрессионный анализ. Это касается только тех случаев, когда исследователь может управлять проведением эксперимента, а именно он имеет возможность выбирать диапазон аргумента. Рассматривается одномерный полиномиальный регрессионный анализ с использованием нормированных ортогональных полиномов Форсайта, приведенных в главе 6 [1]. Критерием улучшения регрессионной модели является уменьшение дисперсий оценок коэффициентов в регрессионном полиноме. В статье приводится аналитическое доказательство того, что при масштабировании интервала, после проведения эксперимента, не повлияет на качество регрессии.

2. Выбор зоны проведения активного эксперимента

Можно выделить шесть условных зон проведения эксперимента:

1) Начало больше нуля, малый интервал. Например, [10;20].
2) Начало больше нуля, интервал большой. Например, [10;1010].
3) Начало в точке ноль, интервал малый. Например, [0;10].
4) Начало в точке ноль, интервал большой. Например, [0;1000].
5) Симметричен относительно точки ноль, интервал мал. Например, [-5;5].
6) Симметричен относительно точки ноль, интервал большой. Например, [-500;500].

В таблицах 1-3 приведены дисперсии оценок коэффициентов регрессионного полинома, рассчитанные по формулам, приведенным в главе 6 [1].

Табл. 1. Дисперсии оценок коэффициентов регрессионного полинома (10 входных точек)

Интервал	0	1	2	3	4	5
[10;20]	$8{,}4\times10^{6}\sigma^2$	$9{,}5\times10^{5}\sigma^2$	$1{,}7\times10^{4}\sigma^2$	$7{,}3\times10\sigma^2$	$7{,}7\times10^{-2}\sigma^2$	$1{,}3\times10^{-5}\sigma^2$
[10;1010]	$5\times10\sigma^2$	$1\times10^{-2}\sigma^2$	$2{,}4\times10^{-7}\sigma^2$	$1{,}1\times10^{-12}\sigma^2$	$1\times10^{-18}\sigma^2$	$1{,}3\times10^{-25}\sigma^2$
[0;10]	$3{,}7\times10\sigma^2$	$8{,}5\times10\sigma^2$	$2{,}1\times10\sigma^2$	$1\sigma^2$	$9{,}8\times10^{-3}\sigma^2$	$1{,}3\times10^{-5}\sigma^2$
[0;1000]	$3{,}7\times10\sigma^2$	$8{,}5\times10^{-3}\sigma^2$	$2{,}1\times10^{-7}\sigma^2$	$1\times10^{-12}\sigma^2$	$9{,}8\times10^{-19}\sigma^2$	$1{,}3\times10^{-25}\sigma^2$
[-5;5]	$4\times10^{-1}\sigma^2$	$2{,}5\times10^{-1}\sigma^2$	$4{,}3\times10^{-2}\sigma^2$	$7{,}5\times10^{-3}\sigma^2$	$1{,}4\times10^{-4}\sigma^2$	$1{,}3\times10^{-5}\sigma^2$
[-500;500]	$4\times10^{-1}\sigma^2$	$2{,}5\times10^{-5}\sigma^2$	$4{,}3\times10^{-10}\sigma^2$	$7{,}5\times10^{-15}\sigma^2$	$1{,}4\times10^{-20}\sigma^2$	$1{,}3\times10^{-25}\sigma^2$

Табл. 2. Дисперсии оценок коэффициентов регрессионного полинома (100 входных точек)

Интервал	0	1	2	3	4	5
[10;20]	$3{,}3\times10^{5}\sigma^2$	$4\times10^{4}\sigma^2$	$7{,}7\times10^{2}\sigma^2$	$3{,}5\sigma^2$	$4\times10^{-3}\sigma^2$	$7\times10^{-7}\sigma^2$
[10;1010]	$6{,}1\times10^{-1}\sigma^2$	$2{,}1\times10^{-4}\sigma^2$	$7{,}1\times10^{-9}\sigma^2$	$4{,}2\times10^{-14}\sigma^2$	$4{,}7\times10^{-20}\sigma^2$	$7\times10^{-27}\sigma^2$
[0;10]	$4{,}3\times10^{-1}\sigma^2$	$1{,}7\sigma^2$	$6{,}1\times10^{-1}\sigma^2$	$3{,}8\times10^{-2}\sigma^2$	$4{,}5\times10^{-4}\sigma^2$	$7\times10^{-7}\sigma^2$
[0;1000]	$4{,}3\times10^{-1}\sigma^2$	$1{,}7\times10^{-4}\sigma^2$	$6{,}1\times10^{-9}\sigma^2$	$3{,}8\times10^{-14}\sigma^2$	$4{,}5\times10^{-20}\sigma^2$	$7\times10^{-27}\sigma^2$
[-5;5]	$3{,}5\times10^{-2}\sigma^2$	$2{,}3\times10^{-2}\sigma^2$	$2{,}2\times10^{-3}\sigma^2$	$5{,}7\times10^{-4}\sigma^2$	$4{,}5\times10^{-6}\sigma^2$	$7\times10^{-7}\sigma^2$
[-500;500]	$3{,}5\times10^{-2}\sigma^2$	$2{,}3\times10^{-6}\sigma^2$	$2{,}2\times10^{-11}\sigma^2$	$5{,}7\times10^{-16}\sigma^2$	$4{,}5\times10^{-22}\sigma^2$	$7\times10^{-27}\sigma^2$

Табл. 3. Дисперсии оценок коэффициентов регрессионного полинома (1000 входных точек)

Интервал	0	1	2	3	4	5
[10;20]	$3{,}2\times10^{4}\sigma^2$	$3{,}9\times10^{4}\sigma^2$	$7{,}5\times10\sigma^2$	$3{,}5\times10^{-1}\sigma^2$	$3{,}9\times10^{-4}\sigma^2$	$7\times10^{-8}\sigma^2$
[10;1010]	$5{,}2\times10^{-2}\sigma^2$	$1{,}9\times10^{-5}\sigma^2$	$6{,}6\times10^{-10}\sigma^2$	$4\times10^{-15}\sigma^2$	$4{,}6\times10^{-21}\sigma^2$	$7\times10^{-28}\sigma^2$
[0;10]	$3{,}7\times10^{-2}\sigma^2$	$1{,}5\times10^{-1}\sigma^2$	$5{,}7\times10^{-2}\sigma^2$	$3{,}6\times10^{-3}\sigma^2$	$4{,}4\times10^{-5}\sigma^2$	$7\times10^{-8}\sigma^2$
[0;1000]	$3{,}7\times10^{-2}\sigma^2$	$1{,}5\times10^{-5}\sigma^2$	$5{,}7\times10^{-10}\sigma^2$	$3{,}6\times10^{-15}\sigma^2$	$4{,}4\times10^{-21}\sigma^2$	$7\times10^{-28}\sigma^2$
[-5;5]	$3{,}5\times10^{-3}\sigma^2$	$2{,}3\times10^{-3}\sigma^2$	$2{,}2\times10^{-4}\sigma^2$	$5{,}7\times10^{-5}\sigma^2$	$4{,}4\times10^{-7}\sigma^2$	$7\times10^{-8}\sigma^2$
[-500;500]	$3{,}5\times10^{-3}\sigma^2$	$2{,}3\times10^{-7}\sigma^2$	$2{,}2\times10^{-12}\sigma^2$	$5{,}7\times10^{-17}\sigma^2$	$4{,}4\times10^{-23}\sigma^2$	$7\times10^{-28}\sigma^2$

Как видно из таблиц 1-3, чем шире интервал, тем меньше дисперсии оценок коэффициентов. Также, дисперсии уменьшаются, когда интервал эксперимента симметричный точке ноль. Если невозможно принимать отрицательные значения, то рекомендуется начинать интервал в точке ноль. Интервал довольно сильно влияет на дисперсии оценок коэффициентов, даже больше чем изменение количества входных точек. Дисперсии оценок коэффициентов могут быть пересчитаны при масштабировании по формуле (32):

$$D\hat{\theta}_j^z = \sigma^2 \sum_{l=r}^{j} \left(\frac{1}{k^j} q_{lj}^x\right)^2 = \left(\frac{1}{k^j}\right)^2 D\hat{\theta}_j^x$$

С учетом того, что на практике количество входных точек и необходимая степень регрессионного полинома могут быть достаточно большими числами, это приводит к выполнению значительного количества расчетов. Перерасчет коэффициентов является достаточно простым процессом и не требует значительных компьютерных мощностей. Рассмотрим перерасчет дисперсий на примере (табл. 4).

Табл. 4. Пример перерасчета дисперсий оценок коэффициентов (1000 входных точек)

	[-5;5]	[-500;500] $k=100$
0	$3{,}5\times10^{-3}\sigma^2$	$\left(\frac{1}{100^0}\right)^2\times3{,}5\times10^{-3}\sigma^2 = 3{,}5\times10^{-3}\sigma^2$
1	$2{,}3\times10^{-3}\sigma^2$	$\left(\frac{1}{100^1}\right)^2\times2{,}3\times10^{-3}\sigma^2 = 2{,}3\times10^{-7}\sigma^2$
2	$2{,}2\times10^{-4}\sigma^2$	$\left(\frac{1}{100^2}\right)^2\times2{,}2\times10^{-4}\sigma^2 = 2{,}2\times10^{-12}\sigma^2$
3	$5{,}7\times10^{-5}\sigma^2$	$\left(\frac{1}{100^3}\right)^2\times5{,}7\times10^{-5}\sigma^2 = 5{,}7\times10^{-17}\sigma^2$
4	$4{,}4\times10^{-7}\sigma^2$	$\left(\frac{1}{100^4}\right)^2\times4{,}4\times10^{-7}\sigma^2 = 4{,}4\times10^{-23}\sigma^2$
5	$7\times10^{-8}\sigma^2$	$\left(\frac{1}{100^5}\right)^2\times7\times10^{-8}\sigma^2 = 7\times10^{-28}\sigma^2$

Таким образом, можно заранее увидеть эффективность регрессионного анализа, и подобрать необходимые исследователю диапазоны. Аналогично, по формуле (16) можно пересчитать значения коэффициентов нормированных ортогональных полиномов Форсайта: что требует значительно меньше вычислений, чем расчет этих значений заново по формулам (3)-(8).

$$q_{jl}^{z} = \frac{1}{k^{l}} q_{jl}^{x}, \forall j = \overline{0,r}, \forall l = \overline{0,j}$$

Такой пересчет требует значительно меньше вычислений, чем расчет этих значений заново по формулам (3)-(8).

Однако, проведение масштабирования уже полученных входных данных не приведет к улучшению результата, дальше это будет доказано.

3. Масштабирование входных данных

Как показали эксперименты, при масштабировании уже имеющихся входных данных, качество регрессионного полинома не изменится. То есть, несмотря на то, что изменятся оценки коэффициентов и их дисперсии, значение регрессионного полинома при заданном аргументе останется таким же. Для доказательства этого используем формулы из главы 6 [1]. Оценки коэффициентов регрессионного полинома получаются с помощью нормированных ортогональных полиномов Форсайта.

$$\hat{\theta}_j = \hat{w}_r q_{rj} + \cdots + \hat{w}_j q_{jj}, j = \overline{0,r} \qquad (1)$$

В формуле (1) $\hat{\theta}_j$ – оценки коэффициентов регрессионного полинома, $\hat{w}_j$ – оценки весовых коэффициентов, полученные методом наименьших квадратов, q_{jj} – коэффициенты нормированных ортогональных полиномов Форсайта. Расчет оценок весовых коэффициентов происходит по формуле (2).

$$\hat{w}_j = \sum_{i=1}^{n} y_i \theta_j(x_i), j = \overline{0,r}, \qquad (2)$$

где $\theta_j(x_i)$ – значение $j^{ого}$ нормированного ортогонального полинома Форсайта на аргументе x_i.

Коэффициенты нормированных ортогональных полиномов Форсайта рассчитываются по рекуррентной формуле (3) из двух предыдущих.

$$\lambda\theta_j(x) = x\theta_{j-1}(x) - \alpha\theta_{j-1}(x) - \beta\theta_{j-2}(x), \qquad (3)$$

где α, β, λ – коэффициенты, которые рассчитываются по формулам (4), (5) и (6) соответственно.

$$\alpha = \sum_{i=1}^{n} x_i \theta_{j-1}^2(x_i) \qquad (4)$$

$$\beta = \sum_{i=1}^{n} x_i \theta_{j-1}(x_i)\theta_{j-2}(x_i) \qquad (5)$$

$$\lambda = \sqrt{\sum_{i=1}^{n} (x_i\theta_{j-1}(x_i) - \alpha\theta_{j-1}(x_i) - \beta\theta_{j-2}(x_i))^2} \qquad (6)$$

Первых два ортогональных полинома рассчитываются по формулам (7) и (8) соответственно.

$$\theta_0(x) = \frac{1}{\sqrt{n}} \qquad (7)$$

$$\theta_1(x) = -\frac{\bar{x}}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}} + \frac{x}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}} \qquad (8)$$

Для доказательства того, что при масштабировании качество входных данных регрессионного полинома не изменится, будем рассматривать коэффициенты нормированных ортогональных полиномов Форсайта отдельно. Тогда из (3), (7) и (8) получаем выражения (9) - (15).

$$q_{00} = \frac{1}{\sqrt{n}} \qquad (9)$$

$$q_{10} = -\frac{\bar{x}}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}} \qquad (10)$$

$$q_{11} = \frac{1}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}} \qquad (11)$$

$$q_{j0} = \frac{\alpha q_{j-1,0} - \beta q_{j-2,0}}{\lambda} \qquad (12)$$

$$q_{jl} = \frac{q_{j-1,l-1} - \alpha q_{j-1,l} - \beta q_{j-2,l}}{\lambda}, l = \overline{1, j-2} \qquad (13)$$

$$q_{j,j-1} = \frac{q_{j-1,j-2} - \alpha q_{j-1,j-1}}{\lambda} \qquad (14)$$

$$q_{j,j} = \frac{q_{j-1,j-1}}{\lambda} \qquad (15)$$

Формулы (12) - (15) выполняются $\forall j = \overline{2,r}$.

Предположим, что было выполнено масштабирование исходных данных следующим образом $z = kx$, где k – некоторый коэффициент масштабирования, который является действительным числом. Обозначим коэффициенты при начальном масштабировании с помощью

q^x, а при новом – q^z. Покажем, что связь коэффициентов нормированных ортогональных полиномов при разном масштабировании соответствует формуле (16).

$$q_{jl}^z = \frac{1}{k^l} q_{jl}^x, \forall j = \overline{0,r}, \forall l = \overline{0,j} \qquad (16)$$

Рассмотрим связь между средним значением аргументов начального и нового масштабирования, поскольку оно используется в формулах (10) и (11).

$$\bar{z} = \frac{1}{n}\sum_{i=1}^{n} z_i = \frac{1}{n}\sum_{i=1}^{n} kx_i = k\bar{x} \qquad (17)$$

Рассмотрим коэффициенты первых двух нормированных ортогональных полиномов Форсайта

$$q_{00}^z = \frac{1}{\sqrt{n}} = q_{00}^x \qquad (18)$$

$$q_{10}^z = \frac{k\bar{x}}{\sqrt{\sum_{i=1}^{n}(kx_i - k\bar{x})^2}} = q_{10}^x \qquad (19)$$

$$q_{11}^z = \frac{1}{\sqrt{\sum_{i=1}^{n}(kx_i - k\bar{x})^2}} = \frac{1}{k} q_{11}^x \qquad (20)$$

Следовательно, для коэффициентов первых двух ортогональных полиномов выполняется формула (16). Рассмотрим значения ортогонального полинома при заданном аргументе, для которых выполняется формула (16).

$$\theta_j^z(z_i) = q_{j0}^x + \cdots + \frac{1}{k^j} q_{jj}^x k^j x_i = \theta_j^x(x_i) \qquad (21)$$

Рассмотрим связь коэффициентов α, β, λ при различных масштабированиях.

$$\alpha^z = \sum_{i=1}^{n} kx_i \theta_{j-1}^2(x_i) = k\alpha^x \qquad (22)$$

$$\beta^z = \sum_{i=1}^{n} kx_i \theta_{j-1}(x_i)\theta_{j-2}(x_i) = k\beta^x \qquad (23)$$

$$\lambda^z = \sqrt{\sum_{i=1}^{n}(kx_i\theta_{j-1}^x(x_i) - k\alpha\theta_{j-1}^x(x_i) - k\beta\theta_{j-2}^x(x_i))^2} = k\lambda^x \qquad (24)$$

Рассмотрим связь между коэффициентами полиномов $j = \overline{2,r}$ при различных масштабированиях.

$$q_{j0}^z = \frac{k\alpha^x q_{j-1,0}^x - k\beta^x q_{j-2,0}^x}{k\lambda^x} = q_{j0}^x \qquad (25)$$

$$q_{jl}^z = \frac{\frac{1}{k^{l-1}} q_{j-1,l-1}^x - k\alpha^x \frac{1}{k^l} q_{j-1,l}^x - k\beta^x \frac{1}{k^l} q_{j-2,l}^x}{k\lambda^x} = \frac{1}{k^l} q_{jl}^x, l = \overline{1, j-2} \qquad (26)$$

$$q_{j,j-1}^z = \frac{\frac{1}{k^{j-2}} q_{j-1,j-2}^x - k\alpha^x \frac{1}{k^{j-1}} q_{j-1,j-1}^x}{k\lambda^x} = \frac{1}{k^{j-1}} q_{j,j-1}^x \qquad (27)$$

$$q_{j,j}^z = \frac{\frac{1}{k^{j-1}} q_{j-1,j-1}^x}{k\lambda^x} = \frac{1}{k^j} q_{j,j}^x \qquad (28)$$

Итак, для ортогональных полиномов Форсайта формула (16) справедлива, при масштабировании $z = kx$.

Рассмотрим связь оценок весовых коэффициентов при различных масштабах.

$$\hat{w}_j^z = \sum_{i=1}^{n} y_i \theta_j^x(x_i) = \hat{w}_j^x \qquad (29)$$

Исходя из формул (16) и (29), связь между оценками коэффициентов регрессионного полинома при различных масштабированиях будет следующая.

$$\hat{\theta}_j^z = \hat{w}_r \frac{1}{k^j} q_{rj}^x + \cdots + \hat{w}_j \frac{1}{k^j} q_{jj}^x = \frac{1}{k^j}\hat{\theta}_j^x \qquad (30)$$

Исходя из (30), связь между значениями регрессионного полинома при заданном аргументе и различных масштабированиях будет следующая.

$$f^z(z_i) = \sum_{j=0}^{r} \frac{1}{k^j}\hat{\theta}_j^x k^j x_i = f^x(x_i) \qquad (31)$$

Следовательно, значение регрессионного полинома, при различных масштабированиях имеющихся входных данных останется одинаковым, хотя вид самого регрессионного полинома изменяется согласно формуле (30).

Рассмотрим связь между дисперсиями оценок коэффициентов регрессионного полинома при различных масштабированиях.

$$D\hat{\theta}_j^z = \sigma^2 \sum_{l=r}^{j} (\frac{1}{k^j} q_{lj}^x)^2 = (\frac{1}{k^j})^2 D\hat{\theta}_j^x \qquad (32)$$

Следовательно, дисперсии оценок коэффициентов регрессионного полинома изменяются по формуле (32).

4. Вывод

При проведении активного эксперимента с получением входных данных регрессии, дисперсии оценок коэффициентов регрессионного полинома будут зависеть от интервала, на котором он проводится. Чем ближе центр интервала до точки ноль, тем меньшими будут дисперсии. Однако значительного снижения дисперсий можно достичь путем расширения интервала. Однако, при масштабировании уже имеющихся входных данных, качество регрессионного полинома не изменится, хотя изменится его коэффициенты и их дисперсии.

Список литературы

1. М.З. Згуровский, А.А. Павлов Принятие решений в сетевых системах с ограниченными ресурсами. – К.: Наукова думка, – 2010. – 569 с.

УДК 004.052.042

МАРКОВСЬКИЙ О.П.,
ІВАНОВ Д.Г.,
ВЕЛИКИЙ М.М.,
НЕВДАЩЕНКО М.В.

МЕТОД РЕЗЕРВУВАННЯ ТА ПРИСКОРЕНОГО ВІДНОВЛЕННЯ ДАНИХ В СИСТЕМАХ ЇХ ВІДДАЛЕНОГО ЗБЕРІГАННЯ

В статті запропоновано метод резервування і відновлення даних користувачів, що зберігаються на віддалених вузлах. Розроблений метод дозволяє відновлювати дані в ситуаціях, що найчастіше зустрічаються на практиці: при втраті доступу до трьох довільних носіям. Детально описані математична ідея метода та процедура відновлення фрагментів даних з трьох носіїв, доступ до яких втрачено. Розроблена процедура відновлення илюструється прикладами. Наведені теоретичні та експериментальні оцінки ефективності запропонованого методу.

In paper the method of reservation and recovering of user data stored on remote memory units is proposed. Developed method makes it possible to recover data for frequent occurrence in practice situation: in case of access loosing for any three storage devices. The mathematical idea of proposed method and procedure for recovering of data fragments from three access lost storage devices are described in details. A numerical example for developed recovering procedure are given. The theoretical and experimental estimations of proposed method effectiveness are given.

Вступ

Поступальний розвиток технології комп'ютерної обробки інформації нерозривно пов'язаний з проблемою забезпечення високого рівня надійності всіх компонентів обчислювальних систем, в тому числі засобів зберігання даних на носіях. Високі темпи зростання можливостей систем передачі даних та прогрес мережевих технологій дозволили в останнє десятиліття вийти на якісно новий рівень розвитку комп'ютерних систем, визначальною особливістю якого стала можливість віддаленого надання обчислювальних ресурсів широкому колу користувачів. Розподілені системи зберігання та обробки інформації дозволяють гнучко концентрувати значні за обсягом обчислювальні ресурси на вирішення прикладних задач, значно розширюючи можливості ефективного використання комп'ютерних технологій в усіх сферах людської діяльності.

В останнє десятиліття широкого розповсюдження набули розподілені системи віддаленого зберігання інформації користувачів. Такі системи дозволяють ефективно використовувати об'єм носіїв, сконцентрованих в рамках вузлів зберігання інформації і забезпечують якісно новий рівень надійності зберігання даних. Динамічне розширення використання прогресивних технологій віддаленого надання широкому колу користувачів ресурсів потужних комп'ютерних систем вимагає вирішення низки проблем, чільне місце серед яких займає задача забезпечення надійності доступу до даних в розподілених системах їх віддаленого зберігання.

Ключовими чинниками ефективності розподілених систем віддаленого зберігання інформації користувачів є забезпечення високої надійності та неперервності доступу до даних. Втрата доступу до даних може бути спричинена як втратою, з тих чи інших причин, даних на носіях, так і тимчасовою неможливістю доступу до вузла зберігання даних, зумовленою, здебільшого, перевантаженням вузла або мережі. З ростом кількості користувачів систем віддаленого зберігання їх інформації проблема забезпечення неперервності віддаленого доступу до даних загострюється.

Для того, щоб в повній мірі реалізувати потенціал прогресивних технологій розподіленого віддаленого зберігання інформації, потрібні нові методи резервування та забезпечення доступності даних в таких системах.

Таким чином, проблема підвищення ефективності систем резервування та відновлення даних користувачів при їх віддаленому зберіганні

є актуальною для сучасного етапу розвитку інформаційних технологій.

Огляд та аналіз існуючих засобів резервування та відновлення даних

Висока надійність зберігання даних на віддалених від користувача вузлах зумовлена можливістю ефективного застосування в них технологій резервування та відновлення даних. Проблема відновлення даних, що досліджується в роботі, фактично лежить в рамках загальної тенденції забезпечення живучості комп'ютерних систем і їх компонентів на мікро і макро рівнях в умовах динамічного зростання їх складності

До теперішнього часу створено низку методів резервування та відновлення даних при їх віддаленому зберіганні [1-6].

В якості критеріїв ефективності систем резервування найчастіше виступають:

- обчислювальна складність процедури відновлення даних з використанням резервних носіїв;
- кількість носіїв (КВ), дані з яких можуть бути відновлені в разі втрати доступу до них;
- відношення кількості резервних носіїв (КР) до числа носіїв, дані з яких можуть бути відновлені (=КР/КВ).

Найбільшого розповсюдження на практиці здобули технології резервування та відновлення, створені в рамках системи RAID [1]. Якщо перші системи RAID-1 резервування створені на основі простого дублювання даних, то більш розвинуті версії, включаючи RAID-10, передбачають використання корегуючих та erasure-кодів [2]. На відміну від корегуючих кодів, характерною особливістю останніх є те, що локалізація втрачених даних відома. Оскільки ця умова виконується для більшості систем віддаленого зберігання даних на дискових накопичувача, то саме erasure-коди вважаються [3] найбільш ефективними в рамках зазначених вище критеріїв для відновлення даних.

Подібно корегуючим кодам, можна виділити лінійні та циклічні erasure-коди [4]. Недоліком перших вважається високе значення коефіцієнту надмірності резервування при відновленні даних з більшого ніж один числа носіїв. Фактично, залежність між кількістю резервних носіїв та кількістю тих, дані з яких можуть буди відновлені носить нелінійних характер. Процес відновлення зводиться до розв'язання систем лінійних рівнянь, що визначає складність процесу відновлення.

Циклічні erasure-коди, що, в своїй більшості, являють собою модифікації кодів Ріда-Соломона, побудовані на основі алгебри полів Галуа і забезпечують менший, в порівнянні з лінійними кодами. рівень надлишковості, проте мають високу часові складність процесу відновлення.

При віддаленому зберіганні даних користувача вони розподіляються по окремим вузлам зберігання інформації. В рамках окремого вузла організовано розподілення даних по носіях, доступ користувачів до даних, їх захист, резервування в разі втрати доступу до одного або декількох носіїв.

Загальною рисою більшості з відомих технологій відновлення даних з носіїв, до яких втрачено доступ є те, що вони реалізовані в рамках окремого вузла зберігання інформації. Це означає, що в разі втрати доступу до вузла в результаті тимчасового виходу його з ладу, перевантаження, вірусної атаки, відключення від мережі, техногенних або природних катаклізмів, відомі механізми відновлення даних або доступу до них для конкретного користувача не спрацьовують [4].

Вказаний недолік виключається при застосуванні методів резервування, які передбачають розділене зберігання основної та резервної інформації. Зокрема, в роботі [5] запропоновано метод резервування та відновлення даних, відмінністю якого є формування відновлюючого коду у вигляді сум за модулем 2 даних користувача з однойменних носіїв всіх вузлів, на яких зберігається його інформація, а також суми за модулем 2 попарних поліноміальних добутків даних користувача з усіх можливих пар однойменних носіїв всіх вузлів, за рахунок чого досягається можливість відновлення даних користувача при постійній або тимчасовій втраті доступу до окремого вузла зберігання інформації, або при втраті доступу до двох будь-яких носіїв різних вузлів, на яких зберігаються дані користувача з використанням теоретично мінімальної кількості додаткових носіїв.

Проте згаданий метод принципово обмежує кількість носіїв, дані з яких можуть бути відновлені двома.

Окремим різновидом лінійних erasure-кодів можна розглядати діагональні коди [6]. Ці коди передбачають представлення інформації користувача у вигляді матриці, рядки якої співвідносяться з носіями, і формування резервних кодів у вигляді логічних сум стовпців, а також розташованих під різними кутами висхідних та

низхідних діагоналей матриці, що забезпечує можливість рекурсивного відновлення даних без розв'язання систем рівнянь за рахунок чого зменшується обчислювальна складність процедури відновлення даних та досягається теоретичний мінімум надлишковості резервування. Фактично, діагональні коди за рахунок спеціальної системи обчислення резервних кодів забезпечують зведення задачі відновлення даних з декількох носіїв до рекурсивного виконання відновлення даних з одного носія. Саме це забезпечує теоретичний мінімум надлишковості резервування і простоту обчислювальної реалізації процесу відновлення даних.

З іншого боку, рекурсивний характер відновлення даних зумовлює основний недолік методу діагональних кодів - для відновлення окремого фрагменту даних потрібно відновити весь попередній об'єм інформації, що передує йому. Це зумовлює необхідність для відновлення фрагменту даних отримання через Інтернет і обробки значної за обсягом інформації з усіх носіїв, на яких зберігаються дані користувача. Відповідно, процес відновлення навіть невеликого за обсягом фрагменту даних потребує значних затрат часу.

Для усунення вказаного недоліку ефективного методу відновлення даних з обмеженої кількості віддалених носіїв потрібно розробити спеціальні методи.

Таким чином, відомі методи резервування даних при їх віддаленому зберіганні мають обмежену ефективність, зумовлену тим, що вони не забезпечують високої швидкості відновлення доступу до фрагментів даних в типових для практики ситуаціях, коли об'єм фрагменту значно менший за інформаційну ємкість носія.

Ціль досліджень полягає в підвищенні оперативності відновлення доступу до даних, що зберігаються на віддалений носіях за рахунок зменшення об'єму інформації та складності обчислень, що потребуються для відновлення окремого фрагменту даних.

Метод відновлення даних з носіїв *вузла, до якого втрачено доступ*

Інформація користувача зберігається на m віддалених носіях. Інформація на кожному з них може розглядатися як така, що складається з n фрагментів. Поняття фрагменту не прив'язано до певної фізичної природи і фрагментом може бути сектор жорсткого диску або блок інформації довжина якого співпадає з розрядністю процесора. Відповідно кожен j-ий фрагмент, $j \in \{1,..,n\}$, i-го віддаленого носія може бути позначений як $s_{i,j}$. Тоді інформація користувача, що зберігається на віддалених носіях може бути представлена у вигляді матриці S:

$$S = \begin{pmatrix} s_{1,1} & s_{1,2} & \dots & s_{1,n} \\ s_{2,1} & s_{2,2} & \dots & s_{2,n} \\ \dots & \dots & \dots & \dots \\ s_{m,1} & s_{m,2} & \dots & s_{m,n} \end{pmatrix}$$

В такій матриці кожен стовбець містить однойменні фрагменті всіх носіїв, а рядок – фрагменти одного носія.

Для відновлення інформації з трьох довільних віддалених носіїв до яких втрачено доступ пропонується спеціальний метод. Метод потребує використання чотирьох резервних носіїв.

Призначення методу є підвищення швидкості відновлення конкретних фрагментів віддалених носіїв, до яких був втрачений доступ.

Метод передбачає, що всі однойменні фрагменти кожного віддаленого носія розділяються на блоки розміром p, $p \in \{m-2, .., n\}$. Відповідно кожний h-ий блок, $h \in \{1,..,n/p\}$, може бути позначений як B_h. Ці блоки незалежні і можуть відновлюватись окремо, і навіть паралельно. Тоді інформація в h-му блоку може бути представлена у вигляді матриці B_h:

$$B_h = \begin{pmatrix} s_{1,1+(h-1)\cdot p} & s_{1,2+(h-1)\cdot p} & \dots & s_{1,h\cdot p} \\ s_{2,1+(h-1)\cdot p} & s_{2,2+(h-1)\cdot p} & \dots & s_{2,h\cdot p} \\ \dots & \dots & \dots & \dots \\ s_{m,1+(h-1)\cdot p} & s_{m,2+(h-1)\cdot p} & \dots & s_{m,h\cdot p} \end{pmatrix}$$

Запропонований метод дозволяє відновлювати будь-який блок даних носіїв до яких втрачено доступ незалежно від інших. Тому, для спрощення викладу методу, не відступаючи від принципу узагальнення, можна розглядати метод відносно задачі відновлення першого блоку всіх носіїв, кількістю не більше 3-х, до яких втрачено доступ:

$$B_1 = \begin{pmatrix} s_{1,1} & s_{1,2} & \dots & s_{1,p} \\ s_{2,1} & s_{2,2} & \dots & s_{2,p} \\ \dots & \dots & \dots & \dots \\ s_{m,1} & s_{m,2} & \dots & s_{m,p} \end{pmatrix}$$

Для відновлення даних використовуються чотири резервних носія, дані на яких пропонується формувати наступним чином.

Фрагменти першого резервного носія, позначені як $d_{1,1}$, $d_{1,2}$, .., $d_{1,p}$, пропонується формувати з фрагментів, що належать висхідним діагоналям матриці, а формально як суму за модулем два фрагментів, сума індексів яких перевищує на 1 порядковий номер фрагмента першого носія:

$$\forall j \in \{1,..,m\}: d_{1,j} = \bigoplus_{q=1}^{j} s_{j-q+1,q} \quad (1)$$

$$\forall j \in \{m+1,..,n\}: d_{1,j} = \bigoplus_{q=1}^{m} s_{m-q+1,q+j-m} \quad (2)$$

Фрагменти другого резервного носія, позначені як $d_{2,1}$, $d_{2,2}$, .., $d_{2,p}$, пропонується формувати з фрагментів, що належать низхідним діагоналям матриці, а формально як суму за модулем два фрагментів, модуль різниці індексів яких дорівнює модулю різниці кількості віддалених носіїв і порядкового номера фрагмента другого носія:

$$\forall j \in \{1,..,m\}: d_{2,j} = \bigoplus_{q=1}^{j} s_{m-j+q,q} \quad (3)$$

$$\forall j \in \{m+1,..,n\}: d_{2,j} = \bigoplus_{q=1}^{m} s_{q,j-m+q} \quad (4)$$

Фрагменти третього резервного носія, позначені як $d_{3,1}$, $d_{3,2}$, .., $d_{3,p}$, пропонується формувати з фрагментів, що належать стовбцям матриці, а формально як суму за модулем два однойменних фрагментів всіх інформаційних носіїв:

$$\forall j \in \{1,..,n\}: d_{3,j} = \bigoplus_{q=1}^{m} s_{q,j} \quad (5)$$

Фрагменти четвертого резервного носія, позначені як $d_{4,1}$, $d_{4,2}$, .., $d_{4,m-2}$, пропонується формувати з фрагментів, що належать ламаним діагоналям з p елементів матриці, а формально як суму за модулем два фрагментів, модуль різниці індексів яких дорівнює модулю різниці кількості віддалених носіїв і суми порядкового номера фрагмента четвертого носія та розміру блоку або модулю різниці порядкового номера фрагмента четвертого носія і суми кількості віддалених носіїв та розміру блоку:

$$\forall j \in \{1,..,m-2\}: d_{4,j} = \bigoplus_{q=1}^{m-j} s_{q,j+q+p-m} \oplus \bigoplus_{q=1}^{j} s_{m-q+1,p-j+q-1} \quad (6)$$

При формуванні фрагментів четверного резервного носія передбачена виключна ситуація за умови $p=m-2$. За цієї умови перший, позначений як $d_{4,1}$ і останній, позначений як $d_{4,m-2}$, фрагменти утворюються за наступними формулами.

Перший фрагмент четвертого резервного носія, пропонується формувати з фрагментів, що належать ламаній діагоналі з m-1 елементів матриці, а формально як суму за модулем два фрагментів, модуль різниці індексів яких дорівнює модулю різниці розміру блоку збільшеного на одиницю і кількості віддалених носіїв або сума індексів яких дорівнює сумі кількості віддалених носіїв і розміру блоку зменшеного на одиницю:

$$j = 1: d_{4,1} = \bigoplus_{q=1}^{m-2} s_{q+1,q+p-m+2} \oplus s_{m,p-1} \quad (7)$$

Останній фрагмент четвертого резервного носія пропонується формувати з фрагментів, що належать ламаній діагоналі з m-1 елементів матриці, а формально як суму за модулем два фрагментів, модуль різниці індексів яких дорівнює модулю різниці 2 і розміру блоку або сума індексів яких дорівнює розміру блоку збільшеного на двійку:

$$j = m-2: d_{4,m-2} = s_{1,p-1} \oplus \bigoplus_{q=1}^{m-2} s_{m-q,q+p-m+2} \quad (8)$$

Запропонований порядок формування фрагментів резервних носіїв може бути ілюстровано прикладом, який зображено на рис. 1 та рис. 2. В рамках цього прикладу вважається, що загальна кількість віддалених носіїв дорівнює семи (m=7), а розмір блоку – п'яти (p=5=m-2).

Запропонований метод передбачає послідовне, починаючи з першого, відновлення для заданого блоку фрагментів з носіїв, до яких втрачено доступ. Для визначеності, номери носіїв, до яких втрачено доступ, позначено через a, b і c, причому $a<b<c$. В процесі відновлення однойменних j-их фрагментів першим реконструюється фрагмент $s_{a,j}$. Фрагмент $s_{a,j}$, в залежності від значення a і порядкового номеру j, належить висхідній діагоналі, сума якої зберігається в першому резервному носієві, або ламаній діагоналі, сума якої зберігається в четвертому резервному носієві. І в першому, і в другому випадку фрагмент $s_{a,j}$ може бути реконструйований, оскільки фрагменти, що належать діагоналі і лежать на ній вище $s_{a,j}$ зберігаються на непошкоджених носіях з номерами меншими a. Фрагменти, що лежать на діагоналі нижче $s_{a,j}$ мають номери менші за j, тобто відновлені раніше або належать доступним носіям.

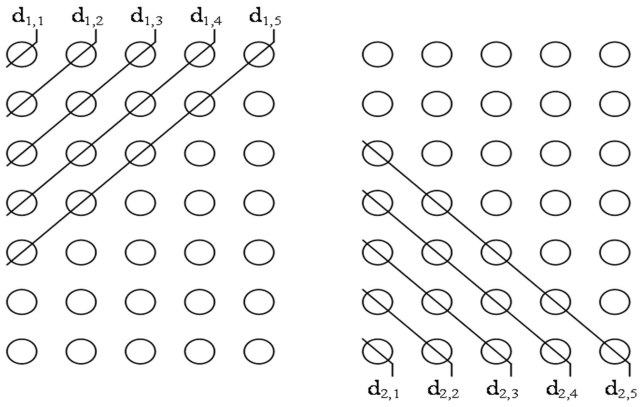


Рисунок 1. Формування фрагментів першого і другого резервних носіїв

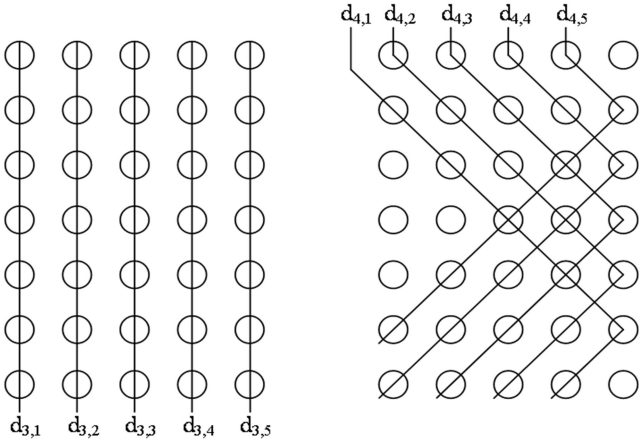


Рисунок 2. Формування фрагментів третього і четвертого резервних носіїв

На наступному кроці реконструюється фрагмент $s_{c,j}$. Фрагмент $s_{c,j}$, в залежності від значення c і його порядкового номеру j, належить низхідній діагоналі, сума якої зберігається в другому резервному носієві, або ламаній діагоналі, сума якої зберігається в четвертому резервному носієві. В обох випадках фрагмент може бути реконструйований, оскільки фрагменти, що знаходяться на діагоналях нижче $s_{c,j}$ не пошкоджено, а фрагменти, що знаходяться вище $s_{c,j}$ на вказаних діагоналях – відомі або вже відновленні, оскільки мають номери менші за j.

На останньому кроці відновлення реконструюється фрагмент $s_{b,j}$. Фрагмент $s_{b,j}$ завжди належить вертикальній прямій, сума якої зберігається в третьому резервному носієві. Фрагмент може бути реконструйований так, як $s_{a,j}$ та $s_{c,j}$ вже відновлені, а інші фрагменти відомі.

Процес реконструкції ілюструється прикладом, в рамках якого вважається, що загальна кількість віддалених носіїв дорівнює п'яти, відповідно, блоки складаються з трьох однойменних фрагментів кожного носія. Прийнято, що було втрачено доступ до першого, третього і четвертого віддаленого носія (a=1, b=3 і c=4).

Зобразимо схематично фрагменти певного блоку, де колом позначені відомі фрагменті, кругом – невідомі, а лініями – фрагменти за допомогою яких формуються фрагменти резервного носія.

На рис. 3 зображено послідовне відновлення $s_{1,1}$, $s_{4,1}$, $s_{3,1}$. Для відновлення $s_{1,1}$ використовується фрагмент резервного носія $d_{1,1}$, для $s_{4,1}$ – $d_{2,2}$, а для $s_{3,1}$ – $d_{3,1}$.

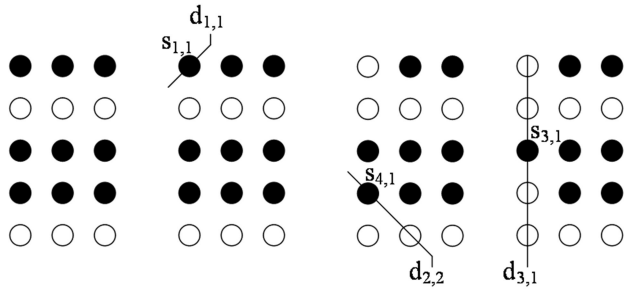


Рисунок 3. Відновлення перших фрагментів кожного носія

На рис. 4 зображено послідовне відновлення $s_{1,2}$, $s_{4,2}$, $s_{3,2}$. Для відновлення $s_{1,2}$ використовується фрагмент резервного носія $d_{1,2}$, для $s_{4,2}$ – $d_{2,3}$, а для $s_{3,2}$ – $d_{3,2}$.

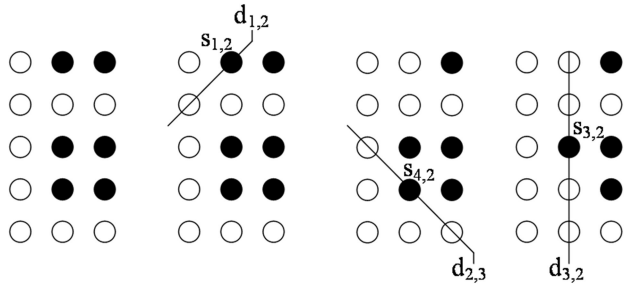


Рисунок 4. Відновлення других фрагментів кожного носія

На рис. 5 зображено послідовне відновлення $s_{1,3}$, $s_{4,3}$, $s_{3,3}$. Для відновлення $s_{1,3}$ використовується фрагмент резервного носія $d_{1,3}$, для $s_{4,3}$ – $d_{4,1}$, а для $s_{3,3}$ – $d_{3,3}$.

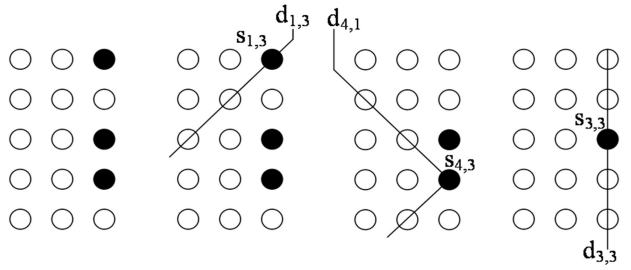


Рисунок 5. Відновлення третіх фрагментів кожного носія

В загалом для отримання інформації з першого блоку задача зводиться до відновлення наступних фрагментів віддалених носіїв $s_{a,1}..s_{a,p}$, $s_{b,1}..s_{b,p}$, $s_{c,1}..s_{c,p}$.

Запропонований спосіб відновлення передбачає реконструювання пошкоджених секторів в наступному порядку: $s_{a,1}$, $s_{c,1}$, $s_{b,1}$, $s_{a,2}$, $s_{c,2}$, $s_{b,2}$, ..., $s_{a,p}$, $s_{c,p}$, $s_{b,p}$.

Запропонований порядок відновлення фрагментів формально представимо у вигляді наступного плану:

1. Встановити номер стовпця j=1.
2. Відновити фрагмент $s_{a,j}$ за однією з наступних формул:

$p=m-2$:

$$\forall j \in \{1,..,m-a-1\}: s_{a,j} = \bigoplus_{\substack{q=1 \\ q\neq j}}^{a+j-1} s_{a+j-q,q} \oplus \oplus d_{1,a+j-1} \quad (9)$$

$$j = m-a: s_{a,j} = s_{1,m-3} \oplus \bigoplus_{\substack{q=1 \\ q\neq j}}^{m-2} s_{m-q,q} \oplus d_{4,m-2} \quad (10)$$

$$\forall j \in \{m-a+1,..,m-2\}: s_{a,j} = \bigoplus_{q=1}^{-m+1+a+j} s_{q,2*m-4-a-j+q} \oplus \oplus \bigoplus_{\substack{q=1 \\ q\neq m+1-a}}^{2*m-1-a-j} s_{m-q+1,-m-1+a+j+q} \oplus d_{4,2*m-2-a-j} \quad (11)$$

$p \in \{m-1, .., m\}$:

$$\forall j \in \{1,..,p-a+1\}: s_{a,j} = \bigoplus_{\substack{q=1 \\ q\neq j}}^{a+j-1} s_{a+j-q,q} \oplus \oplus d_{1,a+j-1} \quad (12)$$

$$\forall j \in \{p-a+2,..,p\}: s_{a,j} = \bigoplus_{q=1}^{-p+a+j-1} s_{q,2*p-a-j+q} \oplus \oplus \bigoplus_{\substack{q=1 \\ q\neq m+1-a}}^{m+p-a-j+1} s_{m-q+1,-m-1+a+j+q} \oplus d_{4,m-a-j+p} \quad (13)$$

$p \in \{m+1, .., n\}$:

$$\forall j \in \{1,..,m-a+1\}: s_{a,j} = \bigoplus_{\substack{q=1 \\ q\neq j}}^{a+j-1} s_{a+j-q,q} \oplus \oplus d_{1,a+j-1} \quad (14)$$

$$\forall j \in \{m-a+2,..,p-a+1\}: s_{a,j} = \bigoplus_{\substack{q=1 \\ q\neq m-a+1}}^{m} s_{m-q+1,q+a+j-m-1} \oplus d_{1,a+j-1} \quad (15)$$

$$\forall j \in \{p-a+2,..,p\}: s_{a,j} = \bigoplus_{q=1}^{-p+a+j-1} s_{q,2*p-a-j+q} \oplus \oplus \bigoplus_{\substack{q=1 \\ q\neq m+1-a}}^{m+p-a-j+1} s_{m-q+1,-m-1+a+j+q} \oplus d_{4,m-a-j+p} \quad (16)$$

3. Відновити фрагмент $s_{c,j}$ за однією з наступних формул:

$p=m-2$:

$$\forall j \in \{1,..,c-2\}: s_{c,j} = \bigoplus_{\substack{q=1 \\ q\neq j}}^{m-c+j} s_{c-j+q,q} \oplus d_{2,m-c+j} \quad (17)$$

$$j = c-1: s_{c,j} = \bigoplus_{\substack{q=1 \\ q\neq j}}^{m-2} s_{q+1,q} \oplus s_{m,m-3} \oplus d_{4,1} \quad (18)$$

$$\forall j \in \{c,..,m-2\}: s_{c,j} = \bigoplus_{\substack{q=1 \\ q\neq c}}^{m+c-j-2} s_{q,-c+j+q} \oplus \oplus \bigoplus_{q=1}^{-c+j+2} s_{m-q+1,m-5+c-j+q} \oplus d_{4,-c+j+2} \quad (19)$$

$p \in \{m-1, .., m\}$:

$$\forall j \in \{1,..,c+p-m\}: s_{c,j} = \bigoplus_{\substack{q=1 \\ q\neq j}}^{m-c+j} s_{c-j+q,q} \oplus \oplus d_{2,m-c+j} \quad (20)$$

$$\forall j \in \{c+p-m+1,..,p\}: s_{c,j} = \bigoplus_{\substack{q=1 \\ q\neq c}}^{p+c-j} s_{q,-c+j+q} \oplus \oplus \bigoplus_{q=1}^{m-p-c+j} s_{m-q+1,2*p-m+c-j+q-1} \oplus d_{4,-c-p+m+j} \quad (21)$$

$p \in \{m+1, .., n\}$:

$$\forall j \in \{1,..,c\}: s_{c,j} = \bigoplus_{\substack{q=1 \\ q\neq j}}^{m-c+j} s_{c-j+q,q} \oplus d_{2,m-c+j} \quad (22)$$

$$\forall j \in \{c+1,..,c+p-m\}: s_{c,j} = \bigoplus_{\substack{q=1 \\ q\neq c}}^{m} s_{q,q+j-c} \oplus \oplus d_{2,m-c+j} \quad (23)$$

$$\forall j \in \{c+p-m+1,..,p\}: s_{c,j} = \bigoplus_{\substack{q=1 \\ q\neq c}}^{p+c-j} s_{q,-c+j+q} \oplus \oplus \bigoplus_{q=1}^{m-p-c+j} s_{m-q+1,2*p-m+c-j+q-1} \oplus d_{4,-c-p+m+j} \quad (24)$$

4. Відновити фрагмент $s_{b,j}$ за наступною формулою:

$$\forall j \in \{1,..,p\}: s_{b,j} = \bigoplus_{\substack{q=1 \\ q \neq b}}^{m} s_{q,j} \oplus d_{3,j} \qquad (25)$$

Нижче представлений формальний розв'язок на прикладі, який проілюстрований вище і в рамках якого вважається, що загальна кількість віддалених носіїв дорівнює п'яти, а блоки складаються з трьох однойменних фрагментів кожного носія. Для повноцінного розв'язку фрагментом в даному випадку виступатиме блок інформації, що складається з 4 бітів.

