УДК 004.032.24:519.614.2

ГРИБЕНКО Д.В.,
СТІРЕНКО С.Г.

РЕАЛІЗАЦІЯ РОЗВ'ЯЗАННЯ ЗАДАЧІ ЗНАХОДЖЕННЯ ВЛАСНИХ ЧИСЕЛ ТА ВЕКТОРІВ МЕТОДОМ ЛАНЦОША НА КЛАСТЕРНІЙ СИСТЕМІ

В статті розглянуто супутні проблеми, що виникають при реалізації методу Ланцоша на ЕОМ, а саме: вибір способу зберігання розрідженої матриці, оптимізація коду за допомогою застосування SIMD команд сучасних процесорів. Виконано реалізацію алгоритмів множення матриці на вектор в різних форматах та їх експериментальне порівняння. Розроблено програмне забезпечення знаходження власних чисел та власних векторів дійсних симетричних розріджених матриць для кластерної обчислювальної системи. Застосовано бібліотеку MPI для передачі повідомлень та бібліотеку LAPACK для пост-обробки результатів алгоритму Ланцоша. ПЗ підтримує введення матриць в форматі Matrix Market, що забезпечує сумісність з існуючими пакетами чисельних обчислень.

This article discusses questions that arise during implementation of Lanczos method for clusters: choice of sparse matrix storage format and code optimization with SIMD instructions of modern CPUs. Discussed algorithms of sparse matrix-vector multiplication have been implemented and an experimental comparison was carried out. A parallel cluster eigenvalue and eigenvector solver cluster for real symmetric sparse matrices was developed. It uses MPI for communication and LAPACK for post-processing. It supports the de-facto standard Matrix Market format.

Вступ

Типовою задачею лінійної алгебри є знаходження для заданої матриці $A \in R^{n \times n}$ власних чисел $\{\lambda_i\}$ та відповідних власних векторів $\{x_i\}$, або, іншими словами, розв'язків рівняння $Ax = \lambda x$.

Необхідність знаходження заданої кількості найбільших за абсолютною величиною власних чисел та відповідних векторів розріджених матриць великої розмірності виникає, наприклад, при дослідженні коливань у фізиці. [1] Коливання пружних конструкцій, коливань в аеродинаміці, коливання в електричних колах, коливання молекул та атомів у фізиці елементарних часток – всі ці задачі є різноманітними прикладними аспектами однієї і тієї самої математичної задачі.

Іншою прикладною задачею, що потребує знаходження власних векторів великих розріджених матриць є задача ранжування результатів веб-пошуку, одним із способів вирішення якої є визначення міри центральності вузлів графу веб-сторінок. [2] Якщо задати граф G, вузлами якого є веб-сторінки, а ребрами якого є гіперпосилання, то міра центральності кожного вузла буде характеризувати імовірність того, що людина, читаючи веб-сторінки та переходячи по гіперпосиланням, опиниться на відповідній веб-сторінці. Так, алгоритм PageRank, що використовується пошуковою системою Google, базується саме на визначенні міри центральності кожної веб-сторінки (яку розробники назвали PageRank цієї веб-сторінки). Звичайно, кількість веб-сторінок в мережі Інтернет є дуже великою (за оцінками Google [3], в 2008 році кількість веб-сторінок в мережі Інтернет досягла 10^{12}), але кожна веб-сторінка має порівняно невелику кількість гіперпосилань (до сотень), тому матриця суміжності графу G буде розрідженою.

Наївні методи знаходження власних чисел є простими для реалізації, але мають велику обчислювальну складність. Універсальні методи, такі як QR-розклад, також мають велику обчислювальну складність, але є обчислювально стійкими.

Найбільш ефективними та обчислювально стійкими серед ітераційних методів є проекційні методи, особливо ті з них, що виконують проекції на підпростори Крилова. Іншою перевагою цих методів є можливість ефективного розпаралелювання. Так, якщо матриця A має деякі властивості, а саме, є дійсною, симетричною та розрідженою, метод Ланцоша має значні переваги з точки зору обчислювальної складності над іншими методами, що використовують підпростори Крилова.

