УДК 004.67:004.428.4

СТІРЕНКО С.Г.,
ЗІНЕНКО О.І.

ЗБЕРІГАННЯ РОЗРІДЖЕНИХ МАТРИЦЬ ВЕЛИКИХ РОЗМІРНОСТЕЙ В СИСТЕМІ З ЛОКАЛЬНОЮ ПАМ'ЯТТЮ

В цій статті запропоновано ієрархічну схему зберігання розріджених матриць великих розмірностей в системі з локальною пам'яттю, який може бути використаний при розробці застосунків з лінійної алгебри. Проведено аналіз ефективності застосування цієї схеми та порівняння з іншими схемами. Запропоновано метод обробки таких матриць в розподіленій системі.

In this article the scheme of large scale sparse matrix storage in the distributed memory system, which can be useful in linear algebra applications development, is proposed. The comparative efficiency analysis of proposed and existing schemes is made. Also the method of proceeding such matrices in a distributed memory computer system is proposed.

Вступ

Велика кількість наукових задач та прикладних задач при реалізації їх розв'язку на ЕОМ може бути зведена до роботи з матрицями та векторами, для яких існує багато розроблених алгоритмів та структур даних. Зокрема для розв'язку диференційних рівнянь методом скінченних елементів або скінченних різниць, розв'язку систем лінійних алгебраїчних рівнянь, задачі знаходження власних чисел та векторів лінійних операторів тощо, існує декілька математичних методів розв'язку, що базуються на використанні матриць та інших алгебраїчних структур [1].

З розвитком засобів обчислювальної техніки став можливим розв'язок задач, що містять великі обсяги вхідних даних [2], таких як системи рівнянь з декількома мільйонами невідомих. Серед них окремо виділяються задачі, вхідні дані яких є розрідженими матрицями, які містять відносно невелику кількість ненульових елементів. Для таких задач розроблені сучасні швидкі алгоритми, що базуються на теорії підпросторів Крилова, зокрема алгоритми Ланцоша та Відемана для розв'язку задачі власних чисел та розв'язку системи лінійних рівнянь відповідно [3]. Ці методи були успішно застосовані, наприклад, до атаки на криптографічний алгоритм RSA.

Стримуючим фактором ефективного застосування програмних реалізацій алгоритмів розв'язку задач такого класу на ЕОМ є необхідність зберігання великої кількості даних під час обробки. На даному етапі розвитку техніки вхідні матриці не можуть бути розміщені в оперативній пам'яті однієї ЕОМ [4], хоча повністю вони необхідні лише на початковому етапі алгоритму. Тому необхідно запропонувати таку схему зберігання розріджених матриць великої розмірності, яка б дозволила завантажувати всі вхідні дані (або їх частину) в локальну оперативну пам'ять кожного вузла кластера та виконувати їх ефективну обробку для подальшого використання в програмних реалізаціях математичних алгоритмів Крилівського типу. Також необхідно, щоб вхідні дані могли б бути приведені до запропонованої схеми з найбільш поширеного формату зберігання розріджених матриць на зовнішніх носіях Matrix Market [5].

Схеми зберігання розріджених матриць

Всі схеми зберігання розріджених матриць базуються на зменшенні розміру даних, що зберігаються, завдяки зберіганню лише ненульових елементів. Оскільки в розріджених матрицях за визначенням таких елементів значно менше ніж ненульових, використання таких схем дозволяє зберігати матриці більших розмірностей. Позначимо ω кількість ненульових елементів матриці, та $k_\omega = \frac{\omega}{n \times m}$ коефіцієнт розрідженості матриці. Важливою властивістю схеми зберігання є необхідність наявності способу визначити елемент матриці за його рядковим та стовпцевим індексами.

Найпростіший підхід запропоновано в схемі Matrix Market [5]. Вона передбачає збері-

гання впорядкованого за індексом рядка та стовпця в вихідній матриці лінійного масива трійок "елемент, індекс рядка, індекс стовпця".