Припущено, що перший блок фрагментів B_1 включає в себе наступну інформацію:

$$B_1 = \begin{pmatrix} 1101 & 1111 & 0001 \\ 1001 & 0101 & 1001 \\ 0110 & 1111 & 0100 \\ 0111 & 0101 & 0000 \\ 1001 & 1010 & 1110 \end{pmatrix}$$

Сформовано значення фрагментів резервних носіїв за запропонованим методом. Фрагменти першого резервного носія, мають наступний вигляд:

$$d_{1,1} = \bigoplus_{q=1}^{1} s_{2-q,q} = 1101$$

$$d_{1,2} = \bigoplus_{q=1}^{2} s_{3-q,q} = 1001 \oplus 1111 = 0110$$

$$d_{1,3} = \bigoplus_{q=1}^{3} s_{4-q,q} = 1101 = 0110 \oplus 0101 \oplus$$
$$\oplus 0001 = 0010$$

Фрагменти другого резервного носія мають наступний вигляд:

$$d_{2,1} = \bigoplus_{q=1}^{1} s_{4+q,q} = 1001$$

$$d_{2,2} = \bigoplus_{q=1}^{2} s_{3+q,q} = 0111 \oplus 1010 = 1101$$

$$d_{2,3} = \bigoplus_{q=1}^{3} s_{2+q,q} = 0110 \oplus 0101 \oplus 1110 = 1101$$

Фрагменти третього резервного носія мають наступний вигляд:

$$d_{3,1} = \bigoplus_{q=1}^{5} s_{q,1} = 1101 \oplus 1001 \oplus 0110 \oplus$$
$$\oplus 0111 \oplus 1001 = 1100$$

$$d_{3,2} = \bigoplus_{q=1}^{5} s_{q,2} = 1111 \oplus 0101 \oplus 1111 \oplus$$
$$\oplus 0101 \oplus 1010 = 1010$$

$$d_{3,3} = \bigoplus_{q=1}^{5} s_{q,3} = 0001 \oplus 1001 \oplus 0100 \oplus$$
$$\oplus 0000 \oplus 1110 = 0010$$

Фрагменти четвертого резервного носія мають наступний вигляд:

$$d_{4,1} = \bigoplus_{q=1}^{3} s_{q+1,q} \oplus s_{5,2} = 1001 \oplus 1111 \oplus 0000 \oplus$$
$$\oplus 1010 = 1100$$

$$d_{4,2} = \bigoplus_{q=1}^{3} s_{q,q} \oplus \bigoplus_{q=1}^{2} s_{6-q,q} = 1101 \oplus 0101 \oplus 0100$$
$$\oplus 1001 \oplus 0101 = 0000$$

$$d_{4,3} = s_{1,2} \oplus \bigoplus_{q=1}^{3} s_{5-q,q} = 0111 \oplus 1111 \oplus 1001 \oplus$$
$$\oplus 1111 = 1110$$

Прийнято, що було втрачено доступ до першого, третього і четвертого віддаленого носія (a=1, b=3 і c=4)

Встановлено j=1.

Оскільки j∈{1, .., 3} $s_{1,1}$ відновлено за допомогою формули (7), в яку підставлено значення a, m та j. Отримано наступний вираз:

$$s_{1,1} = \bigoplus_{\substack{q=1 \\ q \neq 1}}^{1} s_{2-q,q} \oplus d_{1,1} = d_{1,1} = 1101$$

Оскільки j∈{1, .., 2} $s_{4,1}$ відновлено за допомогою формули (10), в яку підставлено значення c, m та j. Отримано наступний вираз:

$$s_{4,1} = \bigoplus_{\substack{q=1 \\ q \neq 1}}^{2} s_{3+q,q} \oplus d_{2,2} = s_{5,2} \oplus d_{2,2} = 1010 \oplus$$
$$\oplus 1101 = 0111$$

Відновлено $s_{3,1}$ за допомогою формули (13), в яку підставлено значення b, m та j. Отримуємо наступний вираз:

$$s_{3,1} = \bigoplus_{\substack{q=1 \\ q \neq 3}}^{5} s_{q,1} \oplus d_{3,1} = s_{1,1} \oplus s_{2,1} \oplus s_{4,1} \oplus s_{5,1} \oplus$$
$$\oplus d_{3,1} = 1101 \oplus 1001 \oplus 0111 \oplus$$
$$\oplus 1001 \oplus 0100 = 0110$$

Збільшуємо значення j на 1, нове значення номера стовпця j=2.

Аналогічно відновленню перших однойменних фрагментів кожного носія, до якого було втрачено доступ, виконуємо відновлення других фрагментів. Отримуємо наступні вирази:

$$s_{1,2} = \bigoplus_{\substack{q=1 \\ q \neq 2}}^{2} s_{3-q,q} \oplus d_{1,2} = 1001 \oplus 0110 = 1111$$

$$s_{4,2} = \bigoplus_{\substack{q=1 \\ q\neq 2}}^{3} s_{2+q,q} \oplus d_{2,3} = 0110 \oplus 1110 \oplus$$

$$\oplus 1101 = 0101$$

$$s_{3,2} = \bigoplus_{\substack{q=1 \\ q\neq 3}}^{5} s_{q,2} \oplus d_{3,2} = 1111 \oplus 0101 \oplus$$

$$\oplus 0101 \oplus 1010 \oplus 1010 = 1111$$

Збільшуємо значення j на 1, нове значення номера стовпця j=3.

Аналогічним способом отримуємо наступні вирази:

$$s_{1,3} = \bigoplus_{\substack{q=1 \\ q\neq 3}}^{3} s_{4-q,q} \oplus d_{1,3} = 0110 \oplus 0101 \oplus$$

$$\oplus 0010 = 0001$$

$$s_{4,3} = \bigoplus_{\substack{q=1 \\ q\neq 3}}^{3} s_{q+1,q} \oplus s_{5,2} \oplus d_{4,1} = 1001 \oplus 1111 \oplus$$

$$\oplus 1010 \oplus 1100 = 0000$$

$$s_{3,3} = \bigoplus_{\substack{q=1 \\ q\neq 3}}^{5} s_{q,3} \oplus d_{3,3} = 0001 \oplus 1001 \oplus$$

$$\oplus 0000 \oplus 1110 \oplus 0010 = 0100$$

Збільшуємо значення j на 1, нове значення номера стовпця j=4. j=4 не відповідає умові $j \leq 3$, отже розв'язання завершено.

Оцінка ефективності

Враховуючи безмежну кількість варіантів значень пари m і p та різний підхід до резервування та відновлення даних в залежності від цих значень, оцінка ефективності визначається методом моделювання. За результатами моделювання отримано наступну обчислювальну складність:

$$O(n*(5{,}55*m + 0{,}225*p - 12{,}15))$$

При використанні запропонованого методу розробник такої системи має обрати розмір блоків, на які будуть розділені всі однойменні фрагменти кожного віддаленого носія. Моделювання показало, що зі збільшення розміру блоку зростає середнє значення кількості операції на фрагмент. На рис. 6 зображено графік залежності середнього значення кількості операцій на фрагмент від розміру блоку для m=4.

Виходячи з вище сказаного в теорії найефективнішим відносно швидкості відновлення даних виступає блок найменшого розміру, тобто p=m-2. На практиці ж спиратися треба на кількість процесорів P, які будуть використовуватися у відновленні даних, та середньостатистичне значення кількості недоступних різнойменних фрагментів δ кожного віддаленого носія , до якого втрачено доступ, та їх розташування на носієві. Якщо всі фрагменти розташовані один за одним, тобто належать одній послідовності, то найбільш ефективним буде розмір блоку який відповідає умові:

$$(\delta/P) mod_p = 0 \text{ при } p \rightarrow m\text{-}2 \qquad (26)$$

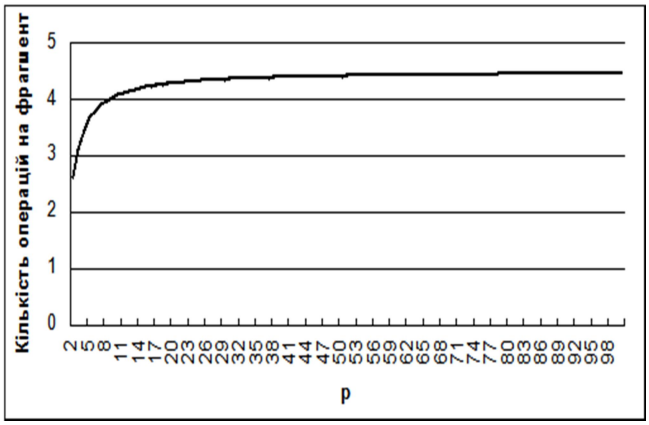


Рисунок 6. Залежність середнього значення кількості операцій на фрагмент відносно розміру блоку

В іншому випадку, коли фрагменти розташовані в r послідовностях, треба вирахувати середнє значення довжини послідовностей, тоді формула (26) набуде наступного вигляду:

$$(\delta / r/P) mod_p = 0 \text{ при } p \rightarrow m\text{-}2 \qquad (27)$$

Висновки

Запропоновано метод організації резервування та прискореного відновлення даних при їх зберіганні на віддалених від користувача носіях з урахуванням потреби відновлення тільки частини даних недоступних не більше як трьох віддалених носіїв.

Дослідження запропонованого методу прискорення відновлення довели, що розподіл всіх фрагментів віддаленого носія на блоки певного розміру і використання чотирьох резервних носіїв відкривають можливість для відновлення кожного блоку даних окремо, що значно збільшує швидкість відновлення відносно малої частини носія.

Розроблений метод може бути ефективно використано в перспективних "хмарних" технологіях віддаленого зберігання даних.

Список посилань

1. Patterson D. Case of Redundant Array of Inexpensive Disk (RAID) / Patterson D., Gibson G., Katz R.- Berkeley: University of California.-1987.- 238 P.
2. Plank J. S., Thomasson M.G., On Practical Use of LDPC Erasure Codes for Distribute Storage Application: Technical Report UT-CS-03-510.- Department of Computer Science, University of Tennessee.-2004.
3. Berger C.R. Optimizing Joint Erasure- and Error-Correction Coding for Wireless Packet Transmissions // Willett P., Pattirati K. / IEEE Transactions on wireless communications.- Vol.7 - No. 11 .- 2008.- PP. 4586-4595.
4. Иванов Д. Г. Организация резервирования в системах распределенного хранения данных // Вісник Національного технічного університету України ”КПІ” Інформатика, управління та обчислювальна техніка, – Київ: ВЕК+ – 2012 – № 56. с.160-164.
5. Марковський О.П. Організація резервування та відновлення даних при їх віддаленому зберіганні / Марковський О.П., Іванов Д.Г., Ванчугов Б.Ю. // Вісник Національного технічного університету України ”КПІ” Інформатика, управління та обчислювальна техніка, – Київ: ВЕК+ – 2013. – № 59. - С. 50-55.
6. Иванов Д.Г. Метод восстановления данных на основе скошенных матриц в системах распределенного хранения / Иванов Д.Г. Луцкий Г.М. // Известия высших учебных заведений. Проблемы полиграфии и издательского дела. Информационные технологии.- М.:УПИПК МГУП им. И.Федорова.- 2013.- № 2. - С.47-52

УДК 004.451.24

СТІРЕНКО С.Г.

МЕТОД ПЕРЕБАЛАНСУВАННЯ НАВАНТАЖЕННЯ В КЛАСТЕРНИХ КОМП'ЮТЕРНИХ СИСТЕМАХ В ДИНАМІЧНОМУ РЕЖИМІ

В роботі розглядаються підходи до організації відкладення моменту передачі даних, що ефективно використовуються під час виконання перебалансування навантаження. В межах розглянутих підходів висовуються гіпотези щодо вибору оптимального моменту затримки часу передачі даних.

This paper provides approaches to the deposition time of data that are being used during the execution of rebalancing the load. Within the approaches put forward hypothesis on choosing the best moment of delay time data.

Підходи до організації відкладення передачі даних при перебалансуванні

Відкладення передачі даних, що були введені або обчислені в одному вузлі, а будуть використані в іншому вузлі, є загальною ідеєю яка може бути декількома різними способами в межаш запропонованої моделі [1,2]. Зокрема для її реалізації необхідно запропонувати методи визначення часу прийняття рішення про відкладення, методи визначення необхідності відкладення, методи визначення тривалості відкладення тощо. При цьому також необхідно враховувати необхідність виконання запропонованих методів та алгоритмів, що їх реалізують, в динаміці так, щоб їх виконання не займало істотну кількість часу, оскільки він буде врахований в загальний час виконання обчислень і може негативно вплинути на коефіцієнт ефективності. Розглянемо декілька відносно простих підходів до визначення цих значень.

Підхід 1: *без відкладення передачі даних*. Цей підхід пропонується для підтвердження сумісності відкладення передачі даних з існуючими моделями програмування обчислювальних систем з локальною пам'яттю. Він не передбачає власне відкладення передачі даних, а лише передачу даних безпосередньо після їх готовності $t_{comm,k}^{i\to j} = t_{rdy,k}^{i}$. Реалізація підходу без відкладення передачі даних може застосовуватись для перевірки працездатності програмної реалізації та коректності проведення обчислень. Рішення про передачу даних приймається у момент готовності даних.

Гіпотеза 1. Застосування підходу без відкладення передачі даних знизить загальний коефіцієнт ефективності через наявність додаткових розрахунків по визначенню часу початку передачі даних.

Підхід 2: *передача даних по запитам*. Цей підхід є прямо протилежним до підходу без відкладення передачі даних, так як передбачає початок передачі даних одразу після того, як надійшов запит на їх використання, якщо дані вже готові на цей момент часу, або передачу даних одразу по готовності, якщо запит надійшов раніше.

$$t_{comm,k}^{i\to j} = \begin{cases} t_{req,k}^{j}, \text{якщо } t_{req,k}^{j} \geq t_{rdy,k}^{i}; \\ t_{rdy,k}^{i}, \text{інакше.} \end{cases}$$

Застосування цього підходу за умови наявності балансування навантаження потенційно може зменшити час очікування введення та виведення даних у ситуаціях, аналогічних до розглянутих у попередньому підрозділі. Додатковим ефектом може бути також зменшення часу безпосереднього введення та виведення даних за мережею завдяки запобіганню повторних передач даних.

Гіпотеза 2. Застосування підходу передачі даних по запитам може збільшити обсяги пам'яті, що використовуються у вузлах, які переважно відправляють дані.

Гіпотеза 3. Застосування підходу передачі даних по запитам не знижує час очікування введення та виведення даних за відсутності балансування навантаження у випадках, коли запит на використання даних відбувся пізніше ніж вони були готові (в цьому разі можна розпочати передачу у такий момент часу, щоб вона завершилась раніше запиту на використання).

Підхід 3: *відкладення передачі даних на певний час*. Даний підхід передбачає відкладення початку передачі даних на певний фіксований час td за умови, що на момент готовності даних запит на їх використання ще не надійшов. В інших випадках відкладення передачі даних лише збільшить час очікування введення та

виведення, тому виконувати відкладення передачі даних може бути недоцільно.

$$t_{comm,k}^{i\to j} = \begin{cases} t_{rdy,k}^{j} + t_d, \text{якщо } t_{req,k}^{j} > t_{rdy,k}^{i}; \\ t_{rdy,k}^{i}, \quad \text{інакше.} \end{cases}$$

Рішення про відкладення передачі даних приймається у момент готовності даних. Однак можливі варіанти підходу, з повторним відкладанням на фіксований час після виконання балансування навантаження, що вплинуло на вже відкладену передачу. В рамках цього підходу необхідно розглянути декілька способів визначення часу затримки t_d.

Спосіб 3.1: статичне задання часу затримки. Час затримки t_d = const задається статично для всього процесу обчислень. Перевагою цього способу є відсутність обчислень для визначення часу t_d, однак істотним недоліком є неможливість урахування динамічних характеристик обчислень. Конкретне значення константи може бути розраховане попередньо на основі тестових запусків певного ряду обчислень.

Гіпотеза 4. Застосування відкладення передачі даних на статично заданий час затримки при малих значеннях часу затримки наближається за значенням коефіцієнту ефективності до способу без відкладення передачі даних.

Гіпотеза 5. Застосування відкладення передачі даних на статично заданий час затримки при великих значеннях часу затримки, тобто таких, що перевищують середнє значення різниці між моментом використання даних та моментом їх готовності, знижує коефіцієнт ефективності. Через те, що час початку передачі в середньому буде більше часу передачі, що збільшить сумарний час очікування введення та виведення.

Спосіб 3.2: задання часу затримки як функції від різниці середнього часу між моментом запиту даних та моментом їх готовності. При чому, доцільно розглядати лише випадки, у яких запит на використання даних отриманий пізніше ніж дані готові до використання, так як в іншому випадку середнє значення прямуватиме до нуля. Визначимо

$$t_{diff,k} = \begin{cases} t_{req,k}^{j} - t_{rdy,k}^{i}, \text{якщо } t_{req,k}^{j} > t_{rdy,k}^{i}; \\ 0, \quad \text{інакше.} \end{cases} \tag{1}$$

Тоді середня різниця

$$\hat{t}_{diff} = avg_k t_{diff,k},$$

в цьому разі можемо визначити час затримки як функцію середньої різниці $t_d = f(\hat{t}_{diff})$.

Гіпотеза 6. Якщо $f(\hat{t}_{diff}) > \max_k t_{diff,k}$, то передача даних розпочнеться завжди через певний час після запиту за визначенням (1), що призведе до зниження коефіцієнту ефективності. Якщо ж $f(\hat{t}_{diff}) < \text{avg}_k t_{diff,k}$, то коефіцієнт ефективності має збільшитись за умови виконання обмеження обсягів використовуваної пам'яті.

Найбільш перспективними для дослідження функціями f обчислення часу затримки є монотонно неспадні функції на проміжку [0; +∞) з обмеженням $f(t) \le \max_k t_{diff,k}$. Серед таких функцій можна розглянути лінійні функції вигляду $kt + b$, нелінійні монотонні функції вигляду:

$$at^k + b: \ k, b \in \mathbb{R},$$
$$alog_k t + b: \ a, b \in \mathbb{R}, k \in \mathbb{N},$$
$$ab^t + c: \ a, b, c \in \mathbb{R}.$$

До розрахунку середнього часу різниці входять лише ті передачі, які завершились до цього моменту, тому необхідність в оцінюванні часу майбутніх передач або часу завершення певних обчислень відсутня.

Гіпотеза 7. Відкладення передачі даних в перші 5-10% часу виконання обчислень недоцільне, оскільки у цей час переважно виконується передача вхідних даних та похідних від них, які досить швидко використовуються для подальших обчислень. Однак, застосування способу 3.2 за умови $f(0) = 0$ позбавлене даного недоліку.

Оскільки обчислення виконуються в системі з локальною пам'яттю, необхідно розглянути декілька стратегій по обчисленню необхідних середніх значень. При цьому основний вплив має фактор (де-)централізованості збирання статистики. Оскільки розглядається обчислювальна система з потенційно неоднорідними зв'язками між вузлами, середній час різниці $t_{diff}^{i\to j}$ може суттєво відрізнятися для різних пар вузлів i, j, тому доцільно при визначенні часу затримки перед безпосереднім початком передачі даних t_d враховувати лише ті значення, які відповідають передачам даних між тією ж самою парою вузлів. Таким чином, кожний вузол має збирати значення t_{diff} для даних, які передавалися з та на нього, при цьому обмін цими даними між вузлами не потрібен. Останнє дозволяє також зменшити час, що витрачається на роботу інфраструктури даної системи тобто меншим чином негативно вплинути на коефіцієнт ефективності обчислень.

Підхід 4: *відкладення передачі з оцінкою часу передачі та обчислень.* Даний підхід передбачає можливість відкладення початку передачі даних на певний час td, аналогічно до підходу 3

$$t_{comm,k}^{i\to j} = \begin{cases} t_{rdy,k}^{j} + t_d, \text{якщо } t_{req,k}^{j} > t_{rdy,k}^{i}; \\ t_{rdy,k}^{i}, \quad \text{інакше}, \end{cases}$$

однак час затримки перед безпосередньою передачею даних є функцією оціненої тривалості передачі та/або обчислень на вузлі, з якого передаються дані, та вузлі, на який передаються дані. В даному підході необхідно оцінювати тривалість передачі та час, через який може відбутися перше звернення до певних даних, так як точні значення на момент прийняття рішення про відкладення невідомі: запит на звернення ще не надійшов, передачу даних ще не виконано. Існує ряд методів, які дозволяють оцінити цей час [3,4], однак найбільш простим з них є застосування моделі з абстракціями задач, підзадач та пов'язаних з ними даних, запропоноване в [5-7].

В цій моделі вважається, що система розв'язує певну задачу T_0 (тобто послідовність обчислювальних дій), яку можна розбити на частини для їх паралельного виконання, причому кожна частина може бути розбита на менші частини рекурсивно. Для визначення частин вводиться поняття підзадача z-го порядку T_z, який характеризує кількість рекурсивних розбиттів до отримання даної підзадачі. Розбиття підзадачі визначається за допомогою оператора split

$$split\ T_z^{\langle j,\dots,k\rangle} \to \left\{T_{(z+1)}^{\langle j,\dots,k,i\rangle}\right\},$$
$$i \in \left(1, N_{(z+1)}^{\langle j,\dots,k\rangle}\right), \qquad (1)$$

де $N_{(z+1)}^{\langle j,\dots,k\rangle}$ – кількість підзадач, на які була розбита вихідна підзадача $T_{(z+1)}^{\langle j,\dots,k\rangle}$.

Послідовне застосування оператора розбиття до підзадач, представлених у вигляді множини послідовностей дій, призводить до формування у кожної підзадачі послідовності Г: $T_{z)}^{\langle \gamma 1,\gamma 2,\gamma 3,\dots\rangle}$, довжина якої визначає порядок підзадачі z = dimГ, а кожний i-тий елемент послідовності характеризує номер підзадачі і-го порядку, з якої було отримано дану підзадачу. Послідовність Г називається шляхом розбиття. Для вихідної задачі, яку для збереження спільності можна вважати підзадачею нульового порядку, шлях розбиття є пустою послідовністю ⟨⟩.

Використання подібної моделі програмування дозволяє при прийнятті рішення про відкладення передачі даних розглядати інформацію про порядок підзадачі та шлях розбиття, використовуючи яку можна формулювати оцінку часу виконання обчислень. Для оцінки часу виконання передачі даних, можна використовувати як обсяг інформації, що передається, так і близькість відповідних шляхів розбиття.

При застосуванні даного підходу можна розглянути декілька способів обчислення часу затримки перед безпосереднім відправленням даних.

Спосіб 4.1. Виконати відкладення на максимально можливий час мінус оцінений час передачі. Для цього необхідно оцінити час передачі поточних даних та час, у який відбудеться запит на їх використання. Для оцінки часу запиту на використання певних даних необхідно визначити час початку обчислень, що їх використовують. В моделі з використанням підзадач, цей час може бути оцінений як сума часу виконання всіх підзадач, що знаходяться в черзі перед тією, що використовує поточні дані $\sum t_c^l$. Причому час виконання підзадачі може бути оцінений як функція часу виконання всіх підзадач на цьому вузлі з урахуванням їх шляху розбиття

$t_c^{\Gamma^l} = f(t_c^{\Gamma^{l-1}}, t_c^{\Gamma^{l-2}}, \dots, t_c^{\Gamma^1}, \Gamma^{l-1}, \Gamma^{l-2}, \dots, \Gamma^1)$. Можна вважати, що час виконання обчислень підзадач однакового порядку підпорядкований нормальному розподілу $N(\mu,\sigma^2)$ за центральною граничною теоремою, оскільки на нього впливає велика кількість факторів, де математичне очікування μ є зваженим середнім арифметичним

$$\mu = \frac{\sum_{l=1}^{L} d_{\Gamma,\Gamma^l} t_c^{\Gamma}}{\sum_{l=1}^{L} d_{\Gamma,\Gamma^l}};$$

середньоквадратичне відхилення σ є зваженою характеристикою

$$\sigma = \sqrt{\frac{1}{\sum_{l=1}^{L} d_{\Gamma,\Gamma^l}} \sum_{l=1}^{L} d_{\Gamma,\Gamma^l} (t_c^{\Gamma^l} - \mu)^2};$$

$t_c^{\Gamma^1}$ – відомий час виконання обчислень підзадачі із шляхом розбиття Γ^l; L – кількість підзадач, що були обчислені на даний момент; Г = ⟨γ_i⟩ – шлях розбиття підзадачі, час виконання якої оцінюється; d_{Γ,Γ_l}– ваговий коефіцієнт, що обчислюється за формулою

$$d_{\Gamma,\Gamma_l} = b^{\dim\Gamma - \dim\Gamma_l} \sum_{i=1}^{\min(\dim\Gamma - \dim\Gamma_l)} \delta(\gamma_i - \gamma_i^l),$$

δ – функція Дірака, b – коефіцієнт розгалуження задач на підзадачі, тобто середня кількість підзадач після розбиття. Вагова функція враховує той факт, що близькі за шляхом розподілу підзадачі найбільш ймовірно мають близький час виконання обчислень для більшості задач. При цьому, якщо порівняння виконується з підзадачею меншого порядку, її час розрахунків необхідно розділити на відповідну кількість підзадач більшого порядку, а у випадку порівняння з підзадачею більшого порядку – помножити на відповідну кількість підзадач, що враховується показниковою функцією коефіцієнту розгалуження.

Оцінка часу запиту k-го блоку даних $t_{req,k}^{j}$ береться як сума випадкових значення з нормальним розподілом N_C^j (μ,σ) для всіх підзадач, що мають бути виконані j-тим вузлом. Аналогічним чином визначається нормальний розподіл для оціненого часу передачі даних $N_T^{i\to j}$ (μ,σ), за яким вибирається оцінка часу передачі даних $t_T^{i\to j}$. Після чого час затримки початку передачі даних визначається як

$$t_d = t_{req,k}^{i} - t_{T,k}^{i\to j}.$$

Гіпотеза 8. Відкладення на максимально можливий час з використанням оцінки часу за нормальним розподілом може зменшувати коефіцієнт ефективності у випадку задач, що розбиваються на підзадачі з обсягами обчислень, що підпорядковуються деякому періодичному закону.

Спосіб 4.2 Виконувати передачу в середині проміжку $[t_{rdy,k}^{\mathrm{i}}, t_{req,k}^{\mathrm{j}}]$, тобто

$$t_{comm,k}^{i\to j} = \begin{cases} t_{req,k}^{j} - t_{rdy,k}^{i}, \text{якщо } t_{req,k}^{j} > t_{rdy,k}^{i}; \\ t_{rdy,k}^{i}, \qquad \text{інакше.} \end{cases}$$

При цьому оцінка часу, у який відбудеться запит на використання даних виконується з використанням нормального розподілу аналогічно до попереднього способу.

Гіпотеза 9. Застосування відкладення передачі даних до середини оціненого проміжку очікування демонструє більші значення коефіцієнта ефективності, аніж відкладення на максимально можливий час для випадків, в яких останній є малоефективним через нерівномірність обсягів обчислень.

Для більш точних оцінок часу майбутнього запиту на використання даних та часу передачі даних між вузлами, існує можливість в рамках даного підходу використати різні ймовірнісні моделі, зокрема графові. Крім того, для визначення параметрів цих моделей можна застосувати широко вивчені на даний момент методи машинного навчання [8,9].

Слід зазначити, що після виконання балансування навантаження або відміни частини обчислень, необхідно заново розглянути можливість відкладення даних для обох запропонованих способів, оскільки оцінка часу передачі даних та часу запиту на їх використання могла змінитися через зміну вузла, на якому будуть виконуватись обчислення.

Висновки

Запропоновано ряд підходів до визначення часу початку передачі даних. Серед них слід відзначити два підходи, які є відповідно нижнім і верхнім обмеженням на доцільні значення часу початку передачі даних: передавати дані відразу по готовності, тобто фактично на виконувати відкладення, та передавати дані за запитом, тобто відкладати передачу на максимально можливий термін. Обмеження знизу є гарантією збереження коректності обчислень, оскільки дані будуть підготовані перед їх пересилкою для використання, а вихід за обмеження зверху через необхідність додаткового очікування після запиту на використання даних вносить у час виконання затримку, під час якої обчислення не виконуються, що в результаті знижує коефіцієнт ефективності. Запропоновано також два підходи до більш гнучкого обчислення часу початку передачі даних, один з яких базується на статистичній обробці попередніх значень часу очікування введення та виведення та обсягів пам'яті під час розв'язання даної задачі, а інший – на ймовірнісній оцінці часу першого запиту даних та часу, необхідного для передачі цих даних між вузлами. В рамках кожного з підходів може бути запропоновано декілька різних способів визначення невідомих величин, а саме можна змінювати алгоритми статистичної обробки або ймовірнісні моделі прогнозування. Для останніх найбільш простою є використання моделі програмування, що базується на розбитті на підзадачі, яка якісно відображає задачі з великою кількістю даних для обробки.

Список посилань

1. Стіренко С.Г. Модель технології програмування паралельних систем // Вісник НТУУ «КПІ» Інформатика, управління та обчислювальна техніка. – 2013. – Т.58. – С.158-164.
2. Стіренко С.Г. Організація відкладених обчислень в кластерних системах // Вісник НТУУ «КПІ» Інформатика, управління та обчислювальна техніка. – 2013. – Т.59. – С.12-16.
3. Hennessy, John L. Computer architecture: a quantitative approach / John L Hennessy, David A Patterson. – Amsterdam, Noord-Hooland, Netherlands: Elsevier, 2011. – 824 p.
4. Tanenbaum, Andrew S. Distributed Systems: Principles and Paradigms (2nd Edition) / Andrew S. Tanenbaum, Maarten van Steen. – Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 2006.
5. Представление задач в системах распараллеливания с изменяемой зернистостью / Г. М. Луцкий, С. Г. Стиренко, А. И. Зиненко, Д. В. Грибенко // Вісник НТУУ «КПІ». Інформатика, управління та обчислювальна техніка. – 2013. – № 55. – С. 212.
6. Стиренко С. Г. Модель организации вычислений в распределённой системе / С. Г. Стиренко, А. И. Зиненко, Д. В. Грибенко // Вісник НТУУ «КПІ». Інформатика, управління та обчислювальна техніка.– 2012,– № 57. – С. 101-109.
7. Pheatt, Chuck. Intel ® threading building blocks / Chuck Pheatt //Journal of Computing Sciences in Colleges.– 2008.– Vol. 23, no. 4.– P. 298–298.
8. Russell, S.J. Artificial intelligence: a modern approach / S.J. Russell, P. Norvig. Prentice Hall series in artificial intelligence. –New York, NY, USA: Pearson Education/Prentice Hall, 2010.
9. Koller, D. Probabilistic graphical models: principles and techniques / D. Koller, N. Friedman. Adaptive Computation and Machine Learning Series. – Boston, MA, USA: Mit Press, 2009.

УДК 004.056.5

МАРКОВСЬКИЙ О.П.,
ГУБСЬКА А.О.,
ЧЕЧЕЛЬ В.Р.,
ФЕДОРЕЧКО О.І.

МЕТОД КОРЕКЦІЇ ДВОХ ПОМИЛОК СИНХРОНІЗАЦІЇ В АСИНХРОННИХ ЛІНІЯХ ПЕРЕДАЧІ ЦИФРОВИХ ДАНИХ

В статті запропоновано метод виправлення двох помилок, що виникають в асинхронних каналах передачі цифрових даних. Запропонований метод оснований за використанні системи зважених контрольних сум, що утворюють систему рівнянь, розв'язання яких дозволяє локалізувати помилки синхронізації і визначити їх тип. Метод використовує логічні операції. Детально представлено математичну ідею методу та процедури класифікації та корекції пари помилок. Використання процедури корекції помилок синхронізації ілюструється прикладами. Наведено теоретичні та експериментальні оцінки ефективності запропонованого методу.

In article method of double errors correcting in asynchronous digital data transmission channels is proposed. The proposed method is based on the use system of weighted checksum that creating a system of equations, the solution which allow to localize the synchronization errors and determine their type. Method is based on logical operations. The method mathematical basis and procedure of detecting and correcting of double errors are presented in detail. The use of the procedure double errors correction is illustrated via the thorough presentation of an example of erroneous data transmission. The theoretical and experimental estimations of effectiveness of proposed method are given.

Вступ

Досягнутий в останнє десятиліття розвиток засобів комп'ютерної обробки інформації все більшою мірою визначається розповсюдженням технологій розподілених обчислень. Необхідною передумовою ефективності розподіленої обробки даних є високий рівень розвитку засобів обміну та передачі цифрових даних. Домінуюче місце серед них посідають послідовні інтерфейси, які здатні забезпечити високу швидкість передачі цифрових даних в широкому діапазоні відстаней. Для передачі даних на відносно невеликі відстані (до десятків метрів) застосовуються послідовні канали з кодово-імпульсною модуляцією, в той час, як передача цифрової інформації на більші відстані реалізується послідовними інтерфейсами з модуляцією синусоїдального сигналу (канали зі спектральною модуляцією).

Задля досягнення високої швидкості передачі даних в послідовних каналах з кодово-імпульсною модуляцією здебільшого використовується асинхронне кодування, яке при заданому граничній частоті зміни потенціалу в лінії забезпечує, як мінімум, вдвічі більшу інформаційну швидкість передачі даних. Суттєвим недоліком асинхронного кодування даних є ризики виникнення помилок синхронізації - помилкового сприйнятті на боці приймача довжини серії однакових бітів, передача яких не синхронізована, тобто не супроводжується зміною потенціалу в лінії. Ймовірність виникнення помилок синхронізації суттєво зростає зі збільшенням довжини серії бітів, передача яких не синхронізується, підвищенні частоти передачі даних, а також зі збільшенням різниці температур на приймачі та передавачі [1].

Динамічний розвиток систем розподіленої обробки даних стимулює неперервне зростання швидкості передачі даних в паралельних каналах, насамперед за рахунок збільшення частоти слідування сигналів. Так, в новому стандарті послідовного інтерфейсу USB 3.0 тактова частота передачі в порівнянні з попереднім збільшилася в 10 раз [1]. Таке відчутне збільшення частоти передачі має наслідком зростання ймовірності виникнення помилок синхронізації в силу того, що темпи росту частоти помітно випереджають збільшення точності роботи тактових генераторів, яка є визначальним чинником того, що довжина несинхронізованої послідовності бітів на стороні передавача і на стороні приймача сприймається однаковою [2]. До цього слід додати, що збільшення частоти передачі в сучасних послідовних інтерфейсах призводить до різкого зростання негативного впливу міжсигнальної інтерференції, яка, в свою чергу, стимулює спотворення границь сигналу несинхронізованої передачі послідовності бітів і, тим

самим, провокує виникнення помилок синхронізації [2].

Вказані чинники диктують необхідність адекватного удосконалення засобів контролю та виправлення помилок синхронізації в послідовних каналах обміну даними комп'ютерних систем. Зокрема, практичного значення набуває задача корекції двох помилок синхронізації при передачі одного блоку даних.

Таким чином, задача підвищення ефективності корекції помилок синхронізації в послідовних каналах обміну даними в ракурсі розширення класу помилок, що можуть бути виправлені, є актуальною для сучасного етапу розвитку технології передачі інформації в комп'ютерних системах.

Аналіз існуючих засобів виявлення та корекції помилок синхронізації

Динамічний розвиток засобів передачі даних мав наслідком збільшення на порядки швидкості передачі даних. Виходячи з реалій сьогоднішнього дня тенденція зростання швидкості передачі буде зростати і далі.

За цих умов відбувається переоцінка значимості традиційних критеріїв ефективності засобів виправлення помилок. Так, критерій кількості контрольних розрядів за умов стрімкого зростання швидкості передачі втрачає свою значимість. На противагу йому, зростає значимість такої характеристики як часова складність обчислювальних процедур корекції, що визначає час корекції помилок, що виникли при передачі.

В послідовних інтерфейсах комп'ютерних систем використовуються дві базові технології виправлення виникаючих в процесі передачі даних помилок:

- виявлення помилок за допомогою спеціальних кодів та їх виправлення шляхом повторної передачі;
- корегування помилки безпосередньо на стороні приймача з використанням даних контрольного коду.

Перша з наведених технологій також передбачає додаткову передачу контрольного коду, який використовується лише для виявлення помилок. Здебільшого, в якості контрольного коду використовується CRC (Control Redundancy Codes) [2], який здатен гарантовано виявляти бітові помилки, кратність яких не перевищує трьох. Інші помилки CRC виявляє зі ймовірністю $1\text{-}2^{-n}$, де n-розрядність контрольного коду. В умовах досягнутого в останні роки значного збільшення об'ємів даних, що передаються між компонентами комп'ютерних систем ймовірність того, що помилка синхронізації не буде виявлена приймає реальні значення. Наприклад, при використанні CRC-32 і передачі 128 Мбайт ймовірність виникнення непоміченої помилки становить 10^{-3} [3]. Головною вадою виправлення помилок в прийнятому блоці даних шляхом його повторної передачі є великі затрати часу на перевірку правильності передачі, посилку запису на повторну передачу і саму повторну передачі. Обчислення контрольного коду CRC виконується рекурсивно, розряд за розрядом, тобто має принципово послідовний характер, що зумовлює значні часові затрати на його реалізацію.

Надсилання приймачем запиту на повторну передачу і сама повторна передача також потребує значного часу. І до того ж, вона не гарантує, що при повторній передачі не матимуть місце помилки [4]. Тому, в сучасних умовах використання повторної передачі, як засобі виправлення виникаючих помилок, стає все менш ефективним. Особливо це стосується комп'ютерних систем управління, що працюють в реальному часі.

Виходячи з критерію часу корекції помилок, більш ефективним є використання корегуючих кодів. Проте, класичні корегуючі коди, такі як коди Хемінга, коди Голея, циклічні коди, включаючи БЧХ і коди Ріда-Соломона орієнтовані на виправлення однієї або обмеженої кількості помилок і не здатні виправляти помилки синхронізації [2]. Тому, на практиці використовуються методи попередження виникнення помилок синхронізації і спеціалізовані методи їх виправлення [3].

Найбільш поширеним методом попередження виникнення помилок синхронізації є використання бітового стаффінгу - тобто вставка біту, передача якого супроводжується зміною потенціала на лінії в послідовності несинхронізованих бітів при досягненні ними критичної довжини [1]. Реалізація механізму бітового стаффінгу потребує передачі додаткових бітів переривання послідовностей несинхронізованих бітів, а також часу для аналізу та вилучення з прийнятого блоку даних бітів переривання. Причому, час на розпізнавання і видалення біту переривання на боці приймача витрачається завжди.

Для виправлення одиночної помилки синхронізації використовуються спеціальні методи, більшість яких має за основу використання ло-

гічних та арифметичних зважених контрольних сум [4,5].

Зокрема, в методі [4] для корекції одиночної помилки синхронізації використовується двокомпонентний контрольний код. Перша бітова компонента розраховується як сума за модулем два всіх бітів інформаційного блоку. Значення другої компоненти контрольної суми розраховується як арифметична сума порядкових номерів усіх позицій, на яких у вихідному блоці знаходяться одиничні біті.

Виявлення факту передачі інформації з помилкою, визначення її типу, локалізація та корекція відбувається на приймачі шляхом порівняння та аналізу кількості отриманих біт у інформаційному повідомленні з очікуваною кількістю, а також розрядів контрольних кодів: отриманого від передавача, та розрахованого на приймачеві. По різниці перших компонент контрольного коду виконується класифікація типу помилки синхронізації (збільшення чи зменшення довжини прийнятої на боці приймача послідовності відносно переданої), а різниця других компонентів контрольного коду на приймачі і передавачі локалізує послідовність несинхронізованих бітів, при прийомі якої приймач невірно визначив її довжину.

Метод, крім помилок синхронізації, забезпечує виправлення одиничних помилок в рамках моделі двійкових симетричних каналів. Проте метод не дозволяє виправляти дві помилки синхронізації. Отже, відомий метод [4] корекції помилок синхронізації має вузький клас помилок синхронізації, що можуть бути виправлені.

Як було зазначено вище, в сучасних умовах зростання частоти передачі несучих сигналів в послідовних лініях передачі даних, ймовірність виникнення двох помилок синхронізації стає цілком реальною. Крім того, відомий метод виправлення помилок синхронізації використовує звичайну арифметику, що ускладнює апаратну реалізацію корекції помилок синхронізації в порівнянні з використанням логічних операцій, які не потребують формування та обробки сигналів переносів.

Таким чином, існує необхідність в розробці спеціального методу виявлення та виправлення помилок синхронізації, кратність яких більша ніж забезпечують відомі методи.

Ціль досліджень полягає в розробці способу ефективної корекції подвійних помилок, що виникають в асинхронних каналах обміну цифровими даними між компонентами комп'ютерних систем.

Метод корекції подвійних помилок синхронізації

Для досягнення поставленої мети пропонується метод корекції пари помилок синхронізації. Розроблений метод передбачає наступну модель виникнення помилок синхронізації при передачі n-бітового блоку $B = \{b_1, b_2, ..., b_n\}$, $\forall k \in \{1..n\} : b_k \in \{1,0\}$ даних в асинхронному каналі. В сучасних асинхронних каналах обміну даними синхронізуються лише біти одного знаку – одиниці чи нулі [1]. Задля визначеності, без втрати загальності, можна вважати, що не синхронізована передача одиничних бітів, як це, зокрема, має місце в каналах USB [1]. Тоді помилка синхронізації полягає в тому, що при несинхронізованій передачі фрагменту, що складається з j одиниць, кількість їх на стороні передавача та приймача можуть відрізнятися. На практиці ця різниця становить не більше одиниці [2]. Ймовірність виникнення помилки синхронізації зростає зі збільшенням довжини l фрагменту одиниць, що слідують підряд. Якщо довжина l фрагменту досягає певної межі h, ймовірність виникнення помилки синхронізації стає критичною, тобто такою, що виникає реальний ризик виникнення помилки, що зумовлює необхідність в спеціальних засобах для її недопущення чи виправлення. Зокрема, для послідовного порту USB такою критичною межею є значення h =6 [1]. Конкретне значення критичної межі h довжини фрагменту залежить від способу передачі даних, характеристик каналу, швидкості передачі та вимог до надійності, зумовлених специфікою застосування.

Розроблений метод корекції помилок синхронізації передбачає виділення всіх фрагментів, що складаються не менш як з h-1 одиниць, в блоці даних B та їх нумерацію, починаючи з одиниці. Виділені вказаним чином фрагменти позначаються як $E_1, E_2, ..., E_m$, де m-кількість фрагментів одиниць довжиною більшою, ніж h-2 в n-бітовому блоці B. Кількість одиниць в цих фрагментах відповідно дорівнює: l_{1s}, $l_{2s}, ..., l_{ms}$: $\forall i=1,...,m$: $l_{is} \geq h$-1.

Під час передачі i-го фрагменту E_i, $i \in \{1,...,m\}$, що складається з l_{is} одиниць можливі два типи помилок: втрата одиничного біту, і, як наслідок, отримання приймачем фрагменту, що має на біт меншу довжину відносно відпра-

вленого, тобто $l_{ir}= l_{is}$ - 1, та поява зайвого одиничного біту, що призводить до збільшення отриманого фрагменту відносно відправленого на один біт, тобто $l_{ir}= l_{ir}+ 1$.

В обох випадках необхідно виявити факт появи помилки, визначити її точне місце розташування та виконати корекцію відповідно тому, з'явився зайвий біт чи зник значущий.

Для вирішення поставленої задачі пропонується застосовувати контрольний код, який складається η+1 η-бітових компонентів C_0, $C_1,...,C_\eta$ та η бітових компонентів ν_0, $\nu_1,...,\nu_\eta$, $\eta=\lceil \log_2 m \rceil$, $\forall j\in\{0,1,...,\eta\}$: $\nu_j\in\{0,1\}$, де $\lceil x \rceil$ - найменше ціле, що більше або дорівнює x.

Формування контрольного коду на стороні передавача і стороні приймача пропонується виконувати наступним чином. Нульова компонента C_0 обчислюється як сума за модулем два порядкових номерів послідовностей одиниць, довжина яких більша за h-2 і є непарною:

$$C_0 = \bigoplus_{j=1}^{m} j \cdot l_j \bmod 2 . \qquad (1)$$

Бітову компоненту ν_0 пропонується обчислювати як суму за модулем 2 других розрядів коду довжини всіх послідовностей одиниць, в яких міститься більше ніж h-2 одиниць:

$$\upsilon_0 = \bigoplus_{j=1}^{m} (l_j / 2) \bmod 2 . \qquad (2)$$

Кожна j-та з компонент $C_1,C_2,...,C_\eta$, j=1,2,...,η, обчислюється як сума за модулем два таких порядкових номерів послідовностей одиниць, довжина яких більша за h-2 і є непарною, при тому, що їх j-тий розряд дорівнює одиниці. Тобто, якщо представити номер i послідовності одиниць у двійковій системі: $i=i_1 + i_2\cdot 2 + i_3\cdot 2^2 +...+ i_m\cdot 2^{m-1}$, де то компоненти $C_1,C_2,...,C_\eta$ контрольного коду обчислюються за наступною формулою:

$$\forall j = 1,2,...,\eta : C_j = \bigoplus_{i=1}^{m} i \cdot i_j \cdot l_j \bmod 2 . \qquad (3)$$

Кожна j-та з бітових компонентів $\nu_1,\nu_2,...,\nu_\eta$, j=1,2,...,m, обчислюється як сума за модулем два других розрядів коду довжини всіх послідовностей одиниць, j-тий розряд порядкових номерів яких дорівнює одиниці:

$$\forall j = 1,2,...,\eta : \upsilon_j = \bigoplus_{i=1}^{m} i_j \cdot (l_i / 2) \bmod 2 . \qquad (4)$$

Запропонований порядок формування контрольних кодів блоку даних може бути ілюстровано наступним прикладом. Можна припустити, що критична межа h існування реального ризику помилки синхронізації дорівнює h=6. Передається 64-бітовий блок B={1101 1011 0**111 1111 111**0 0111 10**11 111**0 **1111 111**0 0011 0111 0**111 111**0 00**11 1111**}. Згідно з наведеним вище, виділяються послідовності одиниць, довжина яких не менша ніж h-1=5. Блок B містить 5 (m=5) таких послідовностей E_1, $E_2,...,E_5$, що виділені жирним шрифтом, кількість одиниць в яких відповідно становить l_1 = 10, l_2 = 5, l_3 = 7, l_4 = 6 та l_5 = 6. Розрядність η номерів виділених послідовностей одиниць становить $\eta=\lceil \log_2 m \rceil = \lceil \log_2 5 \rceil$ = 3. Компонента C_0 обчислюється як сума по модулю 2 порядкових номерів послідовностей, довжина яких непарна : $C_0 = 2 \oplus 3 =1$. Компонента C_1 обчислюється як сума порядкових номерів послідовностей непарної кількості одиниць молодший розряд котрих дорівнює одиниці. Оскільки тільки номер $3=11_2$ має одиницю в молодшому розряді, то : $C_1 = 3$. Аналогічно, $C_2 = 2 \oplus 3 = 1$, $C_3 = 0$. Нульова бітова компонента ν_0 обчислюється як сума за модулю 2 других розрядів двійкового представлення довжин ($l_1=10=10\mathbf{1}0_2$, $l_2=5=1\mathbf{0}1_2$, $l_3 =7=1\mathbf{1}1_2$, $l_4=6=1\mathbf{1}0_2$, $l_5=6=1\mathbf{1}0_2$) всіх виділених послідовностей: $\nu_0 = 1\oplus 0 \oplus 1 \oplus 1 \oplus 1 = 0$. Перша бітова компонента ν_1 обчислюється як сума за модулю 2 других розрядів двійкового представлення довжин лише тих з виділених послідовностей, молодший розряд порядкових номерів яких дорівнює одиниці, тобто 1, 3 та 5-ї послідовностей: $\nu_1 = 1\oplus 1 \oplus 1 = 1$. Друга бітова компонента ν_2 обчислюється як сума за модулю 2 других розрядів двійкового представлення довжин лише тих з виділених послідовностей, другий розряд порядкових номерів яких дорівнює одиниці, тобто 2 та 3-ї послідовностей: $\nu_2 = 0 \oplus 1 = 1$. Аналогічно, третя бітова компонента ν_2 обчислюється як сума за модулю 2 других розрядів довжин послідовностей, номер яких має одиничний третій розряд, тобто 4-ї та 5-ї послідовностей: $\nu_3 = 1 \oplus 1 = 0$.