Метод Ланцоша для знаходження власних чисел

Метод Ланцоша призначений для знаходження власних чисел симетричної матриці. Вперше був опублікований в 1950 році, [4] практичне застосування було розпочато після

Вхідні дані : $A \in \mathbb{R}^{n\times n}$, причому $A = A^T$, m
Вихідні дані: $Q \in \mathbb{R}^{n\times m}, T \in \mathbb{R}^{m\times m}$

```
1  begin
2      for i ← 1 to n do
3          q_{1,i} ← rand() ;                          // n разів по 1 операції
4      end
5      q_1 ← q_1 / sqrt(⟨q_1, q_1⟩) ;                  // 3n операцій
6      β_1 = 0
7      q_0 = 0
8      for j ← 1 to m do
9          v_j = A q_j − β_j q_{j−1} ;                 // m разів по 2n + U операцій
10         α_j = ⟨v_j, q_j⟩ ;                          // m разів по 2n операцій
11         v_j ← v_j − α_j q_j ;                       // m разів по 2n операцій
12         β_{j+1} = sqrt(⟨v_j, v_j⟩) ;                // m разів по 2n операцій
13         q_{j+1} = v_j / β_{j+1} ;                   // m разів по n операцій
14     end
15 end
```

Рис. 1. Обчислювальний алгоритм методу Ланцоша

того, як були знайдені оцінки похибок. [5] З сучасної точки зору, в контексті знаходження апроксимацій власних чисел, метод Ланцоша є поєднанням методу Релея–Рітца, процесу Грама–Шмідта (або іншого методу ортогоналізації набору векторів) та підпросторів Крилова. [6]

Викладення усіх міркувань, що приводять до методу Ланцоша, є досить довгим та не доречним в даній статті, тому через брак місця наведемо тільки обчислювальну процедуру методу Ланцоша та стислий її аналіз. Вхідними даними є симетрична дійсна матриця A розміром $n \times n$ та необхідна кількість власних чисел m . Метод Ланцоша задає рекурентну послідовність, що визначає елементи тридіагональної матриці T_m, власні числа якої є наближеннями власних чисел матриці A. [4, 7] Введемо наступні позначення для матриці T_m: діагональні елементи позначимо $\alpha_1, \alpha_2, \ldots, \alpha_m$, позадіагональні елементи позначимо $\beta_2, \beta_3, \ldots, \beta_m$:

$$T_m = \begin{pmatrix} \alpha_1 & \beta_2 & 0 & 0 & 0 \\ \beta_2 & \alpha_2 & \beta_3 & 0 & 0 \\ 0 & \ddots & \ddots & \ddots & 0 \\ 0 & 0 & \beta_{m-1} & \alpha_{m-1} & \beta_m \\ 0 & 0 & 0 & \beta_m & \alpha_m \end{pmatrix},$$

Власне обчислювальний алгоритм у вигляді, зручному для реалізації на ЕОМ, представлений на рис. 1. Проведемо аналіз обчислювальної складності цього алгоритму. В нотатках справа від алгоритму показана обчислювальна складність кожного кроку. В рядку 9 вказана деяка величина U, тому що обчислювальна складність множення матриці на вектор залежить від того, чи є матриця щільною чи розрідженою, а в останньому випадку – ще й від способу зберігання.

У випадку, коли матриця A є щільною, множення матриці на вектор потребує $U = 2n^2$ операцій. Тоді обчислювальна складність алгоритму Ланцоша для щільних матриць становить $2mn^2 + 9mn + 4n = O(mn^2)$.

У випадку, коли матриця A є розрідженою, та при використанні відповідних ефективних способів зберігання матриці в пам'яті, множення матриці на вектор потребує $U = 2k$ операцій, де k – кількість ненульових елементів в матриці. Тоді обчислювальна складність алгоритму Ланцоша для розріджених матриць становить $9mn + 2mk + 4n = O(m(n + k))$.