Більш складний, проте більш ефективний метод запропоновано в схемах CSR (compressed sparse row) та CSC (compressed sparse column) [6], які є симетричними та відрізняються лише використанням рядків або стовпців в якості основного індексу. В схемі CSR зберігається лінійний масив пар "елемент, індекс стовпця" та лінійний масив індексів в головний масив, що показує початок кожного рядка.

Схема Кнута-Ларкомба [6] є прикладом зв'язної схеми зберігання. Для симетричних матриць в ній застосовується структура з елемента та двох покажчиків на правий та верхній ненульові елементи матриці. Для матриць загального вигляду легко розробити модифікацію, що містить інші покажчики. В залежності від потреб алгоритму, що реалізується, існує можливість використати декілька покажчиків за необхідними напрямами обходу. Ця схема потребує достатньо великого обсягу пам'яті для службових даних та незручна в зберіганні на зовнішніх носіях, але є найшвидшою при виконанні ітерацій за ненульовими елементами.

Гіперматрична схема

Дамо визначення гіперматриці. Гіперматрицею k-го порядку називається така матриця, елементами якої є гіперматриці $(k-1)$-го порядку. Гіперматрицею першого порядку є така матриця, елементами якої є скаляри. Гіперматриця k-го порядку може бути представлена в вигляді гіперматриці $(k+1)$-го порядку шляхом групування її сусідніх елементів. Оскільки матриця з одним елементом з точки зору алгебри залишається матрицею, процес групування та підвищення порядку може продовжуватися нескінченно. Наведемо приклад гіперматриці

$$\begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{pmatrix} \leftrightarrow \begin{pmatrix} \begin{pmatrix} \begin{pmatrix} 1 \\ 5 \end{pmatrix} & \begin{pmatrix} 2 \\ 6 \end{pmatrix} \end{pmatrix} & \begin{pmatrix} \begin{pmatrix} 3 \\ 7 \end{pmatrix} & \begin{pmatrix} 4 \\ 8 \end{pmatrix} \end{pmatrix} \\ \begin{pmatrix} \begin{pmatrix} 9 & 10 \end{pmatrix} \\ \begin{pmatrix} 13 & 14 \end{pmatrix} \end{pmatrix} & \begin{pmatrix} \begin{pmatrix} 11 & 12 \end{pmatrix} \\ \begin{pmatrix} 15 & 16 \end{pmatrix} \end{pmatrix} \end{pmatrix}.$$

Розглянемо обернений процес: мається гіперматриця k-го порядку, з якої необхідно отримати гіперматрицю першого порядку. Для цього виконати наступні ітерації. Одночасно замінити всі гіперматриці першого порядку на їх елементи, сформувавши таким чином нові гіперматриці першого порядку замість таких другого порядку, але більшої розмірності. При цьому порядок загальної гіперматриці зменшиться на одиницю. Продовжувати ітерації доки порядок загальної гіперматриці не дорівнюватиме одиниці.

Однак при такому підході існує наступна проблема: гіперматриця за визначенням є більш складною структурою ніж просте групування елементів матриці, тому може містити матриці-елементи різних розмірностей. В цьому випадку, їх знищення і використання лише їх елементів напряму неможливе. Таким чином необхідно перевіряти умови знищення: сумарна кількість елементів в кожному рядку та кожному стовпці після знищення матриць-елементів має бути відповідно однаковою. Для спрощення, розглядатимемо тільки такі гіперматриці, які задовольняють більш строгим умовам: всі матриці-елементи в рядку гіперматриці мають містити однакову кількість елементів в стовпці; всі матриці-елементи в стовпці гіперматриці мають містити однакову кількість елементів в рядку.

Застосуємо цей підхід до зберігання даних в ЕОМ з локальною пам'яттю. Нехай задана матриця великої розмірності. Виділимо підматриці однакової розмірності зі скалярів та збережемо їх за певною схемою. Додатково створимо службову структуру даних, в якій зберігатиметься розмірність таких підматриць та їх взаємне розташування. Таким чином отримали гіперматричну схему зберігання другого порядку. Отримані підматриці (або гіперматриці першого порядку) можуть зберігатися в локальній пам'яті окремих вузлів незалежно; необхідно лише кожному вузлу надати інформацію про розташування інших частин матриці, якщо це необхідно за алгоритмом.