На передавачеві обчислення контрольних кодів виконується по даним блоку B_S даних, що передаються, а на приймачі, відповідно, по прийнятому блоку B_R даних. Відповідно, контрольні коди, обчислені на стороні передавача позначаються індексом S, а обчислені на боці приймача - індексом R.

Після обчислення по прийнятому блоку B_R контрольних кодів $C_{0R}, C_{1R}, ..., C_{\eta R}$ та ν_{0R}, ν_{1R},..., $\nu_{\eta R}$ на боці приймача також обчислюються різниці відповідних кодів передавача та приймача згідно з наступної формули:

$$\forall j = 0,1,..,\nu : \Delta_{jR} = C_{jS} \oplus C_{jR}, \delta_j = \upsilon_{jS} \oplus \nu_{jR} \quad (5)$$

При виникненні двох помилок синхронізації, для визначеності, при передачі p-ї та q-ї послідовностей одиниць, причому, $q>p$, значення різниць контрольних кодів можуть бути представлені у наступному вигляді:

$$\Delta_0 = p \oplus q, \quad \delta_0 = \xi_p \oplus \xi_q$$
$$\forall j = 1,2,..,\eta : \Delta_j = p \cdot p_j \oplus q \cdot q_j, \quad (6)$$
$$\delta_j = p_j \cdot \xi_p \oplus q_j \cdot \xi_q$$

де p_j та q_j значення j-го розрядів відповідно номерів p і q, а ξ_p та ξ_q - біти, що дорівнюють одиниці, якщо змінилися другі розряди довжини відповідно p-тої і q=тої послідовностей одиниць. Формально: $\xi_p = (l_{pS} \oplus l_{pR})/2 \bmod 2$ і $\xi_q = (l_{qS} \oplus l_{qR})/2 \bmod 2$.

Наведене, в рамках розглянутого вище прикладу, може бути ілюстровано наступним чином. Нехай, в процесі передачі блоку на боці приймача були невірно сприйняті довжини першої (p=1) та п'ятої (q=5) послідовностей одиниць, передача яких не синхронізована, причому перша послідовність сприйнята приймачем як така, що складається з 9-ти одиниць, а п'ята - як така, що містить 7 одиниць. $l_{1R} = 9 = 10\mathbf{0}1_2$, $l_{2R} = 5=1\mathbf{0}1_2$, $l_{3R} = 7=1\mathbf{1}1_2$, $l_{4R} = 6=1\mathbf{1}0_2$ та $l_{5R} = 7=1\mathbf{1}1$. Очевидно, що для першої із послідовностей довжина на передавачі (l_{1S} =10 = $10\mathbf{1}0_2$) і приймачі ($l_{1R} = 9 = 10\mathbf{0}1_2$) змінилася у другому розряді, так, що $\xi_p = \xi_1 = 1$. Другий розряд довжини 5-ї послідовності на передавачі ($l_{5S} = 6=1\mathbf{1}0_2$) та приймачі ($l_{5R} = 7=1\mathbf{1}1_2$) не змінився: відповідно $\xi_q = \xi_5 = 0$.

Відповідно, контрольні коди на стороні приймача обчислюються наступним чином. $C_{0R} = 1\oplus 2 \oplus 3 \oplus 5 = 5$, $C_{1R} = 1 \oplus 3 \oplus 5 = 7$, $C_{2R} = 2$ та $C_{3R} = 5$. Нульова бітова компонента контрольного коду на приймачі має такий вигляд: $\nu_{0R} = 0 \oplus 0 \oplus 1 \oplus 1 \oplus 1 = 1$. Перша бітова компонента обчислюється на приймачі як сума за модулем 2 других бітів кодів довжин виділених одиничних послідовностей номери яких мають одиницю в молодшому розряді, тобто 1-ї, 3-ї та 5-ї послідовностей: $\nu_{1R} = 0 \oplus 1 \oplus 1 = 1$, аналогічно друга бітова компонента обчислюється як сума за модулем 2 других розрядів довжин одиничних послідовностей, номери яких мають одиницю у другому розряді, тобто 2-ї та 3-ї послідовностей: $\nu_{2R} = 0 \oplus 1 = 1$; третя бітова компонента дорівнює сумі за модулем 2 других розрядів довжин 4-ї та 5-ї послідовностей, оскільки тільки їх номери мають одиницю у третьому розряді: $\nu_{3R} = 1 \oplus 1 =1$.

Різниці контрольних кодів передавача та приймача, обчислені на боці останнього мають такий вигляд:

$\Delta_0 = C_{0S} \oplus C_{0R} = 1 \oplus 5 = p \oplus q = 4$
$\Delta_1 = C_{1S} \oplus C_{1R} = 3 \oplus 7 = p \oplus q = 4$
$\Delta_2 = C_{2S} \oplus C_{2R} = 2 \oplus 2 =0$
$\Delta_3 = C_{3S} \oplus C_{3R} = 0 \oplus 5 = q = 5$

Різниці бітових компонентів контрольного коду передавача та приймача, обчислені на боці останнього мають такий вигляд:

$\delta_0 = \nu_{0S} \oplus \nu_{0R} = 0 \oplus 1 = 1 = \xi_p \oplus \xi_q = 1$
$\delta_1 = \nu_{1S} \oplus \nu_{1R} = 1 \oplus 0 = 1 = \xi_p \oplus \xi_q = 1$
$\delta_2 = \nu_{2S} \oplus \nu_{2R} = 1 \oplus 1 = 0$
$\delta_3 = \nu_{3S} \oplus \nu_{3R} = 1 \oplus 1 = 0 = \xi_q = 0$

Отримані на боці приймача різниці контрольних кодів (6) являють собою системи булевих рівнянь, які дозволяють однозначно визначити значення порядкових номерів спотворених при асинхронній передачі послідовностей одиниць - p і q, а також визначити значення бітів ξ_p і ξ_q , які, в свою чергу, дають змогу визначити тип кожної з помилок синхронізації.

Виходячи з того, що $q > p$ завжди існує таке $1\leq g \leq \eta$, що в розрядах старших за g номери p та q однакові, а g-тий розряд p менший за g-тий розряд коду q, тобто: $\forall\ 1\leq i<g : p_i = q_i, p_g < q_g$, або $p_g = 0$ і $q_g = 1$. Це означає, що g-те рівняння системи (6) завжди має вигляд $\Delta_g = q$, а так як $\Delta_0 = p\oplus q$, то значення p може бути визначено у вигляді $p= \Delta_0 \oplus \Delta_g$. Для $\forall\ 1\leq i<g : p_i = q_i$, і, відповідно, $\Delta_i = 0$, якщо $p_i=q_i=0$ або $\Delta_i = p \oplus q = \Delta_0$, якщо $p_i=q_i=1$. Так, для наведеного прикладу $g = 3$, відповідно, $\Delta_3 = q = 5$.

Визначення значень p та q - номерів порушених при передачі послідовностей одиниць дає змогу визначити тип помилки. Дійсно, оскільки, згідно з (6) $\delta_g = p_g \cdot \xi_p \oplus q_g \cdot \xi_q = \xi_g$, то значення біту $\xi_q = \delta_g$. Відповідно, значення $\xi_p= \delta_0 \oplus \xi_q = \delta_0 \oplus \delta_g = 1\oplus 0 = 1$. Якщо позначити через γ_p молодший розряд довжини p-ї послідовності одиниць на приймачі: $\gamma_p = l_{pR} \bmod 2$, то пара значень ξ_p, γ_p однозначно визначають тип помилки синхронізації при передачі p-ї послідов-

ності одиниць. В таблиці 1 наведено всі можливі ситуації зміни довжини несинхронізованої послідовності при її аналізі на боці приймача. В таблиці 1 символом d позначено операцію зміни довжини послідовності при її помилковому сприйнятті на боці приймача.

Табл. 1. Типи помилок синхронізації

l_S mod 4	d	l_R mod 4	ξ_p	γ_p
0 0	+1	0 1	0	1
0 1	+1	1 0	1	0
1 0	+1	1 1	0	1
1 1	+1	0 0	1	0
0 0	-1	1 1	1	1
0 1	-1	0 0	0	0
1 0	-1	0 1	1	1
1 1	-1	1 0	0	0

Аналіз даних, наведених в таблиці 1 дозволяє зробити наступний висновок.

Якщо $\xi_p \oplus \gamma_p = 1$, то довжина p-тої послідовності на приймачеві сприймається збільшеною на одиницю, тобто $l_{pR} = l_{pS}+1$. В протилежному випадку, якщо $\xi_p \oplus \gamma_p = 0$, довжина p-тої несинхронізованої послідовності одиниць сприймається на стороні приймача на одиницю меншою: $l_{pR} = l_{pS}$ -1. В прикладу, що розглядається, для першої послідовності $l_{1R} = 9 = 10\mathbf{0}1_2$ тобто $\gamma_p = 1$. Оскільки $\xi_p = \xi_1 = 1$, то $\xi_p \oplus \gamma_p = 0$; відповідно, при передачі першої послідовності її довжина на боці приймача сприйнята на одиницю меншою. Аналогічним чином визначається тип помилки і для q-тої послідовності несинхронізованих при передачі одиниць.

Запропонована процедура корекції двох помилок синхронізації формально може бути представлена у вигляді наступної послідовності дій.

1. Якщо різниці всіх компонентів контрольного коду передавача і приймача дорівнюють нулю, тобто за умови $\forall j \in \{0,1,...,\eta\}$: $\Delta_j=0$ і $\delta_j=0$, вважається, що блок передано без помилок. Перехід на п.6.

2. Якщо всі різниці $\Delta_1, \Delta_2,...,\Delta_\eta$ компонентів контрольного коду передавача і приймача приймають тільки два значення: 0 та Δ_0, тобто, якщо $\forall j \in \{1,2,..,\eta\}$: $\Delta_j \in \{0,\Delta_0\}$, то має місце одна помилка синхронізації, причому порядковий номер p послідовності одиниць, довжина якої невірно сприйнята приймачем дорівнює Δ_0: $p = \Delta_0$. Корекція цієї послідовності виконується наступним чином: визначається молодший розряд γ_p коду довжини послідовності p на стороні приймача: якщо $\xi_p \oplus \gamma_p = \delta_0 \oplus \gamma_p = 0$, то до послідовності додається одиниця, а якщо $\delta_0 \oplus \gamma_p = 1$, то довжина послідовності зменшується на одиницю. Перехід на п.6.

3. Лічильник i розрядів різниць контрольних кодів встановлюється в η: $i = \eta$.

4. Якщо $\Delta_i = \Delta_0$ або $\Delta_i = \Delta_0$ перехід на п. 5, інакше порядковий номер q другої з пошкоджених при передачі послідовностей одиниць обчислюється як $q = \Delta_i$, а номер p першої невірно прийнятої на боці передавача послідовності одиниць визначається як $p = \Delta_0 \oplus q = \Delta_0 \oplus \Delta_i$. Корекція цієї послідовності виконується наступним чином: визначається молодший розряд γ_p коду довжини послідовності з порядковим номером p на стороні приймача; визначається біт зміни передостаннього розряду довжини послідовності $\xi_p = \delta_0 \oplus \delta_i$. Якщо $\xi_p \oplus \gamma_p = \delta_0 \oplus \delta_\eta \oplus \gamma_p = 0$, то до послідовності додається одиниця, а якщо $\xi_p \oplus \gamma_p = \delta_0 \oplus \delta_i \oplus \gamma_p = 1$, то довжина послідовності p зменшується на одиницю. Корекція другої послідовності несинхронізованих при передачі символів з порядковим номером q виконується аналогічним чином: визначається молодший розряд γ_q коду довжини послідовності q на стороні приймача, визначається біт ξ_q зміни передостаннього розряду довжини послідовності q: $\xi_q = \delta_i$: якщо $\xi_q \oplus \gamma_q = \delta_i \oplus \gamma_q = 0$, то до послідовності q додається одиниця, а якщо $\delta_i \oplus \gamma_q = 1$, то довжина послідовності з порядковим номером q зменшується на одиницю. Перехід на п. 6.

5. Виконується декремент значення лічильника i розрядів різниць контрольних кодів: $i = i$-1 і повернення на п.4.

6. Кінець процедури корекції.

В рамках наведеного вище прикладу робота викладеної процедури корекції може були проілюстрована наступним чином.

Оскільки значення компонентів контрольного коду $\Delta_0 = 4$, $\Delta_1 = 4$, $\Delta_2 = 0$, $\Delta_3 = 5$ та $\delta_0=1$, $\delta_1=1$, $\delta_2=0$ та $\delta_3=0$ не дорівнюють нулю, то мають місце помилки синхронізація.

Згідно з п.2 виконується перевірка того, що всі $\Delta_1=4$, $\Delta_2=0$, $\Delta_4=5$ належать множині $\{0,4\}$, яка показує, що ця умова не виконується - це значить, що має місце помилка синхронізації кратності більше ніж один.

Згідно п. 3 лічильник $i = \eta = 3$.

Згідно п.4 виконується перевірка $\Delta_i = \Delta_3$ на рівність нулю або Δ_0. Оскільки $\Delta_3 \neq \Delta_0$ і $\Delta_3 \neq 0$,

то порядковий номер q другої з пошкоджених при передачі послідовностей одиниць обчислюється як $q = \Delta_3 = 5$. Номер p першої невірно прийнятої на боці передавача послідовності одиниць визначається як $p = \Delta_0 \oplus q = 4 \oplus 5 = 1$. Корекція цієї послідовності виконується наступним чином: визначається молодший розряд γ_p коду l_{1R}=9: γ_p =1; визначається біт ξ_1 зміни передостаннього розряду довжини послідовності $\xi_1 = \delta_0 \oplus \delta_3 = 1$. Так як $\xi_1 \oplus \gamma_1 = 1 \oplus 1 = 0$, то до послідовності з номером 1 на боці приймача додається одиниця. Визначається молодший розряд γ_5 коду довжини l_{5R}= 7 послідовності з номером q=5 на стороні приймача: γ_5=1; визначається біт ξ_5 зміни передостаннього розряду довжини послідовності з номером 5 : $\xi_5 = \delta_3 = 0$. Оскільки $\xi_5 \oplus \gamma_q = 0 \oplus 1 = 1$, то довжина послідовності з порядковим номером q=5 збільшується на одиницю. Отже, обидві помилки синхронізації зкореговано.

Таким чином, для досягнення поставленої цілі: корекції двох помилок синхронізації в асинхронних каналах розроблено метод, оснований на використанні позиційних корегуючих сум.

Аналіз ефективності

Основною перевагою запропонованого методу корекції помилок синхронізації в порівнянні з існуючими є те, що він дозволяє розширити клас помилок, що гарантовано виправляються забезпечуючи вправлення двох помилок синхронізації на противагу відомим методам [3-5], які розраховані на корекцію лише однієї помилки.

Кількість контрольних розрядів k, що додатково передаються визначається довжиною n блоку даних. Якщо припустити, що в блоці B даних з рівною ймовірністю зустрічаються нулі і одиниці, то можна вважати, що послідовність бітів блоку генерується за допомогою LFSR (Linear Feedback Shift Register) розрядністю r ($n \geq 2^r$). Тоді математичне очікування кількості послідовностей з i слідуючих підряд одиниць, $i \in \{1,..,r\}$, визначається як сума кількості комбінацій при яких послідовність знаходиться на початку чи в кінці блоку: $2 \cdot 2^{r-i-1} = 2^{r-i}$ та кількості комбінацій коли i-бітова послідовність, обмежена зліва і справа нулями, знаходиться посередині послідовності: $(n-r-2) \cdot 2^{r-i-2}$. Враховуючи, що при послідовностях, які складаються з більш ніж h-1 одиниць, значення i змінюється від h-1 до n, математичне очікування s числа послідовностей одиниць, довжиною від h-1 до n визначається як сума:

$$s = 3 + \sum_{j=2}^{r-h} 2^{j-2} \cdot (j+3). \qquad (7)$$

Відповідно, розрядність ρ номеру послідовності в блоці даних становить $\lceil \log_2 s \rceil$, а загальна кількість k розрядів контрольного коду визначається формулою:

$$k = \rho \cdot (\rho + 1) = \lceil \log_2 s \rceil \cdot (\lceil \log_2 s \rceil + 1). \qquad (8)$$

Наприклад, при типовому значенні h=6, контролюється передача блоку довжиною 256 байт (n=2048), то значення r = 11, середня кількість послідовностей бітів, передача яких несинхронізована становить s=112, відповідно розрядність порядкового номеру послідовностей ρ = 7, а загальна кількість k контрольних розрядів дорівнює k = 56, що становить 2.7 % від розміру блока. За цих умов, використання бітового стаффінгу потребує передачі s=112 додаткових бітів, що становить 5.5 % об'єму блоку. Таким чином, запропонований метод потребує приблизно вдвоє меншого об'єму даних, що передаються разом з даними.

Ефективність запропонованого методу зростає зі збільшенням довжини блоку. Так при передачі блоку довжиною 1 Кбайт (n=8192), r = 13, середня кількість s послідовностей бітів, передача яких несинхронізована дорівнює s= 576, відповідно розрядність порядкового номеру послідовностей ρ = 10, розрядність контрольного коду становить k = 110, що становить 1.3 % від розміру блока.

Важливою перевагою запропонованого методу в порівнянні з відомими є простота та використання логічних операцій на противагу арифметичним. Як слідує з наведеного вище формалізованого опису процедури корекції, для локалізації послідовностей, довжина яких невірно сприйнята на боці приймача, середня кількість циклів дорівнює математичному очікуванню m_ρ числа старших розрядів, значення яких співпадають для двох випадкових ρ-розрядних чисел:

$$m_\rho = \sum_{j=1}^{\rho-1} \frac{j}{2^j} < \sum_{j=1}^{\infty} \frac{j}{2^j} = 2. \qquad (9)$$

Відповідно, в середньому, для локалізації двох помилок потребується всього дві операції порівняння і ще одна операція додавання для знаходження позиції другої невірно сприйнятої на приймачі послідовності. Отже, запропонована процедура корекція гранично проста в реалізації і кількість операцій для локалізації поми-

лок практично не залежить від довжини блоку. Базовою операцією розробленої процедури корекції є операція логічного додавання (XOR), яка ефективно реалізується апаратними засобами і, зокрема, ПЛІС.

Важливою перевагою запропонованого методу є також те, що він дозволяє шляхом незначної модифікації викладеної вище процедури корегувати помилки синхронізації більшої кратності. Зокрема, експериментально доведено, що метод дозволяє корегувати до 99.2% потрійних помилок, 82% 4-х помилок синхронізації і близько 76% п'ятикратних помилок синхронізації.

Проведені експериментальні дослідження апаратної реалізації запропонованого методу з використанням VHDL-моделей довели, що його простота невисока часова складність дозволяють корегувати помилки синхронізації практично в темпі передачі даних.

Висновки

В результаті проведених досліджень запропоновано метод корекції подвійних помилок синхронізації, що виникають в послідовних асинхронних каналах.

Відмінністю розробленого та дослідженого метод корекції двократних помилок синхронізації в асинхронних каналах обміну даними, полягає в тому, що локалізація помилково сприйнятих на боці приймача послідовностей бітів, передача яких не синхронізується, виконується на основі аналізу сум підмножин порядкових номерів послідовностей критичної довжини, що дозволяє розширити клас помилок, що виправляються в порівнянні з відомими методами, які орієнтовані на виправлення лише однократної помилки синхронізації.

На відміну від відомих, запропонований метод не тільки гарантовано виправляє одно і двократні помилки синхронізації, але й дозволяє виправляти значну частину помилок синхронізації більшої кратності.

Важливою перевагою запропонованого методу в порівнянні з відомими [4] є те, що він значно простіший і орієнтований на логічні операції, що забезпечує високу ефективність апаратної реалізації на ПЛІС.

Запропонований метод орієнтовано для швидкісних послідовних каналів обміну цифровими даними між компонентами комп'ютерних систем широкого призначення.

Список посилань

1. Агуров П.В. Интерфейс USB. Практика использования и программирования / П.В. Агуров - СПб.:БХВ-Петербург, 2005.- 576 с.
2. Klove T. Error Detecting Codes: General Theory and Their Application in Feedback Communication Systems / T. Klove, V. Korzhik.- Norwell, MA: Kluwer, 1995. – 433 p.
3. Марковский А.П. Использование взвешенных контрольных сумм для обнаружения ошибок в линиях передачи с асинхронным кодированием данных/ А.П. Марковский , Пуя Солеймани Нежадиан, Мулки Ахмед Яссин Ал Бадайнех. //Современные информационные и электронные технологи: 9-тая междунар. науч.-техн. конф., 19-22 мая. 2008 р.: тезисы докл. - Одесса., 2008.- С.201.
4. Марковський О.П., Федоречко О.І., Коротенко А.А. Метод корекції одиночної помилки синхронізації в асинхронних лініях передачі цифрових даних // Вісник Національного технічного університету України "КПІ" Інформатика, управління та обчислювальна техніка, – Київ: ВЕК+ – 2012 – № 57. с. 70-75.
5. Fedorechko O., Markovskyi O.P., Doukas N., Bardis N. Synchronization Error Detection of Data Transmission Errors in Asynchronous Channels // Recent Advances in Electrical Engineering Series -37. Latest trends on Systems.- Vol.1.- Proceeding of the 18-th International Conference on Systems - CSCC-14. - Santorini Island, Greece., July 17-21. 2014.- ISSN: 1790-5117, ISBN: 978-1-61804-243-9. P.179-183.

УДК 004.77

РОКОВОЙ А.П.,
ГАБИНЕТ А.В.

СПОСОБ ПРОГНОЗИРОВАНИЯ ДИСПЕРСИИ ЗАДЕРЖКИ ПРИ ПЕРЕДАЧЕ ГОЛОСОВЫХ ДАННЫХ

Рассмотрены существующие способы прогнозирования трафика в IP сетях. Предложен способ прогнозирования дисперсии задержки при передаче голосовых данных на основе сингулярно-спектрального анализа временного ряда. Определены критерии оценки качества способа прогнозирования. Выполнено сравнение предложенного способа прогнозирования с прогнозом на основе модели ARIMA для стационарной и мобильной сети.

Existing methods of traffic forecasting in IP networks were reviewed. The method of jitter forecasting based on singular spectrum analysis of time series is proposed. The criteria for accuracy evaluation are determined. The forecasting accuracy of the proposed method was compared with the forecasting based on ARIMA model for fixed and mobile networks.

1. Введение

Передача голосовых данных в IP сети требует соблюдения определенных ограничений, связанных с задержкой передачи и процента потери пакетов. Желательно, чтобы задержка передачи в одну сторону не превышала 150 мс, и количество теряемых пакетов не превышало 4 % [1]. В большинстве случаев эти параметры находятся в обратной зависимости, то есть чем меньше задержка пакетов может быть обеспечена, тем больше процент отбрасываемых (теряемых на конечном узле) пакетов. Обратное утверждение также справедливо. Связано это с наличием на приемном узле джиттер-буфера, который компенсирует флуктуацию задержки прибывающих пакетов.

Чем больше флуктуация задержки, тем больше размер джиттер-буфера требуется для компенсации, иначе часть пакетов будет отброшена, если пакеты придут позже времени воспроизведения. При максимальном размере джиттер-буфера появляется возможность свести количество отбрасываемых пакетов к минимуму, но при этом увеличивается время задержки. При минимальном размере джиттер-буфера время задержки уменьшается, но при этом увеличивается количество отбрасываемых пакетов.

Размер джиттер-буфера определяется значением дисперсии времени задержки прибывающих пакетов. Для сетевого трафика характерно изменение этого параметра во времени [2]. Следовательно, размер джиттер-буфера должен меняться во времени по алгоритму, учитывающему текущее состояние сети.

Существуют различные способы адаптивного регулирования джиттер-буфера [3-8]. В работе [2] предложено использовать способ на основе прогнозирования дисперсии задержки. Он позволяет поддерживать размер джиттер-буфера на минимальном уровне для компенсации флуктуации времени задержки прибывающих пакетов, обеспечивая допустимое количество их потерь.

Проблеме прогнозирования сетевого трафика посвящен ряд работ [9,10]. В них даны общие рекомендации по выбору и использованию моделей прогнозирования сетевого трафика. Недостатком этих моделей является то, что они не учитывают особенностей прогнозирования его временных параметров, следовательно, не обеспечивают требуемую точность прогноза.

Таким образом, целью настоящей работы является повышение точности прогноза временных параметров сетевого трафика за счет разработки специального способа прогнозирования.

2. Способ прогнозирования временных параметров трафика

Для построения качественного прогноза необходима разработка адекватных моделей, учитывающих особенности прогнозирования временных параметров трафика. В работе [2] показано, что временной ряд задержек передачи пакетов является процессом с медленно убывающей зависимостью (МУЗ), то есть не является стационарным. Это свойство ряда характерно для большей части случаев, наблюдаемых в IP сетях при передаче голосовых данных. Поэтому предлагается использовать мо-

дель на основе сингулярно-спектрального анализа (ССА), которая позволяет строить прогноз для нестационарных временных рядов [11].

Способ прогнозирования на основе сингулярно-спектрального анализа включает следующие действия.

1. В исходном временном ряду выделяется N последних значений:

$$\{x_i\}_{i=1}^{N} \quad (1)$$

2. Выбирается некоторое значение M (длина окна), такое что $M < N$, и строится траекторная матрица X по формуле 2.

$$X = (x_{ij})_{i,j=1}^{k,M} = \begin{pmatrix} x_1 & x_2 & x_3 & \cdots & x_M \\ x_2 & x_3 & x_4 & \cdots & x_{M+1} \\ x_3 & x_4 & x_5 & \cdots & x_{M+2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_k & x_{k+1} & x_{k+2} & \cdots & x_N \end{pmatrix} \quad (2)$$

3. Выполняется сингулярное разложение траекторной матрицы исходного ряда. Для этого определяется корреляционная матрица R по формуле 3.

$$R = \frac{1}{k}\sum_{i=1}^{k}\frac{1}{s_i s_j}(x_{i+l-1} - \bar{x}_i)\left(x_{j+l-1} - \bar{x}_j\right) \quad (3)$$

где:

$\bar{x}_i$ – среднее арифметическое значение по столбцам траекторной матрицы, которое определяется по формуле 4.

s_i – стандартное отклонение, которое определяется по формуле 5.

$$\bar{x}_i = \frac{1}{k}\sum_{i=1}^{k} x_{i+j-1} \quad (4)$$

$$s_j = \sqrt{\frac{1}{k}\sum_{i=1}^{k}(x_{i+j-1} - \bar{x}_j)^2} \quad (5)$$

4. Определяются собственные числа и собственные векторы, то есть выполняется разложение корреляционной матрицы R по формуле 6.

$$R = P\Lambda P^T \quad (6)$$

где:

Λ – диагональная матрица собственных чисел $\Lambda = \begin{pmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_M \end{pmatrix}$

- P – ортогональная матрица собственных векторов матрицы R,

$$P = (p_1, p_2, \dots, p_M) = \begin{pmatrix} p_{11} & p_{21} & \cdots & p_{M1} \\ p_{12} & p_{22} & \cdots & p_{M1} \\ \vdots & \vdots & \ddots & \vdots \\ p_{1M} & p_{2M} & \cdots & p_{MM} \end{pmatrix}$$

5. Определяются главные компоненты Y по формуле 7 и выполняется отбор значимого количества r.

$$Y = \sum_{i=1}^{k}\frac{1}{s_i}(x_{i+l-1} - \bar{x}_i)\,P \quad (7)$$

6. Выполняется восстановление многомерного ряда $\tilde{X}^*$ по формуле 8.

$$\tilde{X}^* = \sum_{l=1}^{r} y_{i_r} p_{i_r}^T \quad (8)$$

7. Выполняется денормирование матрицы $\tilde{X}^*$ путем умножения на диагональную матрицу S, состоящую из стандартных отклонений и децентрирование путем добавления к элементам матрицы соответствующих средних значений по формуле 9.

$$\tilde{X} = \tilde{X}_0^* + \sum_{l=1}^{M}\tilde{X}_l^* S = \sum_{l=0}^{M}\tilde{X}_l^* S \quad (9)$$

В результате получена исходная матрица диагональной структуры $\tilde{X}$ в виде суммы $(M+1)$ матриц.

8. Переход к исходному ряду может быть выполнен за счет усреднения по побочным диагоналям с помощью оператора усреднения по формуле 10.

$$\tilde{x}_s = \begin{cases} \frac{1}{s}\sum_{i=1}^{s}\tilde{x}_{s-i+1,i} & 1 \le s \le M, \\ \frac{1}{M}\sum_{i=1}^{M}\tilde{x}_{s-i+1,i} & M \le s \le k, \\ \frac{1}{N-s+1}\sum_{i=1}^{N-s+1}\tilde{x}_{k-i+1,i+s-k} & 1 \le s \le M. \end{cases} \quad (10)$$

9. Процедура прогнозирования на L точек для исходного ряда выполняется путем установления последнего номера элемента ряда в $N + L$ для формулы 10.

Точность предложенного способа прогнозирования определяется следующими параметрами: глубина прогноза, длина окна, выборка главных компонент. Эмпирическим способом были определены наилучшие значения этих параметров для исследуемых временных рядов: глубина прогноза – 100 точек; длина окна – по-

ловина глубины прогноза; количество главных компонент – 0,1-0,15 длины окна.

3. Экспериментальная оценка качества предложенного способа прогнозирования

Для определения качества способа прогнозирования временных параметров сетевого трафика предлагается использовать следующие критерии:

- коэффициенты переоценки K^+ и недооценки K^-, которые определяются по формулам 11 и 14 соответственно;

- средняя относительная ошибка прогноза MAPE, определяется по формуле 16.

$$K^+ = \frac{M[e_t^+]}{M[x_t]} = \frac{\sum e_t^+}{\sum x_t} \quad (11)$$

где:

e_t^+ – абсолютная ошибка прогноза, определяется по формуле 12.

$$e_t^+ = \begin{cases} e_t, \text{если } e_t \geq 0 \\ 0, \text{если } e_t < 0 \end{cases} \quad (12)$$

где:

e_t – абсолютная ошибка прогноза, определяется по формуле 13.

$$e_t = x_t - \widehat{x_t} \quad (13)$$

где:

$\widehat{x_t}$ – значение прогноза.

$$K^- = \frac{M[e_t^-]}{M[x_t]} = \frac{\sum e_t^-}{\sum x_t} \quad (14)$$

где:

e_t^- – абсолютная ошибка прогноза, определяется по формуле 15.

$$e_t^- = \begin{cases} |e_t|, \text{если } e_t < 0 \\ 0, \text{если } e_t \geq 0 \end{cases} \quad (15)$$

$$MAPE = \frac{1}{n}\sum_{t=1}^{n} \frac{|e_t|}{x_t} 100 \quad (16)$$

У лучшего способа прогнозирования временных параметров сетевого трафика должны быть минимальными и близкими по значению коэффициенты переоценки и недооценки, а также минимальной относительная ошибка прогноза.

Предложенный способ прогнозирования предлагается сравнить со способом на основе модели авторегрессии скользящего среднего – ARIMA [12], который считается одним из наиболее точных для прогнозирования сетевого трафика [13].

Модель ARIMA(p, d, q) для нестационарного временного ряда X_t представлена формулой 17.

$$\Delta^d X_t = c + \sum_{i=1}^{p} a_i \Delta^d X_{t-i} + \sum_{j=1}^{q} b_j \varepsilon_{t-j} + \varepsilon_t \quad (17)$$

где:

ε_t – стационарный временной ряд;

c, a_i, b_j – параметры модели;

Δ^d – оператор разности временного ряда порядка d;

p – порядок авторегрессии;

q – порядок скользящего среднего.

Для построения прогноза дисперсии задержки пакетов с помощью модели ARIMA(p, d, q) необходимо определить значения параметров p, d, q, глубину и горизонт прогнозирования.

В работе [14] показано, что для прогнозирования объема сетевого трафика на 20 и более шагов вперед высокую точность показывает модель ARIMA(1,1,0). С учетом особенностей прогнозирования временных параметров трафика предлагается глубину и горизонт прогноза выбрать в 100 и 20 точек соответственно.

Для оценки качества предложенного способа прогнозирования были проведены эксперименты в условиях мобильной и стационарной сетей.

В первом случае голосовой трафик проходил через IP сеть, в которой один из сегментов был представлен системой наземной мобильной связи стандарта CDMA2000 с технологией передачи данных EV-DO Rev. A. Во втором случае голосовой трафик проходил через IP сеть, которая состояла только из стационарных наземных сегментов.

На одном из промежуточных узлов была запущена утилита tcpdump, захватывающая и сохраняющая в файл IP пакеты, проходящие через интерфейсы узла. На оконечных узлах с помощью стандартных средств передачи голоса в IP сети был сгенерирован трафик. Файлы с захваченными голосовыми данными переданы для анализа в приложение Wireshark [15]. В Wireshark выделен временной ряд джиттера для всех пакетов каждого потока.

На рисунке 1 сплошной ломанной показаны значения джиттера для пакетов в стационарной сети, точками – прогноз джиттера, рассчитанный с помощью предложенного способа (прогноз ССА), пунктиром – прогноз джиттера, рассчитанный с помощью модели ARIMA. Прогноз рассчитывался в итеративном режиме с усреднением со 190 до 215 пакета с анализом 100 предыдущих пакетов на горизонт в 20 пакетов.

На рисунке 2 сплошной ломанной показаны значения джиттера для пакетов в мобильной сети, точками – прогноз джиттера, рассчитанный с помощью предложенного способа (прогноз ССА), пунктиром – прогноз джиттера, рассчитанный с помощью модели ARIMA. Прогноз рассчитывался в итеративном режиме с усреднением со 154 до 175 пакета с анализом 100 предыдущих пакетов на горизонт в 20 пакетов.

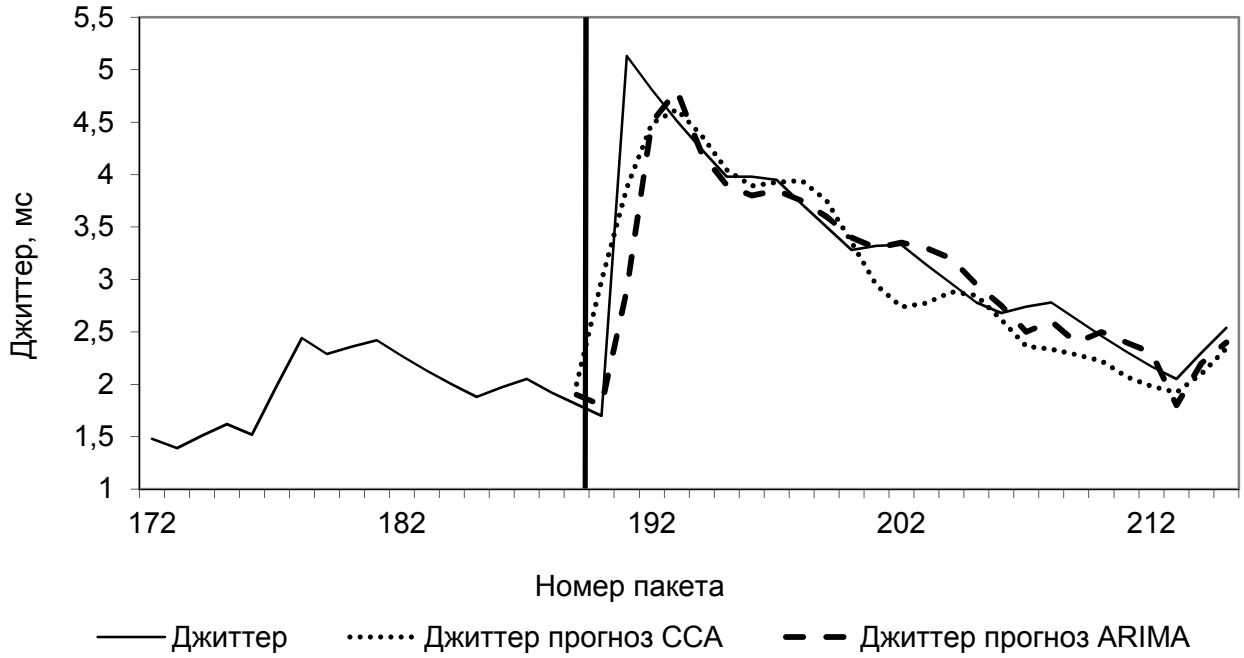


Рис.1. Сравнение методов прогнозирования ССА и ARIMA для стационарной сети

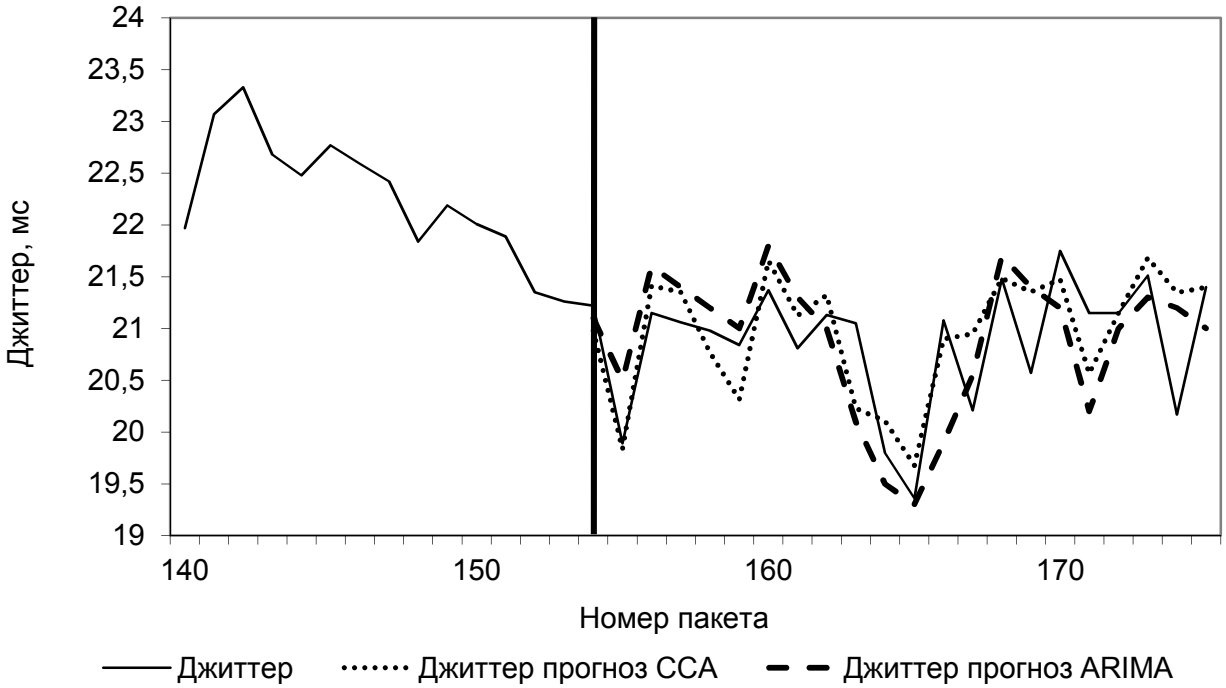


Рис. 2. Сравнение методов прогнозирования ССА и ARIMA для мобильной сети

В таблице 1 представлены коэффициенты переоценки и недооценки, а также средняя относительная ошибка для предложенного способа прогнозирования и прогноза на основе модели ARIMA для стационарной и мобильной сети.

Для стационарной сети предложенный способ прогнозирования немного уступает по точности способу на основе модели ARIMA, а в мобильной сети – превосходит его. Поэтому предложенный способ прогнозирования рекомендуется применять в мобильных сетях. В

стационарных сетях целесообразным является использование способа на основе модели ARIMA, который требуют меньше накладных расходов на реализацию.

Таблица 1. Количественные оценки способов прогнозирования

Параметр	Стационарная сеть		Мобильная сеть	
	ARIMA	CCA	ARIMA	CCA
K^+	0,061	0,062	0,101	0,085
K^-	0,062	0,062	0,095	0,081
MAPE, %	14,3	14,5	21,8	19,5

4. Заключение

Был предложен и обоснован способ прогнозирования дисперсии задержки при передаче голосовых данных на основе сингулярно-спектрального анализа временного ряда. Эмпирически определены параметры: глубина прогноза, длина окна, выборка главных компонент, которые обеспечивают требуемую точность прогнозирования. Было выполнено сравнение точности прогнозирования предложенного способа со способом на основе модели ARIMA для стационарной и мобильной сети. По результатам сравнения даны рекомендации по применению предложенного способа прогнозирования.

В качестве дальнейших исследований необходимо разработать аналитический способ определения начальных параметров прогнозирования в зависимости от характеристик временного ряда задержек передачи пакетов.

Список литературы

1. Definition of categories of speech transmission quality, ITU-T Recommendation G.109, Jan 2007.
2. Роковой А.П. Способ адаптивного регулирования джиттер буфера в VoIP // Вісник НТУУ «КПІ». Інформатика, управління та обчислювальна техніка: збірник наукових праць - К.:"Век+", 2012.- № 55, с.199-204.
3. R. Ramjee, J. Kurose, D. Towsley, H. Schulzrinne, Adaptive playout mechanisms for packetized audio applications in wide-area networks, INFOCOM ’94, Networking for Global Communications, 13th Proceedings IEEE, 1994.
4. M. Narbutt, L. Murphy, A new VoIP adaptive playout algorithm, Telecommunications Quality of Services: The Business of Success, QoS 2004, IEEE, 2004.
5. K. Fujimoto, S. Ata, M. Murata, Playout control for streaming applications by statistical delay analysis, Proceedings of IEEE International Conference on Communications (ICC), 2001.
6. C. Sreenan, J.-C. Chen, P. Agrawal, B. Narendran, Delay reduction techniques for playout buffering, IEEE Transactions on Multimedia, 2000.
7. K. Fujimoto, S. Ata, M. Murata, Adaptive playout buffer algorithm for enhancing perceived quality of streaming applications, Global Telecommunications Conference, GLOBECOM ’02, IEEE, vol. 3, 2002.
8. L. Sun, E. Ifeachor, Prediction of perceived conversational speech quality and effects of playout buffer algorithms, IEEE International Conference on Communications (ICC), vol. 1, 2003.
9. Bo Zhou, Dan He, Zhili Sun, Traffic Modeling and Prediction using ARIMA/GARCH Model, Modeling and Simulation Tools for Emerging Telecommunication Networks 2006, p. 101-121.
10. Sangjoon Jung, Chonggun Kim, Younky Chung, A Prediction Method of Network Traffic Using Time Series Models, Computational Science and Its Applications - ICCSA 2006, p. 234-243.
11. Golyandina N., Nekrutkin V., and Zhigljavsky A. Analysis of Time Series Structure: SSA and Related Techniques. London: Chapman & Hall/CRC, 2001. 305 p.
12. William W.S. Wei, Time Series Analysis: Univariate and Multivariate Methods (2nd Edition), Pearson Addison Wesley, 2006.
13. B. Zhou, D. He and Z. Sun, "Traffic modeling and prediction using ARIMA/GARCH model", Modeling and Simulation Tools for Emerging Telecommunication Networks, Springer-Verlag, New York, NY, 2006.
14. G. Rutka., Network Traffic Prediction using ARIMA and Neural Networks Models// Electronics and Electrical Engineering - Kaunas: Technologija, 2008.-№ 4(84), p. 47-52.
15. http://www.wireshark.org/

УДК 004.652.5

БУЗОВСКИЙ О.В.,
ПОДРУБАЙЛО А.А.

МЕТОДЫ И АЛГОРИТМЫ ОБЪЕДИНЕНИЯ ТАБЛИЦ ДЛЯ РАСПРЕДЕЛЕННЫХ ХРАНИЛИЩ ДАННЫХ В ОПЕРАТИВНОЙ ПАМЯТИ

В данной статье приведен обзор существующих методов распределенного объединения таблиц. Предложен метод и алгоритм модифицированного путевого объединения, оптимизированный для работы с распределенными хранилищами данных в оперативной памяти. Данный метод отличается от известных аналогов тем, что минимизирует объем пересылаемых по сети данных.

This paper provides an overview of existing distributed join methods and proposes modified track join method and algorithm, optimized for usage with in-memory data grids. This method differs from known analogs minimizing data size sent across the network.

Введение

В последнее десятилетие, в связи с постоянно повышающимися требованиями к скорости доступа к данным, широкое распространение получили распределенные хранилища данных в оперативной памяти (in-memory data grid, IMDG). Первоначальная идеология таких хранилищ предусматривала выборку данных из IMDG исключительно методом доступа по ключу (т.н. NoSQL подход). Также была предусмотрена возможность выполнения пользовательских функций непосредственно на серверах хранилища. Однако опыт применения таких хранилищ показал необходимость выполнения SQL-подобных запросов. На текущий момент в наиболее распространенных реализациях IMDG (VmWare Gemfire, Oracle Coherence, Hazelcast) частично поддерживается объектный язык запросов (object query language, OQL), но ни одна из реализаций распределенных хранилищ в оперативной памяти не предполагает возможности выполнения слияний распределенных таблиц. Операция join (объединение) поддерживается только для таблиц, копии которых хранятся на узлах хранилища целиком (реплицированные таблицы)[1]. Таким образом, в случае необходимости выполнения объединения распределенных таблиц, пользователь вынужден выполнять эту операцию самостоятельно.

Современные технологии передачи данных могут быть медленными по сравнению с обработкой данных в оперативной памяти. Сеть, построенная по технологии InfiniBand с номинальной пропускной способностью 40 Гбит/сек, показала 3 Гбит/сек на узле во время выполнения секционирования данных при помощи хэш-функции. При выполнении в оперативной памяти, секционирование на несколько тысяч частей выполняется со скоростью, приблизительно равной пропускной способности оперативной памяти [2,3]. Следовательно, алгоритм объединения таблиц для хранилищ в оперативной памяти должен минимизировать пересылки данных по сети.

Анализ существующих решений

В большинстве распределенных баз данных на долговременных запоминающих устройствах для объединения таблиц используется Grace-метод объединения хешированием[4,5]. Особенностью этого метода является разбиение входных таблиц на части (партиции) с последующей записью их на диск, и последовательным чтением. Очевидно, что данный метод неэффективен для хранилищ в оперативной памяти. Применительно к IMDG, за основу можно взять метод гибридного объединения хешированием, являющийся модификацией Grace-метода. Гибридное объединение предусматривает хранение партиций в оперативной памяти, причем, во избежание переполнения памяти предусмотрена запись части партиций на диск. Такой подход позволяет, при наличии достаточного количества свободной памяти, выполнять объединение таблиц без использования медленных устройств. С другой стороны, при высокой интенсивности операций объединения или при дефиците памяти будет использовано устройство долговременного хранения, что позволит обработать запрос даже при недостатке ресурсов. Однако объединение хешированием далеко от оптимального использования сетевых ресурсов, так как при этом способе обе

таблицы-операнда передаются по сети практически полностью [6]. Использование предопределенных хэш-функций гарантирует балансировку нагрузки, но ограничивает вероятность того, что хэшированный кортеж не будет передан по сети до 1/N, где N – количество узлов распределенного хранилища.

Метод путевого объединения (track join), представленный в [6], минимизирует количество пересылок кортежей по сети. Основная идея этого метода состоит в определении цели отправки каждого конкретного кортежа. Метод путевого объединения существенно снижает использование сетевых ресурсов по сравнению с другими известными методами выполнения объединений распределенных таблиц. Однако он ориентирован на использование в традиционных базах данных, где хранимые кортежи строго типизированы и фактический объем хранимых данных в разных строках одной таблицы отличается несущественно. В отличие от них, IMDG хранят объекты, способные, в свою очередь, содержать другие объекты. Таким образом, объем хранимых данных в разных кортежах одной таблицы может отличаться на несколько порядков (а, теоретически, эта разница может быть и больше). Поэтому простой подсчет количества пересылаемых кортежей, используемый методом путевого объединения для оценки использования сетевых ресурсов, применительно к распределенным хранилищам в оперативной памяти не дает объективной количественной оценки объема пересылаемых данных.