Так як $k \gg n$ та $k \gg m$, то найбільше часу витрачається на виконання операції множення розрідженої матриці на вектор в рядку 9 (рис. 1). Тому оптимізація цієї виконання операції є першочерговим завданням для отримання високопродуктивної реалізації. Питання роботи з розрідженими матрицями, включаючи вибір форматів зберігання, аналіз алгоритмів множення розрідженої матриці на вектор, розпаралелювання цих алгоритмів, необхідно розглянути окремо в наступних розділах.

Табл. 1. Характеристики форматів зберігання розріджених матриць

Формат	Спеціалізований	Вимоги до пам'яті	Обчислювальна складність множення матриці на вектор	Додаткові операції при розпаралелюванні
Координатний	Ні	$3k$	$O(k)$	$n(p-1)$
CSR	Ні	$2k+n+1$	$O(k)$	–
CSC	Ні	$2k+n+1$	$O(k)$	$n(p-1)$
CSR для симетричних матриць	Ні	$2r+2n+1$	$O(r+n)$	$n(p-1)$
BCSR	Так	$n_b b^2 + n_b + \frac{n}{b} + 1$	$O(n_b)$	–

Вибір формату зберігання розрідженої матриці

При виборі методу зберігання розріджених матриць необхідно звернути увагу на наступні характеристики кожного методу:

- вимоги до пам'яті;
- складність побудови структур даних під час завантаження матриці в пам'ять;
- обчислювальна складність виконання необхідних операцій (в контексті реалізації методу Ланцоша – множення матриці на вектор);
- наявність можливості розподілу даних по вузлам кластерної системи (тобто, відсутність необхідності зберігати повну копію матриці в кожному вузлі) та можливість паралельного виконання необхідних операцій;
- чи виникає необхідність виконання додаткових обчислень після паралельного виконання операції для отримання фінального результату, їх обчислювальна складність;
- ефективність використання кеш-пам'яті при виконанні операцій.

Було розглянуто наступні формати: координатний, [8, 9] CSR (Compressed Sparse Rows), [10] CSC (Compressed Sparse Columns), [9, 10] модифікація формату CSR для симетричних матриць, BCSR (Block Compressed Sparse Row). [11]

В таблиці 1 наведені основні характеристики форматів зберігання розріджених матриць, причому використано наступні позначення:

- n – розмірність квадратної матриці;
- k – кількість ненульових елементів матриці;
- r – кількість ненульових елементів симетричної матриці, що лежать нижче головної діагоналі ($r < \frac{k}{2}$);
- b – розмір щільного блоку;
- n_b – кількість блоків, що мають хоча б одним ненульовим елементом;
- p – кількість процесорів.

Задача зберігання та обробки великих розріджених матриць сама по собі не є тривіальною; до цієї задачі часто неможливо застосувати традиційні прийоми, що застосовуються при обробці щільних матриць. При виконанні операцій над розрідженими матрицями складно досягти високої продуктивності обчислень (на рівні з продуктивністю операцій над щільними матрицями) з наступних причин:

- структури даних є більш складними;
- алгоритми виконання операцій є складними, так як оброблюють складні структури даних;
- при обході елементів матриці є необхідність застосовувати непряму адресацію пам'яті;

- продуктивність обчислень сильно залежить від вхідних даних (можна побудувати такі структури даних, обробка яких буде відбуватись повільніше, ніж в «середньому» випадку).