Схема зберігання підматриць або гіперматриці в цьому випадку може бути будь-якою. Але при застосуванні, наприклад, традиційної щільної схеми, використання запропонованого методу недоцільне, оскільки потребує лише додатової пам'яті для службових даних, і має перевагу лише при використанні блочних ал-

горитмів, для яких такий формат представлення є зручним. У випадку розріджених матриць з великою кількістю нульових елементів підматриця може виявитися складеною повністю з нульових елементів. Зберігати таку підматрицю недоцільно, тому в структурі гіперматриці можна зберігати лише сигнальне значення. Таким чином, гіперматриця другого порядку також може виявитися розрідженою, і до неї може бути застосована одна з схем зберігання розріджених матриць. Таким чином процес підвищення порядку розріджених гіперматриць можна продовжувати, доки наступна може бути ефективно збережена за розрідженою схемою або не буде досягнуто певного порядку. Проте недоліком збільшення порядку є збільшення часу доступу до окремого елементу за відомими індексами. Аналіз ефективності та швидкості роботи запропонованої схеми наведено далі.

Порівняльний аналіз схем зберігання

Наведемо теоретичні розрахунки ефективності згаданих схем та перевіримо їх теоретично. Для спрощення аналізу, вважатимемо що розглядається збереження матриць чисел з плаваючою комою розмірністю 64 двійкових розряди на 64-розрядній системі, оскільки такі системи є найбільш розповсюдженими існуючими кластерними системами. Тобто розмір як елемента даних, так і покажчика або індекса становитиме 8 байт.

Схема Matrix Market потребує зберігати для кожного елемента безпосередньо значення та два індекси: рядковий та стовпцевий. Тому для зберігання за цією схемою необхідно

$$M_{MM} = 8 \cdot 3 \cdot \omega \quad \text{байт.}$$

Однак класична реалізація цієї схеми як впорядкованого за індексами лінійного масиву потребуватиме $O(\log_2 k_\omega n \cdot \log_2 k_\omega m)$ операцій для пошуку елемента за індексами, $O(1)$ – для вибору наступного елемента в рядку та $O(k_\omega \cdot m)$ – для вибору наступного елементу стовпця.

Схема CSR потребує окрім значення ненульового елементу зберігання також його стовпцевого індексу та окремого масиву індексів початку рядків, тобто необхідний обсяг пам'яті

$$M_{CSR} = 8 \cdot (2 \cdot \omega + n) \quad \text{байт.}$$

При чому пошук елемента за індексами може бути виконаний за $O(\log_2 k_\omega m)$ операцій, перехід до наступного елементу в рядку за $O(1)$ операцій, а до наступного елементу в стовпці за $O(\log_2 k_\omega m)$ операцій.

Схема Кнута-Ларкомба з застосуванням лише двох покажчиків: на елементи знизу та справа – тобто формуванням однозв'язних списків, та застосуванням додаткових масивів покажчиків на початки рядків та стовпців потребуватиме

$$M_{KL} = 8 \cdot (3 \cdot \omega + n + m) \quad \text{байт.}$$

Для цієї схеми пошук елементів за індексами матиме складність $O(\min(k_\omega m, k_\omega n))$ операцій, але перехід до наступного елементу як в рядку, так і в стовпці – лише $O(1)$ операцій.

Безперечною перевагою гіперматричних схем є можливість застосування різних схем, як розріджених, так і щільних, при зберіганні гіперматриць різних порядків. Тому теоретично можливо розрахувати необхідні обсяги лише якщо всі такі схеми відомі, або в термінах обсягів пам'яті, необхідних для зберігання за зазначеними схемами. Для зберігання матриць найбільших розмірностей доцільно використовувати таку схему, яка потребує найменшого обсягу пам'яті. З розглянутих схем в загальному випадку це CSR схема. Знайдемо необхідний обсяг пам'яті для збереження гіперматриці k-го порядку за такою схемою

$$M_H = 8 \cdot \left(\sum_{i=1}^{p_1} (2 \cdot \omega_{1,i} + \frac{n}{p_1^{0.5}}) + \sum_{k=2}^{g} \sum_{i=1}^{p_k} (2 \cdot \omega_{k,i} + \sqrt{p_k}) \right),$$

Де $\omega_{i,k}$ – кількість ненульових елементів в i-тій підматриці гіперматриці k-го порядку, p_k – кількість елементів гіперматриці k-го порядку, g – найбільший порядок гіперматриці.