Цель

Целью данной работы является разработка метода объединения распределенных таблиц, позволяющий количественно оценить и минимизировать фактический объем пересылаемых по сети данных при выполнении операции объединения в хранилищах данных в оперативной памяти.

Изучению подвергается внутреннее симметричное объединение по признаку равенства ключа таблиц распределенного хранилища данных в оперативной памяти.

Исходными данными являются две таблицы, R и S, произвольно распределенные по N узлам хранилища. Каждый узел может взаимодействовать со всеми остальными и все каналы передачи данных между узлами обладают одинаковой пропускной способностью.

Описание метода

Рассмотрим три версии модифицированного метода путевого объединения для хранилищ в оперативной памяти, по мере возрастания их сложности. Для всех трех случаев примем, что три процесса (R и S, работающие с локальными для каждого узла частями таблиц R и S соответственно, а также управляющий процесс T) работают одновременно на каждом узле, но только один раз на каждом узле.

Двухфазное путевое объединение – это простейшая версия метода. Она определяет узлы хранилища, содержащие как минимум один подходящий кортеж для каждого из ключей объединения (значения поля, по равенству которого выполняется объединение кортежей). Затем одна из таблиц кортеж за кортежем передается на все узлы, содержащие соответствующий ключ объединения. Такую процедуру передачи будем называть селективным широковещанием.

Необходимо отметить, что в данной статье описан оригинальный алгоритм двухфазного путевого объединения, представленный в [6]. Описание этого алгоритма приведено для облегчения понимания 3х- и 4х-фазного алгоритмов модифицированного путевого объединения.

Метод двухфазного путевого объединения состоит из двух этапов. Первый этап – определение путей передачи, второй - селективное широковещание.

Во время первой фазы обе таблицы R и S проецируются по ключу объединения, затем проекции пересылаются по сети. Узел назначения при этом определяется при помощи хеш-функции, так же, как пересылаются кортежи при обычном объединении хешированием. Каждый узел принимает уникальные ключи и хранит их вместе с идентификатором отправившего их узла. Эта операция выполняется отдельным процессом T, который также генерирует расписания передачи. Расположение процесса T определяется хэш-функцией ключа объединения, таким образом, распределяя процесс управления по узлам хранилища.

Во второй фазе выполняется передача кортежей таблицы R на узлы, хранящие соответ-

ствующие им по ключу объединения кортежи таблицы S. При этом процесс T отправляет сообщения на каждый узел, содержащий хотя бы один кортеж таблицы R, для которого было найдено совпадение в таблице S. Каждое сообщение содержит ключ объединения и соответствующее ему множество идентификаторов узлов, содержащих части таблицы S.

Выбор передаваемой таблицы (R или S) в этом случае должен быть выполнен оптимизатором до начала выполнения запроса, подобно выбору внутреннего и внешнего отношений в традиционных алгоритмах объединения хешированием.

Метод двухфазного путевого объединения (или однонаправленного селективного широковещания) передает по сети данные только одной из таблиц. В случае, когда входные таблицы имеют в основном уникальные значения ключей объединения, а селективность высока, объем пересылок данного алгоритма равен сумме стоимости путевого поиска и минимума кардинальных чисел множеств R и S.

Блок-схемы процессов R, S и T представлены на рис. 1 – 3.

Трехфазное путевое объединение

Метод модифицированного трехфазного путевого объединения (или двунаправленного селективного широковещания) позволяет во время выполнения определить, кортежи которой из таблиц (R или S) следует пересылать. Это решение принимается индивидуально для каждого из значений ключа объединения. Для этого каждый узел высылает координирующему процессу информацию не только о наличии определенного ключа объединения, но и о связанном с ним объеме хранимых данных.

Двунаправленное селективное широковещание позволяет определить случаи, когда пересылка кортежей таблицы S означает передачу меньшего объема информации, чем в случае пересылки кортежей таблицы R. Решение о направлении пересылки (R к S или наоборот) принимается для каждого уникального значения ключа объединения независимо, таким образом, оптимизатор запросов освобождается от необходимости при каждом выполнении оценивать общую выгодность и выбирать единственное направление пересылки для всей операции объединения.

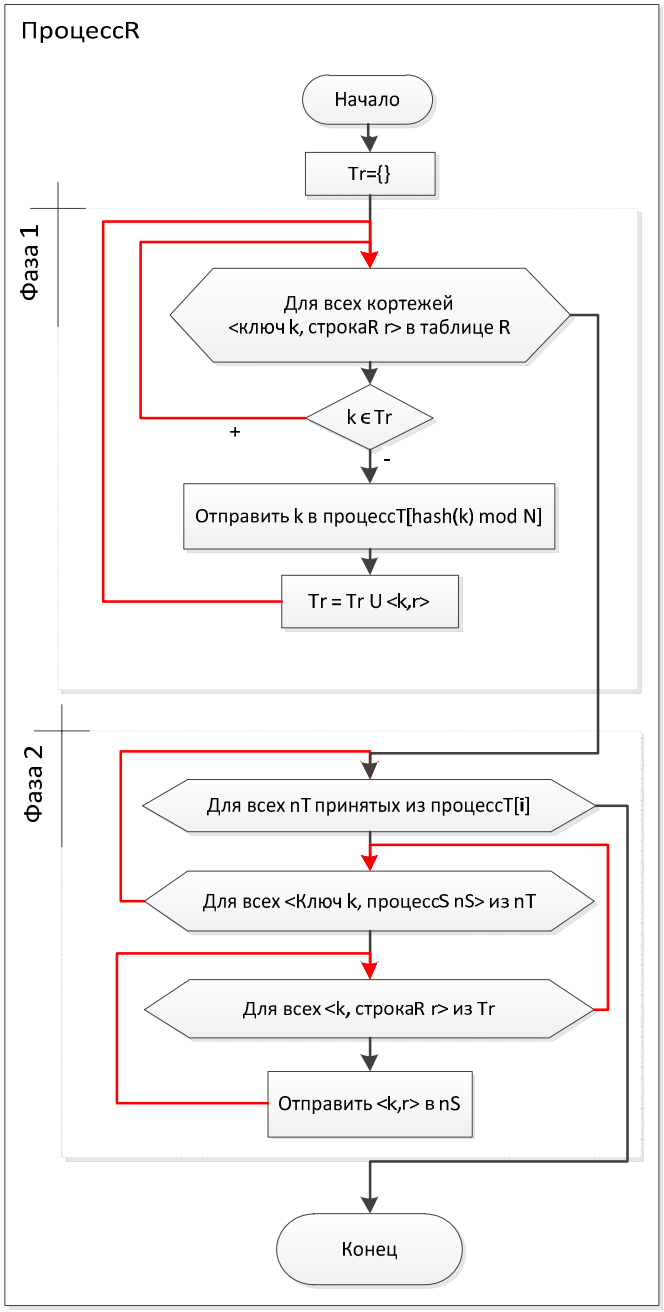


Рис 1. Алгоритм работы процесса R для двухфазного путевого объединения

В трехфазном алгоритме процессы, оперирующие таблицами R и S, симметричны. Блок-схема алгоритма такого процесса (на примере процесса R) приведена на рис. 4. Для того, чтобы получить блок-схему алгоритма процесса S, необходимо на рис.4 заменить все буквы r на s, и наоборот.

Алгоритм управляющего процесса представлен на рис. 5. Как видно, он использует процедуру определения стоимости однонаправленного селективного широковещания.

Метод выполнения оценки симметричен по направлению пересылки. Приведем оценку стоимости пересылки кортежей R к кортежам S.

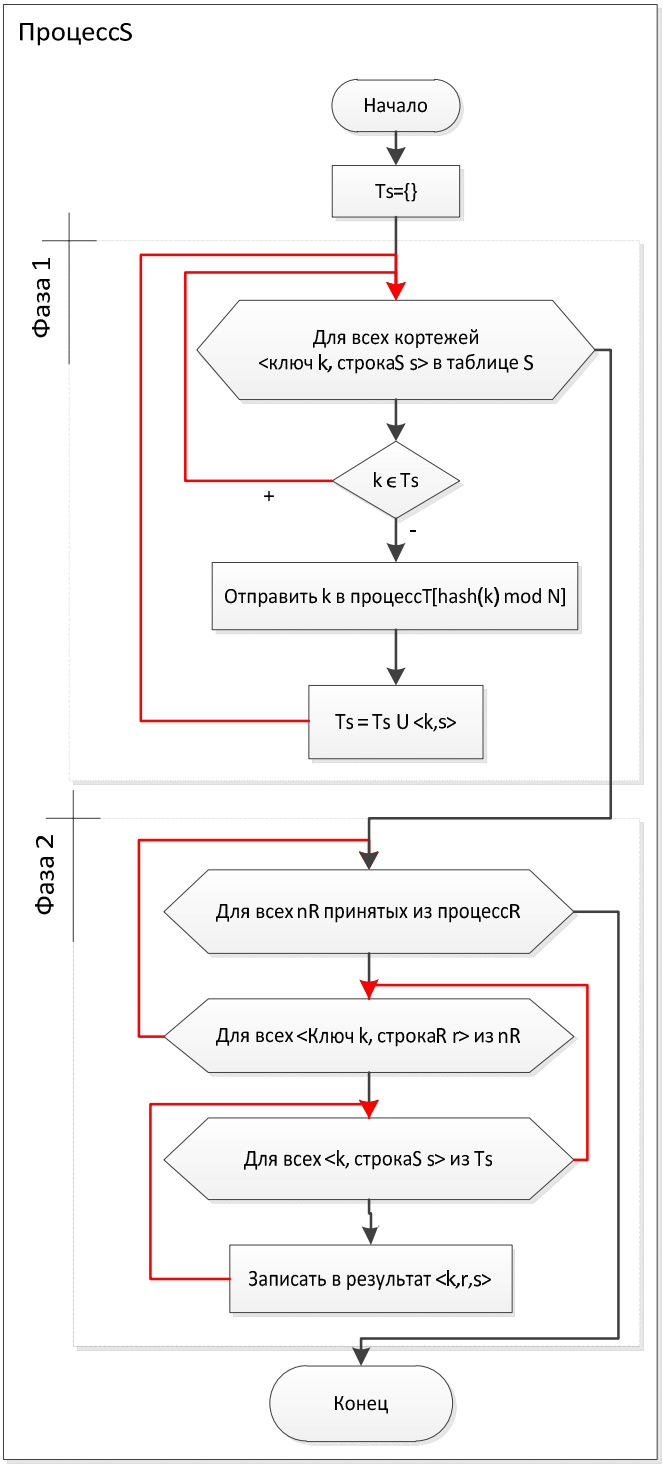


Рис 2. Алгоритм работы процесса S для двухфазного путевого объединения

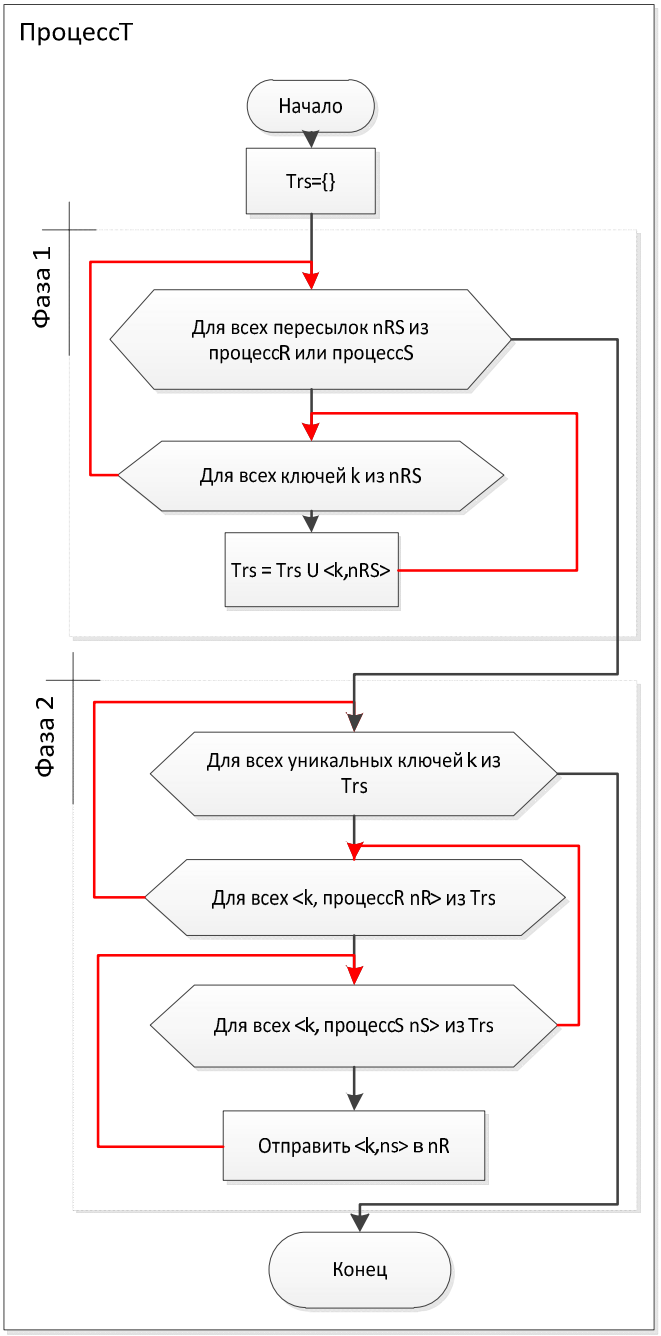


Рис 3. Алгоритм работы процесса T для двухфазного путевого объединения

В качестве исходных данных примем массивы R = {<k, id_r, c_r>} и S = {<k, id_s, c_s>}, каждый кортеж которых содержит значение ключа объединения, идентификатор узла, хранящего этот ключ и суммарный объем хранимых узлом кортежей, которые содержат ключ k. Оценка стоимости пересылки выполняется для каждого значения ключа k независимо, т.е. в каждом конкретном случае множества R и S содержат кортежи с единственным значением k.

Общий объем хранимых в таблице R данных, содержащих ключ k:

$$R_{общ} = \sum_{i=1}^{N} c_{r_i} . \qquad (1)$$

Здесь и далее N – количество узлов в распределенном хранилище, содержащих ключ объединения k.

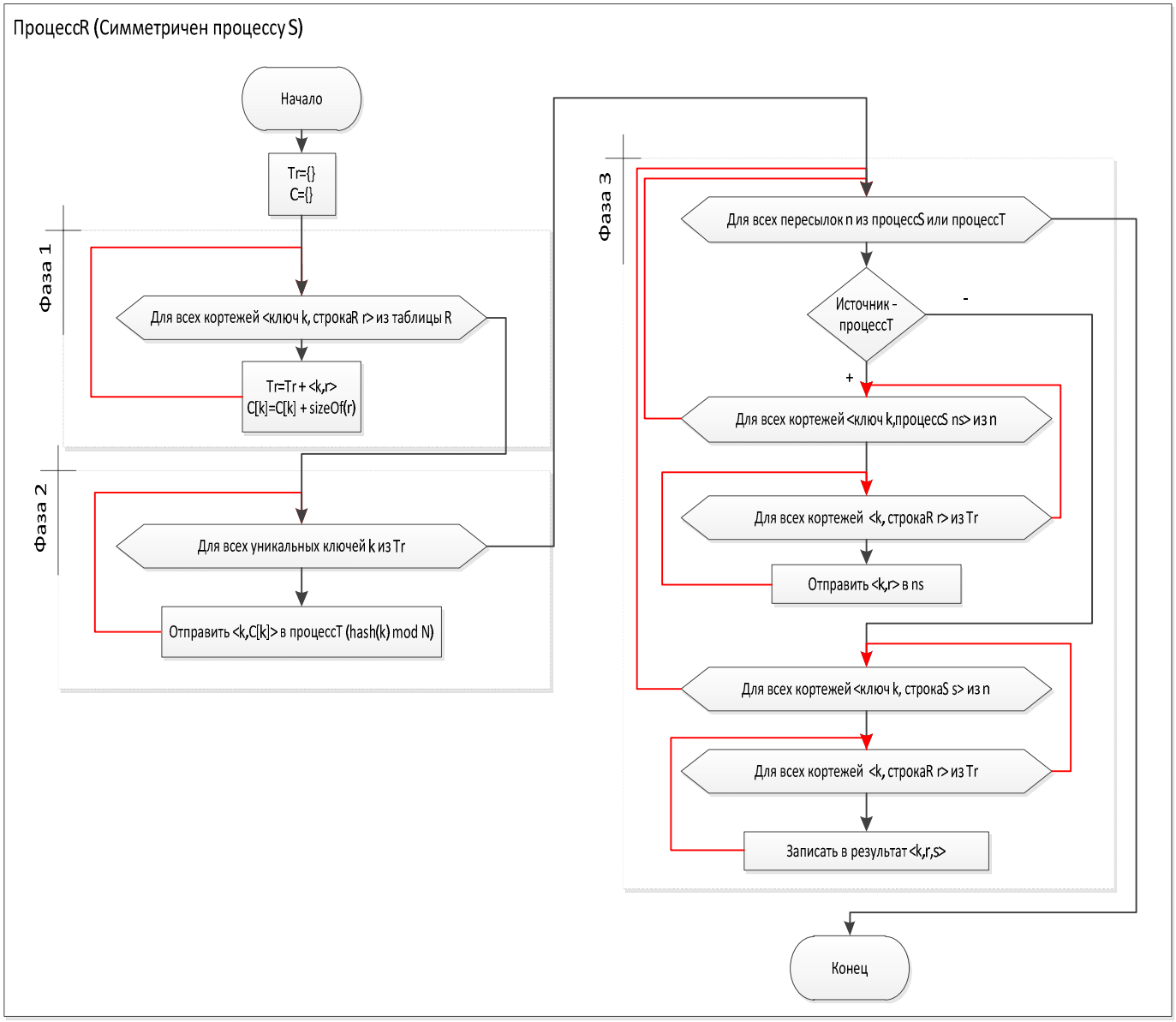


Рис 4. Блок-схема алгоритма процесса R в модифицированном трехфазном путевом объединении

Объем кортежей R, хранимых на одном узле с соответствующими им по ключу объединения кортежами таблицы S:

$$R_{лок} = \sum_{i=1}^{N} q_i \quad (2) \qquad где$$

$$q_i = \begin{cases} c_{r_i}, c_{s_i} > 0 \\ 0, c_{s_i} = 0 \end{cases} \quad (3)$$

Количество узлов, которые необходимо оповестить управляющему процессу:

$$N_r = \sum_{i=1}^{N} k_{r_i}, \quad (4) \qquad где$$

$$k_{r_i} = \begin{cases} 1, c_{r_i} > 0 \vee процессT \notin узел_i \\ 0, c_{r_i} = 0 \vee процессT \in узел_i \end{cases} \quad (5)$$

Количество узлов, содержащих кортежи таблицы S:

$$N_s = \sum_{i=1}^{N} k_{s_i} \quad (6)$$

$$Здесь \quad k_{s_i} = \begin{cases} 1, c_{s_i} > 0 \\ 0, c_{s_i} = 0 \end{cases} \quad (7)$$

Стоимость селективного широковещания от R к S равна сумме объемов пересылаемых кортежей первой таблицы, кроме локально хранимых с кортежами второй таблицы, и объема генерируемых управляющим процессом уведомлений об отправке:

$$RS_{cost} = R_{общ} \cdot N_s - R_{лок} + N_r \cdot N_s \cdot M, \quad (8)$$

Где M – размер уведомления об отправке, содержащего идентификатор узла-получателя и ключ объединения.

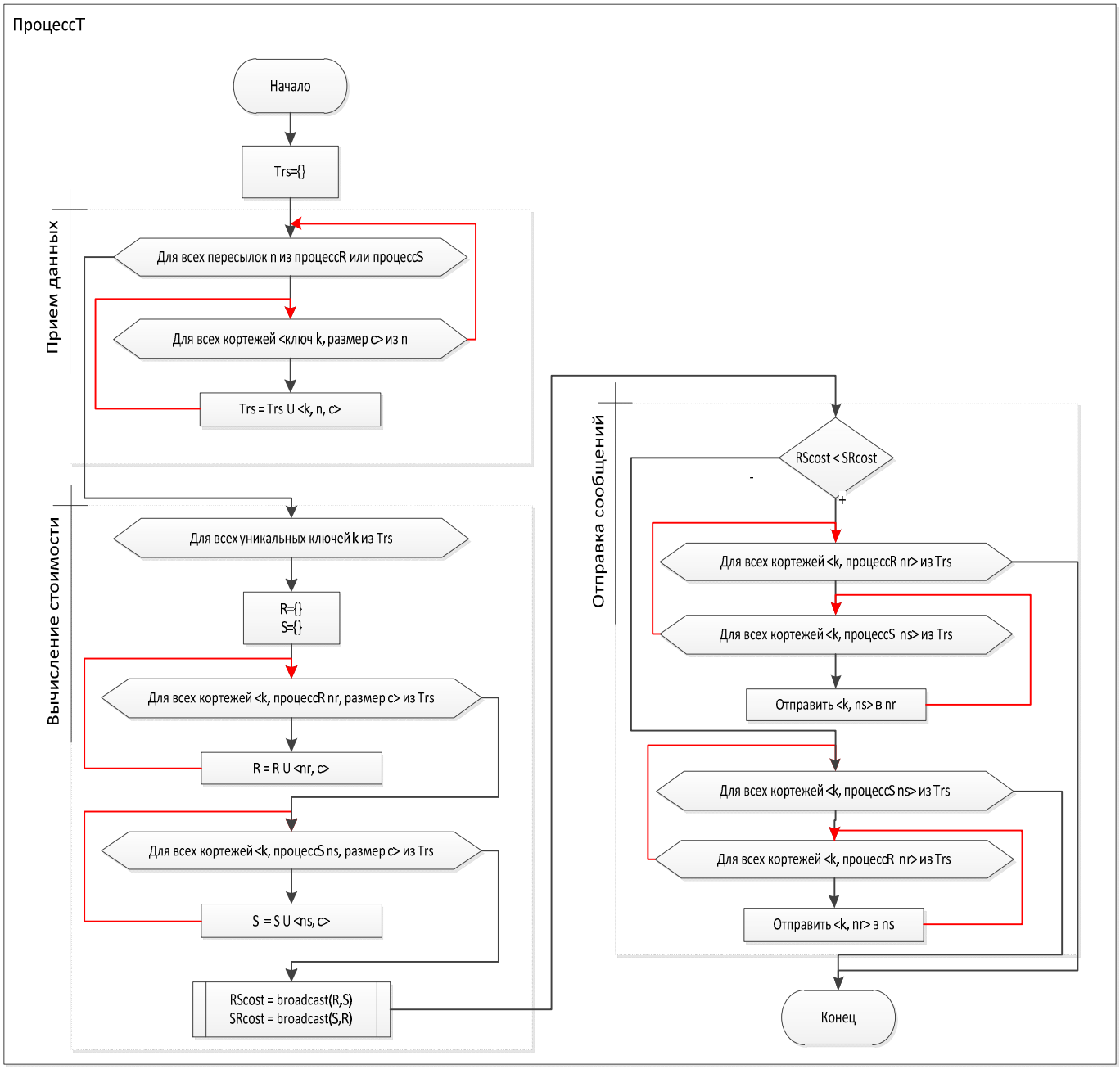


Рис. 5. Блок-схема управляющего процесса алгоритма модифицированного трехфазного путевого объединения

Количество выполняемых операций меньше количества анализируемых кортежей, следовательно, алгоритм является линейным по классу вычислительной сложности (O(N)).

Четырехфазное путевое объединение

В полной версии метода путевого объединения объем пересылок по сети оптимизируется за счет учета возможностей предварительных группировок кортежей одной из таблиц, содержащих определенный ключ объединения. Это достигается путем введения более сложного управляющего алгоритма, содержащего фазу миграции. Задачей этой фазы является определение подмножества узлов, содержащих кортежи одной из таблиц-операндов объединения, для которого временная сложность предварительного переноса кортежей первой таблицы и последующего селективного широковещания со второй будет наименьшей.

Блок-схема симметричных процессов R и S представлена на рис.6, управляющего процесса T – на рис.7.

Управление пересылками в четырехфазном методе иногда демонстрирует поведение, сходное с методом объединения хешированием, где все кортежи пересылаются на один узел хранилища. Даже в этом случае алгоритм путевого объединения, чтобы минимизировать стоимость

пересылок, выберет узел с наибольшим количеством уже хранимых кортежей.

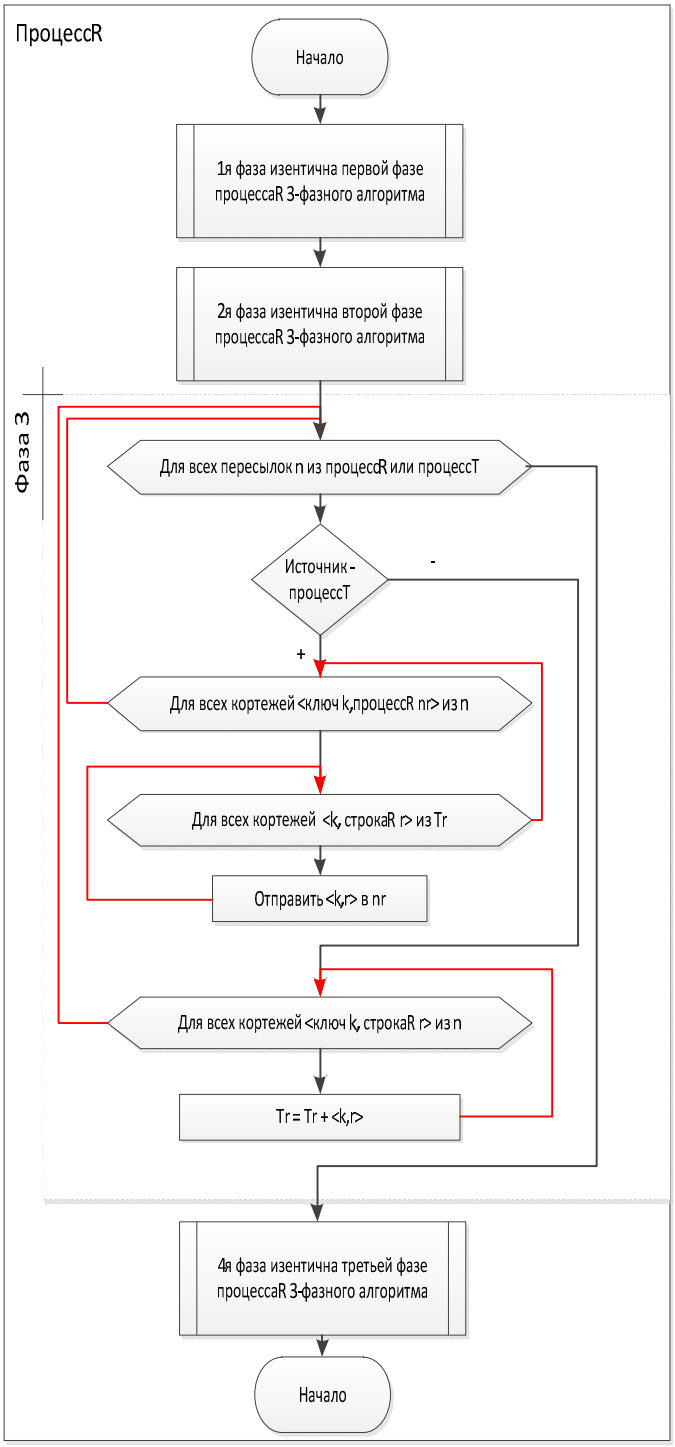


Рис 6. Блок-схема алгоритма процесса R в модифицированном четырехфазном путевом объединении

Введем функцию оценки временной сложности объединения двух таблиц по одному значению ключа объединения. Обозначим через x_{ij} булево решение о пересылке кортежей таблицы R с узла i на узел j, а y_{ij} – решение о пересылке кортежей S с i на j. Примем c_{r_i} как общий объем хранимых на узле i кортежей таблицы R, содержащих данный ключ объединения. Аналогично c_{s_j} обозначает объем кортежей таблицы S на узле j. Тогда объем пересылок для объединения кортежей по одному значению ключа будет

$$Q = \sum_i \sum_{j \neq i} x_{ij} \cdot c_{r_i} + y_{ij} \cdot c_{s_j} \qquad (9)$$

Где $\forall i, j \sum_k x_{ik} \cdot y_{kj} \geq 1$

Таким образом, определение способа объединения таблиц, оптимально использующего сетевые ресурсы, сводится к нахождению минимума функции Q.

Четырехфазный метод путевого объединения решает задачу минимизации сетевого трафика путем миграции, т.е. перегруппировки кортежей таблицы, выступающей получателем кортежей второй таблицы. Так, перед селективным широковещанием R→S, происходит миграция кортежей S таким образом, чтобы уменьшить объем последующей пересылки кортежей R.

Алгоритм определения стоимости пересылки $RS_{\cos t}$ и узлов, кортежи с которых должны мигрировать S_{migr} (в случае направления пересылки R→S), представлен на рис. 8. Алгоритм симметричен по направлению пересылки.

Рисунок 9 иллюстрирует схему сетевых пересылок для объединения таблиц хешированием, а также для разных версий путевого объединения. Для хеш-объединения примем, что хэш-функцией был выбран первый узел. Алгоритм двухфазного объединения пересылает кортежи таблицы R к кортежам таблицы S. Трехфазный алгоритм выполняет пересылку в обратном направлении, т.к. это снижает объем пересылок. Четырехфазный алгоритм выполняет предварительную миграцию кортежей R, также выполняя затем селективное широковещание кортежей S.

Оценка объема пересылок

Для упрощения примем равномерное распределение кортежей по узлам распределенного хранилища. Это наихудший случай для метода путевого соединения, так как он исключает локализацию данных[6].

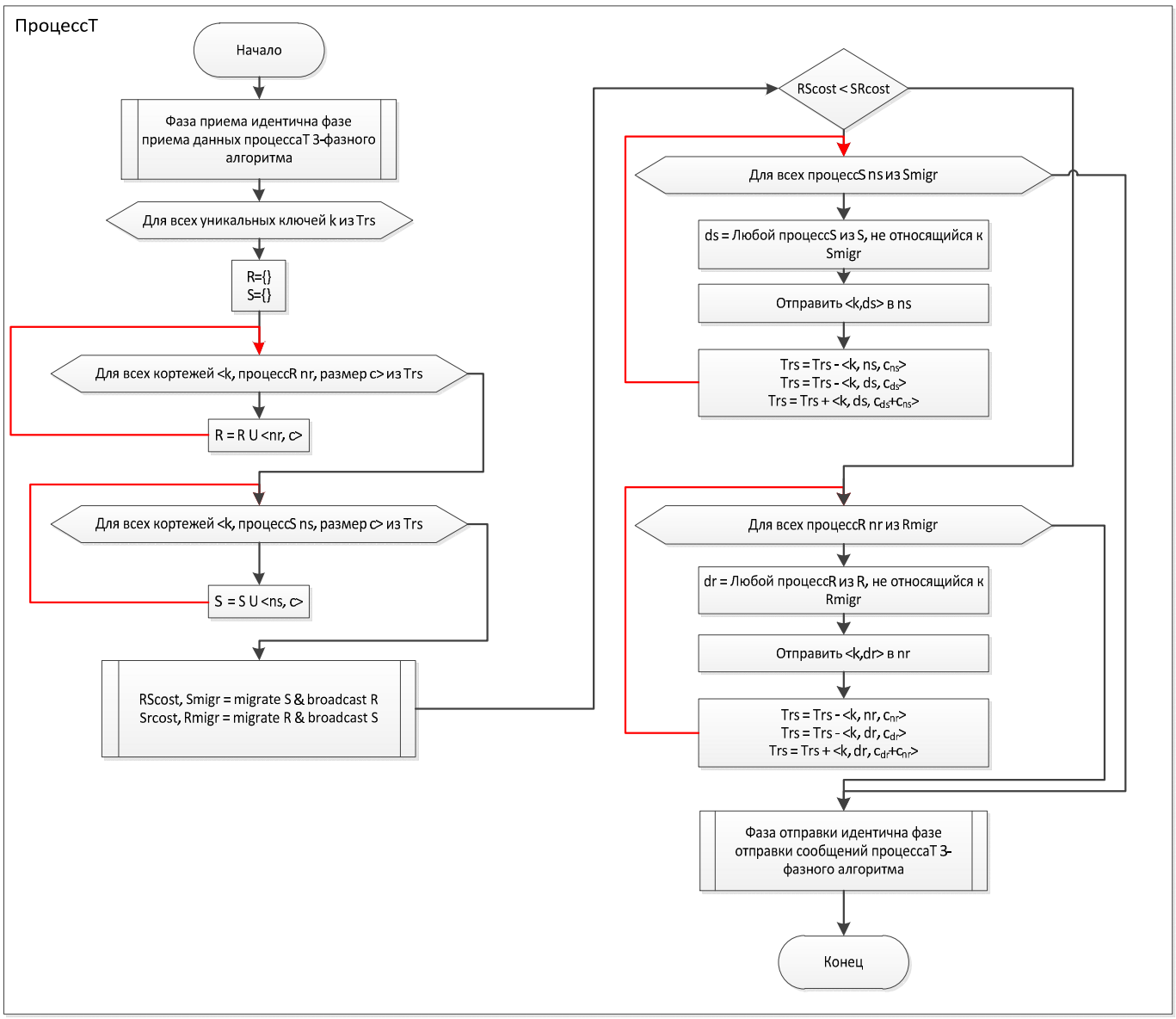


Рис. 7. Блок-схема управляющего процесса алгоритма модифицированного четырехфазного путевого объединения

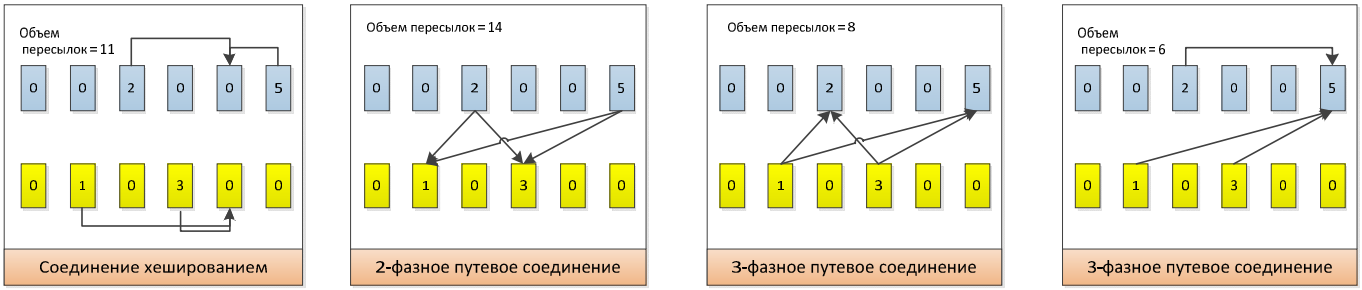


Рис. 9. Сравнение выполнения объединения путевыми алгоритмами и алгоритмом хеширования

Оценка объема пересылаемых по сети кортежей приводится применительно к распределенным хранилищам данных в оперативной памяти, использующим объектную модель.

Объем пересылок в случае использования метода соединения хешированием:

$$w_k \cdot (t_r + t_s) + \sum (c_{r_i} + c_{s_i})$$

Где w_k - объем памяти, занимаемый ключом объединения

t_r и t_s – кардинальные числа множеств R и S соответственно

c_{r_i} и c_{s_i} – совокупные объемы хранимых на i-м узле подлежащих объединению данных из таблиц R и S соответственно.

Для оценки объема пересылок в двухфазном путевом соединении, прежде всего, необходимо определить объем пересылок при определении пути. Количество узлов хранилища, содержащих конкретный ключ объединения, ограничено сверху количеством узлов N, и, в худшем случае, т.е. при равномерном распределении равных ключей по всем узлам, составляет t/d, где t – количество кортежей и d – количество уникальных ключей объединения. Мы будем обозначать количество узлов, содержащих конкретный ключ соединения как $n_R \equiv \min(N, t_R / d_R)$ и $n_S \equiv \min(N, t_S / d_S)$.

Введем понятие входной селективности (s_R и s_S) как процент кортежей одной таблицы, для которых существуют совпадения в другой таблице. Исходя из того, что совпадения существуют для всех значений ключа объединения, определим количество узлов, содержащих совпадения: $m_R \equiv \min(N, t_R \cdot s_R / d_R)$ и $m_S \equiv \min(N, t_S \cdot s_S / d_S)$

Исходя из этого, объем пересылок при использовании двухфазного метода путевого объединения будет

$(d_R \cdot n_R + d_S \cdot n_S) \cdot w_k$ (поиск путей)

$+ d_R \cdot m_S \cdot w_k$ (передача идентификаторов узлов с кортежами S)

$+ t_R \cdot s_R \cdot m_S \cdot (w_k + \sum c_{R_i} / d_R)$ (передача кортежей R к кортежам S)

Трехфазное путевое объединение предусматривает передачу общего объема хранимых данных, содержащих данный ключ объединения. Отношение $t/(d \cdot s)$ показывает среднее количество повторений ключей на каждом узле при равномерном распределении кортежей. Тогда максимальные размеры счетчиков объема пересылаемых по ключу данных будут $q_R = \log_2(\max(c_{R_i}) \cdot t_R / (d_R \cdot s_R))$ и $q_S = \log_2(\max(c_{S_i}) \cdot t_S / (d_S \cdot s_S))$.

Таким образом, объем пересылок при трехфазном путевом объединении составляет

$d_R \cdot n_R \cdot (w_k + q_R) + d_S \cdot n_S (w_k \cdot q_S)$ (поиск путей)

$+ d_{R_1} \cdot m_{S_1} \cdot w_k + t_{R_1} \cdot s_{R_1} \cdot m_{S_1} \cdot (w_k + \sum c_{R_{1_i}} / d_{R_1})$

$(R_1 \to S_1)$

$+ d_{R_2} \cdot m_{S_2} \cdot w_k + t_{R_2} \cdot s_{R_2} \cdot m_{S_2} \cdot (w_k + \sum c_{R_{2_i}} / d_{R_2})$

$(S_2 \to R_2)$

Здесь и далее R_1, S_1, R_2, S_2 – подмножества кортежей соответствующих таблиц, разделенные по направлению передачи.

Оценка объема пересылок для четырехфазного путевого объединения имеет вид

$d_R \cdot n_R \cdot (w_k + q_R) + d_S \cdot n_S (w_k \cdot q_S)$ (поиск путей)

$+ d_{R_3} \cdot m_{R_1} \cdot w_k + t_{R_3} \cdot s_{R_3} \cdot m_{R_1} \cdot (w_k + \sum c_{R_{3_i}} / d_{R_3})$

(миграция $R_3 \to R_1$)

$+ d_{S_3} \cdot m_{S_2} \cdot w_k + t_{S_3} \cdot s_{S_3} \cdot m_{S_2} \cdot (w_k + \sum c_{S_{3_i}} / d_{S_3})$

(миграция $S_3 \to S_2$)

$+ d_{R_1} \cdot m_{S_1} \cdot w_k + t_{R_1} \cdot s_{R_1} \cdot m_{S_1} \cdot (w_k + \sum c_{R_{1_i}} / d_{R_1})$

$(R_1 \to S_1)$

$+ d_{R_2} \cdot m_{S_2} \cdot w_k + t_{R_2} \cdot s_{R_2} \cdot m_{S_2} \cdot (w_k + \sum c_{R_{2_i}} / d_{R_2})$

$(S_2 \to R_2)$

Здесь R_3, S_3 обозначают множества кортежей, мигрирующие на другие узлы с целью минимизации объема пересылок.

Результаты экспериментов

Для сравнительной оценки использования методов модифицированного путевого объединения была создана система имитационного моделирования.

В системе были реализованы алгоритмы объединения хешированием, оригинальный 2х-, 3х- и 4х-фазный алгоритм путевого соединения и модифицированный алгоритм путевого соединения, представленный в данной статье.

Моделируемая система хранения данных содержит 16 узлов, на которых размещается в общей сложности 10^9 кортежей. В рамках модели было принято, что таблицы R и S содержат одинаковое количество кортежей. Изменяемым параметром в модели является допустимое отклонение размеров кортежей относительно минимального значения. Рассматривались значения 1 (т.е. все кортежи равны, и каждый кортеж содержит только примитивные типы данных), 2 и 10. Допустимое отклонение в 10 раз означает, что размер кортежа может быть от x до 10x, где x – заданное минимальное значение. В качестве минимального размера кортежа для таблицы R было принято 60 байт, для таблицы S – 40 байт.

Результаты имитационного моделирования приведены на рис. 10.

Проведенные исследования показали, что использование модифицированных методов путевого объединения целесообразно только при наличии вариативности размера пересылаемого кортежа. Таким образом, использовать его следует в случаях, когда пересылаемый кортеж содержит в себе объектные структуры данных. В случае же, когда кортеж содержит только примитивные типы, целесообразнее использовать оригинальный метод путевого объединения.

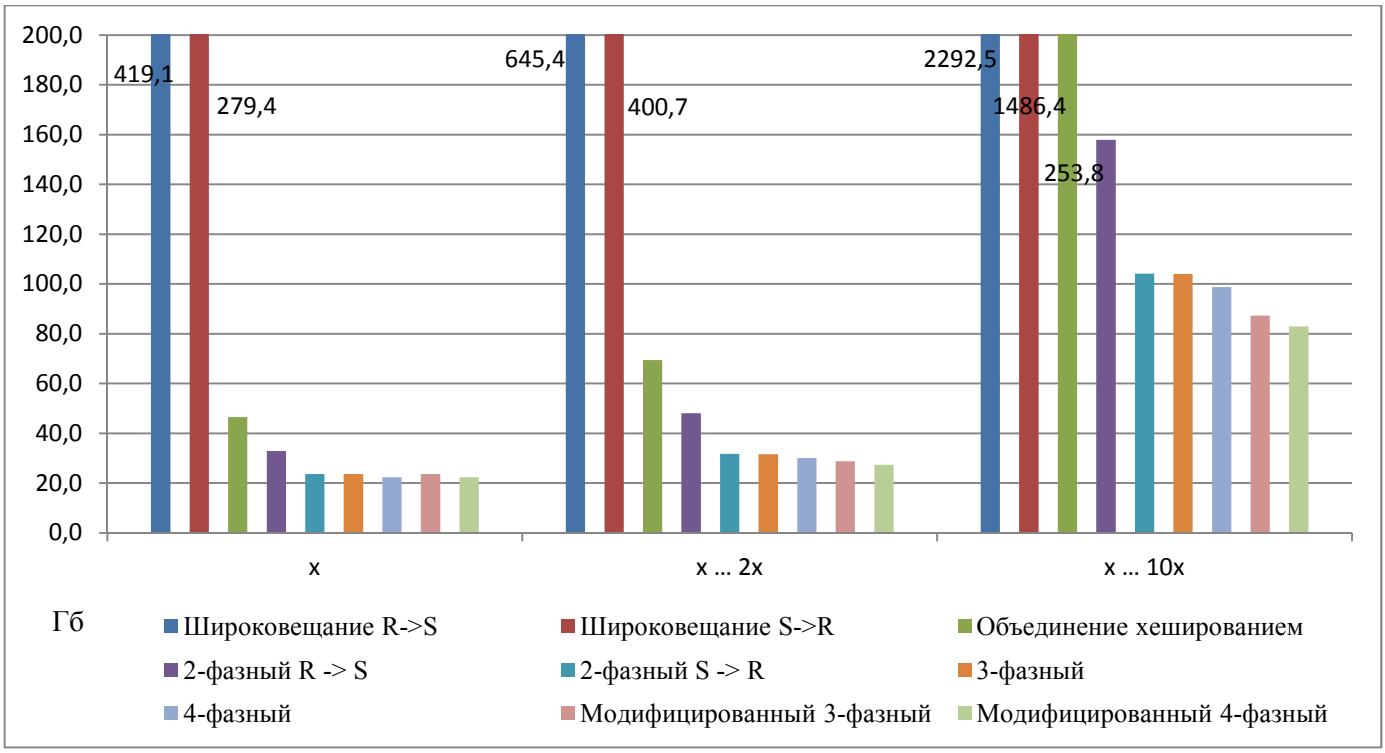


Рис. 10. Объем пересылок при использовании различных методов объединения

Вывод

Предложенные в данной статье модифицированные методы и алгоритмы путевого объединения распределенных таблиц позволяют уменьшить объем пересылаемых данных в хранилищах, использующих объектную модель данных, прежде всего, в хранилищах данных в оперативной памяти. Их применение позволяет значительно ускорить время выполнения операции объединения в распределенных хранилищах в оперативной памяти, а также увеличить количество одновременно выполняемых в хранилище операций.

Список литературы

1. А. Александров. Ударные СУБД в оперативной памяти // Открытые системы. СУБД – 2008 – №7 – С.36-44.
2. M.-C. Albutiu, A. Kemper, and T. Neumann. Massively parallel sort-merge joins in main memory multi-core database systems. // PVLDB, 5(10) –2012 – p.1064-1075.
3. C. Balkesen et al. Multicore, main-memory joins: Sort vs hash revisited. // PVLDB, 7(1) – Sept. 2013 – p.85-96.
4. D. J. DeWitt et al. The Gamma database machine project. // IEEE Trans. Knowl. Data Engin., 2(1) – 1990 – p.44-62.
5. M. Kitsuregawa, H. Tanaka, and T. Moto-Oka. Application of hash to data base machine and its architecture. //New Generation Computing – 1983 – p.63-74.
6. O. Polychroniou, R. Sen and K. Ross. Track join: distributed joins with minimal network traffic. // SIGMOD Conference – 2014 – p. 1483-1494.

УДК 004.7

ЛУЦЬКИЙ Г.М.,
ДОЛГОЛЕНКО О.М.,
АКСЬОНЕНКО С.В.,
СТОРОЖУК В.О.

МОДЕЛЮВАННЯ ОБМЕЖЕНОЇ РЕАЛІЗАЦІЇ АРХІТЕКТУРИ ПОТОКУ ДАНИХ В СТРУКТУРІ СУПЕРСКАЛЯРНОГО ПРОЦЕСОРА

У статті представлені результати дослідження особливостей обмеженої реалізації архітектури потоку даних (RDF) у структурі ядра суперскалярного процесора, промодельовані схемотехнічні рішення, пов'язані з реалізацією RDF, уточнено кількість рядків у кожній з трьох станцій резервування досліджуваного ядра.

The results of the study features a restricted implementation of a data flow architecture (RDF), in the structure of the superscalar processor core modeled circuit solutions related to the implementation RDF, the quantity of rows in each of three stations of reservation of the studied core is specified.

Вступ

Архітектура Data Flow [1], в якій відсутній лічильник команд, а момент готовності команд до виконання визначається моментом готовності їхніх даних поки не знайшла свого впровадження у вигляді, запропонованому J. Dennis, через складність її реалізації. Складність пов'язана з тим, що визначення моменту готовності до виконання кожної окремо взятої команди здійснюється на основі використання активної комірки пам'яті для її подання. Для реалізації великої кількості активних комірок пам'яті не тільки потрібні значні апаратурні витрати, але й ускладнюється організація розпаралелювання виконання команд при настанні одночасної готовності великої їх кількості до виконання.

Однак, досить швидко низкою дослідників [2,3] була запропонована обмежена реалізації архітектури потоку даних (Restricted Data Flow (RDF)) в структурі суперскалярного процесора, яка функціонує на основі розширеного алгоритму Tomasulo і дозволяє використовувати дрібнозернистий паралелізм при виконанні RISC операцій (Reduced Instruction Set Computing - обчисленнях зі скороченим набором команд) [4]. Так, наприклад, починаючи з мікроархітектури Intel P6 (1995 р.), в суперскалярних мікропроцесорах фірми Intel реалізовується технологія OoOE (Out-of-Order Execution), заснована на використанні RDF.

Метою роботи є дослідження особливостей реалізації RDF в структурі суперскалярного процесора, розробка схемотехнічних рішень, пов'язаних з реалізацією RDF.

Архітектура CISC-RISC

В даний час домінуючою архітектурою, для побудови мікропроцесорів [4], стала архітектура CISC-RISC («CISC-outside RISC-inside», CISC - Complex Instruction Set Computing). Наприклад, останні процесори Intel: Nehalem, Sandy Bridge, Ivy Bridge, Haswell, а також AMD: K10, Bulldozer, Piledriver, Jaguar побудовані по архітектурі CISC-RISC.

У цих процесорах паралелізм на рівні команд (ILP-Instruction-Level Parallelism) досягається шляхом використання методик реалізації неявного динамічного паралелізму, присутнього на етапі виконання CISC команд, декодованих на безліч RISC-операцій (їх часто називають μops, або подібними термінами). Однією з таких методик є використання RDF.