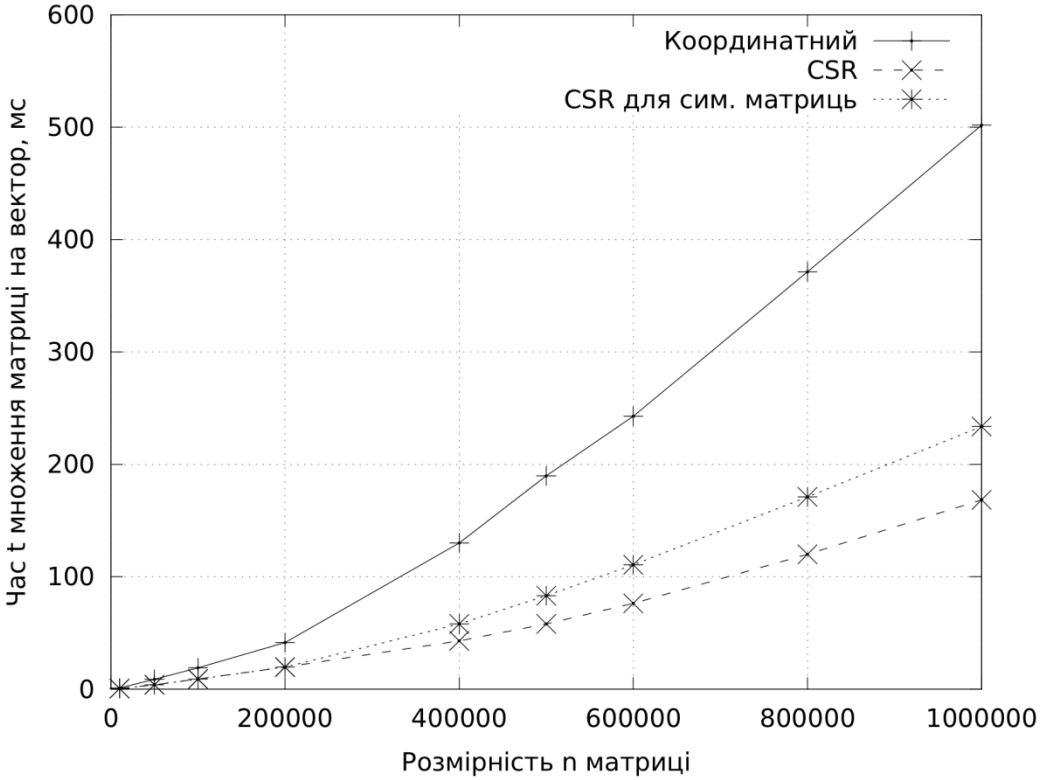

Рис. 2. Залежність часу множення розрідженої матриці на вектор від розмірності матриці при постійній середній кількості ненульових елементів у рядку (30)

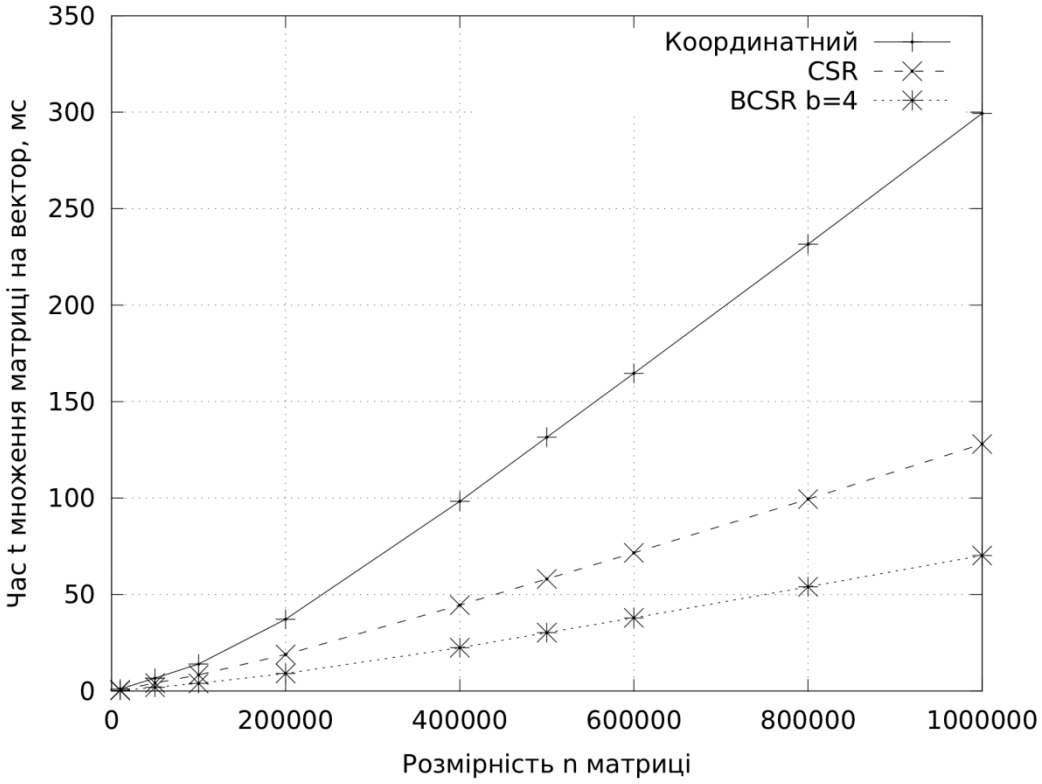

Рис. 3. Залежність часу множення розрідженої матриці з блочно-щільною структурою на вектор від розмірності матриці при постійній середній кількості ненульових елементів у рядку (44)