Очікувані з теоретичних розрахунків обсяги використаної пам'яті в залежності від розрідженості наведені на рис. 1. Розглядалися квадратні розріджені матриці, в яких ненульові елементи розподілені рівномірно. Вважаємо, що розмірність матриць постійна, а кількість ненульових елементів змінюється, що призводить до зміни коефіцієнту розрідженості. При розрахунках для гіперматричної схеми, вважаємо, що розмір гіперматриці першого порядку визначається як $\sqrt[g]{n}$, де g – максимальний порядок.

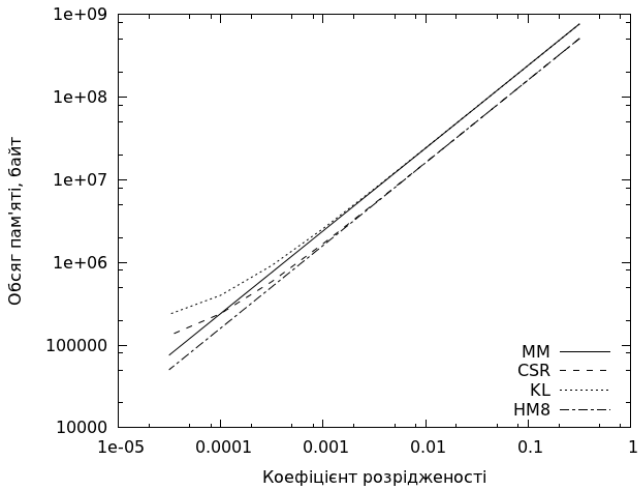

Рис. 1. Обсяг пам'яті, необхідна для зберігання матриці розміром $10^4 \times 10^4$ елементів.

Тут і далі в графіках використано наступні позначення: MM – схемаMatrix Market; CSR – розріджена рядкова схема; KL – схема Кнута-Ларкомба; HMi – гіперматрична схема і-го порядку.

Також було розраховано ефективність застосування запропонованого методу у порівнянні з переліченими вище класичними методами (рис. 2).

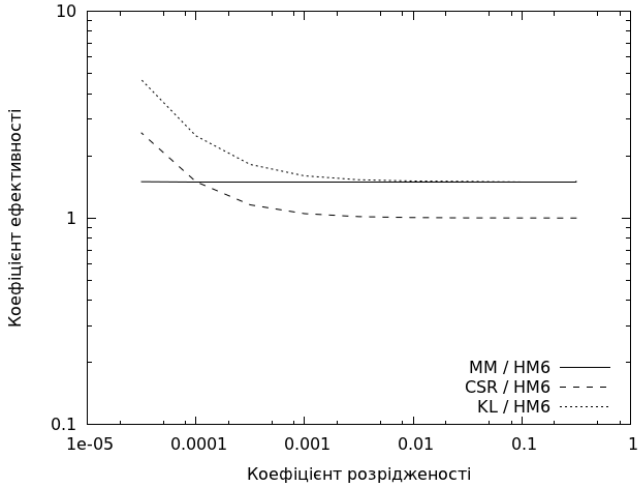

Рис. 2. Відношення пам'яті, необхідної для існуючих методів, до пам'яті, що потребується для запропонованого методу.

Для підтвердження теоретичних розрахунків та практичного порівняння схем було розроблено програми, що реалізують відповідні схеми зберігання розріджених матриць на мові C++ з використанням бібліотеки паралельних обчислень для систем з локальною пам'яттю MPI. Робота виконувалась на кластері центру суперкомп'ютерних обчислень НТУУ "КПІ". Для перевірки часу роботи алгоритмів зі схемами зберігання, що порівнюються, було також розроблено програмну реалізацію генерації підпросторів Крилова, що включає виконання матрично-матричних та матрично-векторних операцій. Було проведено серію запусків для різних згенерованих довільним чином розріджених матриць великих розмірностей з фіксованою кількістю ненульових елементів. Ненульові елементи розподілені рівномірно по всій матриці в першій серії експериментів, та нормально відносно головної діагоналі. Результати приведені на рис. 3 та рис. 4.