Для реалізації паралелізму на рівні потоків CISC команд (TLP - Thread-Level Parallelism) або мультитредової архітектури - ядра цих процесорів одночасно обробляють два потоки команд, кожен з яких реалізується відповідно до принципів ILP - архітектури.

Архітектура CISC-RISC отримала назву суперскалярной архітектури [4].

У модельованій мікроархітектурі RISC - ядра є новітні аспекти вирішення проблеми оптимізації процесів виконання команд в конвеєрі та підвищення його пропускної спроможності. Стало майже універсальною нормою споряджати процесори центральним контролюючим і управляючим пристроєм, який на усіх стадіях перевіряє правильність обробки команд в конвеєрі. Цей контролер визначає і координує зміни станів команд, що проходять через конвеєр.

У модельованій мікроархітектурі ядра застосовано децентралізовану схему пристрою контролю і управління процесами виконання команд в конвеєрі, що розподілена по всіх сегментах конвеєра. Ядро одночасно обробляє 4 потоки CISC команд.

Для досягнення ILP, модельована мікроархітектура декодує CISC команди x86-64 в прості стилізовані елементарні обчислювальні операційні одиниці, що названі RISC–операціями. При цьому мікроархитектура може змінювати свої машинні стани в єдиний спосіб – шляхом виконання RISC – операцій.

Мікроархітектура RISC-ядра є глибоко конвеєрною, відносно відомих реалізацій. Глибокий конвеєр реалізовано як декілька коротких сегментних потоків, сполучених чергами. Такий підхід дає можливість застосувати вищу тактову частоту, а негативні наслідки глибокої конвеєризації знижуються за рахунок застосування вдосконаленого пристрою передбачення галужень, надшвидкісного доступу до L2–кеш (другого рівня) і гранично високої тактової частоти.

Модельована мікроархітектура також є архітектурою з динамічним прогнозуванням галужень (speculations). Результати динамічно прогнозованого виконання (виконання, що працює з припущеннями відносно ще не обчисленої умови галуження) не повинні змінювати архітектурні стани процесора доти, доки не буде обрахований результат, припущення відносно якого і викликало це динамічне прогнозування. Після перевірки припущень динамічно прогнозовані обчислення або стають не динамічно прогнозованими, або – недійсними.

Декодер команд

Схема декодера команд (ID) показана на *Рис.1*. На першій стадії його роботи здійснюється запам'ятовування байтів командного вікна в черзі поточного треду буферу розмічених команд. Байти командного вікна надходять з буфера-переддекодера (IFU) через вирівнювач.

Завантаження нового вікна команд оброблюваного потоку проводиться при будь-якій з наступних умов:

- Через неправильне передбачення галужень, внаслідок чого проводиться скидання ID;
- Правильно передбачене за допомогою цільового буфера галужень (BTB) галуження, яке повинне прийматися;
- Усі повні команди, що знаходяться в даний момент у буфері успішно декодовані.

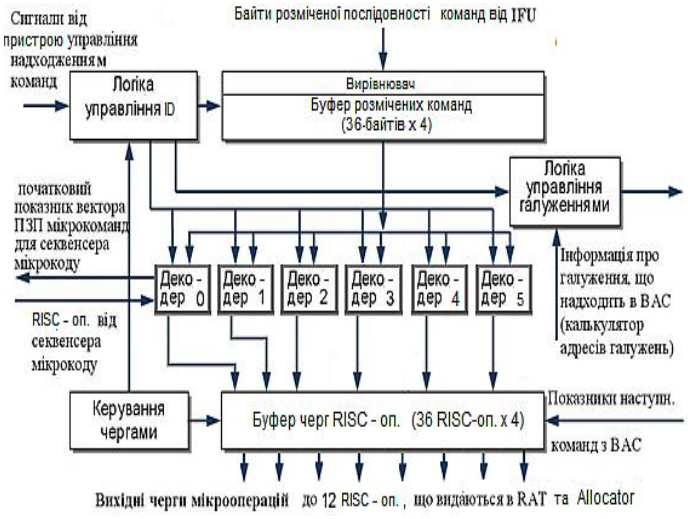


Рис. 1. Декодер команд

З черг буферу розмічені команди спрямовуються на два комплексні декодери та чотири прості. Комплексні декодери: декодер 1 використовуються для декодування довгих команд та макрокоманд, що декодуються не більше ніж в 4 RISC-операції; декодер 0 використовується для декодування складних команд, котрі декодуються в 4 і більше RISC-операції і потребують використання секвенсора мікрокоду (MS). Декодери 2, 3, 4 та 5 призначені для декодування тільки простих команд і макрокоманд, які завжди складають більшість у потоках команд.

Через змінну довжину x86-64 команд ускладняється подача в одному такті шістьох правильно вирівняних команд на шість декодерів. Навіть саме визначення довжини однієї команди не просте, оскільки спочатку повинні декодуватися перші байти, щоб інтерпретувати подальші байти. Оскільки процес подачі шістьох команд змінної довжини по суті послідовний, то необхідно заздалегідь знати розташування меж кожної команди. Пристрій виявлення довжини команд, що знаходиться в буфері-переддекодері IFU виконує функцію попереднього декодування; тобто визначення меж знаходження команд за допомогою сканування потоку байтів команд, маркування першого байта коду операції і останнього байта команди. Крім того, IFU відмічає байти, щоб вказати BTB на передбачення галужень і на контрольні точки можливих запинок коду.

Кожна з чергових шести команд командного вікна подається на один з шести декодерів, що можуть її декодувати. Якщо буфер команд не

містить шести повних команд, то в декодери буде подано стільки розпізнаних і вирівняних команд, скільки їх залишилося в черзі. Логіка управління подачею команд використовує маркери першого байта коду операції для вирівнювання і одночасної подачі команд декодерам.

Оскільки в черги командного буферу може одночасно поступати до 32 команд оброблюваного потоку, то для декодування усіх їх може знадобитися декілька тактів. Розташування початкового байту чергових шести команд, що подаються в деякому такті на декодери може знаходитися де завгодно у черзі буферу. Але спеціальна апаратура вирівнює шість команд і спрямовує їх на шість відповідних декодерів.

Хоча шість команд одночасно подаються на декодери за один цикл, деякі з команд можуть не декодуватися до кінця. Коли команда успішно не декодована, то певний декодер скидається і усі RISC-операції, отримані від цього декодера стають недійсними і анулюються. Це може зайняти множину циклів, що витратяться на успішне декодування усіх команд, котрі знаходяться у черзі буфера. До анулювання RISC-операцій приводять наступні ситуації, внаслідок чого проводиться повторна подача відповідних команд на декодери в подальших циклах:

Якщо на декодері 0 повинна декодуватися макрокоманда чи складна команда, яка вимагає перемикання її на MS з ПЗП мікрокоду (UROM), а він ще не завершив декодування попередньої команди.

Якщо зустрічається галуження, то всі RISC-операції, що вироблені наступними декодерами, стають недійсними. В одному циклі може декодуватися тільки одне галуження.

Число команд і їх складність, які можуть одночасно декодуватися, безпосередньо не мають відношення до числа виданих ID RISC-операцій, тому що буфер черг RISC-операцій декодера запам’ятовує RISC-операції та видає їх пізніше.

Складні команди

Складні команди – це ті команди, які для свого декодування потребують використання MS. В модельованому ядрі такі команди може оброблювати тільки декодер 0. Виклик роботи MS відбувається за двома умовами:

Якщо команда декодується в більше ніж 4 RISC-операції, то 4 перші з них виробляються декодером 0, а наступні – формує MS. При цьому MS отримує початкову адресу UROM від декодера 0 і починає виробляти послідовність RISC-операції, поки не досягне кінця потоку мікрокоду.

Коли зустрічаються повільні команди з великим числом циклів, декодер 0 не видає ніяких RISC-операцій і передає управління MS, який формує послідовність RISC-операцій за допомогою UROM.

Після декодування RISC - операції передаються далі по конвеєру разом з усією інформацією, необхідною для їх планування, диспетчеризації, виконання та вилучення.

Таблиця альтернативних імен регістрів

Використання таблиці альтернативних імен регістрів (RAT) забезпечує перейменування регістрів – цілочислових та з плаваючою точкою, а також регістрів ознак (прапорів), в процесі виконання команд, трансформованих в RISC-операції. Це дає можливість розширити невеликий набір архітектурний регістрів, початково закладений в x86-64. Але головною функцією перейменування регістрів є усунення залежностей між командами і забезпечення більш високого рівня ILP. В RAT логічні адреси регістрів операндів і регістру результату RISC-операції відображаються на відповідні фізичні адреси комірок буферу переупорядковування (ROB). При цьому відповідний масив в RAT, що відповідає поточному потоку команд оновлюється новою фізичною адресую приймача результату даних для кожної нової RISC-операції. Ці нові адреси надаються пристроєм призначення ресурсів (allocator).

Принцип перейменувань регістрів для одного потоку за допомогою RAT показано на *Рис. 2*. У кожному тактовому циклі RAT по своїм масивам повинна швидко відшукувати фізичні адреси ROB, які відповідають логічним посиланням кожної з RISC-операцій. Такі фізичні посилання стають частиною стану RISC-операцій і з цього моменту переміщуються разом з ними, стаючи їх атрибутами. Будь-який машинний стан, який буде модифіковано RISC-операцією також перейменовується, через інформацію, надану пристроєм виділення. Це посилання на фізичний регістр одержувача результату RISC-операції записується в RAT для використання подальшими RISC-операціями, джерела операндів яких посилаються до того ж логічного регістру. Оскільки значення фізичного приймача результату буде унікальне (в результаті перейменування воно вже не співпадає з логічним значенням) для кожної RISC-операції, воно використовується як ідентифіка-

тор для RISC-операції скрізь, де проводиться її позачергове виконання. Усі перевірки і посилання на RISC-операцію виконуються з використанням цього фізичного приймача (physical destination – PDst), як його нове ім'я.

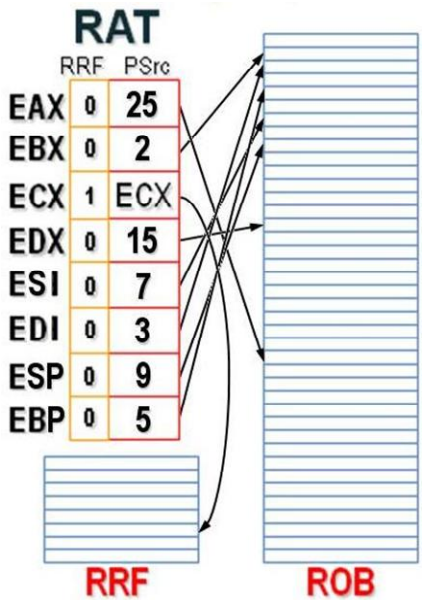


Рис. 2. Принцип перейменування регістрів за допомогою RAT для одного з оброблюваних потоків команд та знаходженням даних в ROB і RRF для цього потоку

Логічні регістри, з якими оперують RISC-операції, повинні перейменовуватися при динамічному прогнозованому виконанні. Якщо виникає пряма залежність за даними, наприклад така:

RISC0: ADD EAX, EBX; src 1 = EBX, src 2 = EAX, dst = EAX;

RISC1: ADD EAX, ECX; src 1 = ECX, src 2 = EAX, dst = EAX;

RISC2: ADD EAX, EDX. src 1 = EDX, src 2 = EAX, dst = EAX;,

то RAT повинна надати в ході обробки, причому "на льоту" – не зупиняючись, перейменовані адреси джерел операндів за допомогою спеціальної логіки обхідних передач. Ця логіка безпосередньо забезпечить для RISC1 джерело регістрового операнда src 2, EAX, щоб уникнути необхідності очікування запису результату в приймач RISC0 того ж EAX, який записуватиметься в RAT і потім читатиметься як джерело операнда RISC1 src.

Коли відбувається неправильне прогнозування галуження, RAT повинна скинути непотрібний стан, який було накопичено і знову повернутися до відображень логічних адрес регістрів на фізичні, які проводитимуться вже з наступним набором RISC-операцій. Схема відновлення неправильного прогнозування галуження гарантує, що RAT не виконуватиме ніяких нових перейменувань, поки ядро позачергового виконання не скине весь непотрібний неправильно прогнозований стан. Це означає, що регістрові посилання тепер знаходитимуться в регістровому файлі вилучення результатів (retirement register file – RRF) та відповідатимуть логічним значенням доти, поки не почнуть з'являтися нові динамічно прогнозовані RISC-операції. Всі незавершені RISC-операції, що ще очікують свого виконання в станціях резервування (RS) і мають посилання на неправильно прогнозовані фізичні регістри, стають недійсними та звільняють RS.

Реалізація RAT

Архітектура процесорів x86-64 дозволяє проводити читання і записи з різними форматами регістрових даних: байти, півслова, слова, подвійні слова з цілочисловими регістрами загального призначення (наприклад: AL, AH, AX, EAX, RAX), що представляє проблему при перейменуванні регістрів. Ця проблема виникає, коли запис часткової ширини стався перед читанням регістрових даних з більшою довжиною формату. В цьому випадку дані, потрібні при читанні формату більшої довжини, можуть бути перекриті множиною попередніх записів до різних частин регістра.

Вирішення цієї проблеми вимагає, щоб RAT запам'ятовувала ширину кожного вхідного запису цілочислового формату, що туди надходить. Це виконується за допомогою додаткового 2-бітового поля для кожного запису, що надходить, в окремі банки менших і більших цілочислових форматів. Дворозрядне кодування допомагає розрізнити чотири розміри записів в регістри шириною: 64, 32, 16 та 8 біт. RAT використовує інформацію про розмір регістрів, щоб визначити чи необхідне більше регістрове значення, ніж воно було раніше записане. В деяких випадках RAT повинна генерувати призупинення при виконанні часткових записів.

Іншим випадком, поширеним при 16-бітовому кодуванні, є незалежне використання 8-бітових регістрових даних. Якби тільки одним псевдонімом підтримувався доступ до цілочислових регістрів для усіх розмірів, то незалежне використання 8-бітових піднаборів регістрів викликало б величезне число помилкових

залежностей. Наприклад, розглянемо послідовність RISC-операцій:

RISC0: MOV AL,#DATA1
RISC1: MOV AH,#DATA2
RISC2: ADD AL,#DATA3
RISC3: ADD AH,#DATA4.

RISC-операції 0 і 1 пересилають незалежні дані в AL і AH. RISC-операції 2 і 3 використовують операнди в AL і AH для виконання підсумовувань. Якщо для регістра "A" було б призначено тільки один псевдонім, то дані в AH операцією RISC1 перезаписувалися б поверх даних, що записуються операцією RISC0 в AL. Потім, коли RISC2 спробує прочитати AL, RAT вже не буде містити правильний показник і виконання RISC2 і RISC3 повинно бути зупинене, поки повністю не завершиться RISC1. Операнд-джерело для RISC2 буде втрачено внаслідок запису в AL. Для відновлення цієї інформації знадобилося б відновлення даних і послідовна обробка команд, що привело б до різкого падіння продуктивності.

Для запобігання цьому, в RAT підтримуються два цілочислові регістрові банки даних (меншого і більшого форматів). Для 32-бітових і 16-бітових доступів в RAT, дані читаються тільки з меншого банку, але запис проводиться в обидва банки одночасно. Для 8-бітових доступів, тільки відповідний більший або менший форматний банк проводить запис або читання, залежно від доступу до старшого або молодшого байту півслова (наприклад, до AH або AL). Тому, старший або молодший байти регістрів використовують різні записи надходжень, але обидва вони можуть бути перейменовані незалежно. Банк великих форматів містить тільки 15 масивів вхідних надходжень, тому що вісім 64-бітових цілочислових регістрів, а саме: R8 - R15, шість 16-бітових сегментних регістрів, а саме: CS, DS, ES, FS, GS, SS та 18-бітовий регістр станів EFlags, відповідно до специфікації x86-64, не розділяються на старші і молодші частини і не мають будь-яких часткових доступів. Масиви вхідних надходжень для шістнадцяти регістрів 128-бітових векторних операндів та операндів з плаваючою комою, а саме: XMM0 – XMM15 не мають часткових доступів, але послідовно можуть попарно об'єднуватися, утворюючи вісім регістрів для 256-бітових векторних операндів команд AVX.

Показники фізичних джерел операнда (PSrc) RAT вказує на комірки в регістровому масиві ROB, де знаходяться необхідні для обчислень дані. Фактично дані не з'являються в ROB, поки RISC-операція не сформує результат виконання і не запише його назад в ROB по шині зворотного запису. Поки не виконається зворотний запис на місце PSrc, ROB містить недійсні значення ("сміття").

Кожен запис в RAT має біт RRF, котрий вказує на місцезнаходження даних архітектурного регістру, що відповідає цьому запису: в RRF, чи в ROB (див. рис. 2). Якщо біт RRF встановлено, то дані знаходяться в RRF і фізична адреса в RAT вказує на відповідний регістр в RRF. Якщо біт RRF скинуто, то дані знаходяться в ROB і фізична адреса вказує на коректну позицію в ROB. По 10-бітовому полю фізичної адреси можна отримати доступ до будь-якого із записів в ROB. Шини влаштовані так, щоб RRF міг передавати початкові дані так само, як і ROB.

RAT здійснює перейменування логічних джерел-операндів (LSrc's) в RISC-операціях, що видаються з буферу черг RISC-операцій. Ці 24 джерела використовуються як індекси для цілочислового масиву RAT. RAT має 24 порти читання, щоб дозволити одночасне читання всіх LSrc's для чергового надходження RISC-операцій.

Після завершення фази читання, цілочисловий масив RAT має бути оновлений новими фізичними приймачами результатів (PDst's) від allocator, що є місцями запису результатів поточних оброблюваних RISC-операцій в ROB, чи RRF. Оскільки можливі залежності усередині циклів виділення приймачів, застосована пріоритетна схема для записів, щоб гарантувати, що для кожного приймача записується коректний PDst. Так, для прикладу з попереднього розділу: вищий: - фізичні приймачі результату в ROB для поточних RISC-операцій RISC3, RISC2, RISC1 та RISC0; нижчий – логічний приймач результату в RRF для будь-якої RISC-операції, що завершується та вилучається з ROB.

Розглянемо дії RAT при завершенні RISC-операції та вилученні з ROB її результату (retirement) і фіксації її кінцевих архітектурних станів (committing) у місці їх постійного зберігання у RRF. ROB повідомляє RAT, що місце розміщення результату приймача RISC-операції (PSrc), котра завершується, звільняється, а її результат повинен бути прийнятим в RRF. По цьому сигналу співпадаючий запис, або мно-

жина записів, в RAT із отриманим значенням поля PSrc від ROB анулюється і надалі звернення до відповідного логічного регістру вказуються на RRF (див. рис. 2).

Механізм вилучення RISC-операції вимагає, щоб в RAT виконалися асоціативні зіставлення отриманого значення поля PSrc від ROB з полем PSrc для кожного з 8 цілочислових масивів RAT. Для усіх знайдених збігів скидаються відповідні записи у масивах, що надалі вказують на RRF. У вилучення є найвищий пріоритет в пріоритетному механізмі зворотного запису – вилучення повинно відбутися раніше, ніж будь-яка нова RISC-операція проведе зворотний запис в ROB. Тому, записи результатів виконання RISC-операцій у ROB здійснюються після скидання RISC-операцій, що вилучаються.

Скидання в перейменуваннях регістрів з плаваючою точкою (FP) ускладнене через застосування стекової організації FP-регістрів в Intel Architecture x86-64. Для використання вершини стека (top-of-stack – TOS) в посиланнях на FP-регістри передбачені спеціальні апаратні ресурси. В RAT підтримується таблиця FP-регістрів, що знаходяться в RRF (retirement FP RAT table – RfRAT), яка містить інформацію про стан регістрового стека для операцій з плаваючою точкою. Кожен запис в RfRAT має 5-бітову ширину: 1 біт виділено як біт недійсності регістру при його вилученні із стека (retired stack valid) і 4 біта для показника адреси регістра в RRF. Застосування RfRAT забезпечує можливість відновлення з неправильно передбачених галужень шляхом запам’ятовування в спеціальному буфері виконаних з динамічним прогнозуванням макрооперацій FXCH.

Макрооперація FXCH з операндом переставляє (виконує свопінг) вершини стека регістрів з плаваючою точкою з «операндом», ставлячи в вершину стека номер регістра, що є операндом в макрооперації FXCH, а номер регістра, що до цього розміщувався у вершині – розміщує у стеку на місці «операнда». Макрооперація FXCH без операнда виконує циклічний зсув стеку на одну позицію вгору.

Макрооперація FXCH могла б бути реалізована як три RISC-операції MOV із застосуванням регістрів для тимчасового зберігання проміжних результатів. Але починаючи з процесора Pentium код обчислень з плаваючою точкою оптимізовано під використання макрооперації FXCH для впорядкування даних та їх подальшої обробки виконавчими блоками. Використання трьох RISC-операцій для реалізації свопінгу FXCH збільшило б продуктивність процесора, але погіршило б оптимізацію коду при обчисленнях з плаваючою точкою, тому в модельованій мікроархітектурі макрооперація FXCH реалізована за допомогою однієї RISC-операції.

Обробка RISC-операції FXCH в RAT здійснюється як перестановка елементів масиву RfRAT. У зв’язку з тим, що ця RISC-операція не потребує ресурсів виконавчих пристроїв, FXCH маркірується як “завершена” в ROB, як тільки ROB отримує її від RAT. Таким чином, FXCH не забирає ресурси RS і виконується за цикл.

В RfRAT може з динамічним прогнозуванням переставлятися будь-яке число її записів, перш ніж станеться неправильне передбачення галуження. При цьому всі раніше видані на виконання RISC-операції повинні бути анульованими. Після неправильно передбаченого галуження ROB видає сигнал, вказуючи, що всі наступні RISC-операції, починаючи з RISC-операції неправильно передбаченого галуження, мають бути вилучені. Це означає, що стан мікроархітектури має бути відновлений до машинного стану, що існував під час неправильного передбачення галуження. Таке відновлення в RAT здійснюється шляхом запам’ятовування всіх динамічно прогнозованих FXCH та інших RISC-операцій з плаваючою крапкою в спеціальному буфері. Вміст цього буферу, або скидається при підтвердженні передбачення, або використовується для відновлення масиву RfRAT при неправильному передбаченні галуження з наступним скиданням вмісту цього буферу. При неправильному передбаченні галуження регістри-приймачі всіх RISC-операцій буферу, крім FXCH, вилучаються із стеку (біт retired stack valid відповідних регістрів в стекові установлюється в “0”).

Перевизначення джерел операндів в RISC-операціях

Коли PDst RISC-операції, що вилучається, все ще викликається RISC-операціями, які тільки проводять читання з таблиць в RAT, в якості PSrc, то ці виклики переадресовуються до відповідного регістру в RRF. Це означає, що при вилученні RISC-операція, що завершується, повинна мати пріоритет над RISC-операціями, які тільки проводять читання з таблиць. Для

цього RISC-операція, що вилучається, виконується як обхідна після апаратного табличного читання. Таким чином, прочитані з таблиць дані будуть перевизначені (overridden) самою останньою інформацією в поточних RISC-операціях.

Механізм перевизначення вимагає виконання асоціативного зіставлення значення обох PSrc всіх RISC-операціями, які проводять читання з таблиць зі всіма показниками PDst, що вилучаються з таблиць в RAT.

Для здійснення одночасного перейменування LSrc's і LDst's у всіх одночасно оброблюваних RISC-операціях та забезпечення широких можливостей швидкого перевизначення PSrc's, RAT реалізовано відповідно до схеми, представленої на *Рис. 3.*

Розроблювана схема RAT, крім вищерозглянутих перевизначень PSrc's, здійснює також перевизначення PSrc's спрямовані на подолання залежностей RAW серед одночасно оброблюваних RISC-операцій. Розглянемо роботу RAT, коли серед одночасно оброблюваних RISC-операцій є наступні RISC-операції:

RISC0: r1 + r3 -> r3

RISC1: r3 + r2 -> r3

RISC2: r3 + r4 -> r5

В цьому прикладі джерело операнда r3 в RISC1 залежить від приймача результату RISC-операції RISC0 за тим же посиланням. Це означає, що між цими двома RISC-операціями існує залежність RAW. Для її подолання дані з r3 необхідні для RISC1 будуть записані в комірку RS, де буде набувати стану готовності RISC1, із зворотної шини запису результату від функціональних пристроїв після виконання RISC0. Це буде здійснено як перевизначення прочитаного з таблиць RAT значення PSr для r3 на очікування операнда із зворотної шини запису результату від функціональних пристроїв після виконання RISC0. Аналогічно, для RISC2 буде здійснено перевизначення прочитаного з таблиць RAT значення PSr для r3 на очікування операнда із зворотної шини запису від виконання RISC1, що визначається за допомогою allocator.

RAT може призупиняти заміну джерел операндів, за зовнішніми та внутрішніми призупинками.

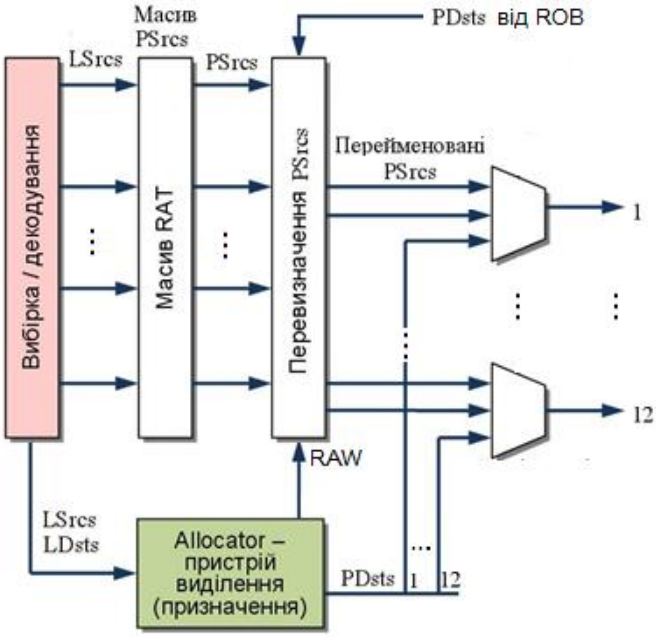


Рис. 3. Одночасні перейменування LSrc's і LDst's та перевизначення PSrc's в RAT

Внутрішні призупинки відбуваються, якщо RAT не може повністю обробити поточний набір RISC-операцій внаслідок наявності часткового регістрового запису в одній із RISC-операцій та наступного читання більшої частини цього ж регістру іншою RISC-операцією набору. В цьому випадку RAT повинен виконати призупинку, поки останній запис часткової довжини необхідного регістра не буде видалено з ROB та здійснено його перенесення в RRF. RAT виконує цю функцію, підтримуючи інформацію про розмір: 8, 16, 32 та 64 біта для кожного регістрового псевдоніма регістрів: A, B, C і D; 16, 32 та 64 біта для кожного регістрового псевдоніма регістрів: Source Index, Destination Index, Base Pointer, Stack Pointer і Instruction Pointer. Згідно специфікації Intel Architecture x86-64 інші регістри не мають часткових записів.

Зовнішнє призупинення RAT відбувається, коли allocator не може обробити чергову порцію RISC-операцій внаслідок відсутності вільних рядків в RS, або комірок в ROB.

Пристрій призначення ресурсів

У кожному тактовому циклі allocator здійснює виділення ресурсів для надходжень, що записуються в: ROB, RS, буфер завантажень (LB – Load Buffer), а також в буфер запам'ятовувань (SB – Store Buffer). Для цього він декодує RISC-операції, що надходять від ID, для визначення потрібних для них ресурсів. У тому випадку, коли allocator не може здійснити виділення потрібних ресурсів для чергової пор-

ції RISC-операцій, подальша видача RISC-операцій призупиняється, поки не стають доступними достатньо ресурсів для подальшого виконання шляхом вилучення з обробки попередніх RISC-операцій.

Призначення для ROB

Для ROB allocator визначає PDst's – номери фізичних регістрів, куди будуть записуватися результати виконаних RISC-операцій. Ці призначення використовуються RAT для створення відповідних записів в таблицях альтернативних імен архітектурних регістрів. RAT використовує PDst's, щоб безпосередньо адресувати ROB. Це означає, що якщо ROB заповнений, allocator повинен встановити сигнал призупинення роботи RAT для запобігання перезапису дійсних даних у ROB.

Для роботи з ROB allocator підтримує список вільних комірок в ROB. Адреси ROB для RISC-операцій allocator призначає послідовно від 0 і до найвищої адреси, а потім пошук вільної адреси для призначення знову починається з 0.

Призначення для станцій резервування

Для RS allocator призначає номери вільних рядків в RS, формує для них біти дозволу запису нових надходжень і передає цю інформацію до RAT. Якщо RS не мають вільних рядків, то allocator виставляє сигнал "призупинення" для запобігання перезапису дійсних даних в RS. Фактично, якщо RS заповнені, то біти дозволу будуть скинуті і, таким чином, ніякий запис, що надходить, не буде дозволено.

Призначення рядків RS здійснюється не так, як призначення для буферу ROB. Оскільки RS диспетчеризують RISC-операції для виконання в позачерговому порядку – у міру готовності їх даних, записи RISC-операцій, що надходять до рядків RS, зазвичай розподілені відповідно до порядку звільнення цих комірок від раніше записаних в них RISC-операцій. Таким чином, модель циклічної буферизації тут не працює. Замість цього використовується схема бітової матриці (побітового відображення), де кожен запис, що поступає, в RS відображається на біт в накопичувачі розміщень в RS. Таким чином, записи в RS можуть заміщатися у будь-якому порядку. Allocator шукає вільні рядки в RS, скануючи бітову матрицю від місця завершення виділення для попередньої порції RISC-операцій і поки не знаходить потрібну кількість перших вільних рядків в RS.

Станції резервування

Станції резервування (RS) у своїй основі є місцем, де RISC-операції, очікують надходжень коректних операндів (тобто поки не вирішаться залежності між RISC-операціями та стануть готовими операнди для їх виконання) і доки не вивільняться потрібні виконавчі пристрої (EU). У кожному тактовому циклі RS здійснюють визначення: достовірності операндів в готових до виконання RISC-операціях, наявності готовності потрібних EU та виконують планування позачергової обробки RISC-операцій. Це планування, або диспетчеризація RISC-операцій, полягає в організації їх відправки на виконання в EU та управлінні ланцюгами обходу (data bypassing) даних, минувши RS і EU, по паралельних обхідних ланцюгах за допомогою мультиплексорів. Надходження в RS можуть містити послідовності RISC-операцій від всіх одночасно оброблюваних потоків, але кожна RS може містити тільки ті RISC-операції, котрі можуть бути виконані на підключених до її портів EU. Всього ядро містить три RS: RS операцій з пам`яттю, RS операцій з фіксованою точкою, RS операцій з плаваючою точкою та з XMM (SSE) регістрами.

Кожна з RS складається із рядків (див. *Рис. 4*). В кожному з рядків зберігається одна RISC-операція, яка набуває в ньому стану готовності, очікуючи своїх операндів. Отримавши операнди, рядок RS, подібно до "активних комірок пам'яті" в архітектурі Деніса [1] "вистрілює на виконання" в EU. Для цього кожен рядок RS розділений на поля.

RISC-операції надходять в рядки RS від RAT. Номер рядка для надходження RISC-операції виділяє allocator на підставі отриманих повідомлень від RS про звільнення рядків (з бітом Busy = 0). При надходженні RISC-операції в рядок RS, значення його біта зайнятості Busy змінюється на протилежне.

Крім того кожен рядок RS містить: поле операції, що має виконатися; поля для запису значень операндів; біти наявності в комірці дійсного значення першого операнда Valid1 та другого – Valid2; біт готовності Ready, який вказує, що всі операнди для цієї RISC-операції надійшли і вона стає готовою до виконання. Рядок містить також поле тегу PDst, що видається разом з результатом виконання RISC-операції на шину зворотного запису і використовується для запису результату в комірку ROB та в інші рядки RS в якості операнда. Для команд завантаження/ запам'ятовування в цьому полі знаходиться адреса.

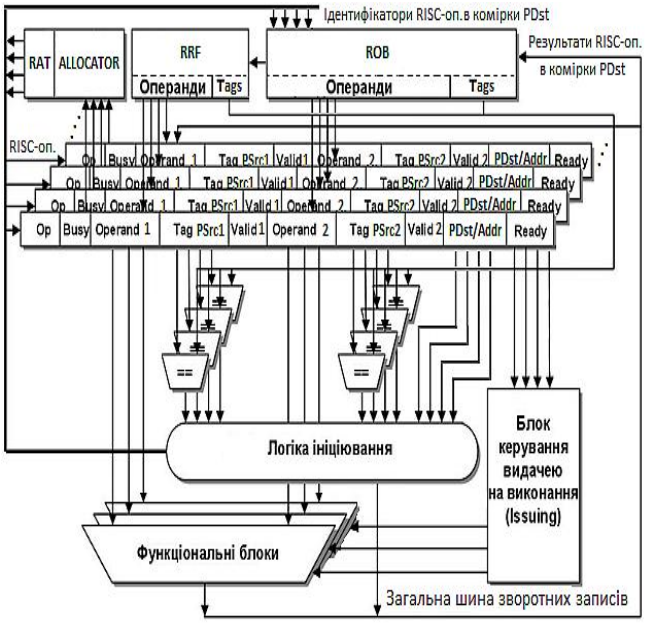


Рис. 4. Надходження RISC-операції до рядків RS, набуття готовності та видача їх на виконання

RISC-операція залишається в RS, поки не стануть дійсними всі джерела її операндів. Чергова порція операндів для RISC-операцій, що зберігаються в RS, може надходити в кожному такті по одинадцяти портам: чотирьом від ROB, чотирьом від RRF і трьом від шини зворотних записів масиву EU. Пристрої адресації пам'яті по вмісту тегів операндів PSrc (content addressable memories – CAMs) в RS управляють "захопленням" операндів з цих портів та записом їх в відповідні поля RISC-операцій з відповідним значенням Src. Запис операнда в поле RISC-операції відбувається при співпадінні бітів тегу фізичного джерела PSrc RISC-операції з бітами тегу фізичного приймача PDst на шині відповідного порту. При цьому RS виконують також наступні операції:

- відшуковують вільні рядки зі значенням біту зайнятості Busy=0 та передають їх номери до allocator;
- встановлюють біт зайнятості рядку Busy=1 при запису в нього нової RISC-операції;
- встановлюють біт наявності в рядку першого операнда Valid1=1 чи біт наявності в рядку другого операнда Valid2=1 при запису в поле операндів рядку значення відповідного операнда;
- встановлюють біти Valid1=Valid2=1 при запису в поле операндів значення операнда для одноперандної RISC-операції;
- формують біт готовності рядку Ready = Valid1&Valid2;
- здійснюють моніторинг бітів Busy всіх пристроїв з масиву EU;
- видають RISC-операцію для виконання в пристрій з масиву EU з бітом Busy=0 з рядку, що має біт Ready = 1 та потребує для свого виконання цього блоку, встановлюючи значення бітів рядку Busy= Valid1=Valid2 = 0.

RS визначають наявність вільних EU та готових до виконання, RISC-операцій, що потребують для обробки цих EU. Для цього вони використовують пріоритетний показник для визначення рядка початку пошуку готових до виконання RISC-операцій. Пріоритетний показник змінюється згідно алгоритму псевдо-FIFO. Це застосовується для збільшення продуктивності RS.

За один тактовий цикл RS можуть диспетчеризувати до дванадцяти RISC-операцій: три операції з плаваючою точкою, три операції з фіксованою точкою, чотири операції формування адрес (AGU) і дві операції запам'ятовування даних (STD).

Деякі RISC-операції можуть диспетчеризуватися для виконання більш ніж до одного порту в RS. Дія по фіксації такої RISC-операції до конкретного порту називається прив'язкою. Прив'язка RISC-операції в RS до порту EU, пов'язаного з відповідним інтерфейсом, виконується за допомогою алгоритму вирівнювання балансу завантаження, при якому відомо скільки RISC-операцій в RS чекають своєї черги на виконання через цей інтерфейс. Цей алгоритм використовується тільки для RISC-операції, які можуть бути виконаними одним з декількох EU. Така диспетчеризація називається "статичної прив'язкою з вирівнюванням балансу завантаження" готових до виконання RISC-операцій.

Буфер переупорядковування (ROB)

Цей циклічний буфер застосовується в мікроархітектурі RISC-ядра суперскалярного процесора для забезпечення двох найважливіших функцій обробки команд: перейменування регістрів і позачергового виконання команд. В деякій мірі ROB аналогічний регістровому файлу в процесорній частині чергового виконання команд, але з додатковими функціями підтримки вилучення команд з приведенням у відпові-

дність всіх архітектурних станів після перейменувань регістрів та динамічно прогнозованого виконання RISC-операцій.

ROB підтримує позачергове виконання, буферизуючи результати виконання RISC-операцій з масиву EU, перш ніж остаточно привести видимі архітектурні стани у відповідність до цих результатів.

ROB містить усі виконувані RISC-операції від усіх виконуваних в даний момент потоків, у тому числі й службового потоку, що знаходяться в виконавчому конвеєрі. Тобто команди, які були диспетчеризовані, але ще архітектурно не завершилися. Вони включають усі RISC-операції, що знаходяться в RS, усі RISC-операції, що знаходяться на стадії виконання в масиві EU, а також ті RISC-операції, що вже завершили своє виконання та їх результати знаходяться у ROB, очікуючи свого завершення та вилучення з одночасним приведенням у відповідність архітектурних станів виконуваного потоку, відповідно до його початкового програмного порядку. Визначення черговості завершення RISC-операцій відбувається на підставі місця її зберігання у ROB, відносно покажчика початку (голови) ROB (див. *Рис. 5 а*) та значення ідентифікаторів RISC-операцій, що містять інформацію про порядок їх декодування (див. *Рис. 5 в*).

Стан кожної команди у ROB може відстежуватися за допомогою декількох бітів в кожному записі до ROB, що надійшов. Кожна команда тут може знаходитися в одному з декількох станів: очікуванні виконання, виконанні операцій і завершенні виконання. У міру того як команда переходить від одного стану до іншого, ці біти станів оновлюються. Одним з бітів є біт динамічно прогнозованого стану для позначення стану прогнозованого напрямку галуження. ROB здійснює перевірку умови галуження. Коли напрям галуження був визначений правильно, раніше динамічно прогнозовані команди відзначаються вже не динамічно прогнозованими.

В іншому випадку, коли напрям галуження був некоректним, результати виконання RISC-операцій становляться недійсними і відкидаються. Завершуватися з приведенням у відповідність усіх архітектурних станів можуть тільки не динамічно прогнозовані RISC-операції.

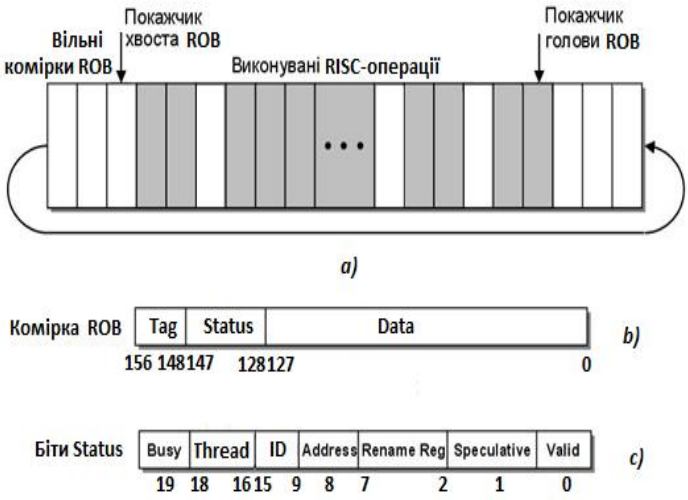


Рис. 5. Структура ROB
a) - завершення і вилучення RISC-операцій з ROB, починаючи з його «голови», та надходження нових RISC-операцій в ROB, починаючи з його «хвоста»;
b) - структура комірки ROB;
c) - структура поля Status в комірці ROB

При неправильно передбаченому галуженні, або виключенні під час виконання CISC-команди, пристрій управління ROB не змінює архітектурні стани у відповідності до отриманих результатів з масиву EU, а просто відкидає результати динамічно прогнозованих RISC-операцій. Для цього він помічає їх тегом Valid = 0 і відправляє allocator повідомлення про звільнення відповідних комірок ROB.

Ключовою функцією ROB є керування процесом вилучення або завершення виконання RISC-операцій. Для підтримки перейменувань регістрів використовується буферний накопичувач результатів з масиву EU. Виконавчі блоки записують дані результатів обчислень в регістри перейменувань. Логіка вилучення з ROB оновлює архітектурні регістри на підставі вмісту кожної перейменованої копії архітектурного регістру, суворо дотримуючись програмного порядку обчислень. RISC-операції, що набувають стану готовності в RS, джерела операндів яких містяться в архітектурних регістрах, можуть отримувати вміст або фактичного архітектурного регістра з RRF, або вміст перейменованого регістра з ROB. Оскільки мікроархітектура RISC-ядра суперскалярна, різні RISC-операції, що використовують в одному циклі один і той же архітектурний регістр, фактично можуть отримати доступ до різних фізичних регістрів.

ROB підтримує out-of-order – позачергове виконання RISC-операції в масиві EU, дозво-

ляючи виконавчим блокам завершувати свої RISC-операції і зворотно записувати результати обчислень незалежно від одночасного виконання інших RISC-операцій. Логіка вилучення RISC-операцій з ROB оновлює архітектурні стани під час вилучення у початковому програмному порядку дотримання, добуваючи потрібну інформацію з ідентифікаторів RISC-операцій, згенерованих декодерами команд і занесених в виділені allocator комірки ROB, минаючи RS і EU.

ROB активний в трьох окремих частинах процесорного конвеєра: на стадіях перейменувань і читання регістрів регістрового файлу; на стадіях виконання і зворотного запису та на стадіях вилучення з приведенням у відповідність архітектурних станів.

ROB сполучений і розташований у безпосередній близькості з allocator і RAT. Allocator виділяє фізичні регістри в ROB для підтримки виконання динамічно прогнозованих операцій і перейменуванням регістрів у RAT. Фактичним перейменуванням архітектурних регістрів в ROB керує RAT. Пристрої allocator і RAT функціонують в схемній частині чергової обробки команд процесорного конвеєра. Таким чином, стадії перейменування і читання регістрів з ROB виконують функції послідовності програмного потоку.

Інтерфейс ROB сполучений з RS, масивом EU та RRF. Дані, прочитувані з ROB в RRF впродовж конвеєрної стадії читання регістрів, можуть бути джерелами операндів RISC-операції в RS, що набувають стану готовності. Ці операнди запам'ятовуються в комірках RS, поки RISC-операції ще не диспетчеризувалися для EU. Виконавчі блоки проводять зворотні записи результатів виконання RISC-операцій в ROB через десять портів зворотного запису (шість портів виділено повністю для зворотних записів, чотири для часткових зворотних записів при виконанні STD і STA). Зворотні записи результатів проводяться в позачерговому порядку відносно порядку декодування RISC-операцій.

Результати обчислень від EU разом з обчисленими бітами регістра EFLAGS (Status register) записуються в комірку ROB, на яку вказує PDst виконуваної RISC-операції. Якщо виявляється, що виконана RISC-операція була помилково динамічно прогнозованою, тоді біти регістра EFLAGS анулюються разом з цією RISC-операцією і іншими виявленими помилково динамічно прогнозованими RISC-операціями. Інакше, ROB аналізує біти регістра EFLAGS RISC-операції що вилучається разом з повідомленням про можливе виключення, закодоване в ідентифікаторі RISC-операції. Якщо виключення має місце, ROB викликає відповідну дію обробки виключення, перш, ніж прийняти рішення про передачу результату виконаної RISC-операції до RRF для приведення у відповідність архітектурних станів.

Обробник вилучення має інтерфейс, що поєднує його з MS. За допомогою цього інтерфейсу ROB повідомляє MS про виникнення виключення, змушуючи його перейти до певної підпрограми мікрокоду обробника виключень.

В конвеєрі процесорного ядра ROB активний на різних стадіях його роботи. Він задіяний на одній з стадій роботи конвеєрного блоку попередньої обробки. На цій стадії ROB отримує від RAT ідентифікатори RISC-операцій, що згенеровані декодерами команд, і заносить їх в виділені allocator свої комірки. Ці ідентифікатори ROB використовуватиме при завершенні і вилученні RISC-операцій, щоб це завершення і вилучення відбувалось відносно до первинного програмного порядку. Для зменшення ширини записів в RS, а також скорочення обсягу інформації, яка має бути виданою в EUs, ці ідентифікатори RISC-операцій заносяться прямо в комірки ROB, куди в наступному будуть занесені результати виконання цих RISC-операцій.

Відразу після розміщення в RS чергової послідовності RISC-операцій, з ROB читаються до RRF результати попередніх RISC-операцій, завершених відповідно до програмного порядку їх виконання. Ці результати можуть бути одночасно операндами для RISC-операцій, що набувають стану готовності в RS. Взагалі операнди для RISC-операцій можуть надходити від одного з трьох місць: від RRF, від ROB, або з шини зворотного запису від масиву EUs.

На конвеєрній стадії читання результатів з EUs результати виконаних RISC-операцій, разом з обчисленими бітами регістра EFLAGS, записуються в комірки ROB (перейменовані фізичні регістри) відповідно до значень PDst виконаних RISC-операцій. Ця стадія зворотного запису відокремлена від стадій перейменування і читання регістрів, тому що RISC-операції видаються з RS в позачерговому порядку. Арбітраж шини зворотного запису здійснюється арбітром блоку EUs.

На конвеєрній стадії завершення виконання RISC-операцій логіка управління вилученням з

ROB фіксує архітектурні стани в RRF відповідного потоку та звільняє комірки ROB, надсилаючи повідомлення про адреси звільнених комірок до allocator. Конвеєрна стадія вилучення відокремлені від стадії зворотного запису, тому що зворотні записи проводяться в позачерговому порядку обробки RISC-операцій відносно початково заданого програмного порядку. Вилучення фактично зворотно переупорядковує RISC-операції та приводить їх у відповідність до чергового порядку виконання машинних команд. Операції вилучення проводиться за два тактові цикли та є повністю конвеєризованими.

Логіка управління вилученням з ROB працює з ним як з FIFO, починаючи з RISC-операцій, які раніше були розміщені згідно з конвеєром в послідовному порядку черги FIFO. При вилученнях з ROB логіка управління вилученням також аналізує ідентифікатори RISC-операцій, що згенеровані декодерами команд. Це гарантує, що вилучення відповідатиме початковому програмному порядку виконання машинних команд і не змінить дійсні машинні стани.

ROB реалізовано як багатопортовий регістровий файл з окремими портами для розміщення записів полів RISC-операцій, необхідних для подальшого вилучення, зворотних записів від EU, читань з ROB операндів для RS і читань для логіки управління вилученнями з ROB.

Кожен з одночасно оброблюваних потоків команд мають свій окремий RRF, де містяться архітектурні регістри, що відображують архітектурні стани оброблюваного потоку. Не всі дійсні стани процесорного RISC-ядра при обробці відповідного потоку зосереджені в RRF, але будь-який перейменований стан там є.

Блок обчислення адреси наступного командного вікна видає архітектурний показник команд для оброблюваного потоку команд.

Коли ROB визначив, що ядро почало виконувати операції по помилковому шляху галуження, то будь-яким RISC-операціям по цьому шляху не дозволено вилучатися зі зміною архітектурних станів оброблюваного потоку. Замість цього встановлюється сигнал “очищення” та відбувається видалення відповідних RISC-операцій.

Висновки

Досліджена реалізація RDF в ядрі суперскалярного процесора, який здійснює одночасне декодування на RISC-операції командного вікна розміром 32 байта одного з 4 оброблюваних потоків CISC команд з набору x86-64. Декодування командних вікон здійснюється почергово, відносно оброблюваних потоків. Декодування командних вікон, відносно команд окремо оброблюваного потоку, здійснюється з прогнозом їх галуження і позачерговим запуском на виконання. В модельованому ядрі застосована децентралізована схема пристрою контролю та управління процесами виконання команд в конвеєрі, яка розподілена по всіх сегментах конвеєра.

У наслідок буферизації операндів в RS можна уникнути зупинок конвеєра, пов'язаних з погрозами збоїв WAR, WAW і RAW.

Уточнено кількість рядків у кожній з трьох станцій резервування досліджуваного ядра та кількість комірок в ROB.