Експериментальне порівняння форматів загального призначення

Аналіз алгоритмів множення розрідженої матриці (в різних форматах) на вектор не дає кількісних оцінок для порівняння часу роботи. Тому доцільним є проведення експериментального порівняння реалізацій цих алгоритмів з метою визначити різницю в часі роботи та дати рекомендації для подальшої реалізації алгоритмів програмного забезпечення. Було виконано реалізацію послідовного варіанту вказаних методів зберігання розріджених матриць та відповідних алгоритмів множення матриці на вектор. Для вимірювання часу множення необхідні вхідні дані. Були згенеровані симетричні матриці розмірністю n =10, 50, 100, 200, 400, 500, 600, 800, 1000 тисяч без будь-якої характерної структури портрету з 30 ненульовими елементами (в середньому) в рядку.

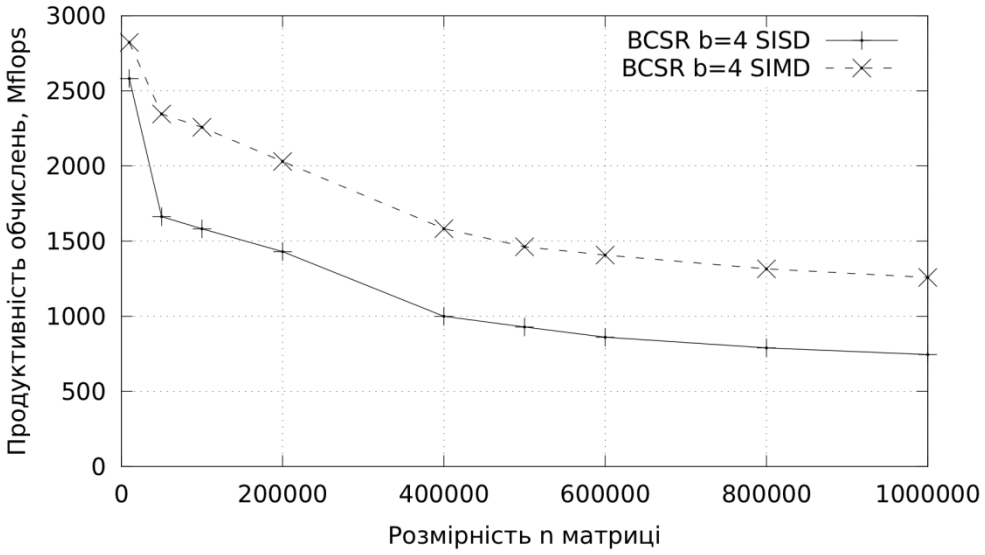

Рис. 4. Залежність продуктивності обчислень від розміру матриці для SISD та SIMD реалізацій операції множення матриці в форматі CSR на вектор