Експериментальні дані також показують переваги запропонованого методу зберігання над існуючими методами при його застосуванні до розріджених матриць, розмірності яких вимірюються мільйонами елементів, а коефіцієнт розрідженості відносно невеликий і становить відповідно $10^{-6} \div 10^{-4}$ для різних розмірностей.

Зі зменшенням коефіцієнту розрідженості доцільно збільшувати порядок використовуваних гіперматричних схем, оскільки збільшується ймовірність того, що гіперматриці також будуть розрідженими.

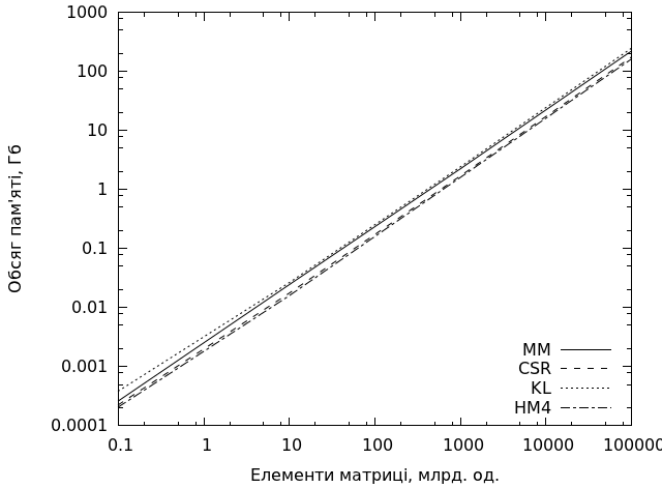

Рис. 3. Середній обсяг пам'яті, необхідний для зберігання розріджених матриць з коефіцієнтом розрідженості $k_\omega = 0.0001$.

Також підтверджено емпіричний висновок, що застосування схем зберігання розріджених матриць доцільне лише для таких матриць, які містять таку кількість ненульових елементів, яка більша ніж одна з розмірностей матриці не більше ніж на декілька порядків (див. рис. 2).

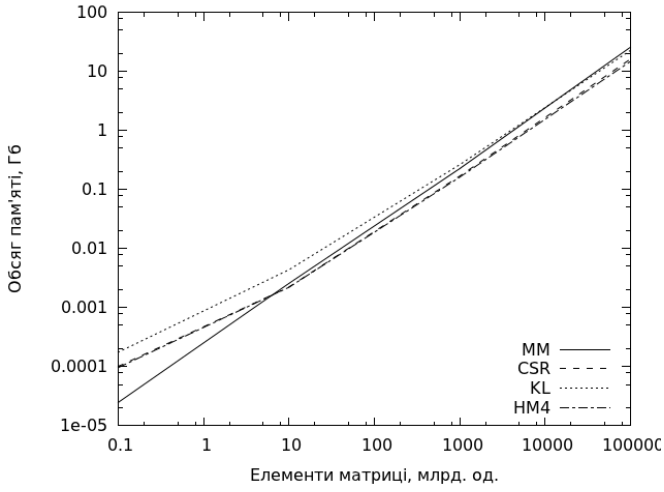

Рис. 4. Середній обсяг пам'яті, необхідний для зберігання розріджених матриць з коефіцієнтом розрідженості $k_\omega = 0.0001$.

Метод організації обчислень та деталі програмної реалізації

Однією з поставлених при розробці схеми задач була можливість ефективного використання запропонованої схеми при розробці програм для копм'ютерної системи з локальною пам'яттю. Оскільки вирішення сучасних задач потребує використання матриць, що містять декілька трильйонів елементів, розмістити всі вихідні дані в пам'яті одночасно неможливо. Запропонована схема дозволяє розбити вихідні дані на незалежні частини, які можуть зберігатися окремо в локальній пам'яті різних вузлів. Тобто, якщо вихідна матриця була представлена в вигляді гіперматричної схеми g-го порядку, то гіперматриці всіх нижчих порядків можуть зберігатися окремо, за умови наявності у всіх вузлів інформації про них. Алгоритмічна обробка найпоширеніших операцій може виконуватись за допомогою блочних методів, таких як методи Фокса, Ланцоша та Відемана-Копперсміта.