Список посилань

1. J. Dennis: Data Flow Supercomputers; IEEE Computer, pp. 48-56, Nov. 1980.
2. M. Simone, A. Essen, A. Ike, A. Krishnamoorthy, T. Maruyama, N. Patkar, M. Ramaswami, M. Shebanow, V. Thirumalaiswamy, D. Tovey (1995). Implementation trade-offs in using a restricted data flow architecture in a high performance RISC microprocessor. New York. pp. 151-162.
3. 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.
4. Hennessy John L., Patterson David A. Computer Architecture. A Quantitative Approach: Fifth Edition.-USA: Morgan Kaufman, 2012.-708.

УДК 004.724.4

КОРОНЕНКО А.М.

ДОСЛІДЖЕННЯ ДИНАМІКИ НАВАНТАЖЕННЯ МУЛЬТИСЕРВІСНОЇ МЕРЕЖІ

В статті виконані дослідження трафіка мультисервісних мереж, наведено результати статистичних замірів двох базових станцій (БС) у м. Києві (Україна). На основі аналізу отриманих даних зроблено висновок про неоднорідність характеру навантаження. Визначено концепції мультисервісної мережі.

The paper studies carried traffic multiservice networks, given the results of statistical measurements of two base stations (BS) in Kyiv (Ukraine). Based on the analysis of the data, concluded that the nature of heterogeneity load. Determined that the concept of multi-service network.

Вступ

Нині в телекомунікаційних мережах спостерігається тенденція зміни характеру та обсягу переданого трафіка, а також надаються інфокомунікаційні послуги, а саме послуги передачі даних, голосу, відео; такі мультимедійні послуги, як IP-телебачення, надання відео за запитом, IP-телефонія, відео- та аудіо-конференції, отже, для надання перерахованих послуг необхідно дотримуватися низки вимог до параметрів якості обслуговування. Найчастіше незнання статистичних характеристик трафіка призводить до неефективного використання мережевих ресурсів операторів а, отже, до низької якості наданої послуги або до низької кількості обслуговуваних абонентів.

Мультисервісні мережі зв'язку є об'єктами високої структурної складності, їхній розвиток дозволив надавати користувачам послуги з фіксованою і зі змінною шириною смуги бітової швидкості передачі, та зі змінними межами пропускної здатності, внаслідок чого виник цілий ряд факторів, що ускладнюють побудову таких мереж [1]. До найважливіших факторів належать: різні профілі трафіка, гетерогенний характер трафіка, якість обслуговування користувачів різних класів. Зважаючи на вище наведене дослідження динаміки навантаження мультисервісної мережі є актуальним. Передача трафіка з дотриманням вимог щодо якості обслуговування є не менш актуальним завданням у сучасних телекомунікації [2]. Неефективне використання ресурсів мережі, велика кількість абонентів і жорсткі вимоги до параметрів QoS можуть стати причинами падіння якості послуг, що надаються в мультисервісних мережах.

Постановка задачі

Огляд досліджень і розробок: В роботі [2] була запропонована стохастична параметризована модель трафіка на основі раніше відомої стохастичної кривої. Запропоновано метод, який дозволяє отримати оптимальні (за коефіцієнтом використання мережі) параметри цієї моделі при відомих вимогах до якості обслуговування і властивостей мультисервісної мережі. Перевага розробленої моделі в порівнянні з традиційними підтверджується проведеними розрахунками та моделюванням з використанням реального мультимедійного трафіка. В роботі [3] розглядаються методи оцінки якості обслуговування в мультисервісних мережах передачі даних зі змішаним типом трафіка. Аналізуються базові фактори, що впливають на якість обслуговування: затримки і втрати. Дані фактори досліджуються в зв'язку з різними типами потоків даних: еластичних, нееластичних. Дослідження [4], в якому запропоновано механізм пріоритезації трафіка і подальшої його обробки в вузлах мережі. Критерієм оптимізації є зменшення затримки пакета в буфері пристрою.

Для реалістичної побудови мультисервісної мережі розглянемо два відомі варіанти: через локальну обчислювальну мережу та через АТМ-мережі. В першому випадку різного роду трафік передається з використанням єдиного обладнання і з урахуванням різних вимог до смуги пропускання. Протоколи резервування ресурсу формування пріоритетних черг і якості обслуговування QoS дозволяють розподіляти послуги, що надаються для різного типу трафіка. Недоліком цього варіанту є те, що інфраструктура не задовольняє функціональні вимоги для локальних обчислювальних мереж, наприклад, в мережі з комутацією пакетів буде відсутнє забезпечення хорошої якості обслуговування, з високими вимогами до полоси пропускання та затримкою, а для нечутливого до затримки трафіка не підійде мережа, орієнтована на тип асинхронної системи передачі. Цю проблему частково можна вирішити, якщо розширити смугу пропускання та проектувати з урахуванням оптимізації використання ресурсу для певного типу

трафіка. Другим варіантом побудови мультисервісної мережі є розгортання АТМ-мереж. Недоліком цього варіанту є проблеми з оптимальним розподілом смуги пропускання для користувачів, плануванням ресурсів, з мережевим адмініструванням, через велику кількість з'єднань, де кожне з них має свій рівень якості обслуговування (QoS). Також необхідно забезпечити сумісність з існуючими інфраструктурами.

Найважливішим питанням є якість обслуговування користувачів та ефективність мережі в цілому, але залежить від збалансованості використання мережевих ресурсів. Саме балансування трафіка з точки зору QoS досягатиметься вздовж різних шляхів передачі трафіка при реалізації багатошляхової стратегії маршрутизації. Підвищення узгодженості в розв'язку задач маршрутизації можна досягти шляхом сумісного розв'язку в єдиній моделі ТКС інших задач управління трафіком, доступом, чергами та резервуванням ресурсів. Для того, щоб вирішити будь-яку задачу відносно мультисервісної мережі, необхідно чітко визначити фундаментальні властивості, переваги, вимоги, можливості та підтримка яких видів послуг в мультисервісній мережі є. В таблиці №1 наведено основні параметри мультисервісної мережі.

Табл. 1. Концепція мультисервісної мережі

Переваги	Фундаментальні властивостями	Підтримка таких видів послуг	Вимоги	Можливості
Надання сучасних високошвидкісних сервісів та мультипротокольна підтримка	Підтримка великого набору послуг, програм та механізмів	Міська комп'ютерна мережа з постійною швидкістю 100 Мбіт/с	Незалежність технологій надання послуг від транспортних технологій	Забезпечення створення, розгортання та управління будь-якого виду служб
Масштабованість	Відділення процесу надання послуги від самої мережі та забезпечення відкритих інтерфейсів, а також функцій управління від можливостей транспортного середовища	Обмін різного роду інформацією між користувачами та доступ до ігрових серверів	Гнучка та динамічна зміна швидкості передачі інформації в широкому діапазоні залежно від поточних потреб користувача	Чіткий поділ між функціями служб і транспортними функціями, надання сервісів, незалежно від типу мереж та доступу, що використовуються
Сумісність з міжнародними стандартами, доступ за загальноприйнятими інтерфейсам, підтримка традиційних мережевих технологій	Взаємодія з успадкованими мережами за відкритими інтерфейсами, різні схеми ідентифікації, які можуть бути реалізовані з використанням IP-адресації в цілях маршрутизації по IP-мережам	Створення віртуальних корпоративних мереж (VPN), комутованих і керованих користувачем	Здатність мережі передавати різного роду інформацію. З необхідною синхронізацією цих компонентів у реальному часі і з використанням складних конфігурацій з'єднань	Вирішення проблем мовних служб, якості обслуговування (QoS), безпеки
Управління якістю обслуговування (QoS)	Пакетний перенос	Високошвидкісний доступ до Internet (до 10 Мбіт/с)	Управління послугою, викликом і з'єднанням з боку користувача або постачальника послуг	Функціональні елементи політики управління, сеансів і т.д. мають бути розподілені по інфраструктурі, включаючи також існуючі мережі
Управління графіком та класифікація видів графіка	Широкосмуговий доступ із забезпеченням якості з кінця в кінець і «прозорості»	IP-телефонія	Організація доступу до послуг незалежно від технології, що використовується	Здійснення міжмережевої взаємодії між мультисервісними та існуючими мережами за допомогою шлюзів
Резервування смуги пропускання	Узагальнена мобільність та конвергенція послуг між мережами фіксованого та рухомого зв'язку	Об'єднання віддалених корпоративних мереж	Участь кількох операторів в процесі надання послуги і поділ їхньої відповідальності відповідно до області їхньої діяльності	Підтримка існуючих і «призначених для роботи з мультисервісними мережами» кінцевих пристроїв

Виклад основного матеріалу дослідження

Стабільна і якісна робота мультисервісної мобільної мережі залежить від її правильного проектування і можливості реконфігурації при її експлуатації. Вибір величини навантаження при проектуванні мережі ґрунтується на статистичних даних, які було отримано з двох базових станцій, розташованих на вулиці Берковецька та вулиці Підгірній в місті Києві, Україна. На рис.1 показано розташування цих базових станцій відносно одна одної. Пункт А – це базова станція по вулиці Берковецькій, а пункт Б – по вулиці Підгірній. Такі станції були обрані, щоб підтвердити, що трафік лишається фрактальним, як в центральних густонаселених районах, так і у віддалених районах міста.

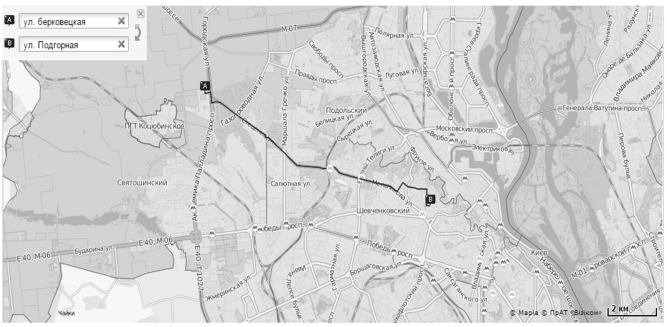
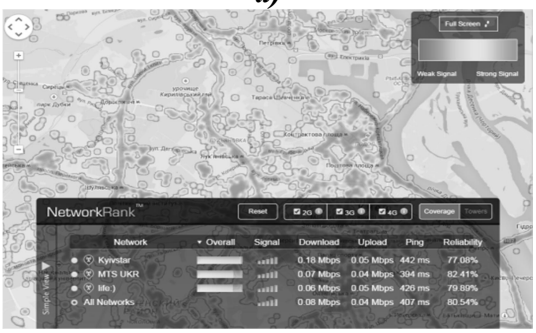


Рис.1. Відстань між базовими станціями

На карті (рис.2) видно статистику для обраної області, де представлено сигнальні середні значення для всіх мереж, які містяться у вказаній області, програмою NetworkRank виділено кольорове маркування, щоб показати області сильного і слабкого сигналу. На рис.2(а) видно, що по вулиці Берковецькій основне навантаження зосереджене вздовж основних доріг: це і будинки, розташовані над дорогою, і мобільні пристрої, якими користуються в транспорті, що рухається дорогою. Найліпший показник надійності зв’язку тут належить MTS UKR – 94,92%, life:) – 94,67%, UA-KYIVSTAR - 93,02%, UTEL3G – 91,80%.

На рис.2(б) видно, що по вулиці Підгірна навантаження зосереджене вздовж основних доріг, будівель типу торгівельних центрів, станцій метро, зупинок транспорту – це значні скупчення, які видно на карті, також присутні і будинки, розташовані над дорогою, і мобільні пристрої якими користуються в транспорті, що рухається дорогою. Найліпший показник надійності зв’язку тут належить MTS UKR – 82,41%, інші – 80,54%, life:) –79,89%, UA-KYIVSTAR – 77,08%.

а)

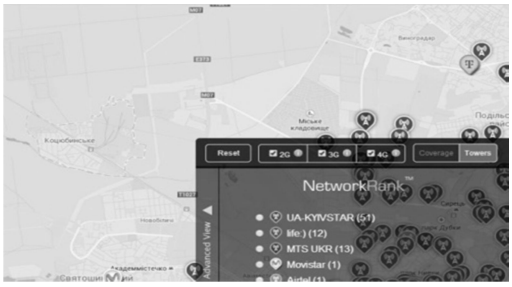


б)

Рис.2. Статистика (а - БС вул. Берковецька, б - БС вул. Підгірна)

Щільність розташування базових станцій в цих географічних областях можна побачити на рис. 3. Очевидно, що рівень зв’язку падає, чим далі мобільний пристрій знаходиться від базової станції, тому для околиць м. Києва зв’язок або зовсім відсутній, або переходить від однієї базової станції до іншої з дуже поганою якістю. Не останню роль в такому стані речей грає проектування та розташування базових станцій, але сучасне телекомунікаційне обладнання спроможне передавати сигнал на досить велику відстань та тривалий час підтримувати QoS (якість обслуговування).

Рухомість мобільного пристрою (МП) в мобільній мережі може спричинити виникнення хендовера, оскільки зміна IP-адреси при зміні БС мережі вимагає оповіщення. Якщо хендовер відбувається коректно, то МП зберігає нерозривність з'єднання під час його пересувань і зміни БС мережі. Відомо два типи хендовера, які визначаються можливістю підключення МП до БС: м'який хендовер і жорсткий хендовер. При жорсткому хендовері поточне з'єднання обривається до того, як обслуговування МП передано до нової БС, це відбувається, коли в суміжних БС різні частотні діапазони і МП не може підтримувати зв'язок з обома БС одночасно, внаслідок чого може статися втрата пакетів. У випадку м'якого хендовера з'єднання з попередньою БС розривається тільки після встановлення з'єднання з доступною БС, МП може одночасно взаємодіяти з обома БС.



а)

б)

Рис.3. Щільність розміщення (а - БС вул. Берковецька, б - БС вул. Підгірна)

Для внесення змін при експлуатації в якості вихідних даних використовуються вже відомі результати вимірювань працюючої мережі. Однак, і в першому, і в другому випадках використовуються усереднені значення, які отримують при статистичних спостереженнях за навантаженням, при цьому виділяючи основні характеристики - навантаження, створене потоком викликів, тобто середню тривалість розмови, інтенсивність навантаження і кількість занять каналів, кількість викликів абонентів за проміжок часу, тривалість паузи між викликами і час обслуговування виклику.

Дослідження характеристик мереж мобільного зв'язку багато разів проводилися раніше, однак вони не проводились для сучасних мультисервісних мобільних мереж, тому не відображають сучасну реальність, що приводить до локальних перевантажень на ділянках мережі.

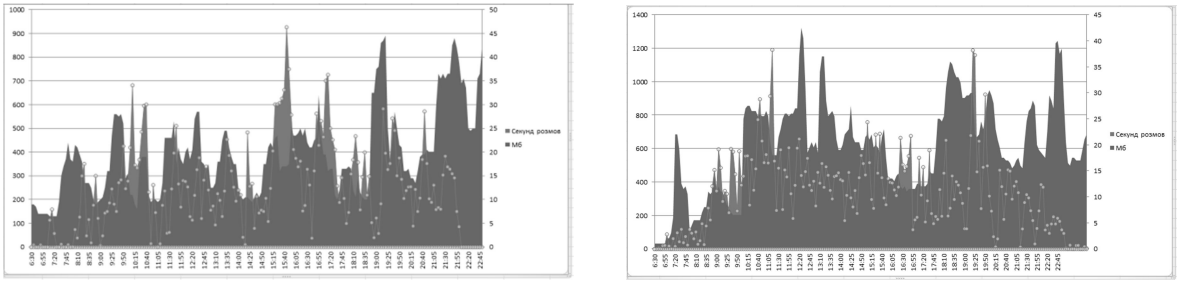
Ця стаття є результатом проведеної дослідницької роботи на мережі оператора мобільного зв'язку PEOPLEnet з вивчення параметрів мультимедійного навантаження в сучасних умовах. При цьому досліджувався характер усього навантаження, що обслуговується протягом місяця квітня 2014 року безперервно кожні 5 хвилин. Вказане дослідження проводилося в м. Києві на вулицях Берковецькій та Підгірній. На момент проведення дослідження проводилися масові заходи, при проведені яких кількість абонентів перевищувала пропускну здатність стільника. Проведення вимірів навантаження здійснювалося шляхом зняття облікової інформації, що надійшла. Ці дані відображають тільки ту частину викликів, які обслуговуються оператором. Однак у випадках, коли абонент зайнятий або він не відповідає, тобто коли виклик не завершений тарифікованою розмовою абонентів, ця частина викликів залишається неврахованою. Таким чином, якщо порівняти враховане, тобто навантаження, що було опрацьоване, то воно виявиться менше того, що реально надійшло. При можливості локального перевантаження мережі найбільш вірогідні випадки отримання відмови в обслуговуванні, коли мережа зайнята через відсутність вільних каналів (тобто зайняті всі або зарезервовані для хендоверних абонентів канали). В цьому випадку виклик, що надійшов, взагалі залишиться незафіксованим навіть за допомогою програмного забезпечення.

На рис. 4 наведено добову діаграму розподілення навантаження з 5-ти хвилинним інтервалом з двох БС. Це другий день тижня – вівторок. Результати досліджень щоденної завантаженості БС показали (рис. 4), що навантаження для повсякденної діяльності в будні дні досить стабільні і зміни незначні. Також з цих рисунків видно, що, не зважаючи на розташування в різних районах з різною кількістю активних користувачів МП, трафік має фрактальний характер. Інваріантна до масштабу, пульсуюча структура є характерною особливістю мобільних мультисервісних мереж.

З рис. 4 видно, що навантаження мережі змінюється протягом доби і цілком відображає повсякденний розпорядок життя [6]. Для встановлення залежності довжини черги від кількості зарезервованих каналів ми використали чисельне моделювання в середовищі MATLAB. Стандартний Пуассонівський потік викликів займав доступні канали, які звільнялись з постійною інтенсивністю. При перевищенні кількості доступних каналів надлишок викликів потрапляв у чергу. Експеримент проводився протягом достатньо великого часу, щоб середня довжина черги стабілізувалася. Експеримент проводився при різній кількості доступних каналів, результати наведені [5]. На рис. 5 наведено добову діаграму розподілення навантаження з 5-ти хвилинним інтервалом з двох БС. Це шостий день тижня – субота. У

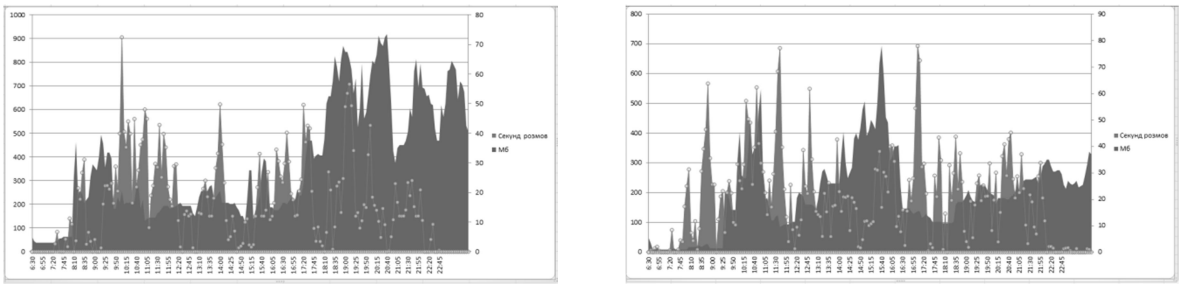
вихідні дні динаміка процесу така: на обох графіках видно, що з 7:20 зростають голосові виклики до 7:30, далі поступово починають зростати о 7:45 і спадають о 8:10, далі знову зростає о 8:35 і спадає о 9:10. Виклики даних починають поступово зростати з 8:10, а потім різко зростають о 9:25 і починають спадати об 11:05. Голосові виклики різко зростають о 9:20 і досягають піку о 10:00 і поступово йдуть на спад до 12:20. В цей час починають зростати виклики даних і досягають свого піку о 15:30 і спадають до 16:30. Невелике зростання голосових викликів відбувається з 12:25 до 12:20 і далі з 13:10 до 13:35. Потім знову зростає з 13:40 і поступово спадає до 14:50. Наступне велике зростання викликів даних відбувається о 16:40 і триває до 20:30, а потім знову зростає з 20:50 і триває до 22:30, і останній приріст викликів відбувається з 22:45 і до 23:30. Щодо голосових викликів, то зростання відбувається з 15:00 і до 16:30, а потім знову різко зростають, наближаючись до найбільшого приросту викликів о 16:40, а спадають о 17:30. Наступне невелике зростання можна прослідкувати о 17:45 і до 18:05, а також з 18:10 до 18:15. Після цього о 18:20 можна прослідкувати зростання, яке спадає о 19:05. Невелике зростання з 19:10 до 19:15, після якого відбувається зростання голосових викликів о 19:20 і спадає о 19:55. З 20:10 до 20:15 знову невеликий приріс голосових викликів, наступний в 20:20, де після різкого зростання з 20:50 починає поступовий спад до 21:25. Два останніх прирости голосових викликів спостерігається з 21:30 до 21:50 і з 21:55 до 22:15.

Таким чином стає очевидно, що динаміка викликів різниться, а місцями голосові виклики повністю спадають і в цей час зростають виклики даних, чергуючись та змінюючи один одного, тому орієнтуватись та створювати моделі розподілення навантаження без урахування того, що динаміка по викликах не збігається, неефективно. Окрім цього, порівнюючи статистику, де відображена динаміка навантаження вихідного дня на двох БС, можна побачити, що вона відрізняється тим, що виклики обох типів починаються значно пізніше. Для викликів даних це 8:10 ранку, а голосові виклики закінчуються о 22:15.



а) *б)*

Рис.4. Динаміка навантаження мережі протягом доби 01/04/14
(а - БС вул. Берковецька, б - БС вул. Підгірна м. Київ)



а) *б)*

Рис.5. Динаміка навантаження мережі протягом доби 06/04/14
(а - БС вул. Берковецька, б - БС вул. Підгірна м. Київ)

Для докладнішого розгляду надходження на БС потоку викликів виділимо два види викликів, які створюють навантаження на БС фіксованого стільника (рис. 6): 1. новий виклик, що виник всередині розглянутого стільника і завершився тут; 2. новий виклик, що виник

всередині розглянутої стільника і перейшов на обслуговування в сусідній стільник.

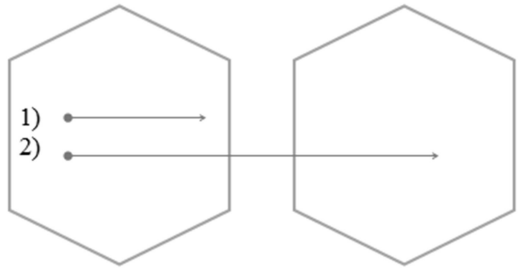


Рис.6. Два види викликів, що надходять на БС стільника

Оскільки в рамках цієї статті ми не розглядаємо випадки, де є хендовер-виклик, що надійшов у розглянутий стільник з сусіднього стільника і завершився в розглянутому стільнику; хендовер-виклик, що надійшов у розглянутий стільник з сусіднього стільника і перейшов на обслуговування в сусідній стільник.

На відміну від моделі двопотокової повнодоступної системи масового обслуговування (СМО) [6] з втратами, де непередбачена зона хендовера та резервування каналів для обслуговування хендовер-викликів, то неповнодоступна модель з безкінечною чергою і нетерплячими заявками дозволяє передбачити зону хендовера. Перейдемо від фізичної моделі до математичної, отже, припустимо, що потоки нових і хендовер-викликів є пуассонівскими потоками (ПП) інтенсивності λ_o (original calls) і λ_h (handover calls) відповідно. Таким чином, сумарний потік викликів, що створюють навантаження на базову станцію стільника, є пуассонівським потоком інтенсивності $\lambda = \lambda_o + \lambda_h$. Будь-який виклик, що обслуговується БС з інтенсивністю μ_1 завершує обслуговування всередині стільника і з інтенсивністю μ_2 переходить у сусідній стільник. Отже, тривалість обслуговування БС виклика, який завершить обслуговування всередині розглянутого стільника, має експоненціальний розподіл з параметром μ_1 , а тривалість обслуговування виклика, якому належить хендовер, - експоненціальний розподіл з параметром μ_2. Таким чином, тривалість заняття каналу є експоненціальною розподіленою випадковою величиною (ВВ) з параметром $\mu = \mu_1 + \mu_2$. Число каналів у стільнику дорівнює C . Застосовується стратегія доступу з резервуванням: на БС стільника g каналів, призначені для обслуговування нових та хендовер-викликів, а решта $C - g$ каналів зарезервовані тільки для обслуговування хендовер-викликів. Передбачено наявність зони хендовера, в якій МП може перебувати не більше випадкового часу, що має експоненціальний розподіл з параметром λ . Якщо в момент, коли надходить виклик на обслуговування, число каналів більше $C - g$ і $0 \le g \le C$, тоді канал займається, інакше губиться. Наступний виклик, якщо немає вільного каналу, очікує на звільнення каналу. Обслуговування відбувається по типу FIFO. Виклик, який очікує в черзі, може покинути СМО з інтенсивністю μ_1, що відповідає закінченню розмови в зоні хендовера, а також з інтенсивністю γ, що відповідає блокуванню хендовер-виклика при спробі передачі обслуговування з сусіднього стільника у розглянутий стільник. Тривалість обслуговування викликів є незалежними випадковими величинами, що мають експоненціальний розподіл з параметром μ . Ймовірність B_o блокування нового виклику відповідає ймовірності π_1 втрати 1-виклику. Відповідно до [6] можна визначити ймовірність втрат, довжину черг, ймовірність блокування B_h.

Для підвищення якості необхідно розрахувати кількість каналів при заданих параметрах довжини черги.

Для цього розрахуємо:

$$Q \sim e^{1/(N-N_0)} \qquad (1)$$

Де N — кількість каналів, N_0 — мінімальна кількість каналів, при якій черга ще не росте необмежено. Величину $N - N_0$ можна сприймати як надлишок каналу — наскільки канал більший за мінімальну величину, при якій всі виклики ще можуть бути опрацьовані.

В реальній мережі голосовий трафік та трафік даних має різний пріоритет. Як показують звіти національних операторів [7], типове співвідношення між показником відмов для даних і голосових викликів становить приблизно 4:1. Таким чином, при оптимізації розподілення каналів ми розраховуємо на різну типову довжину черги для викликів даних і голосових викликів:

$$Q_d = aQ_v \qquad (2)$$

Де a — показник відносного пріоритету, типове значення близько 4.

Зі співвідношення (2) та обмеження на загальну кількість каналів $N = N_d + N_v$, можна за допомогою (1) отримати такий результат:

$$N_d = \frac{N}{2} + \frac{N_d^0 - N_v}{2} + \frac{1}{\ln\alpha} - \\ -\frac{1}{2}\sqrt{(N - N_v^0 - N_d^0) + \frac{4}{(\ln\alpha)^2}} \quad (3)$$

$$N_v = N - N_d$$

У ці формули входять невідомі N_v^0 і N_d^0. З рис. 4, 5 видно, що величина суттєво змінюється протягом доби і має динамічно розраховуватись на основі статистики завантаження каналів таким чином:

$$N_d^0 = \frac{1}{T}\sum_{i=1}^{k_1} \tau_d^i \quad (4)$$

$$N_v^0 = \frac{1}{T}\sum_{j=1}^{k_2} \tau_v^j \quad (5)$$

Де τ_d^i — тривалість i-го виклику даних, τ_v^j — тривалість j-го голосового виклику, T — час, за який ми розраховуємо поточне значення. N_d^0, k_1, k_2 вибираються такими, щоб в суму ввійшли тільки ті виклики, які пройшли через базову станцію за час T.

Висновок

Аналіз отриманих результатів показує, що сучасний трафік є фрактальним. Використання ефективного методу розподілення викликів дозволяє розрахувати кількість каналів при заданих параметрах довжини черги для реального трафіка. Показано, що створювати моделі розподілення навантаження без урахування того, що динаміка за викликами не збігається, неефективно.

Список посилань

1. Гургенидзе, А. Т. Мультисервисные сети и услуги широкополосного доступа / А.Т. Гургенидзе, В.И. Кореш. – СПб.: Наука и техника, 2003. – 400 с.
2. Трещановский П.А. Оптимизация стохастической модели трафика для мультисервисных сетей / П. А. Трещановский // Инженерный вестник Дона. – 2011. – № 3. – С. 1-8.
3. Кравцова Н.А. Модель оценки качества обслуживания в мультисервисных сетях передачи данных, позволяющая описать агрегированный поток информации, потери и задержки / Н. А. Кравцова // Известия ОрелГТУ. – 2007. – №4-3/272(550). С. 74-78.
4. Алиев Р.Т. Методы управления трафиком в мультисервисных сетях / Р. Т. Алиев // Научно-технический вестник СПбГИТМО (ТУ). – 2002. – Выпуск 6. – С. 10–13.
5. Короненко А.М. Метод ефективного динамічного розподілення каналів між голосовими викликами та даними / А. М. Короненко // Electronics and Communications. – 2014. – № 4(81). – С. 83-89.
6. Гайдамака Ю.В. Модели обслуживания вызовов в сети сотовой подвижной связи / Ю.В. Гайдамака, Э.Р. Зарипова, К.Е. Самуйлов . –М.: Издательство Российского университета дружбы народов, 2008 – 72 с.
7. Звіт за договором № 233 від 08 липня 2013 року на надання послуг з проведення вимірювань показників якості послуг рухомого (мобільного) зв’язку в регіонах України [Електронний ресурс] / Науково-технічний центр “ОМНІС” – Режим доступу: http://nkrzi.gov.ua/images/upload/96/4446/34f5f93a82c0dd259e5accf04ce12482.pdf

УДК 681.518

ЗИМЧУК И.В.

РЕДУКЦИЯ МОДЕЛЕЙ ОБЪЕКТОВ УПРАВЛЕНИЯ ДЛЯ СИНТЕЗА ПРОСТЫХ РЕГУЛЯТОРОВ

Изложено методику редуцирования передаточных функций объектов управления, которая основывается на близости частотных характеристик исходной и редуцированной моделей в области частот, определяющих динамические свойства системы. Приведён пример решения задачи редукции с результатами математического моделирования.

Methodology of reduction the transmission functions of management objects is expounded. In basis of methodology the closeness of frequency descriptions is fixed by initial and reduction models in area of frequencies qualificatory dynamic properties of the system. An example of decision of task of reduction is made with the results of mathematical design.

Постановка проблемы

Наличие математической модели объекта управления является необходимым условием для синтеза устройств управления динамическими объектами разных типов. В большинстве случаев такой математической моделью является передаточная функция. При этом сложность регулятора зависит от порядка знаменателя передаточной функции объекта управления. На практике наибольший интерес вызывают регуляторы низкого порядка с фиксированной структурой (простые регуляторы) [1]. Упрощение регулятора актуально не только снижением вычислительной сложности, но и возможностью формировать алгоритмы управления на базе более простых контроллеров [2].

Одним из способов построения простых регуляторов является редуцирование передаточной функции объекта управления [1]. Суть редукции заключается в замене реальной передаточной функции, которая описывает динамику объекта, передаточной функцией низшего порядка, как правило второго [3, 4]. При этом упрощение должно проводиться так, чтобы суммарное значение энергий, моментов или некоторых других характеристик, которые накапливаются в течение переходного процесса, изменялись несущественно [4].

Редуцированию математических моделей объектов управления посвящено ряд публикаций, среди которых работы [1–5]. Однако использовать аналитические подходы [1, 2, 4] не всегда приемлемо из-за сложности математических преобразований, а применение численных методов, которые реализуются с помощью ЭВМ, предусматривает наличие соответствующего программного обеспечения, которое реализует тот или иной численный метод [3, 5].

Альтернативными методами решения задачи редуцирования модели объектов управления являются те, которые не предусматривают сложных математических расчётов [3]. Наиболее простой способ заключается в разложении передаточной функции объекта на элементарные динамические звенья и отбрасывании тех множителей, которые не содержат наибольшую постоянную времени. Если передаточную функцию объекта разложить на звенья не удаётся, то находят корни его характеристического уравнения. Большим значениям корней соответствуют быстро затухающие компоненты переходного процесса, которые мало влияют на поведение системы. Поэтому, корнями, которые на порядок больше остальных, пренебрегают.

Другой подход заключается в замене постоянных времени одним суммарным запаздыванием и сокращении нулей и полюсов исходной передаточной функции объекта [3]. Однако таким подходам присуща низкая точность из-за того, что редукция осуществляется лишь манипулированием постоянными времени и не предусматривают влияние коэффициента преобразования.

В связи с этим в работе предлагается несколько иной подход к решению задачи редукции передаточных функций объектов управления, которому не свойственные указанные замечания.

Формулировка задачи исследования

Задача редукции математических моделей объектов управления ставится следующим образом. Предполагается, что реальная переда-

точная функция описывается выражением

$$K_0(p) = \frac{k_0}{p(1+2\xi T_k p + T_k^2 p^2)} \times \times \frac{1}{\prod_{i=1}^{m}(1+T_i p)}, \quad (1)$$

где k_0 - коэффициент преобразования;

p - оператор Лапласа;

T_i - постоянная времени *i*-го инерционного звена;

T_k, ξ - постоянная времени и коэффициент затухания колебательного звена;

m - количество инерционных звеньев.

Принимая во внимание необходимость сохранения порядка астатизма объекта, в качестве редуцированной [3] выбирается передаточная функция следующего вида

$$K(p) = \frac{k}{p(1+Tp)} \quad (2)$$

здесь k и T - коэффициент преобразования и постоянная времени, которые необходимо рассчитать.

Критерием качества редуцирования является равенство значений интегральных оценок качества переходного процесса исходной J_0 и редуцированной J моделей объектов управления

$$J_0 = \int_0^\infty \varepsilon^2(t)dt, \quad (3)$$

где $\varepsilon(t)$ - рассогласование между входным воздействием и значением переходной характеристики $h(t)$ замкнутой единичной обратной связью системы автоматического управления.

Изложение основного материала

Известно, что адекватность математических моделей исходного и редуцированного объектов, охваченых единичной отрицательной обратной связью, определяется степенью близости их частотних характеристик [3]. Это утверждение положено в основу предлагаемого подхода.

В переходном режиме свойства замкнутой системы автоматического управления определяются в области частоты среза ω_{cp} амплитудно-частотной характеристики разомкнутой части системы [6]. Поэтому реализовать редукцию представляется возможным сохранением у редуцированного объекта присущих реальной модели объекта значений частоты среза ω_{cp} и фазочастотной характеристики на этой частоте $\varphi_0(\omega_{cp})$ как параметров, определяющих показатели качества переходного процесса.

Значение частоты среза может определяться из решения уравнения:

$$K_0(\omega) = 1,$$

где $K_0(\omega)$ - амплитудно-частотная характеристика исходного объекта.

Такой подход предполагает выполнение соответствующих математических преобразований. Во избежание сложности математических расчётов, частоту среза предлагается определять путём построения асимптотической логарифмической амплитудно-частотной характеристики исходного объекта

$$L_0(\omega) = 20\lg K_0(\omega).$$

При выполнении равенства

$$L_0(\omega) = 0,$$

определяется частота среза.

Значение фазочастотной характеристики исходного объекта (1) на частоте среза рассчитываются следующим образом

$$\varphi_0(\omega_{cp}) = -\frac{\pi}{2} - \sum_{i=1}^{m} arctg(\omega_{cp}T_i) - - arctg\frac{2\xi T_k \omega_{cp}}{1-\omega_{cp}^2 T_k^2}. \quad (4)$$

Рассчитанные значения ω_{cp} и $\varphi_0(\omega_{cp})$ используются для расчёта неизвестных значений параметров редуцированной передаточной функции. В соответствии с (2) выражения для амплитудно-частотной $K(\omega)$ и фазочастотной $\varphi(\omega)$ характеристик редуцированного объекта принимают вид

$$K(\omega) = \frac{k\sqrt{1+\omega^2 T^2}}{\omega(1+\omega^2 T^2)}, \quad (5)$$

$$\varphi(\omega) = arctg\left(\frac{1}{\omega T}\right). \quad (6)$$

Подставляя в уравнения (5) и (6) значения ω_{cp} и $\varphi_0(\omega_{cp})$ при выполнении равенств

$$\varphi(\omega_{cp}) = \varphi_0(\omega_{cp}),$$

$$K(\omega_{cp}) = 1,$$

выражения для расчёта коэффициента преобразования и постоянной времени редуцированной передаточной функции принимают следующий вид

$$T = \frac{1}{\omega_{cp} tg(\varphi_0)}, \quad (7)$$

$$k = \frac{\omega_{cp}(1+\omega_{cp}^2 T^2)}{\sqrt{1+\omega_{cp}^2 T^2}}. \qquad (8)$$

Обобщая изложенное, методика расчёта параметров редуцированной передаточной функции (2) включает следующие положения.

1. Определение частоты среза из построения асимптотической логарифмической амплитудно-частотной характеристики исходного объекта.

2. Расчёт значения фазочастотной характеристики исходной передаточной функции объекта управления на частоте среза по выражению (4).

3. Расчёт постоянной времени и коэффициента преобразования редуцированной передаточной функции (2) по формулам (7) и (8).

Практическое использование методики рассматривается на примере. Предполагается, что передаточная функция объекта управления описывается выражением

$$K_0(p) = \frac{5}{p(1+0{,}5p)(1+0{,}1p)(1+0{,}05p)}.$$

Необходимо определить коэффициент преобразования k и постоянную времени T редуцированной передаточной функции (2).

По графику логарифмической амплитудно-частотной характеристики исходной модели (рис. 1) определяется частота среза $\omega_{cp} = 3 рад/с$.

По выражению (4) рассчитывается значение фазочастотной характеристики на частоте среза $\varphi_0(\omega_{cp}) = -170^0$. Подстановкой значений ω_{cp} и $\varphi_0(\omega_{cp})$ в выражения (7) и (8) рассчитывается редуцированная передаточная функция

$$K(p) = \frac{17{,}3}{p(1+1{,}89p)}.$$

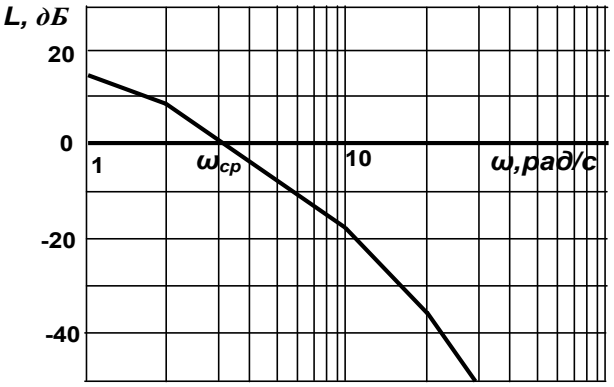


Рис. 1. Логарифмическая амплитудно-частотная характеристика исходной модели

Точность приближения полученного результата к исходной модели проводилась путём математического моделирования. Исследованию подлежали замкнутые единичной обратной связью системы автоматического управления, в которых передаточная функция разомкнутой системы представлена исходной (1) и редуцированной (3) передаточными функциями. Результаты моделирования в виде переходных характеристик систем управления с исходной моделью (1), и редуцированной для случав, когда постоянная времени аппроксимирующей модели равна обобщённой постоянной времени ($T = 0{,}65c$), а коэффициент усиления равен усилению объекта ($k = 5$), и когда величина коэффициента усиления и постоянная времени определялись по изложенной методике, представлены на рис. 2, 3 и 4 соответственно.

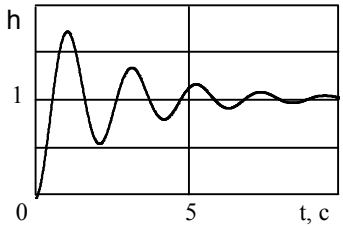


Рис. 2. Переходная характеристика исходной модели

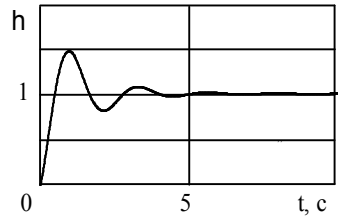


Рис.3. Переходная характеристика известной модели

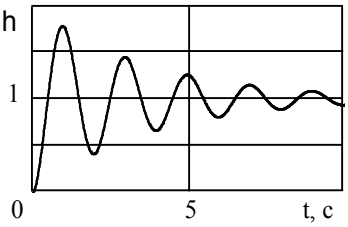


Рис.4. Переходная характеристика синтезированной модели

При этом значения интегральных квадратичних оценок качества переходного процесса (3) соответственно равны: $J_0 = 0{,}9899$; $J_1 = 0{,}4248$; $J_2 = 0{,}9732$.

Из полученных результатов видно, что свойства замкнутой системы, в которой реали-

зована редуцированная по изложенной методике передаточная функция объекта, приближаются к исходной модели наилучшим образом.

Выводы

В работе изложено методику редуцирования математической модели объекта управления, представленной в виде передаточной функции. Предложенный подход основывается на близости частотных характеристик исходной и редуцированной моделей и не предусматривает проведения сложных математических расчётов. Процесс редуцирования иллюстрируется примером, а степень близости полученного результата к исходной модели подтверждена результатами математического моделирования.

Список литературы

1. Бойченко В.А., Курдюков А.П., Тимин В.Н., Чайковський М.М., Ядыкин И.Б. Некоторые методы синтеза регуляторов пониженного порядка и заданной структуры. Управление большими системами / Сборник трудов. Выпуск 19. – М.: ИПУ РАН. – 2007. – С. 23 – 126.
2. Романова И. К. Современные методы редукции нелинейных систем и их применение для формирования моделей движущихся объектов. / Вестник МГТУ им. Н. Э. Баумана. Сер. “Машиностроение”. – 2012. – С. 122 – 133.
3. Гостев В.И., Стеклов В.И., Скляренко С.Н. Оптимальные системы управления с цифровыми регуляторами: Справочник.- К.: КИРЦ «Сенс», 1995. – 484с.
4. Изерман Р. Цифровые системы управления: Пер. с англ. – М.: Мир, 1984. – 541с.
5. Суздаль В.С., Епифанов Ю.М. Редукция модели при синтезе регуляторов для управления кристаллизацией / Восточно-европейский журнал передових технологий. – Выпуск №3. – Том 2. – 2011. – С. 31 – 34.
6. Арсеньев Г.Н., Зайцев Г.Ф. Радиоавтоматика. Ч. 1. Теория линейных непрерывных систем автоматического управления РЭС. Учебник для вузов. – М.: САЙНС-ПРЕСС, 2008. – 480с..

УДК 004.67

ГУСЕВ Е.И.

МАТЕМАТИЧЕСКОЕ МОДЕЛИРОВАНИЕ РАСПРЕДЕЛЁННОЙ КЛАСТЕРНОЙ СИСТЕМЫ ИСПОЛЬЗУЮЩЕЙ SHARED EVERYTHING ПОДХОД (ORACLE RAC)

Предложена математическая модель функционирования распределённой общей памяти на примере Global Cache Fusion (GCF). GCF – реализация Shared Everything подхода в Oracle RAC. Выделены основные характеристики трафика для работы с моделью, а также определена целевая функция для анализа эффективности алгоритмов оптимизации доступа к странице памяти в распределённой системе.

There are proposed a mathematical model of functioning distributed shared memory by the example of the Global Cache Fusion (GCF). GCF is a realization of Shared Everything approach implemented in Oracle RAC. Discovered fundamental characteristics of traffic for using the model, and also defined the objective function for analysis of effectiveness of access to memory page optimization algorithms in a distributed system.

Постановка задачи

Ключевыми задачами системы с точки зрения производительности является увеличение пропускной способности системы и уменьшение времени отклика. В работе [1] ключевым параметром, характеризующим пропускную способность многопользовательской СУБД является SESSTIME (ST) – суммарное время всех задержек во всех сессиях, кроме задержек действий клиента. Приведём формулу:

$$ST = \int_0^t \left(\nu(\tau_{prc})\right) dt = \int_0^t \left(\nu(kt_{sb})\right) dt$$

$$ST = \int_0^t \nu\left(k\left(t_{acc} + t_{prc}\right)\right) dt$$

где τ_{prc} – время обработки одного запроса

t_{sb} – время обработки одного блока

ν – интенсивность запросов к базе

k – количество обрабатываемых в одном запросе блоков

t_{acc} – время доступа к блоку

t_{prc} – время выполнения операций над блоком

Поскольку одной из ключевых задач системы с точки зрения производительности является уменьшение ST при неизменной интенсивности запросов, исследование возможности уменьшения t_{acc} и t_{prc} являются приоритетными задачами. В данной статье мы построим временную модель доступа к блоку в распределённой кластерной системе использующей shared everything подход на примере Oracle RAC. Также заметим, что уменьшение среднего времени доступа к блоку влияет не только на уменьшение ST, но и на время отклика, что представляет интерес в контексте исследования устойчивости на коллапс по блокировкам системы [1].

Важно выделить среди факторов, влияющих на среднее время отклика, базовые характеристики системы, базовые характеристики трафика и производные характеристики. Базовыми характеристиками системы будем называть факторы, зависящие ТОЛЬКО от архитектуры системы и физических возможностей, например, скорость передачи данных между нодами, количество нод или размер кэша на каждой ноде. Базовыми характеристиками трафика назовём характеристики зависящие ТОЛЬКО от набора интенсивностей классифицированных как по типу запроса так и по тому к какому ресурсу запрос. Например, общая интенсивность запросов, соотношение запросов чтения и записи, вероятность конфликта и т. д. И производными параметрами назовём характеристики, которые обусловлены взаимодействием трафика с системой. Например, вероятности состояний блока в кэше, вероятность попадания в кэш, среднее время блокирования и т.д.

1. Описание принципов алгоритма доступа к блоку в Oracle RAC

Следует сказать, что в реальных кластерных системах для взаимосвязи узлов кластера 2PC [4] и 2PL[5] протоколы не используются. Это связано с резким снижением производительности протоколов при увеличении интенсивности конфликта за данные [1,2]. Ключевой особенностью Oracle RAC является Shared Everything подход [3]. Это предполагает как общий диск, так и общую оперативную память на всех нодах. Однако, сразу оговоримся, что «общая память» – в данном случае – условное обозначение. На практике, каждая нода имеет собственную оперативную память, но

осуществляется пересылка блока (страницы) данных от одной ноды к другой по запросу, если блок закеширован на другой ноде [6]. При такой организации возникает задача отслеживания состояния блоков на всех нодах: а именно: какой блок на какой ноде закеширован, и на какой момент времени. Задача решается введением ноды владельца (мастер-ноды) для каждого блока. Все ноды будут слать запросы на изменение состояния блока к этой мастер ноде. Состояние блока может быть XCUR (eXclusive CURrent – эксклюзивное текущее), SCUR (Shared CURrent – разделяемое текущее), CR (Consitent Read – целостное чтение) и PI (Past Image exist – после изменений, удобен при восстановлении, чаще всего следующим действием сконвертируется в CR). Поясним состояния: если мы собираемся изменять блок – мы запрашиваем у мастер ноды блокировку на блок XCUR. Если собираемся его читать – то SCUR. Мастер нода отслеживает каким нодам выданы SCUR блокировки, а какой XCUR. Не более одной копии любого блока закэшированного в XCUR состоянии может быть в каждый момент времени. Для модификации блока, процесс должен получить на него состояние XCUR. CR предполагает, что блок актуален на некоторый момент времени в прошлом, но на данный момент существует более поздняя версия блока. PI – состояние в котором оказывается блок после состояние XCUR, т.е. другая нода запросила XCUR либо SCUR блокировку, и нода, ранее удерживающая XCUR, её отдала. Блок получается в состоянии PI, которое впоследствии может быть сконвертировано в CR. Важно сказать, что один и тот же блок может одновременно находиться на разных нодах в своём состоянии для каждой ноды. С точки зрения исследования, важно выделить 5 состояний блока, с учётом наличия его различных версий и состояний каждой версии на каждой ноде в глобальном кеше:

1. XCUR+n*CR: n – целое, неотрицательное число. В этом состоянии на одной ноде на одной из версий блока XCUR блокировка, а на остальных нодах – версии блока в CR. Количество CR версий блока на каждой ноде произвольно, столько, сколько версий блока понадобилось для работы ноде, причем, версии в CR состояниях могут быть и на ноде, где удерживается XCUR блокировка. Обозначим состояние xcur.
2. k*SCUR+n*CR: k – натуральное число, n – целое, неотрицательное число. На каждой ноде не более одной версии с SCUR блокировкой, но хотя бы одна версия блока в состоянии SCUR существует на какой-либо из нод. Одновременно с этим, в CR состояниях на всех нодах, может быть неограниченное количество версий блока. Обозначим состояние scur.
3. n*CR (XCUR или SCUR вылетел): В глобальном кэше не осталось XCUR или SCUR версий блока. Стоит заметить, что при первого обращения к блоку SCUR или XCUR запросом блок будет зачитан с диска для обработки, после которой останется в состоянии SCUR либо XCUR в зависимости от запроса. Однако, при запросе CR состояния – необходимости зачитывать с диска блок нет, если блок на нужный момент времени остался закэшированным. В данной статье рассмотрен классический GCF алгоритм, позволяющий повторно не собирать блок, а использовать ЛОКАЛЬНО закешированную версию в дальнейшем. Помимо классического подхода в качестве техник оптимизации может быть предложена техника поиска CR блока в ГЛОБАЛЬНОМ кэше. Если мастер нода сохраняет историю получения XCUR блокировок и историю CR запросов к ней она может указать на какой ноде существуют уже собранные на нужный момент CR блоки. Но, как было сказано выше, в этой статье рассматривается классический подход. Обозначим это состояние «cr only».
4. Блока нет в глобальном кэше. Обозначим состояние «нет».
5. Мастер нода при обращении к другой ноде обнаруживает, что ранее закэшированная в состоянии SCUR или XCUR версия блока вылетела. Обозначим состояние «вылетел».