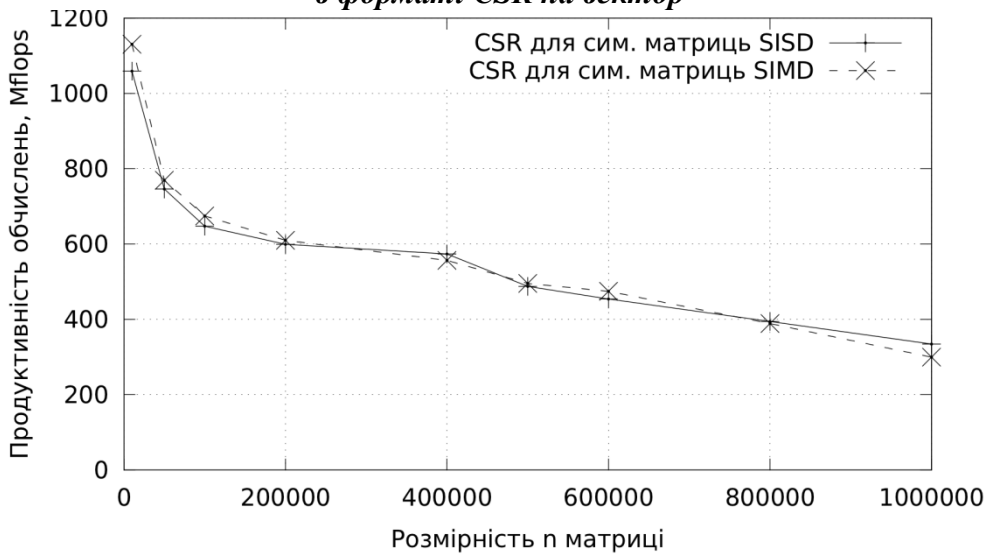

Рис. 5. Залежність продуктивності обчислень від розміру матриці для SISD та SIMD реалізацій операції множення матриці в форматі CSR для симетричних матриць на вектор

Реалізація кожного алгоритму множення була виконана 50 разів з вимірюванням часу, найгірший та найкращий час були відкинуті, а інші 48 результатів були усереднені. Графіки залежності часу виконання від розмірності матриці представлені на рис. 2.

Проаналізуємо ці результати. Так як кількість ненульових елементів $k = 30n$ в даному експерименті (тобто, k пропорційно n), то залежність часу виконання $t(k)$ пропорційна $t(n)$, що побудована на графіку (рис. 2). З графіку видно, що залежності часу виконання для координатного формату, CSR та його модифікації є досить близькими до лінійних, що є в згоді з теоретично обґрунтованою обчислювальною складністю $O(k)$. Найкращий час виконання в усіх випадках показала реалізація алгоритму для формату CSR.

Модифікація формату CSR для симетричних матриць дозволяє зменшити вимоги до пам'яті за рахунок збільшення часу обчислень.

Експериментальне порівняння форматів зберігання блочно-щільних матриць

Для зберігання блочно-щільних матриць можуть бути застосовані формати загального призначення (координатний, CSR) та спеціалізований формат BCSR. Генерування вхідних даних та вимірювання часу були проведені за методикою, аналогічною застосованій в попередньому пункті. Графіки залежності часу виконання від розмірності матриці представлені на рис. 3.

Ці графіки також є близькими до лінійної залежності, а тому є в згоді з теоретично обґрунтованою обчислювальною складністю $O(n_b)$ для алгоритму множення матриці в форматі BCSR. В порівнянні з реалізацією алгоритму для формату CSR, реалізація для формату BCSR показала в середньому в 1,37 разів менший час виконання, що є суттєвим покращенням.

Таким чином, застосування формату BCSR для блочно-щільних матриць є доцільним.