Іншою поставленою задачею була можливість перетворення з існуючих схем на запропоновану за обчислювально простим алгоритмом. Гіперматричні схеми можуть мати підматриці, що зберігаються за різними схемами, тому застосування для гіперматриць першого порядку такої ж схеми, як і для вхідних даних дозволить перетворити їх шляхом тривіального введення коригуючого доданку при індексації. Однак, такий підхід може потенційно збільшити необхідний обсяг пам'яті для зберігання даних. Найбільш ефективним виявився підхід з використанням CSR схеми в якості базової, тому розглянемо перетворення з формату Matrix Market до гіперматричного CSR. Оскільки в першому зберігаються як стовпцеві, так і рядкові індекси, а в останньому в явному вигляді тільки стовпцеві, їх перетворення виконується тривіальним відніманням індексу першого стовпця в гіперматриці першого порядку. Індекси початку рядків можуть бути сформовані під час зміни рядкового індексу в вихідних даних. Таким чином маємо наступне обмеження на розмір вхідних даних: в пам'яті вузла мають бути розміщені один рядок під матриць другого порядку. Як буде показано нижче, це обмеження можна обійти при використанні більш складних алгоритмів роботи обчислювальних процесів.

Читання програмою вихідних даних з зовнішнього накопичувача може бути організоване наступним чином. Оскільки в багатьох кластерних системах фізичний доступ до системи зберігання даних має один або декілька вузлів, в той час як інші отримують дані мережею, недоцільно намагатися виконувати читання паралельно. Натомість рекомендується організувати обчислення за парадигмою «диспетчер-робітник», в якій процес-диспетчер виконуватиме введення даних та їх розсилку робітникам. При цьому він формуватиме загальну інформацію про знаходження певних частих даних в обчислювальних процесах. Оскільки швидкість обробки даних процесором значно вище швидкості передачі мережею, безпосереднє перетворення вхідних даних може виконуватись на керуючому вузлі, після чого дані для виконання алгоритмічної обробки можуть передаватися обчислювальним вузлам. Передачу даних та введення наступної їх частини можна сумістити в часі використовуючи асинхронні передачі, які доступні в MPI.

Разом з тим при виконанні обчислювального алгоритму в залежності від архітектури зв'язків кластера можна виконувати передачу дних як через керуючий вузол, так і між обчислювальними вузлами безпосередньо.

Авторська програмна реалізація виконана з використанням шаблонів мови C++ для оформлення класів різних схем зберігання. Параметризованим є тип елементу матриці. Це дозволяє використовувати каскадування різних схем зберігання в гіперматричну схему і разом

з тим накладає необхідне обмеження на однаковість типів елементів однієї матриці. Використання шаблонів було також викликане відсутністю накладних витрат на перевірку типів під час роботи програми та диспетчеризацію віртуальних викликів.

Архітектурно-залежні оптимізації

В залежності від архітектури процесора цільової системи та організації кластера в цілому можливе застосування деяких оптимізацій або існування додаткових обмежень.

Найбільш простим та ефективним розбиттям є таке, в якому кількість гіперматриць першого порядку кратна кількості обчислювальних процесів комп'ютерної системи. Такий підхід також значно спрощує програмування прикладних алгоритмів. У випадках, коли через інші обмеження така кількість підматриць не може бути створена, можливо до визначити вхідні дані певною кількістю нульових елементів. Це не вплине на роботу більшості алгоритмів, та незначно збільшить необхідний обсяг пам'яті, оскільки нульові елементи не зберігаються безпосередньо.