2. Сценарии алгоритма доступа к блоку в Oracle RAC

Теперь перейдём к описанию последовательностей действий (сценариев), в контексте получения запроса блокировок. Стоит сразу отметить, что сценарии разные в случаях, когда не существует закешированного в глобальном кэше блока, существует один или несколько закешированных блоков с SCUR блокировкой и уже существует закешированный блок с XCUR блоки-ровкой на какой-либо из нод. Таких сценариев 9. Распишем эти сценарии:

Сценарий 1. В случае если блок незакеширован ни на одной из нод (о том что блок незакеширован мастер ноде известно) имеем следующий сценарий:

1. Запрос XCUR/SCUR блокировки у мастер-ноды

2. Ответ мастер ноды об отсутствии блока в глобальном кэше.

3. Чтение блока с общего диска.

Обозначим его как чтение с диска (read from disk, rd1). Распишем временную модель для сценария:

$$t_{rd1} = 2t_{net} + t_{io} \ (1)$$

где t_{io} – время чтения блока с диска

t_{net} – среднее время пересылки служебного сетевого пакета между нодами

Сценарий 2. Если же мастер нода имеет информацию о закешированном блоке, но при обращении оказывается, что блок вылетел из кэша, то сценарий такой:

1. Запрос XCUR/SCUR блокировки у мастер-ноды

2. Отправка ноде, где закеширован блок инструкции, переслать блок запросившей ноде.

3. Ответ ноды, где блок вылетел из кэша, мастер-ноде об отсутствии блока в кэше.

4. Ответ мастер ноды запросившей ноде об отсутствии блока в глобальном кэше.

5. Чтение блока с общего диска.

Обозначим его как чтение с диска (read from disk, rd2)

$$t_{rd2} = 4t_{net} + t_{io} \ (2)$$

Сценарий 3. Если же мастер нода имеет информацию о закешированном блоке и блок в глобальном кэше в состоянии SCUR, имеем:

1. Запрос XCUR блокировки у мастер-ноды

2. Мастер-нода отсылает всем нодам у которых этот блок в состоянии SCUR сообщение, что поступил запрос на изменение блока (XCUR) и с какой ноды.

3. Состояние блока на нодах где был SCUR, меняется на CR.

4. Одна из нод пересылает ноде, запросившей XCUR сам блок.

5. Нода, захватившая XCUR блокировку, подтверждает мастер ноде её захват.

Обозначим сценарий как запись-чтение (write-read, wr) и распишем временную модель. Поскольку начать обработку можно параллельно с последним шагом, имеем:

$$t_{wr} = 2t_{net} + t_{send} \ (3)$$

где t_{send} – время пересылки, включая время записи в память страницы от одной ноды к другой.

Сценарий 4. Если же мастер нода имеет информацию о закешированном блоке и блок в глобальном кэше в состоянии XCUR, имеем:

1. Запрос XCUR блокировки у мастер-ноды

2. Мастер-нода отсылает ноде, у которой этот блок в состоянии XCUR, сообщение, что поступил запрос на изменение блока (XCUR) и с какой ноды.

3. Ожидание завершения обработки

4. Состояние блока на ноде, где был XCUR, меняется на PI.

5. Нода, где был XCUR, пересылает ноде, запросившей XCUR, сам блок.

6. Нода, захватившая XCUR блокировку, подтверждает мастер ноде её захват.

Обозначим сценарий как запись-чтение (write-write, ww), как и в случае выше, обработка начинается одновременно с последним шагом. Имеем:

$$t_{ww} = 2t_{net} + t_{send} + t_{fp} \ (4)$$

где t_{fp} – время ожидания завершения обработки, из-за конфликта за блок

Аналогично распишем 2 сценария получения SCUR блокировки при наличии блока в глобальном кэше. Различие между ними в том, закешированы ли версии блоков с SCUR или XCUR блокировками.

Сценарий 5. Распишем случай, когда на одной или нескольких нодах закешированы версии блока с блокировкой SCUR:

1. Запрос SCUR блокировки у мастер ноды.

2. Пересылка мастер нодой ноде, на которой закэширован блок с SCUR блокировкой, инструкции переслать текущую версию блока запросившей ноде.

3. Пересылка блока с SCUR блокировкой запросившей ноде.

4. Пересылка подтверждения получения блокировки мастер ноде.

Обозначим этот сценарий как чтение-чтение (read-read, rr), также обработка начинается одновременно с последним шагом:

$$t_{rr} = 2t_{net} + t_{send} \ (5)$$

Сценарий 5. Если существует версия блока с XCUR блокировкой на какой либо из нод, то последовательность действий следующая:

1. Запрос SCUR блокировки у мастер ноды.

2. Пересылка мастер нодой ноде, на которой закэширован блок с XCUR блокировкой, инструкции переслать текущую версию блока запросившей ноде.

3. Ожидание завершения обработки

4. Пересылка блока запросившей ноде. При этом XCUR блокировка может быть конвертирована (понижена) в SCUR, и тогда ноде, запросившей SCUR – SCUR и будет предоставлен или же блокировка останется

XCUR, а запросившей ноде будет отдан блок в состоянии CR – но на текущий момент времени. Решение о понижении блокировки принимается на втором шаге сценария мастер-нодой. Отметим, что Oracle RAC понижает блокировку до SCUR в случае повторного запроса SCUR блокировки. С точки зрения временных задер-жек сценария время доступа остаётся неизмен-ным и не зависит от понижения блокировки при выполнении сценария. Однако условия пониже-ния блокировки влияют на то, в каком состоя-нии окажется блок после обработки запроса. Обозначим вероятность понижения блокировки как $p_{down.}$ Заметим, что в формулах этой статьи p_{down} встречаться не будет, тем не менее, выделим эту вероятность как производный параметр. При описании базовых характеристик трафика и вычислении производных параметров алгоритм принятия решения о понижении может быть предметом оптимизации, но в данной статье этот вопрос не рассматривается.

5. Пересылка подтверждения захвата блокировки мастер ноде.

Обозначим этот сценарий как чтение-запись (read-write, rw)

$$t_{rw} = 2t_{net} + t_{send} + t_{fp} \quad (6)$$

В системе существует также сценарий запись на диск (write-disk wd). Но с точки зрения анализа времени отклика или суммарных задержек во всех сессиях, сценарий не представляет интереса, поскольку отложенная запись на диск осуществляется фоновым процессом DBWn асинхронно по мере накопления изменённых блоков в кэшах нод. Анализ представляет интерес с точки зрения исследования конфликтов, поскольку во время записи на диск выставляется XCUR блокировка. В данном исследовании, поскольку задержки фоновых процессов мы не классифицируем как часть ST, время потраченное на фоновую запись на диск в итоговую формулу задержек не войдёт.

Два самых неприятных, с точки зрения моделирования сценария – это запросы CR состояния. Проблема в самой архитектуре Oracle. В данном исследовании мы будем считать что запрос CR состояния может быть преобразован либо в запрос нескольких SCUR блоков, либо, если блок для которого затребованный момент времени совпадает с временем хранимой в кэше копии, в сценарий чтения готового блока из локального (либо глобального) кэша. Таким образом, имеем сценарии:

Сценарий 8 (оптимизированный): нужный CR блок на нужный момент времени оказался в кэше (Обозначим сценарий как crp*)

Сценарий 9: Нужного CR блока не оказалось в кэше и пришлось его собирать. (Обозначим сценарий как crp).

Рассмотрим 9й сценарий (crp): Если в локальном кэше нет требуемого блока на запрошенный момент времени, то для реконструкции блока к моменту времени в прошлом должны быть задействовованы блоки UNDO-сегментов. Система выполняет 5 шагов:

1. Получение SCUR блокировки на изменяемый блок
2. Чтение блока для определения задействованных в изменении блока транзакций.
3. Определение UNDO сегментов, задействованных незавершенными транзакциями
4. Получение SCUR блокировок на блоки этих UNDO сегментов
5. Применение UNDO информации для восстановления блока на требуемый момент времени.

Фактически запрашивается состояние SCUR как на требуемый блок так и еще на один или несколько блоков UNDO – т.е. запрашивается несколько SCUR блоков. Увеличение количества обрабатываемых блоков, а именно получение на большее количество блоков SCUR блокировки, негативно сказывается на эффективности системы. Однако вероятность каскадного запрашивания блоков UNDO обусловлена только характером трафика. Будем считать, в контексте моделирования GCF, что характеристиками трафика является соотношение запросов SCUR, XCUR и CR*, а значит трафик с большим процентом каскадно запрошенных блоков UNDO, с точки зрения модели, рассматриваемой в этой статье – это трафик, в котором больший процент SCUR запросов по сравнению с XCUR и CR*.

Распишем в таблице 1 алгоритм доступа к блоку, описанный выше, и расставим соответствующие 8 сценариев – без сценария wd, поскольку к времени доступа к блоку в пользовательских сессиях он не имеет отношения:

Табл.1. Выбор сценария в зависимости от состояния и запроса состояния

		Существует в глобальном кэше				Нет
		xcur	scur	cr only	Вылетел	
Запрос состояния	cr*	crp*			-	-
	Xcur	ww	wr	-	rd2	rd1
	Scur	rw	rr	-		

Символом «-» в таблице обозначены запрещённые состояния, т.е. Запрос состояния невозможен при таком текущем состоянии в глобальном кэше. Запрещённые состояния возникают по той причине, что из трафика мы выделили подкласс, обрабатываемый crp алгоритмом. Этот подкласс обладает 2 свойствами: «блок должен быть запрошен в CR состоянии» и «блок запрошен на тот момент времени, на который существует копия блока в локальном кэше». Первое запрещённое состояние получается поскольку CR блок невозможно преобразовать в SCUR или XCUR. Поэтому наличие CR блоков никак не связано с получением SCUR и XCUR блокировок. То есть, если запрашивается XCUR и SCUR проверяется также SCUR и XCUR в глобальном кэше и игнорируется проверка есть ли кроме XCUR и CR состояния блоков. Второе запрещённое состояние получается по причине того, что наличие в локальном кэше (условие cr* состояния) автоматически означает его наличие и в глобальном – т.е. блок не может быть в состояниях «нет» (в кэше) и «вылетел» (из кэша).

Анализ временных задержек в различных сценариях доступа к блоку

Следует указать на то, что факторы локального кэша и локальной мастер-ноды существенно сокращают время обработки. Наличие в локальном (для ноды которая выполняет обработку) в нужном состоянии кэше приводит к тому что в сценариях ww, rr, wr и rw не нужно пересылать сам блок. С другой стороны, если нода, обрабатывающая запрос, является мастер-нодой – то не нужно запрашивать состояние блока по сети.

Распишем таблицу задержек с учетом этих условий. За основу возьмём табличную форму алгоритма, добавив в Таблицу 1 факторы локальной мастер-ноды и локального кэша.

Табл.2 Выбор сценария с учетом факторов мастер-ноды и локальноко кэша

	Запрос состоя-ния	Существует в глобальном кэше и в каком состоянии					вылетел	Нет
		xcur		scur		cr only		
		локально	нелокал.	локально	нелокал.			
Мастер	cr*	crp*					rd2	rd1
	xcur	ww	ww	wr	wr	-		
		rw	rw	rr	rr	-		
Не мастер	cr*	crp*					rd2	rd1
	xcur	ww	ww	wr	wr	-		
	scur	rw	rw	rr	rr	-		

В таблице выделены сценарии с классическим временем выполнения. Во всех остальных случаях будут отрабатывать сценарии оптимизации, связанные с фактором локальной мастер-ноды и (или) фактором локального кэша. Подставив времена выполнения из формул 1,2,3, 4, 5,6 занесём времена выполнения в таблицу 3, с учётом оптимизаций мастер ноды и локального кэша для оставшегося трафика после выделения специфического cr* подтрафика:

Табл.3. Времена выполнения сценариев

Запрос состояния		Существует в глобальном кэше				Вылетел	нет
		xcur		scur			
		л	г	л	г		
мастер	xcur	0	$t_{net}+t_{send}+t_{fp}$	0	$t_{net}+t_{send}$	$2t_{net}+t_{io}$	t_{io}
	scur	-(0)	$t_{net}+t_{send}+t_{fp}$	0	$t_{net}+t_{send}$		
немаст.	xcur	0	$2t_{net}+t_{send}+t_{fp}$	$2t_{net}$*	$2t_{net}+t_{send}$*	$4t_{net}+t_{io}$	$2t_{net}+t_{io}$
	scur	-(0)	$2t_{net}+t_{send}+t_{fp}$	0	$2t_{net}+t_{send}$		

* Указано время при отсутствии конфликта. При конфликте, сценарий ww и время обработки соответственно: $2t_{net}+t_{send}+t_{fp}$

С точно такой же структурой распишем вероятности попадания в каждое из состояний, на основе вероятностей ключевых факторов:

Табл.4 Вероятности попадания в сценарий

Запрос состояния		Существует в глобальном кэше				вылетел	нет
		xcur		scur			
		Л	г	л	г		
Мастер	xcur	$p_{gc}p_{lc}p_{xcur}$ $p_{rxcur}p_{mn}$	$p_{gc}(1-p_{lc})p_{xcur}$ $p_{rxcur}p_{mn}$	$p_{gc}p_{lc}p_{scur}$ $p_{rxcur}p_{mn}$	$p_{gc}(1-p_{lc})p_{scur}$ $p_{rxcur}p_{mn}$	$(1-p_{gc})p_{out}p_{mn}$	$(1-p_{gc})(1-p_{out})p_{mn}$
	scur	$p_{gc}p_{lc}p_{xcur}$ $p_{rscur}p_{mn}$	$p_{gc}(1-p_{lc})p_{xcur}$ $p_{rscur}p_{mn}$	$p_{gc}p_{lc}p_{scur}$ $p_{rscur}p_{mn}$	$p_{gc}(1-p_{lc})p_{scur}$ $p_{rscur}p_{mn}$		
Не мастер	xcur	$p_{gc}p_{lc}p_{xcur}$ $p_{rxcur}(1-p_{mn})$	$p_{gc}(1-p_{lc})p_{xcur}$ $p_{rxcur}(1-p_{mn})$	$p_{gc}p_{lc}p_{scur}$ $p_{rxcur}(1-p_{mn})$	$p_{gc}(1-p_{lc})p_{scur}$ $p_{rxcur}(1-p_{mn})$	$(1-p_{gc})p_{out}(1-p_{mn})$	$(1-p_{gc})(1-p_{out})(1-p_{mn})$
	scur	$p_{gc}p_{lc}p_{xcur}$ $p_{rscur}(1-p_{mn})$	$p_{gc}(1-p_{lc})p_{xcur}$ $p_{rscur}(1-p_{mn})$	$p_{gc}p_{lc}p_{scur}$ $p_{rscur}(1-p_{mn})$	$p_{gc}(1-p_{lc})p_{scur}$ $p_{rscur}(1-p_{mn})$		

где p_{gc} – вероятность попадания в глобальный кэш

p_{lc} – вероятность попадания в локальный для ноды обрабатывающей запрос кэш

p_{rxcu} – вероятность того, что запрос будет XCUR.

p_{rscur} – вероятность того, что запрос будет SCUR.

p_{xcur} – вероятность того, что состояние запрошенного блока XCUR

p_{scur} – вероятность того, что состояние запрошенного блока SCUR

p_{mn} – вероятность того, что нода выполняющая запрос является нодой-владельцем (мастер)

p_{out} – вероятность того, что блок оказался в состоянии «вылетел»

Заметим, что поскольку состояния «нет» и «вылетел» определяются p_{gc}, а необходимость использования блока в состоянии «cr only» определяется только во время запросов cr* с вероятностью p_{cr*}, и, поскольку, сценарии рассматривающие срабатывание при вероятностях p_{gc} и p_{cr*} независимы от p_{scur} и p_{xcur} – события нахождения блока в состояниях p_{scur} и p_{xcur} не зависят от p_{gc} и p_{cr*}, а значит и $p_{scur}+p_{xcur}=1$. В контексте того, что после выделения cr* подтрафика из общего трафика, и учитывая что crp запрос распадается на комбинацию rscur и rxcur запросов, имеем $p_{rxcur}+p_{rscur}=1$.

Таким образом, с учетом того, что приведенные выше таблицы 3 и 4 исключают CR* сценарий, имеем 21 сценария, попадание в каждый из которых определяется вероятностями, вычисляемыми по приведенным в таблице 4 формулам на основе 7-ми базовых вероятностей: p_{cr*}, p_{gc}, p_{lc}, p_{rxcur}, p_{xcur}, p_{mn}, p_{out}.

Итоговая формула

Комментируя таблицы 3 и 4 принципиально классифицировать использованные вероятности и времена как базовые

характеристики системы, базовые характеристики трафика и производные характеристики. Базовые характеристики системы это – p_{mn}, t_{net}, t_{send}, t_{io}. Базовые характеристики трафика: p_{rxcur}, p_{rscur} или $(1-p_{rxcur})$, p_{cr*}, p_{pcr} (исключён из формул в модели т.к. может быть представлен комбинацией SCUR и XCUR запросов). И производные характеристики, обусловленные трафиком, системой и алгоритмом доступа к блоку в распределённой системе: p_{xcur}, p_{scur} или $(1-p_{rxcur})$, t_{fp}, p_{gc}, p_{out}, p_{lc}, p_{down}.

Заметим, что хотя интенсивность и не фигурирует в формулах времени доступа напрямую, чаще всего она является основой для вычисления производных параметров. В более широком смысле характеристика трафика – это вектор XCUR, SCUR и CR интенсивностей запросов к каждому ресурсу системы и правила определяющие динамику этих интенсивностей включая правила обратной связи.

Также отметим, что поскольку в формулах не отражен сценарий запись на диск. Задержки отложенной записи, т. е. работа процесса DBWn не включаются в задержки пользовательских сессий, однако могут конкурировать за ресурсы, создавая задержки. При стационар-ной работе системы это допустимо, но при анализе устойчивости на коллапс, этим факто-ром пренебрегать нельзя.

Перемножив из таблиц 3 и 4 соответствующие вероятности на соответствующие времена получаем итоговую формулу для вычисления среднего времени доступа:

$$
\begin{gathered}
t_{acc} = (1-p_{cr*})(\ p_{gc}(1-p_{lc})p_{xcur}p_{rxcur}p_{mn}\left(t_{net}+t_{send}+t_{fp}\right) + \\
+p_{gc}(1-p_{lc})p_{xcur}(1-p_{rxcur})p_{mn}\left(t_{net}+t_{send}+t_{fp}\right) + \\
+p_{gc}(1-p_{lc})p_{xcur}p_{rxcur}(1-p_{mn})\left(2t_{net}+t_{send}+t_{fp}\right) + \\
+p_{gc}(1-p_{lc})p_{xcur}(1-p_{rxcur})(1-p_{mn})\left(2t_{net}+t_{send}+t_{fp}\right) + \\
+p_{gc}p_{lc}p_{scur}p_{rxcur}p_{mn}2t_{net} + p_{gc}(1-p_{lc})p_{scur}p_{rxcur}p_{mn}(t_{net}+t_{send}) + \\
+p_{gc}(1-p_{lc})p_{scur}(1-p_{rxcur})p_{mn}(t_{net}+t_{send}) + \\
+p_{gc}(1-p_{lc})p_{scur}p_{rxcur}(1-p_{mn})(2t_{net}+t_{send}) + \\
+p_{gc}(1-p_{lc})p_{scur}(1-p_{rxcur})(1-p_{mn})(2t_{net}+t_{send})\ \ ) + p_{cr*}\cdot 0
\end{gathered}
$$

Это время и является при неизменных базовых характеристиках системы и трафика критерием, насколько техники оптимизации, внесённые в базовый алгоритм, эффективны. Идеальной является ситуация, когда $t_{acc} = 0$. И снижение t_{acc} является целевой функцией систем с распределённой памятью использующих принципы GCF.

Заключение

Предложенная модель позволяет при различных трафиках вычислять среднее время доступа к блоку в распределённых кластерных системах использующих как основу алгоритм GCF. Необходимость построения модели обусловлена требованиями возможности оценки эффективности различных техник оптимизации алгоритма GCF.

Однако, стоит указать также на направления требующие дополнительного исследования. В первую очередь, это методика описания базовых характеристик трафика и методика вычисления производных параметров.

Среди недостатков модели стоит указать, что не учитываются задержки принятия решения о выборе сценария. Предполагается, что время обмена информацией между нодами существенно больше времени принятия решения о выборе сценария на каждом шаге алгоритма. Это условие справедливо, кроме случаев, когда скорость обмена информацией между нодами сопоставима с временем обработки. В современных системах это справедливо, но с учётом развития технологий коммуникации модель может потерять свою актуальность.

Также следует обратить внимание на то, что процесс каскадного запрашивания блоков UNDO – представляет интерес в контексте анализа системы на коллапс по блокировкам, поскольку обработка блока не завершается до тех пор, пока все вспомогательные блоки UNDO не будут прочитаны. Причем какие блоки UNDO и сколько их, мы можем узнать

только прочитав первый блок с SCUR запросом. То, что необходимость получения очередного блока может быть выявлена только после прочтения предыдущего, приводит к увеличению средней последовательной цепочки блоков при обработке запроса, т. е. обработка блока начинается только по завершению предыдущего. Характер траффика, а именно наличие часто изменяемых т.н. «горячих» блоков определяет среднюю длину цепочки. В предложенной модели, поскольку обработка CR блока распадается на обработку нескольких SCUR блоков, рассматривался весь трафик как набор обработки SCUR, XCUR и CR* блоков. Учитывая, что в OLTP системе запрос выполняется процессом без распараллеливания, рассматривалось среднее количество блоков обрабатываемых запросом, и пренебрегалось зависимостью обработки блоков в цепочке. Оптимизация этого последовательного подхода представляет интерес, однако выходит за рамки данной статьи и требует дополнительного исследования.

Кроме того, предложенная модель может быть использована не только в shared-everything архитектуре, но и в топологиях, в которых доступ к памяти организован как в GCF, а диски на каждой системе собственные, т. е. для моделирования распределенных систем, использующих общую память и работающих с общими ресурсами в глобальных сетях.

Список литературы

1. Гусев Е.И. Исследование области применения неблокирующего алгоритма фиксации распределённых транзакций // Вісник НТУУ "КПІ". Сер. Інформатика, управління та обчислювальна техніка. - 2012. Випуск 57. - С.76 – 80
2. Гусев Е.И. Исключение блокирования общего ресурса в распределённых системах / Гусев Е.И.; Кулаков А.Ю. // Вісник НТУУ "КПІ".Сер. Інформатика, управління та обчислювальна техніка. - 2011. Випуск 54. - С.162 – 166.
3. Oracle® Real Application Clusters Administration and Deployment Guide 11g Release 2(11.2) E16795-08
4. Lampson B., Sturgis H. "Crash recovery in a distributed data storage system," Tech. Rep., Computer Science Lab., Xerox Palo Alto Research Center, Palo Alto, Calif., 1976
5. Menasce D. A., Popek G. J., Muntz R. R. "A locking protocol for resource coordination in distributed databases," ACM Trans. Database Syst. 5, 2 (June 1980), 103-138.
6. Riyaj Shamsudeen. RAC Hack: Deep review of LMS/LGWR process. (July 2011). [Электронный ресурс] . – Режим доступа: http://orainternals.wordpress.com/2011/07/

УДК 004.383

СЕРГІЄНКО А.М.,
СЕРГІЄНКО П.А.

РЕАЛІЗАЦІЯ ФУНКЦІЇ КВАДРАТНОГО КОРЕНЯ У ПЛІС

В статье предложен модифицированный алгоритм извлечения квадратного корня основанный на том, что его первые итерации вычисляются табличным способом, за счет чего уменьшаются аппаратные затраты и уменьшается время вычислений. Алгоритм реализован в Web-приложении, которое генерирует VHDL-модели блоков с заданными параметрами, реализуемые в ПЛИС.

A modified square root algorithm is proposed. In the algorithm the first iterations are calculated by the table functions, that minimizes both hardware volume and speed of calculations. The algorithm is implemented in the parametrized Web application, which generates the VHDL entities for FPGA projects.

1. Вступ

Технологія розробки застосувань на програмованих логічних інтегральних схемах (ПЛІС) основана на описі алгоритму такою мовою, як VHDL та автоматичній трансляції цього опису в опис на рівні логічних таблиць (ЛТ), та інших функціональних компонентів ПЛІС. Але елементарні функції в ПЛІС, як правило, реалізуються як окремі проекти чи замовлені віртуальні модулі, в яких при настроюванні вказується розрядність даних та іноді – внутрішня будова [1].

Відміна ПЛІС останнього покоління полягає у зростанні вхідної розрядності ЛТ з чотирьох до шести. Крім того, вбудовані блоки пам'яті об'ємом в десятки кілобіт також стали застосовуватися як ЛТ [2]. Тому усі напрацювання по побудові віртуальних модулів для реалізації елементарних функцій мають бути переглянуті з метою кращого використання особливостей нових ПЛІС. В даній статті досліджено побудову віртуального модуля для реалізації функції квадратного кореня в ПЛІС.

2. Алгоритми добування квадратного кореня

При виборі алгоритму квадратного кореня слід мати на увазі особливості будови ПЛІС, яка має ресурси ЛТ, мультиплексорів, суматорів, блоків множення, але не має блоків ділення. Функція кореня повинна мати регульовані точність обчислень та розрядність вхідних даних. Для свого швидкого виконання вона має бути реалізована як паралельна структура, що допускає конвеєрну реалізацію.

При визначенні апаратної складності алгоритму Θ_S, яка далі розраховуватиметься у кількості необхідних суматорів, приймається до уваги, що складність блоку множення у ПЛІС для розрядності операндів у межах 16-32 оцінюється як апаратні витрати двадцяти суматорів такої самої розрядності. Часова складність Θ_T алгоритму обчислюється як сумарна затримка комбінаційної схеми блоку, яка приведена до затримки суматора [3].

Далі розглядатимуться алгоритми добування квадратного кореня з оцінкою їх ефективності для 24-розрядних вхідних даних та результатів з фіксованою комою, які можуть претендувати на реалізацію в ПЛІС.

2.1. Поліноміальна апроксимація

Традиційне рішення розрахунку елементарної функції – це обчислення полінома, який є, наприклад, рядом Тейлора, як наступний [4].

$$\sqrt{1+x} = 1 + \frac{1}{2}x - \frac{1}{8}x^2 + \frac{1}{16}x^3 - \frac{5}{128}x^4 + \frac{7}{256}x^5 \ldots .$$

При цьому неможливо досягти похибки обчислення меншої за 0,2%, якщо $x \subset (0;1)$. Крім того, алгоритм потребує виконання багатьох множень. Тому він неприйнятний для реалізації у ПЛІС, хоча, він може згодитись при кусково-поліноміальній апроксимації.

2.2. Ітеративний алгоритм

Відомий наступний ітеративний алгоритм на основі формули Ньютона-Рафсона, який не потребує операцій ділення [4]:

$$y_{i+1} = 1{,}5\, y_i - 0{,}5\, x\, y_i^2 \quad (i = 0, 1, \ldots, n),$$

де $y_0 \approx 1/\sqrt{x}$ – наближене значення функції, $\sqrt{x} \approx x\, y_n$. Кожна наступна ітерація алгоритму приблизно подвоює кількість правильних розрядів результату. Тому для одержання коректного 24-розрядного результату необхідно виконати n = 2 ітерації алгоритму та одержати значення y_0 з

таблиці з семирозрядним входом адреси або одну ітерацію, якщо таблиця має 13-розрядний вхід. Часова Θ_T та апаратна Θ_S оцінки складності цього алгоритму приведені в табл.1. При розрахунку Θ_S вважалось, що згадані таблиці реалізовані у ПЛІС як постійна пам'ять, яка має приблизну складність у 2 та 60 суматорів, відповідно.

1.3. Алгоритм "цифра за цифрою"

Алгоритм, що належить до класу "цифра за цифрою" дає змогу одержувати одну точну цифру результату на кожному кроці. Відомий алгоритм Волдера для обчислення $\sqrt{x}$, але більш конструктивним є алгоритм [5,6]:

$y_0 = x$; $x_0 = x$; $m = 0$; $f = 0$;
for ($i = 1$; $i < n$; $i = i + 1$) {
  $t = x_i + 2^{-m} x_i$;
  $u = t + 2^{-m} t$;
  if ($u \geq 1$) {
    $f = 1$;
    $x_{i+1} = x_i$;
    $y_{i+1} = y_i$;
  }
  else {
    $x_{i+1} = u$;
    $y_{i+1} = y_i + 2^{-m} y_i$;
  }
  if ($f == 1$) $m = m + 1$;
}

Під час виконання алгоритму знаходяться псевдоцифри m результату, причому спочатку, коли $m = 0$, виконується нормалізація операнду x_i з корекцією часткового результату y_i. Потім $m = 1, 2,\ldots, n$ і в процесі конвергенції x_i прямує до одиниці, а y_i – до $\sqrt{x}$.

Для реалізації алгоритму в ПЛІС необхідно блоки нормалізації x_i , зсуву y_i та $3n$ суматорів. Цей алгоритм варто реалізувати у блоці, який виконує кілька функцій за алгоритмом "цифра за цифрою", такому як [7].

2.4. Алгоритм на основі зсуву та віднімання

Метою алгоритму є одержання наближення двійкового числа $B_i = 0{,}b_1b_2\ldots b_i0\ldots0$ до значення $\sqrt{x}$. Він оснований на перевірці знаку остачі $R_i = x - B_i^2$, у результаті чого вибирається чергова цифра b_i. Нехай вважається, що $b_i = 1$, тоді

$$R_i = x - (B_{i-1} + 2^{-i})^2 = R_{i-1} - 2^{-(i-1)}(0{,}b_1\ldots b_{i-1}\,01),$$

а якщо виявиться, що $R_i \leq 0$, то

$$R_i = R_{i-1}\ ;\ b_i = 0\ ;$$

і шукається наступна цифра b_{i+1} .

Алгоритм оснований на відніманні від остачі R_i зсунутого вправо наближення B_i . При його реалізації в ПЛІС він має n ступенів, кожен з яких має суматор в режимі віднімання та мультиплексор для вибору R_i, який керується знаковим розрядом суматора [8].

Даний алгоритм називають алгоритмом з відновленням остачі. Використовують також алгоритм без відновлення остачі, коли R_i є знакозмінним числом. Але він не має суттєвих переваг, оскільки в ньому використовується знакозмінний суматор, який за складністю такий самий, як суматор з мультиплексором [1].

Табл. 1. Витрати для обчислення $\sqrt{x}$

Алгоритм	Блоків множення	Θ_S	Θ_T
Ітеративний, 1 ітерація	2	102	7
Ітеративний, 2 ітерації	4	86	13
"Цифра за цифрою"	–	72	48
Зі зсувом та відніманням	–	24	24
Модифікований зі зсувом та відніманням	–	22	21

Отже, аналізуючи параметри розглянутих алгоритмів у табл.1, можна зробити висновок, що алгоритм на основі зсуву та віднімання реалізується у ПЛІС з мінімальними апаратними витратами. Якщо його виконати у конвейерному блоці, то такий блок матиме високу пропускну спроможність з періодом тактового інтервалу, що дорівнює затримці одного суматора та мультиплексора.

2.5. Модифікований алгоритм зсуву та віднімання

Одна i-та ітерація алгоритму зсуву та віднімання полягає у наступних обчисленнях:

$r_i = R_i - P_i$;
$b_i =$ **not** sign(r_i) ;
$B_{i+1} = (B_i, b_i)$;
$P_{i+1} = 2^{-i+1}(B_{i+1}, 01)$;
if (b_i==1)
  $R_{i+1} = r_i$;
else
  $R_{i+1} = R_i$;

де sign(x) = 1 при $x < 0$, інакше – 0; $i = 1,\ldots,n$; (a,b) – операція конкатенації.

Недолік алгоритму полягає у великій кількості ітерацій, яка дорівнює числу розрядів результату. Помітно скоротити цю кількість можна, якщо перші k ітерацій не виконувати, а результати k – ї ітерації знаходити за допомогою табличних функцій.

Такими табличними функціями є

$$B_k = f_B(x_k) \approx \sqrt{x_k} \quad ; \qquad R_k^{'} = f_R(x_k) \ ,$$

де x_k – $2k$ старших розряди вхідного данного x, причому

$$R_k = R_k^{'} + x - x_k.$$

Наприклад, на рис.1 показана функціональна схема, яка виконує модифікований алгоритм для 12-розрядних вхідних даних, $n = 6$ – розрядного результату та $k = 3$. При цьому табличні функції f_B та f_R реалізовані у блоках постійної пам'яті ROMB та ROMR, відповідно.

Завдяки застосуванню блоків постійної пам'яті, у даному випадку вдалось зменшити кількість ітерацій з шести до чотирьох, перша з яких – це звертання до цих блоків. Відповідно, схема добування кореня стала меншою на три суматори та три мультиплексори.

Недоліком модифікованого алгоритму є те, що блоки постійної пам'яті повинні мати $2k$-розрядні адресні входи. Через це вони мають значний об'єм при великому k.

Проте, цей алгоритм доцільно використовувати в сучасних ПЛІС при $k \le 6$. Так, при $k = 3$ для n = 24-розрядних даних параметри блоку квадратного кореня наведені в табл.1., тобто з застосуванням цього алгоритму параметри блоку квадратного кореня покращуються.

3. Експериментальні результати

Модифікований алгоритм добування квадратного кореня було реалізовано у Web-додатку, призначеному для генерації віртуальних модулів, описаних мовою VHDL, який впроваджено на сайті www.kanyevsky.kpi.ua. У ньому організовано ввід таких параметрів, як розрядність вхідних та вихідних n даних, розрядність $2k$ адрес блоків постійної пам'яті, наявність вхідного та вихідного регістрів, побудова повністю конвеєризованої чи частково конвеєризованої схеми, тобто з n–k або з](n–k)/2[ступенями конвеєра. При цьому користувачу даються оцінки апаратних витрат модуля та максимальної тактової частоти. Наприклад, для 12-розрядних вхідних даних, 6-розрядного результату, $k = 3$ при відсутності конвеєризації генерується наступний поведінковий опис модуля квадратного кореня:

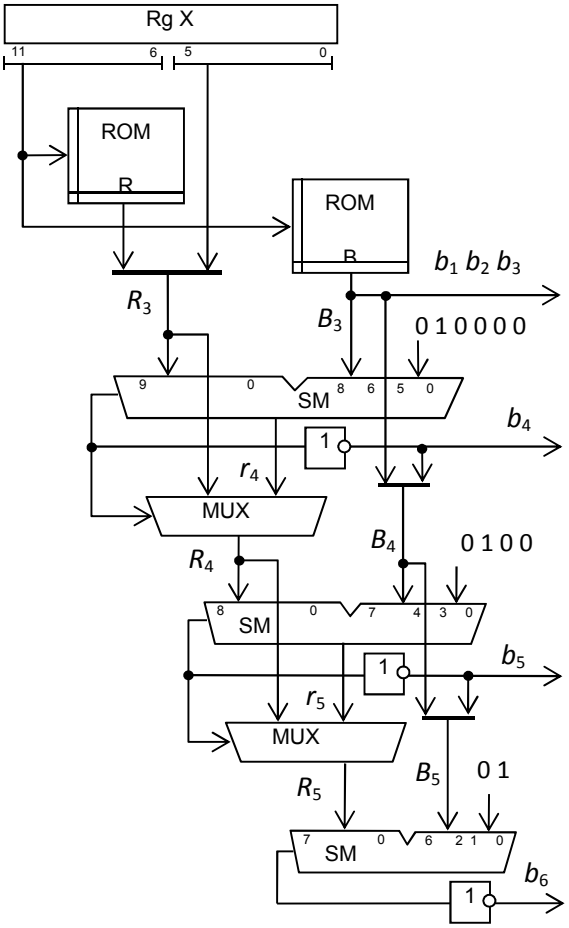


Рис.1. Схема блоку добування квадратного кореня

```
process(CLK) begin
 if rising_edge(CLK) then
  x<=DI;
  DO<=qhr(6)(12 downto 7);
 end if;
end process;

xh<=x(11 downto 6);
qr(3)<= xqh_6(conv_integer(xh))&x(5 downto 0)&"00000";
pr(3)<= "0"&qh_6(conv_integer(xh))&"01"&z(8 downto 0);
qhr(3)<= qh_6(conv_integer(xh))&z(9 downto 0);

Stages:for i in 3 to 5 generate
 rr(i)<=qr(i) - pr(i);
 biti(i) <= '1' when rr(i)(14)='0' else '0';
 qhr(i+1)<=qhr(i)(12 downto 13-i)&biti(i) & z(11-i downto 0);
 pr(i+1)<='0' & qhr(i+1)(12 downto 12-i) &"01"&z(10-i downto 0);
 qr(i+1)<=SHL(rr(i),"1") when rr(i)(14)='0' else SHL(qr(i),"1");
end generate;
```

У ньому масив xqh_6 описує постійну пам'ять для функції f_R, а масив qh_6 – для функції f_B. Цей опис повністю відповідає схемі на рис.1.

Після синтезу та розміщення модулів блоку квадратного кореня у ПЛІС Xilinx Spartan-6 було одержано ряд характеристик апаратних витрат (рис.2) у кількості ЛТ та максимальної тактової частоти, МГц (рис.3) для різних значень k при вхідній розрядності 24 та розрядності результату n = 12. Розглядались модулі у вигляді комбінаційної схеми, повністю конвеєризовані та частково конвеєризовані модулі.

Аналіз цих характеристик показує, що оптимальним за апаратними витратами та за швидкодією є блок з k = 3. Блок з k = 5 має помітно менші апаратні витрати за рахунок застосування окремого блоку пам'яті BlockRAM для реалізації таблиць. Використання значень $k > 6$ недоцільне через значне зростання необхідного об'єму пам'яті та зменшення тактової частоти.

На рис. 4, 5 показані характеристики блоків для k = 3 в залежності від вихідної розрядності n, яка дорівнює вхідній розрядності. Там же для порівняння приведені характеристики блоків, які пропонуються фірмою Xilinx.

Отже, синтезовані модулі для добування квадратного кореня мають помітно кращі показники за модулі, що пропонуються фірмою Xilinx.

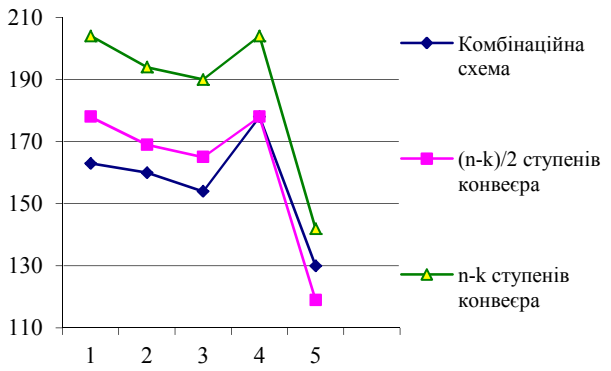


Рис.2. Апаратні витрати блоку $\sqrt{x}$ в залежності від k

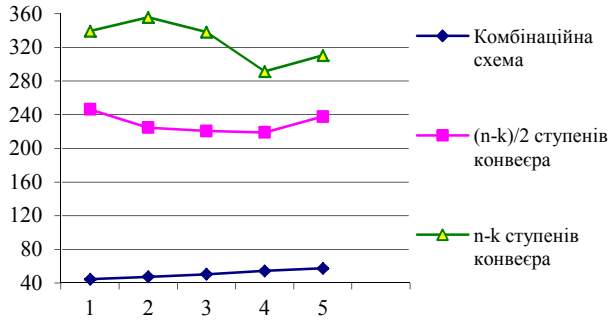


Рис.3. Максимальна тактова частота блоку $\sqrt{x}$ в залежності від k

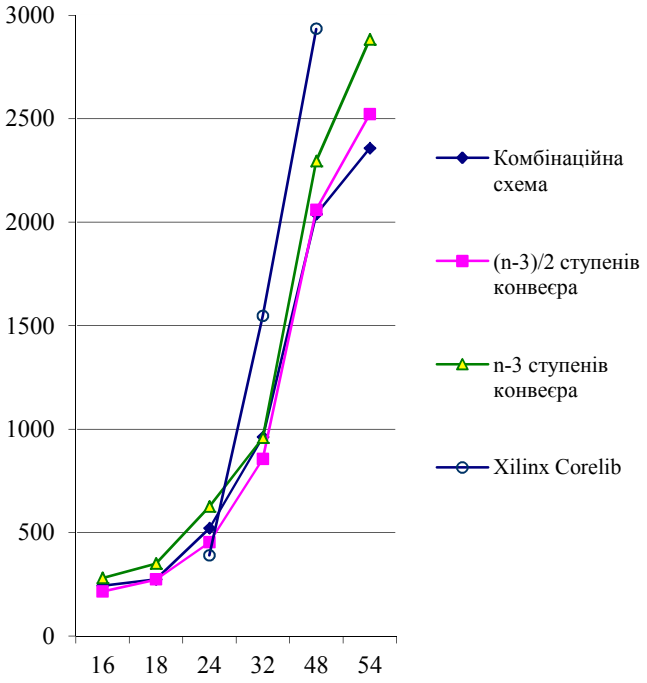


Рис.4. Апаратні витрати блоку $\sqrt{x}$ в залежності від n

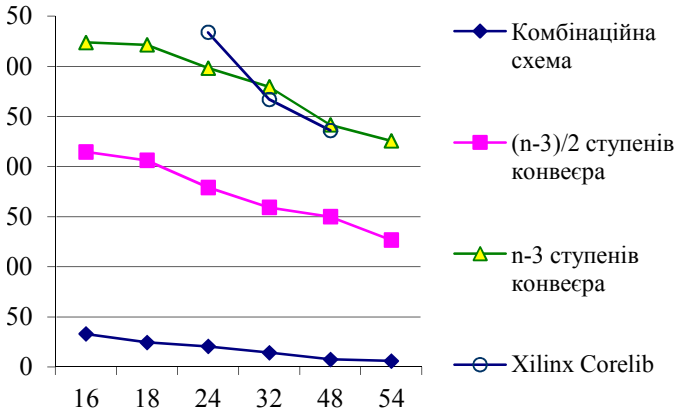


Рис.5. Максимальна тактова частота блоку $\sqrt{x}$ в залежності від n

Модулі обчислення $\sqrt{x}$ часто використовуються у блоках для обчислень з плаваючою комою, для яких критичною є латентна затримка між входом та виходом. Цей параметр дорівнює $T_Л =](n - k)/2[/f_C$ для модулів з частковою конвеєризацією та $T_Л = (n - k)/f_C$ – для модулів з повною конвеєризацією, де f_C – тактова частота. Так, для $n = 24$ і $k = 3$ ця затримка дорівнює $T_Л = 61{,}4$ нс і 70,4 нс, відповідно. Для порівняння, модуль фірми Xilinx має $T_Л = 71{,}9$ нс. Отже, з точки зору зменшення латентної затримки доцільно застосовувати модулі з частковою конвеєризацією, які до того ж мають менші апаратні витрати.

Висновки

Проаналізовано алгоритми обчислення квадратного кореня і вибрано за основу алгоритм зі зсувом та відніманням. Цей алгоритм було модифіковано таким чином, що його перші k ітерацій обчислюються табличним способом. За рахунок цього дещо зменшуються апаратні витрати та час виконання алгоритму.

Модифікований алгоритм добування квадратного кореня було реалізовано у Web-додатку, який генерує модулі, описані мовою VHDL стилем для синтезу. Одержані характеристики ряду згенерованих модулів свідчать про те, що ці модулі є кращими за аналогічні модулі, що пропонуються фірмою-виробником ПЛІС.

Встановлено, що оптимальними є модулі, для яких $k = 3$. Також встановлено, що задля зменшення латентної затримки та апаратних витрат доцільно застосовувати модулі з частковою конвеєризацією.

Список посилань

1. Deshamps J.-P., Bioul G.J.A., Sutter G.D. Synthesis of Arithmetic Circuits: FPGA, ASIC, and Embedded Systems. –John Wiley & Sons, Inc. –2006. – 556 p.
2. FPGA Architecture. White Paper WP-01003-1.0. – Altera Corp. – 2006. – 14 p. Available at http://www.altera.com/
3. Сергиенко А.М., Симоненко В.П. Отображение периодических алгоритмов в программируемые логические интегральные схемы // Электронное моделирование. –2007. –Т.29. –№2. –С. 49–61.
5. Люстерник Л.А., Червоненкис О.А., Янпольский А.Р. Математический анализ. Вычисление элементарных функций. –М.: ГИЗ физ.-мат. лит. – 1963. – 247 с.
6. Chen T.C. Automatic computations of exponentials, logarithms, ratios and square roots. IBM J. Res. and Develop. –1972. –№4. – P. 380-388.
7. Благовещенский Ю.В., Теслер Г.С. Вычисление элементарных функций на ЭВМ. – Киев: Техніка. –1977. – 208 с.
8. Каневский Ю.С., Куц Н.Е., Лозинский В.И., Сергиенко А.М. Устройство для вычисления элементарных функций. А.С. СССР N 1141399, Б.И. N 7, 1985.
9. Каневский Ю.С. Компьютерная арифметика. Конспект лекций. – Киев: Диасофт. – 1994. – 234 с.

УДК 681.518

РОЛИК А.И.

МЕТОДИКА ОПЕРАТИВНОГО УПРАВЛЕНИЯ УРОВНЕМ ИТ-УСЛУГ НА ОСНОВЕ СЕРВИСНО-РЕСУРСНЫХ МОДЕЛЕЙ

Предложена методика управления уровнем ИТ-услуг на основе сервисно-ресурсных моделей. Рассмотрены особенности реализации декомпозиционно-компенсационного способа организации управления уровнем услуг при использовании сервисно-ресурсных моделей.

Method for service level management based on service-resource models was offered. Implementation of the compensation decomposition-compensation method for service level management based on service-resource of models was shown.

Введение

Оперативное управление корпоративной ИТ-инфраструктурой подразумевает решение большого количества разнообразных задач, относящихся к одной из трех обобщенных функциональных областей управления информационно-телекоммуникационными ресурсами, полностью покрывающих весь функционал управления ИТ-инфраструктурой: поддержание качества ИТ-сервисов на согласованном с бизнес-подразделениями уровне, рациональное использование ресурсов ИТ-инфраструктуры, автоматизация операционной деятельности ИТ-подразделения. Для автоматизированного решения этих задач разрабатываются и внедряются системы управления ИТ-инфраструктурой (СУИ). Создание СУИ требует решения множества специфических прикладных задач управления сложными распределенными многообъектными системами. Важнейшей задачей оперативного управления является поддержание качества ИТ-услуг на заданном, согласованном с бизнесом уровне при условии рационального использования ресурсов ИТ-инфраструктуры. Поэтому данная работа, посвященная оперативному управлению уровнем ИТ-услуг, является актуальной.

Постановка задачи исследования

В процессе эксплуатации ИТ-инфраструктуры возникают нештатные ситуации, обусловленные конечной надежностью элементов информационно-телекоммуника-ционной системы (ИТС), нестационарностью количества пользователей и, соответственно, генерируемого ими потока запросов, а также другими факторами. Для поддержания качества ИТ-сервисов на заданном уровне необходимо автоматическое формирование управляющих воздействий или оперативное вмешательство персонала ИТ-подразделения. Каждая нештатная ситуация, которая в теории управления техническими системами рассматривается как возмущающее воздействие, порождает соответствующую прикладную задачу, решением которой является определение вида управляющего воздействия, которое подается на ИТ-инфраструктуру для восстановления качества предоставляемых ИТ-услуг. Решение таких задач осуществляется информационной технологией управления ИТ-инфраструктурой, реализуемой в СУИ. Функционал такой ИТ позволяет работать СУИ в автоматическом или автоматизированном режиме. В последнем случае СУИ рассматривается как система поддержки принятия решений, предоставляющая рекомендации администраторам систем ИТ-инфраструктуры по необходимым оперативным управляющим воздействиям.