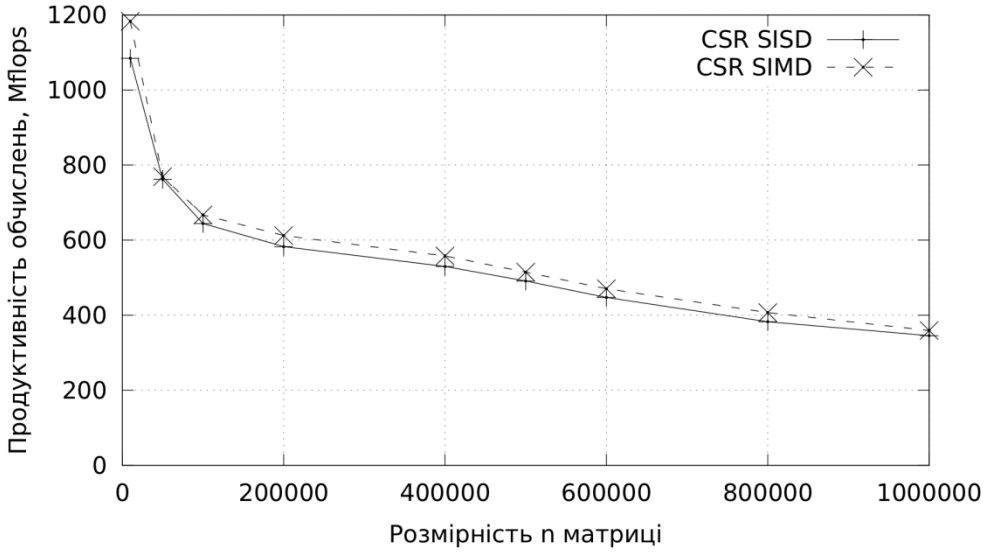

Рис. 6. Залежність продуктивності обчислень від розміру матриці для SISD та SIMD реалізацій операції множення матриці в форматі BCSR при на вектор

Оптимізація алгоритмів множення розрідженої матриці на вектор за рахунок застосування SIMD команд

В даній роботі найбільш цікавою є векторизація операцій множення матриці на вектор з декількох причин. По-перше, ця операція займає більшу частину часу обчислень за методом Ланцоша, і тому її необхідно оптимізувати в першу чергу. По-друге, алгоритми цих операцій є достатньо складними та не реалізуються SIMD командами тривіально.

Була виконана реалізація з використанням наборів команд SSE2, SSSE3, та SSE4 операцій множення матриці на вектор для форматів CSR, модифікації CSR для симетричних матриць, BCSR з фіксованими розмірами блоку $b = 2$, $b = 4$. З набору команд SSE2 було застосовано команди виконання арифметичних операцій, з набору команд SSE4 було використано команду обчислення скалярного добутку DPPD. Набори команд SSE та SSE4 включають команди для зчитування та запису в пам'ять даних в обхід кеш-пам'яті. Це дозволяє більш ефективно використовувати кеш-пам'ять, застосовуючи ці команди для завантаження тих даних, що під час обробки знадобляться тільки один раз.

Експериментальне порівняння SISD та SIMD реалізацій

На рис. 4–6 зображено експериментально отримані залежності продуктивності обчислень від розміру матриці для SISD та SIMD реалізацій операції множення для різних форматів зберігання матриць. Як видно з графіків, найбільший ефект застосування SIMD команд дало для формату BCSR, де підвищення продуктивності склало до 1,7 разів. Для формату CSR підвищення продуктивності становить до 7% Для модифікації формату CSR для симетричних матриць однозначного висновку зробити не можна, але для $n < 200000$ маємо підвищення продуктивності до 5%.

Методика тестування програмного забезпечення

В якості критерію оцінки та порівняння реалізацій алгоритму обрано коефіцієнт прискорення $K_П$ та коефіцієнт ефективності K_E.

Коефіцієнт прискорення показує, у скільки разів менше часу займає виконання паралельної програми в паралельній обчислювальній системі з P процесорами у порівнянні з виконанням послідовної програми в однопроцесорній системі:

$$K_П(P) = \frac{T_1}{T_P}.$$

Коефіцієнт ефективності показує середній рівень завантаження процесорів при виконанні програми в паралельній обчислювальній сис-