При застосуванні гіперматричних схем вищих порядків доцільно обирати їх розміри таким чином, щоб матриця першого порядку (або декілька їх) вміщувалась в кеш-пам'яті першого рівня процесора, а другого порядку – в кеш пам'яті другого рівня відповідно. Наприклад для використовуваних в кластері НТУУ «КПІ» процесорів розміри матриць з елементами з плаваючою комою становитимуть не більше 640x640 та 2560x2560 елементів відповідно (що відповідає розмірам кеш-пам'яті 512 кб та 8 Мб).

Висновки та перспективи

Запропонована схема зберігання розріджених матриць великих розмірностей задовольняє вимогам, що висуваються до таких схем та дозволяє зменшити обсяг пам'яті, що необхідний для зберігання таких даних, на 7-10%.

Рекомендований метод організації обчислень дозволить використовувати локальну пам'ять вузлів кластерної системи для збереження таких даних.

Отримані результати можуть створити передумови для подальшого розвитку програмних комплексів для розв'язку сучасних математичних та фізичних проблем та моделювання. Вони також можуть бути корисні при розробці програмного забезпечення для виконання аналізу криптографічної стійкості алгоритмів шифрування та для аналізу даних методом головних компонент, наприклад, в системах пошуку інформації в мережі.

Експерименти з запропонованою схемою показують, що застосування надмірно великого порядку гіперматриці призводить до використання більшого обсягу пам'яті та збільшення часу доступу до окремого елементу. Рекомендується для матриць розмірністю порядку $10^n \times 10^n$ застосовувати схеми не більше ніж $(n-4)$-го порядку.

В подальшому планується дослідити можливість застосування неквадратних підтматриць в запропонованій схемі, а також використання нерівномірних розбиттів для отримання більшого коефіцієнту ефективності гіперматричної схеми.

Важливим напрямом розвитку запропонованих ідей є розробка евристичного методу визначення областей матриці великої розмірності, що містять лише нульові елементи.

Автори висловлюють подяку суперкомп'ютерному центру НТУУ «КПІ» [7] та лабораторії Intel Manycore Testing Lab [8] за надані ресурси для досліджень.

Перелік посилань

1. *Opsahl, T.* Node centrality in weightednetworks: Generalizing degree and shortest paths [Text] / Opsahl T., Agneessens F., Skvoretz J. // Social Networks. – 2010. – Vol. 32, no. 3. – Pp. 245-251. – ISSN 03788733.
2. *Стиренко С.Г.* Форма представления функциональных программ для автоматического распараллеливания / Стиренко С.Г., Грибенко Д.В., Зиненко А.И. // Вісник НТУУ «КПІ». Інформатика, управління та обчислювальна техніка: Зб. наук. праць. – К.: «Век+», 2009. – №50. – с. 184-187.

3. *Wiedemann D.H.* Solving sparse linear equations over finite fields [Text] // IEEE Transactions of Information Theory. – New York, NY, USA: IEEE Press, 1986. – Vol. 32, no. 1. – Pp. 54-62. – ISSN 0018-9448.
4. *Sutter, H.* Software and concurrency revolution / H. Sutter, J. Laurus // Queue. – New York: ACM, 2005. – Vol. 3, № 7. – P. 54-62. – ISSN 1542-7730.
5. *Boisvert R.F.* Matrix market: a web resource for test matrix collections / R. Boisvert, R. Pozo, K. Remington, et. al. // In Proceedings of the IFIP TC2/WG2.5 working conference on Quality of numerical software: assessment and enhancement, *Ronald F. Boisvert (Ed.)* – : London, UK: Chapman & Hall, Ltd., 1997. – Pp. 125-137.
6. *Montagne E.* An optimal storage format for sparse matrices [Text] / Montagne E., Ekambaram A.// Information Processing Letters. – San Francisco, CA, USA: Information processing society, 2004. – Vol. 90, no. 2. – Pp. 87-92. – ISSN 0020-0190.
7. Суперкомп'ютерний центр НТУУ «КПІ» [електронний ресурс]. – Режим доступу: http://www.hpcc.org.ua/. – Останнє звернення 10.09.2010, назва з екрану.
8. Intel® Manycore Testing Lab [electronic resource]. – Access mode: http://software.intel.com/en-us/articles/intel-many-core-testing-lab/. – Останнє звернення 12.09.2010, назва з екрану.