Для решения задач оперативного управления ИТ-инфраструктурой и создания соответствующей ИТ необходимо разработать математические модели процессов управления ИТ-инфраструктурой и ее компонентами, а также методики управления. Количество решаемых задач управления ИТ-инфраструктурой ввиду ее громоздкости и сложности столь велико, что не позволяет смоделировать все ситуации, которые могут возникнуть в ИТ-инфраструктуре.

Поэтому при автоматизации процессов управления выделяют наиболее актуальные задачи управления и на их решении делается акцент при создании ИТ управления ИТ-инфраструктурой. Одной из самых важных задач управления является обеспечение поддержания качества ИТ-услуг на заданном, согласованном с бизнес-подразделениями уровне, поэтому **целью данной работы** является разработка моделей и методики управления ИТ-инфраструктурой при условии рационального использования ее ресурсов.

Использование сервисно-ресурсных моделей для управления уровнем ИТ-услуг

При управлении уровнем услуг необходимо знать не только текущее количество пользователей ИТ-сервиса и загруженность ИТ-ресурсов, но и иметь представление о том, какие конкретно информационно-телекоммуникационные ресурсы, приложения и другие активы ИТ-инфраструктуры задействованы для предоставления определенного сервиса. Оперативную информацию о загруженности ресурсов и количестве пользователей система управления получает в результате непрерывного мониторинга ИТ-инфраструктуры, а для создания модели сервиса, отображающего его зависимость от конкретных ресурсов и их взаимосвязи, целесообразно использовать предложенные ITIL [1, 2] сервисно-ресурсные модели (СРМ) [3]. СРМ — логическая модель сервиса, описывающая состав и взаимосвязи ИТ-ресурсов, которые совместно обеспечивают предоставление данного сервиса.

СРМ удобно представлять в виде иерархического графа (рис. 1), узлами которого для СУИ являются объекты мониторинга и управления (ОМУ) [4] (на рис. 1 — это ресурсы L_1—L_4) или сервисы (на рис. 1 — s_i), а ребрами — связи между ОМУ и сервисами.

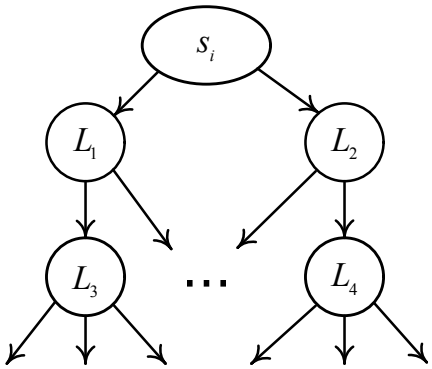


Рис. 1. Граф сервисно-ресурсной модели

СРМ создаются на основе системной методологии в рамках определенного процесса управления ИТ-инфраструктурой. Вид модели зависит от целей процесса, для которого создается модель. Одна из основных задач создания СРМ заключается в мониторинге параметров качества ИТ-услуг. При этом качество услуг определяется комплексом измеримых характеристик [5, 6].

СРМ широко используются в процессном управлении, в то время как применение модели в операционной деятельности при оперативном управлении ИТ-инфраструктурой пока не получило широкого распространения [6]. В данной работе показано, что СРМ не только можно, но и следует применять при оперативном управлении уровнем ИТ-услуг с целью поддержания заданных значений показателей качества, если и не для всех, то для наиболее важных услуг.

При оперативном управлении уровнем ИТ-услуг должна быть известна структура предоставления ИТ-услуг, определены факторы, влияющие на качество предоставления услуг, известно, как можно измерять и влиять на значения параметров качества, должны быть организованы процессы управления услугами. Часть этих вопросов снимается в процессе композиции ИТ-сервиса (рис. 2).

Жизненный цикл сервиса состоит из этапов «планирования», «внедрения», «предоставления» и «поддержки» [6]. С композиционной точки зрения сервис является предоставляемой пользователю управляемой и поддерживаемой ИТ-системой с согласованным уровнем качества [5].

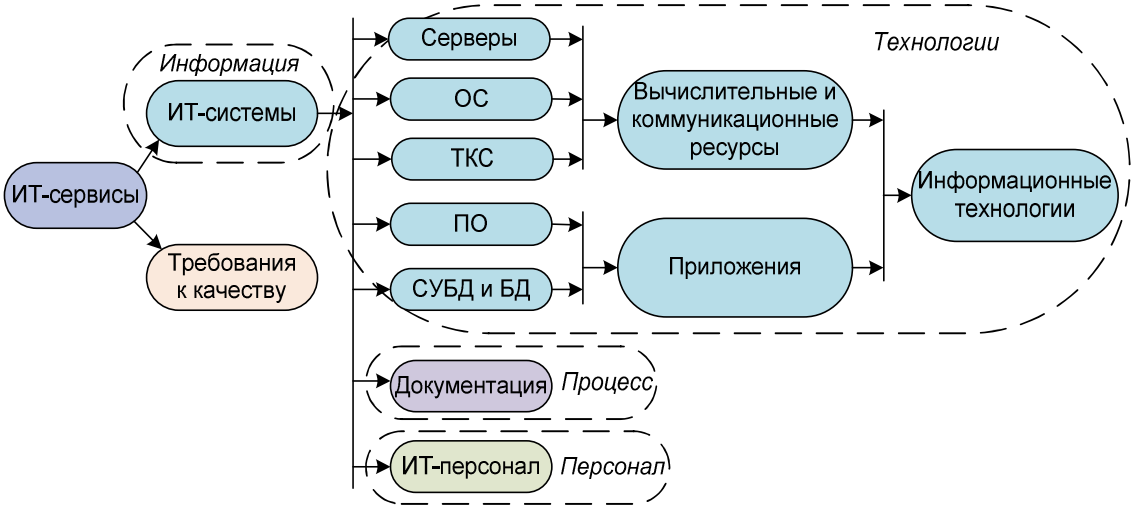


Рис. 2. Композиция ИТ-сервисов с точки зрения ITIL

Композиция сервиса состоит из трех элементов: ИТ-системы, требований к качеству услуг и поддержки. В свою очередь, в соответствии с ITIL ИТ-система включает в себя такие активы: Люди — Процессы — Технологии — Информация (см. рис. 2).

Методика управления уровнем ИТ-услуг на основе сервисно-ресурсных моделей

Предлагаемая методика оперативного управления уровнем услуг на основе СРМ включает следующую последовательность этапов:

1. Формирование каталога ИТ-сервисов.
2. Определение параметров качества услуг.
3. Проектирование СРМ.
4. Определение методов получения значений показателей качества услуг.
5. Мониторинг и анализ значений параметров качества услуг.
6. Управление ИТ-инфраструктурой для поддержания согласованного уровня услуг.

Суть этапов заключается в следующем.

Этап 1. Формирование каталога ИТ-сервисов. При формировании каталога сервисов, например, в ITSM [7, 8] инициализируется процесс «Планирование и реализация новых или измененных услуг», целью которого является гарантирование предоставления новых/измененных услуг и их менеджмента в соответствии с согласованными затратами и качеством услуг.

Бизнес-процесс разделяется на бизнес-операции или функции, для которых определяются бизнес-услуги и технологические услуги. Запускается согласовательный процесс между бизнес-менеджерами и ИТ-дирекцией (рис. 3). Бизнес выдвигает предложения относительно новых $s_i^{н}$ или изменяемых $s_i^{м}$ услуг, руководствуясь ожидаемыми результатами коммерческого воздействия.

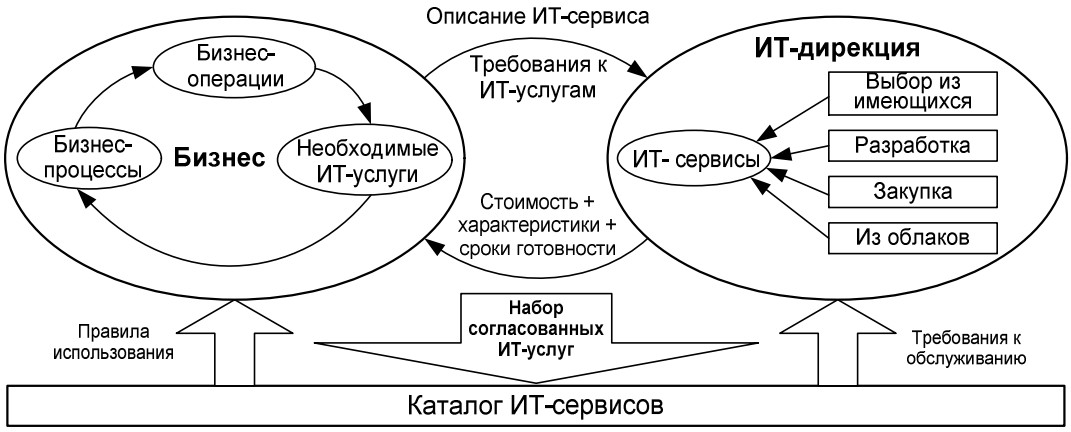


Рис. 3. Процесс формирования каталога ИТ-сервисов

ИТ-дирекция рассматривает различные варианты предоставления новой услуги $s_i^{н}$: комбинация или модификация имеющихся услуг $S = \{s_i\}$, $i = \overline{1, K}$ с учетом стоимости модификации $C_i^{м}$, поддержки $C_i^{п}$ и обслуживания $C_i^{о}$; самостоятельно разрабатываемое ПО с учетом стоимости разработки $C_i^{р}$ и сроков; закупка ПО с анализом стоимости лицензии $C_i^{л}$, сроков запуска, востребованности; поставляемые с использованием облачных вычислений с учетом стоимости $C_i^{Об}$ и надежности. Понимание структуры образования стоимости услуг необходимо для оптимизации портфеля услуг. Выбор пути оптимизации услуг производится с учетом критерия:

$$\min_{C}\{(C_i^{м} + C_i^{п} + C_i^{o}); (C_i^{р} + C_i^{п} + C_i^{o}); (C_i^{л} + C_i^{п} + C_i^{o}); (C_i^{Об})\}$$

Новая услуга заносится в каталог, содержащий информацию о наименовании услуги и значении целевых показателей уровня услуги. Все действия по улучшению услуг вносятся в план совершенствования услуг [9].

Этап 2. Определение параметров качества ИТ-услуг. На этом этапе определяются параметры качества всех услуг s_i, $i = \overline{1, K}$, входящих в каталог S. Этап реализуется в рамках процесса «Управление уровнем услуг (SLM)», обеспечивающего согласование уровня качества и рисков для всех услуг s_i, $i = \overline{1, K}$. Каждый сервис S_b, $b = \overline{1, D}$, где D — количество сервисов, состоит из совокупности услуг $s_{b,i} \in S$, $i = \overline{1, K}$, $b = \overline{1, D}$. Согласованные показатели качества $q_{b,ki}$, $\forall b, k, i$ документируются в SLA. Здесь $q_{b,ki}$ — значение k-го показателя качества i-й услуги, когда она предоставляется вместе с b-м сервисом. SLA находиться под управлением процесса «Управления изменениями». Примерный перечень показателей приведен на рис. 4.

Рис. 4. Примерный перечень показателей ИТ-сервисов

Из всех показателей качества услуг СУИ оперирует только с измеримыми объективными показателями $q_{b,ki}$, $b = \overline{1, D}$, $k = \overline{1, M_i}$, $i = \overline{1, K}$ (см. рис. 4). Пример фрагмента SLA с показателями, находящимися в сфере оперативного управления ИТ-инфраструктурой, приведен в табл. 1.

Табл. 1. Пример фрагмента SLA с показателями уровня услуг

ИТ-сервис	ИТ-услуги	Целевые показатели уровня услуг и их граничные значения					
		Показатели производительности	Значение	Показатели надежности и доступности	Значение	…	…
S_1	$S_{1,1}$	Время открытия формы, не более (с)	1	Доступность, не хуже	99,99	…	…
	$S_{1,т1}$	Загрузка данных, не более (с)	0,5	Доступность, не хуже	99,99	…	…
	$S_{1,т2}$	Печать документа, не более (с)	20	Доступность, не хуже	99,99	…	…
S_2	$S_{2,2}$	Скорость доступа (Мбит/с)	100	Доступность, не хуже	99,9	…	…
	…	…	…	…	…	…	…
	$S_{2,i}$	Время отклика, не более (с)	1	Доступность, не хуже	99,9	…	…
…	…	…	…	…	…	…	…

Функции мониторинга и анализа ИТ управления ИТ-инфраструктурой при управлении уровнем услуг сводятся к определению фактических значений $q^*_{b,ki}$ показателей уровня услуг и сравнения их с целевыми $q_{b,ki}$, $\forall b,k,i$ из табл. 1.

Этап 3. Проектирование СРМ. Несмотря на то, что реализации проектов ITSM часто осуществляются с использованием СРМ, практически отсутствуют методики построения и применения СРМ. В [6] сформулированы некоторые методологические аспекты построения СРМ, которые, однако, не затрагивают процессы оперативного управления.

При согласовании проекта сервиса S_b выделяются его компоненты $s_{b,i} \in S$, связи между компонентами и путем создания СРМ производится отображение услуг на ПО и ресурсы ИТ-инфраструктуры.

При оперативном управлении услугами целесообразно создавать СРМ только для критически важных услуг из S, относительно которых осуществляется управление по поддержанию согласованного уровня услуг.

При создании СРМ предлагается использовать следующие правила:

— модели проектируются сверху вниз — от сервисов к каналам связи;

— в СРМ выделяются уровни, соответствующие уровням ИТ-инфраструктуры;

— связи в модели должны отображать влияние состояний нижерасположенных по иерархии ОМУ на значения параметров качества вышерасположенных ОМУ;

— связи не должны содержать петель;

— определяются параметры, измеряемые в каждом элементе СРМ.

Пример схемы СРМ для сервиса S_1 приведен на рис. 5.

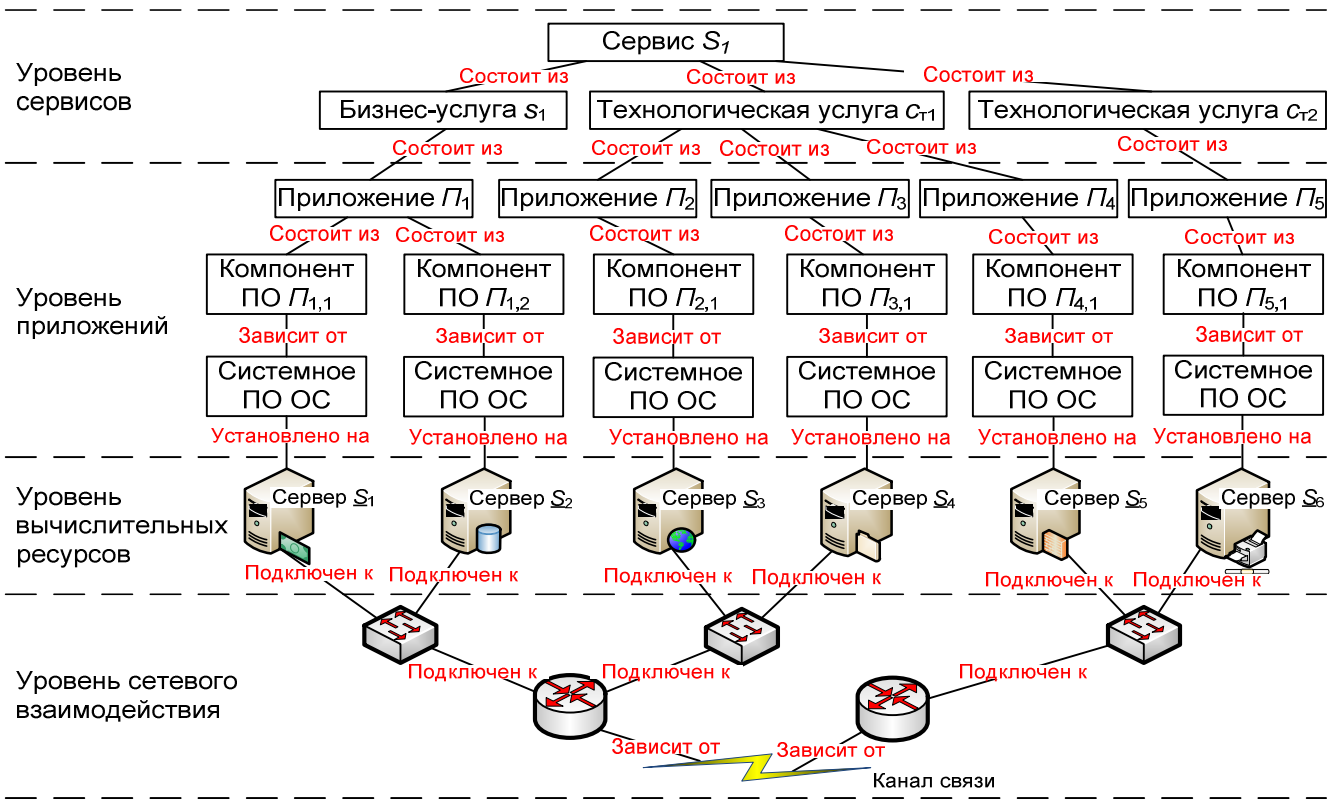


Рис. 5. Пример схемы СРМ

На основе СРМ и требований к параметрам сервиса определяются потребности в ресурсах ИТ-инфраструктуры. Кроме того, СРМ используются для оперативного контроля функционирования сервисов. Для сервисов, функционирование которых оценивается по принципу: работает — не работает, СРМ не содержит ответвлений [6].

Этап 4. Определение методов получения значений показателей качества сервисов. Определение степени снижения уровня услуг $w(\Delta q_{b,ki})$ производится на основе фактических значений $q^*_{b,ki}$, $\forall b,k,i$. Измерение значений $q^*_{b,ki}$ для всех пользователей технически

реализуемо, но трудоемко, поэтому широко используется вычисление $q^*_{b,ki}$ на основании косвенных переменных, характеризующих функционирование ОМУ, участвующих в предоставлении услуги. Для этого решается задача сведения метрик, после чего сведенные значения $q^*_{b,ki}$ сравниваются с целевыми значениями $q_{b,ki}$ показателей уровня услуг.

Рассмотрим в качестве примера СРМ на рис. 5, сконструированную для сервиса S_1 (см. табл. 1). Сервис состоит из услуг $s_{1,1}$, $s_{т,1}$ и $s_{т,2}$. Покажем, как СРМ используется для расчета показателя производительности на примере параметра «Время открытия формы» услуги $s_{1,1}$. Услуга $s_{1,1}$ зависит от: приложения $П_1$ (компоненты ПО $П_{1,1}$ и $П_{1,2}$), сервера приложений $\underline{S}_1$, СУБД и БД на сервере $\underline{S}_2$, программных компонентов серверов и телекоммуникационной сети (ТКС). На значение показателя $q_{1,1,1}$ «Время открытия формы», оказывает влияние производительность сервера $\underline{S}_1$ с установленной на нем ПО и ТКС. Из всех параметров, характеризующих функционирование этих компонентов ИТ-инфраструктуры, выделяются параметры, влияющие на показатель $q_{1,1,1}$: время Δt_{S_1} отклика сервера $\underline{S}_1$ на запрос; время обслуживания запроса $\Delta t_{П_1}$ компонентом ПО $П_{1,1}$; задержка $\Delta t_{ТКС}$ прохождения данных по ТКС. Эти данные измеряются подсистемой мониторинга, которая также постоянно отслеживает количество a_{S_1} открытых пользовательских сессий для контроля доступности услуги $s_{1,1}$ (рис. 6).

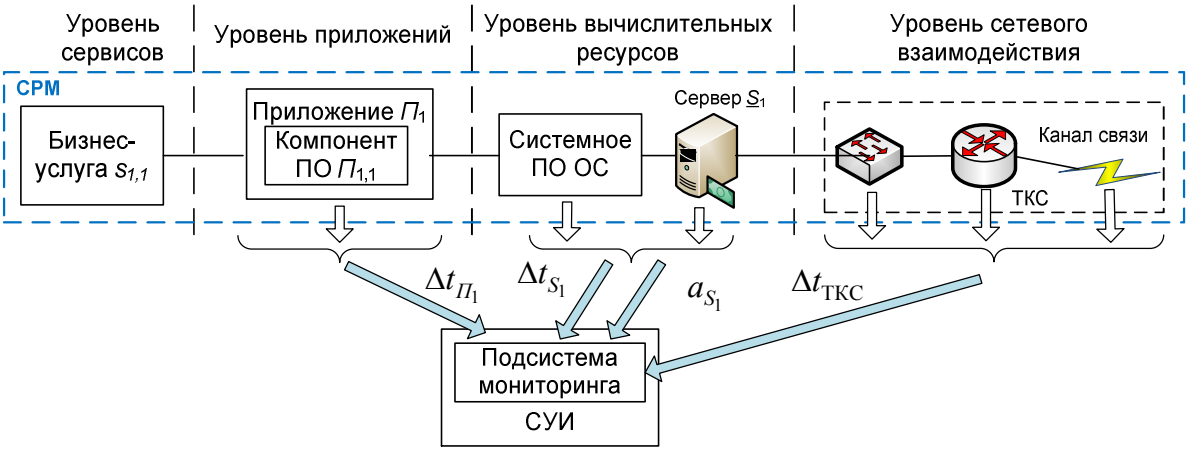


Рис. 6. Измерение значений параметров функционирования ИТ-инфраструктуры, оказывающих влияние на значение показателей уровня услуги $s_{1,1}$

Этап 5. Мониторинг и анализ значений параметров качества услуг осуществляется для установления соответствия текущих значений показателей уровня услуг целевым значениям, т. е. текущего контроля уровня услуг. Выделяются два вида контроля услуг — оперативный и аналитический [6]. Первый необходим для оперативного управления, направленного на поддержание услуг на согласованном уровне. Аналитический контроль используется при процессном управлении ITSM для накопления статистики при коррекции целевых значений уровней услуг.

При оперативном контроле измеряются значения q^*_{ki}, $k=\overline{1,M_i}$, $i=\overline{1,K}$, которые сводятся обобщенному показателю, значение которого сравнивается с целевым значением показателя качества услуг. Например, для услуги $s_{1,1}$ обобщенным показателем будет суммарная задержка передачи данных $\Delta t_{1,1}$:

$$\Delta t_{1,1} = \Delta t_{П_1} + \Delta t_{S_1} + \Delta t_{ТКС}. \quad (1)$$

Значение (1), умноженное на поправочный коэффициент, соответствует фактическому значению $q^*_{1,1,1}$ уровня услуги $c_{1,1}$:

$$q^*_{1,1,1} = k_{1,1,1} \cdot \Delta t_{1,1}. \quad (2)$$

Фактическое значение $q^*_{1,1,1}$ уровня услуги $s_{1,1}$ сравнивается с целевым значением $q_{1,1,1}$, и в случае превышения СУИ генерирует управляющее воздействие.

Этап 6. Управление ИТ-инфраструктурой по поддержанию уровня услуг. Для поддержания уровня услуг $q^*_{b,ki}$, $\forall b,k,i$ сравниваются с $q_{b,ki}$, $\forall b,k,i$. Если значение $q^*_{b,ki}$ превышает предельные значения $q_{b,ki}$, то администратор производит действия, направленные на то, что-

бы показатели качества ИТ-инфраструктуры вернулись к нормативным значениям.

На рис. 7 в качестве примера рассмотрена реализация управления с использованием СРМ для поддержания услуги $s_{1,1}$ на согласованном уровне путем определения значений составляющих выражения (2).

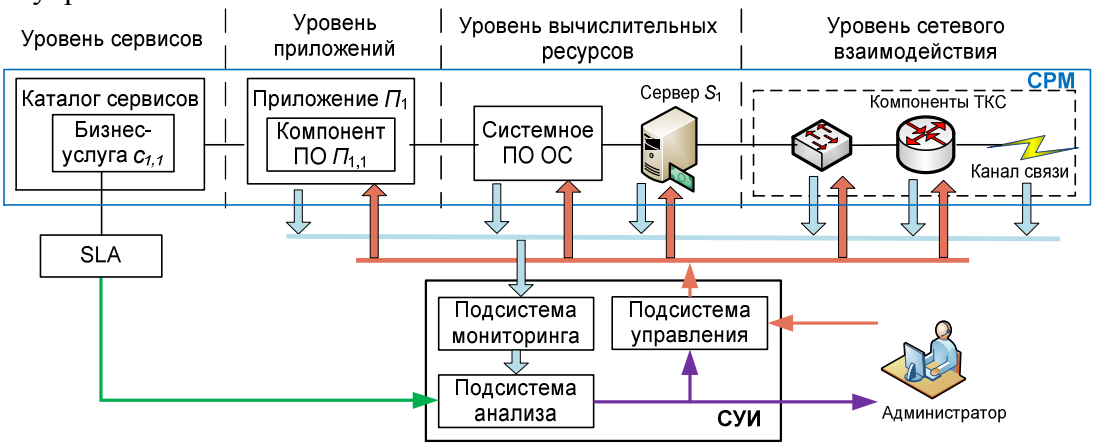


Рис 7. Реализация оперативного управления для поддержания уровня услуг

Особенности декомпозиционно-компенсационного метода управления уровнем услуг при использовании сервисно-ресурсных моделей

Предложенный в [10] декомпозиционно-компенсационный способ оперативного управления уровнем услуг при использовании СРМ будет иметь ряд следующих особенностей.

Во-первых, выделяются услуги S, влияющие на бизнес-сервисы, и определяется перечень параметров q_{ki}, $k=\overline{1,M_i}$, $i=\overline{1,K}$, определяющих их качество. Во-вторых, на основании СРМ выделяются характеристики функционирования ИТ-инфраструктуры, которые оказывают влияние на качество сервиса и которые можно измерять и контролировать, после чего производится сбор и анализ значений этих характеристик. В-третьих, значения показателей функционирования ИТ-инфраструктуры сводятся к обобщенным показателям $q^*_{b,ki}$, $\forall b,k,i$, которые сравниваются с целевыми значениями $q_{b,ki}$, $\forall b,k,i$ показателей уровня услуг. При нарушении SLA подсистема анализа уведомляет об этом администратора ИТ-инфраструктуры. В четвертых, администратор посредством СУИ выбирает решение, позволяющее компенсировать снижение уровня услуги s_i, произошедшее из-за воздействия ряда факторов, например, увеличения количества пользователей a_l услуги A_l, выделением дополнительных ресурсов $\Delta\hat{\rho}_{lj}$, $i=\overline{1,K}$, $l=\overline{1,I}$, $j=\overline{1,m}$ приложению A_l, поддерживающему услугу s_i. Для этого администратор задействует подсистему оптимизации, которая, в свою очередь, использует информацию из базы знаний СУИ.

В качестве примера рассмотрим реализацию управления по поддержанию услуги $s_{1,1}$ на уровне $q_{1,1,1}$ (рис. 8) с использованием СРМ (см. табл. 1).

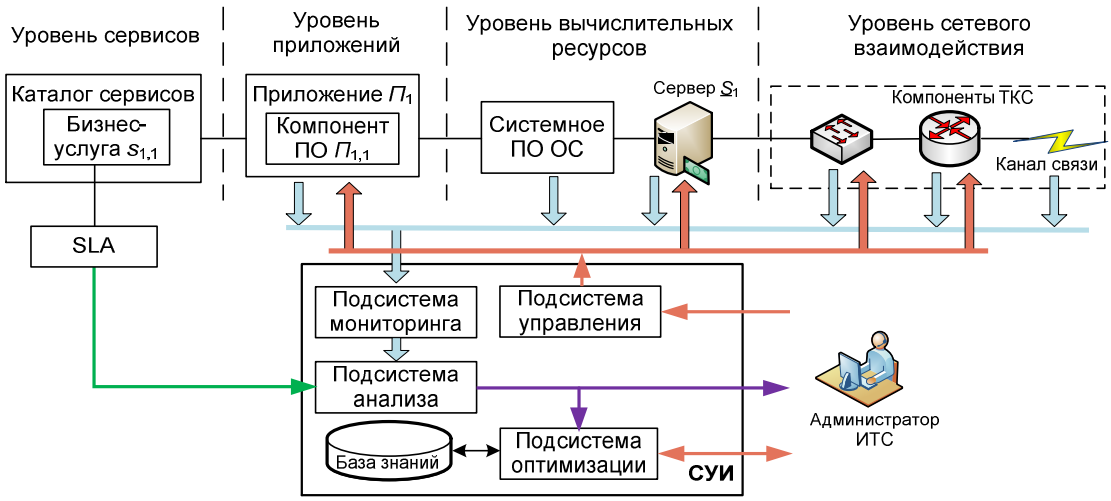


Рис. 8. Схема управление уровнем услуги $s_{1,1}$ с использованием СРМ

Исходя из приведенного выше примера SLA с показателями уровня услуг, ключевым показателем $q_{1,1,1}$ качества услуги $s_{1,1}$ является показатель производительности «Время открытия формы». Фактическое значение $q^*_{1,1,1}$ зависит от значений параметров правой части выражения (1). При $a^*_1 > a_{\max,1}$, где a^*_1 — количества открытых сессий услуги $s_{1,1}$, $a_{\max,1}$ — максимальное количество сессий, оговоренных в SLA, увеличивается время Δt_{S_1} отклика сервера $\underline{S}_1$ на запросы пользователей, а также происходит увеличение других составляющих $\Delta t_{1,1}$, что приводит к снижению качества предоставления услуги $s_{1,1}$. Ограничение a^*_1 до значения $a_{\max,1}$ позволит восстановить уровень услуги $s_{1,1}$ до целевого значения $q_{1,1,1}$, но при этом пострадают интересы бизнеса по причине отключения части пользователей от $s_{1,1}$.

Применение предлагаемого способа организации управления позволяет сохранить как количество открытых сессий при $a^*_1 > a_{\max,1}$, так и качество услуги $s_{1,1}$ на уровне $q_{1,1,1}$. В этом случае СУИ предложит администратору выбрать один из трех методов восстановления уровня услуги $s_{1,1}$ из приведенных в табл. 2.

Табл. 2. Методы компенсации снижения уровня услуги $s_{1,1}$

№ п/п	Обозначение	Суть метода	Возможности	Затраты
1	M_1	Кластеризация приложения $П_1$	Большие	существенные
2	M_2	Выделение компоненту ПО $П_{1,1}$ дополнительных ресурсов на сервере $\underline{S}_1$	Большие	Умеренные
3	M_3	Увеличение полосы пропускания ТКС для данных приложения $П_1$	Ограниченные	Незначительные

Для каждого из методов компенсации имеются графики зависимости степени повышения производительности (уменьшения задержки) от приведенных затрат (рис. 9). Графики могут иметь вид, подобный приведенному на рис. 10, если представляется возможность связать потери бизнеса со снижением качества услуги.

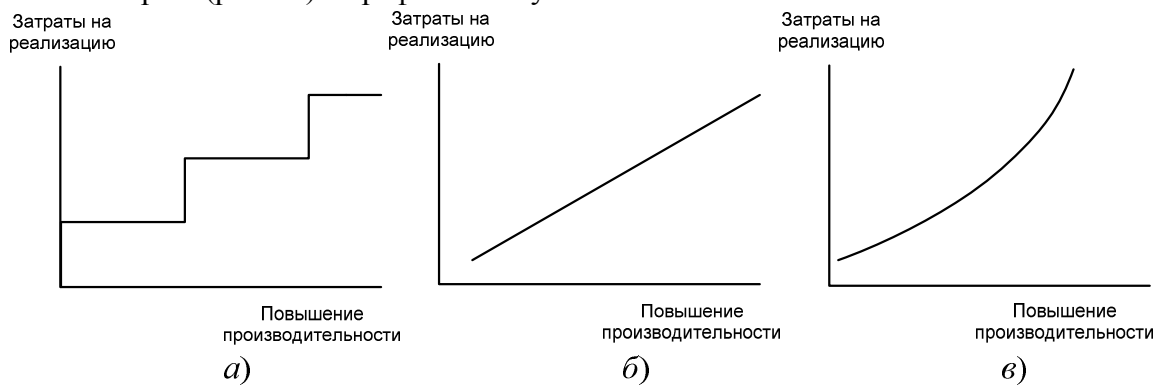


Рис 9. Кривые зависимости повышения производительности от приведенных затрат для: а) метода M_1; б) метода M_2; в) метода M_3

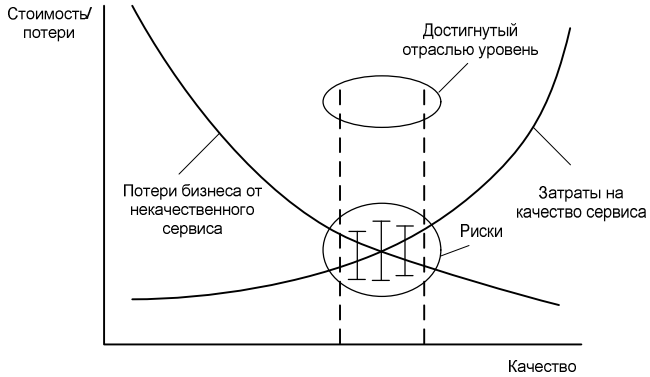


Рис. 10. Нахождение оптимального соотношения уровня качества сервиса и стоимости достижения этого качества

Кривая затрат для метода M_1 (рис. 9, *а*) имеет ступенчатый вид, объясняемый дискретным увеличением затрат на выделение ресурсов $\Delta\hat{\rho}_{1j}$, $j=\overline{1,m}$ для размещения дополнительного экземпляра приложения $П_1$. Повышение производительности методом M_2 (рис. 9,*б*) пропорционально зависит от приведенных затрат, что объясняется примерно пропорциональным повышением производительности компонента ПО $П_{1,1}$ от выделенных для него ресурсов на сервере $\underline{S}_1$.

Для выбора одного из этих методов компенсации СУИ приводит графики к единому мас-

штабу и при заданной степени повышения производительности $w(\Delta q_{ki})$, достаточной для приведения уровня услуги $s_{1,1}$ к уровню $q_{1,1,1}$, позволит администратору выбрать метод, требующий меньших затрат для своей реализации с учетом рисков (рис. 11). Так, для повышения производительности до точки A_1 следует выбирать метод M_1, до A_2 — M_2, до A_3 — M_3.

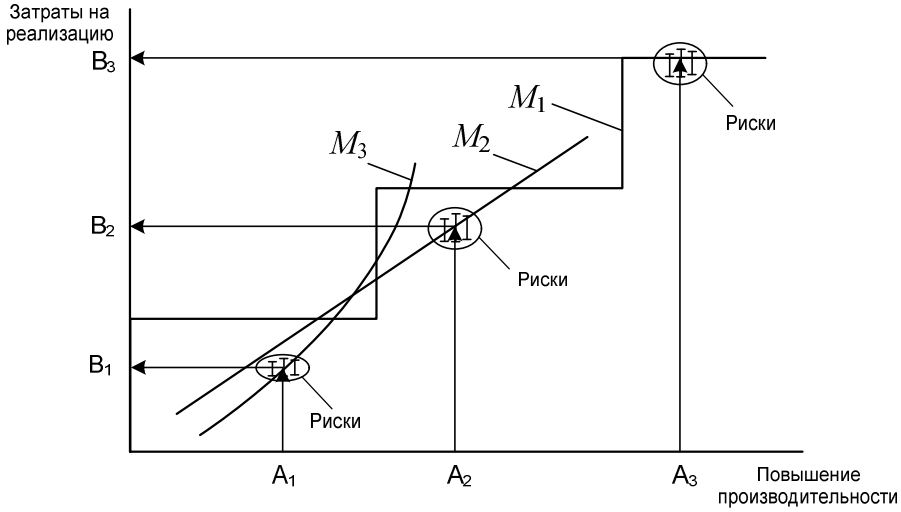


Рис. 11. Выбор метода компенсации для услуги $c_{1,1}$

Кроме учета приведенных затрат при выборе метода компенсации ИТ управления ИТ-инфраструктурой руководствуется следующими соображениями: при соизмеримых затратах отдается предпочтение методу, компенсирующему увеличение соответствующей составляющей в правой части выражения (1); при выборе метода учитывается загруженность ТКС, поскольку при перегруженной ТКС методы компенсации M_1 и M_2 не дадут желаемого результата, а при незагруженной ТКС выбор метода M_3 не повлияет на повышение уровня сервиса. Реализация управления осуществляется посредством правил, подобных (3):

$$
\begin{aligned}
&< система_правил >::=[< продукция >]\\
&< продукция >::=< условие > \rightarrow < следствие >\\
&< условие >::=[< простое_условие >]\\
&< простое_условие >::=< объект >< атрибут >< предикат >< значение >\\
&< объект >::= УП_g,\ i=\overline{1,n}\\
&< следствие >::=[< указание >|< формула >|< программа >]\\
&< предикат >::=|\neq|<|>|\leq|\geq\\
&< атрибут >::=< приложение >|< приоритет_приложения >|< услуга >|\\
&|< бизнес_процесс >|< важность_процесса >|< параметр >|< статус_ИТС >
\end{aligned}
\tag{3}
$$

На рис. 12 изображен фрагмент CPM с замкнутым контуром управления поддержания согласованного уровня услуг. В этом случае СУИ вместе с управлением по отклонению использует принцип управления по возмущению.

Подсистема мониторинга измеряет параметры, оказывающие влияние на качество услуги s_i, предоставляемой приложением $П_l$. Модуль анализа сравнивает фактическое значение T_l^* показателя производительности услуги s_i с целевым значением T_l. При $\Delta T_l = T_l^* - T_l > 0$ модуль анализа выдает сообщение о несоответствии уровня услуги s_i, оговоренному в SLA. В ходе дальнейшего анализа сравнивается фактическое количество a_l^* пользователей услуги s_i со значением $a_{\max,1}$. Кроме этого, проверяется загруженность процессора P_l^* и используемая память ВМ с приложением $П_j$. Если $\Delta a_l = a_l^* - a_l > 0$ и $\Delta P_l = P_l^* - P_l > 0$, то администратор, анализируя степень ухудшения сервиса по значению ΔT_l, превышение допустимого количества пользователей Δa_l услуги s_i и загруженность процессора ΔP_l ВМ, принимает

решение об увеличении процессорной емкости приложению $П_l$. Управление на уровне ресурсов может осуществляться автоматически (с использованием координатора), когда сигнал от подсистемы анализа инициирует автоматическое выделение дополнительной процессорной емкости приложению $П_l$.

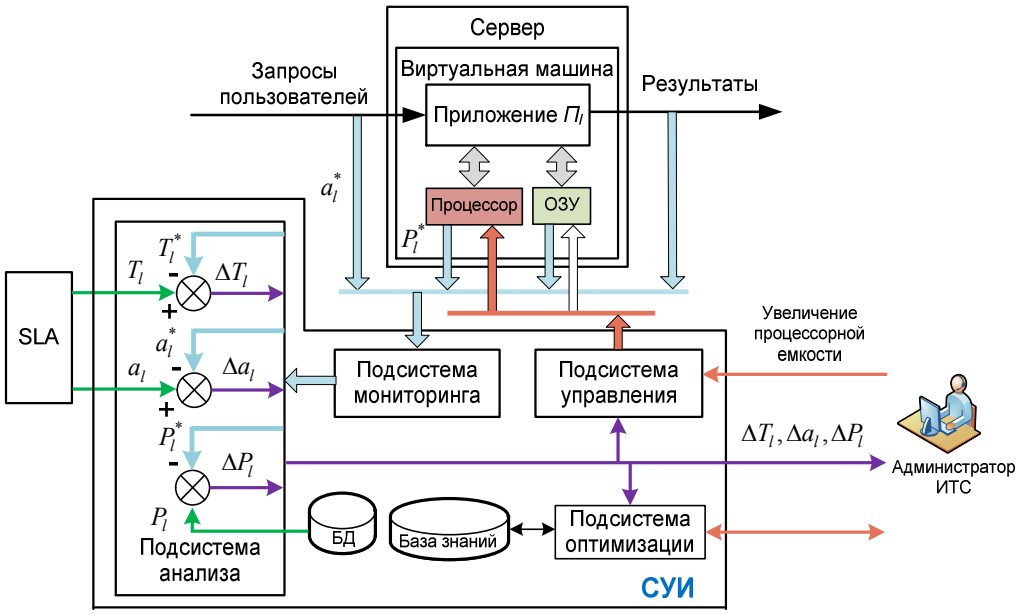


Рис. 12. Замкнутый контур поддержания услуг на согласованном уровне с комбинированным принципом управления

Для реализации декомпозиционно-компенсационного способа организации управления уровнем услуг должны быть разработаны математические модели и алгоритмы кластеризации приложений, распределения и перераспределения ресурсов и управления потоками в ТКС. Решение этих вопросов рассматривается во многих работах, например, в [11, 12].

Выводы

Предложена методика управления уровнем услуг на основе сервисно-ресурсных моделей, создаваемых на основе системной методологии в рамках процесса управления ИТ-инфраструктурой, что позволяет не только легко выявлять факторы, приведшие к снижению уровня услуг, но и определять рациональные компенсирующие воздействия для восстановления согласованного уровня услуг. Рассмотрены особенности реализации декомпозиционно-компенсационного способа организации управления уровнем услуг при использовании сервисно-ресурсных моделей.

Список литературы

1. Foundations of IT Service Management, based on ITIL/ ITSMF-NL// Van Haren Publishing, Second edition. – 2005. – 231 p.
2. Service Support (IT Infrastructure Library)/ Office of Government Commerce (OGC). Stationery Office. – 12th impr. – 2005. – 312 p.
3. Cannon D. ITIL Service Strategy. 2011 Edition / D. Cannon. – UK: TSO, Norwich. – 2011. – 496c.
4. Ролик А.И. Система управления корпоративной информационно-телекоммуникационной инфраструктурой на основе агентского подхода / А.И. Ролик, А.В. Волошин, Д.А. Галушко, П.Ф. Можаровский, А.А. Покотило // Вісник НТУУ «КПІ»: Інформатика, управління та обчислювальна техніка. – К.: «ВЕК+», 2010. – № 52. – С. 39–52.
5. Будкова Л. Методическое руководство для подготовки к профессиональным экзаменам ISO 20000 Foundation и ISO 20000 Foundation Bridge» / Л. Будкова, Р. Журавлёв – М.: Клеверикс, 2010. – 123 с.
6. Голубцов В. Сервисно-ресурсная модель. От теории к практике / В. Голубцов, М. Федоренко // Альманах ITSM 2012. – itSMF России. – 2012.
7. Hubbert E. TechRadar™ For I&O Professionals: IT Service Management Processes, Q1 2012 / E. Hubbert, J.P. Garbani, G. O'Donnell, S.Mann, J. Rakowski. – Forrester Research, Inc. – 2012. – Feb. 7. – 44 p.
8. IT Service Management: An Introduction// J.V. Bon, G. Kemmerling, D. Pondman, Publisher: Van Haren Publishing. – 2002. – 217 p.

9. Information technology. Service management. Part 1: Specification: ISO/IEC 20000-1:2005. – ISO/IEC, 2005. – 16 p.
10. 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.
11. Теленик С.Ф. Управління навантаженням і ресурсами центрів оброблення даних при виділених серверах / С.Ф. Теленик, О.І. Ролік, М.М. Букасов, Р.В. Римар, К.О. Ролік // Автоматика. Автоматизація. Електротехнічні комплексі та системі. – 2009. – №2 (24). – С. 122–136.
12. Теленик С.Ф. Моделі і методи розподілу ресурсів в системах з серверною віртуалізацією / С.Ф. Теленик, О.І. Ролік, М.М. Букасов, О.А. Косован., О.І. Кобец // Зб. наук. праць ВІТІ НТУУ «КПІ». – Випуск № 3. – Київ: ВІТІ НТУУ «КПІ», 2009. – С. 100–109.

Відомості про авторів

Аксьоненко С. В.	науковий співробітник кафедри обчислювальної техніки НТУУ "КПІ"
Амонс О. А.	к.т.н., доцент автоматики та управління в технічних системах НТУУ "КПІ"
Бузовський О. В.	д. т. н., професор кафедри обчислювальної техніки НТУУ "КПІ"
Великий М. М.	студент кафедри обчислювальної техніки НТУУ «КПІ»
Габінет А. В.	аспірант кафедри обчислювальної техніки НТУУ "КПІ"
Губська А. О.	студентка кафедри обчислювальної техніки НТУУ "КПІ"
Гусев Е. И.	здобувач кафедри обчислювальної техніки НТУУ "КПІ"
Долголенко О. М.	к.т.н., с.н.с., доцент кафедри обчислювальної техніки НТУУ «КПІ»
Зимчук И. В.	к.т.н., доцент кафедри автоматизованих систем управління Житомирського військового інституту ім. С. П. Корол ьова
Іванов Д. Г.	здобувач кафедри обчислювальної техніки НТУУ «КПІ»
Калашник В. В.	студент кафедри автоматизованих систем обробки інформації та управління НТУУ «КПІ»
Короненко А. М.	аспірант кафедри обчислювальної техніки НТУУ "КПІ"
Луцький Г. М.	д. т. н., професор кафедри обчислювальної техніки НТУУ "КПІ"
Марковський О. П.	к.т.н., доцент кафедри обчислювальної техніки НТУУ «КПІ»
Мисюра Е. Б.	к.т.н, с.н.с. НДІ інформаційних процесів НТУУ «КПІ»
Невдащенко М. В.	старший викладач обчислювальної техніки НТУУ "КПІ"
Павлов А. А.	д.т.н., професор кафедри автоматизованих систем обробки інформації та управління НТУУ «КПІ»
Подрубайло О. О.	аспірант кафедри обчислювальної техніки НТУУ "КПІ"
Роковий О. П.	старший викладач кафедри обчислювальної техніки НТУУ "КПІ"
Ролік О.І.	д.т.н. професор кафедри автоматики та управління в технічних системах НТУУ "КПІ"
Сергієнко А.М.	д.т.н., с.н.с. кафедри обчислювальної техніки НТУУ "КПІ"
Сергієнко П. А.	студент кафедри спеціалізованих комп'ютерних систем НТУУ "КПІ"
Стіренко С. Г.	к.т.н., доцент кафедри обчислювальної техніки НТУУ «КПІ»
Сторожук В. О.	студент кафедри обчислювальної техніки НТУУ «КПІ»
Терещенко С. Ю.	студент кафедри обчислювальної техніки НТУУ «КПІ»
Федоречко О. І.	аспірант кафедри обчислювальної техніки НТУУ «КПІ»
Чечель В. Р.	студентка інституту комп'ютерних та інженерно-технологічних наук національного університету кораблебудування ім. С.Й. Макарова

SUMMARY

Pavlov A.A., Misura E.B. The PDC-algorithms for one machine scheduling problem to minimize total earliness and tardiness…… 4

Amons O.A., Data synchronization in distributed information systems ……...……. 20

Markovskiy O.P., Tereshchenko S. Y., Fedorechko O.I. Method for correction of triple errors during data transmission in bimary symmetric channels……...…… 33

Pavlov A.A., Kalashnik V.V. Recommendations on the selection of the area of the active experiment for the one-dimensional polynomial regression analysis…….….. 41

Markovskiy O.P., Ivanov D. G., Velikiy M.M., Nevdashenko M.V. Method for data reservation and shortcut recovering in remote memory systems ……..……... 46

Stirenko S. G. The load rebalancing method in cluster systems……. 55

Markovskiy O.P., Gubska A.O., Chechel V.R., Fedorechko O.I. Method for double synchronization errors correcting in asynchronous digital data transmission…….. 60

Rokovoy A. P., Gabinet A. V. Method of forecasting the delay dispersion in transmission of voice data 68

Buzovskyi O. V., Podrubailo O. O. Methods and algorithms of distributed join for in-memory data grids ..…….……... 73

Loutskii H. M., Dolgolenko O. M., Aksyonenko S. V., Storozhuk V. O. Modeling restricted omplementation of data flow architecture in the structure of the superscalar processor core……...…… 83

Koronenko A. M. Research of dynamics load multiservice Networks …….. 95

Zymchuk I. V. Reduction of control object models for the synthesis of simple regulators ……...…… 102

Gusev E.I. Mathematical modeling of distributed cluster system which uses shared everything approach (Oracle RAC) ……...…….. 106

Sergiyenko A.M., Sergiyenko P.A. Implementation of square root in FPGA…….…… 114

Rolik O.I. Method of service level management on service-resource models based……. 119

Information about authors…….. 130

Summary…… 131