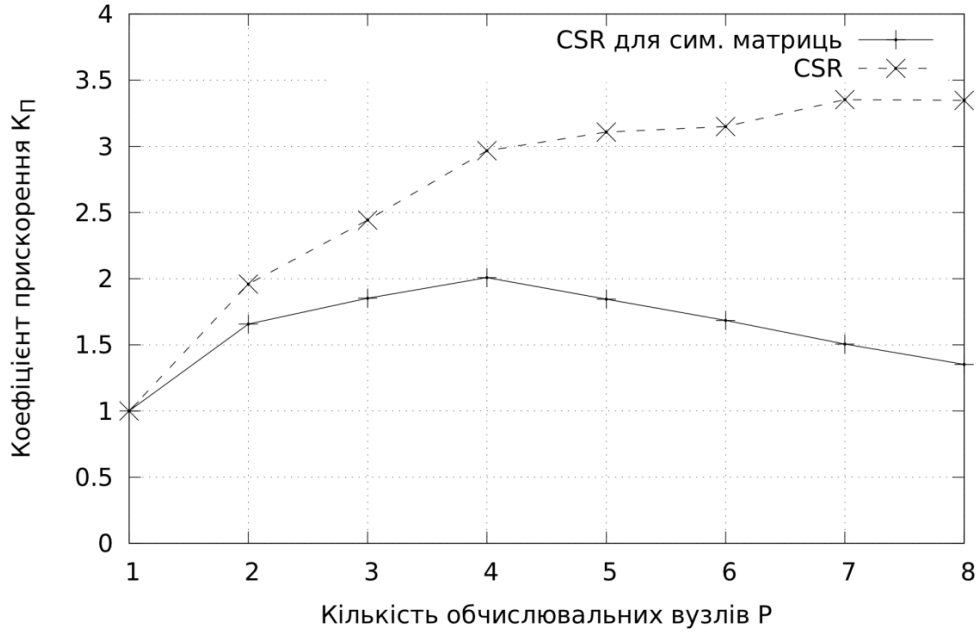

Рис. 7. Залежність коефіцієнту прискорення від кількості запущених задач, система «Б»

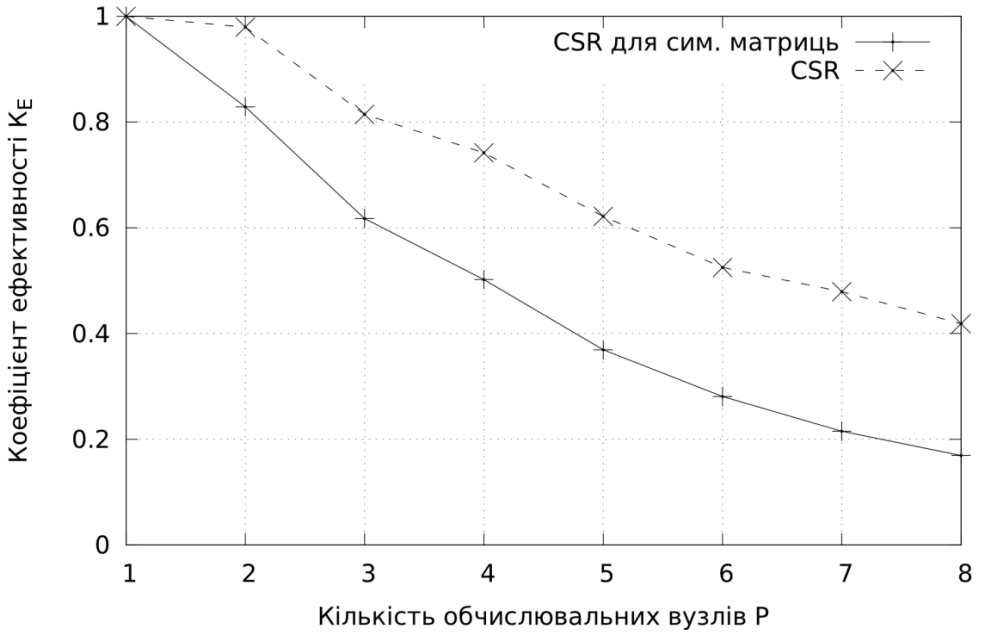

Рис. 8. Залежність коефіцієнту ефективності від кількості запущених задач, система «Б»

темі:

$$K_E(P) = \frac{K_П(P)}{P}$$

Для вимірювання часу виконання використовувався системний виклик clock_gettime(2).

Вимірювався час виконання 100 ітерацій алгоритму Ланцоша для матриці $n = 400000$, використовувався формат CSR.

Наведемо характеристики обчислювальних систем.

- процесор: Intel Core i5-750 (2,66 ГГц, 4 ядра, 8 Мб кеш-пам'яті третього рівня);
- оперативна пам'ять: DDR3 1600 МГц, 2 × 2048 Мб.

В якості програмного забезпечення використовувались:

- операційна система: Debian GNU/Linux «Sid» x86_64;
- компілятор C++: g++ 4.6.0.

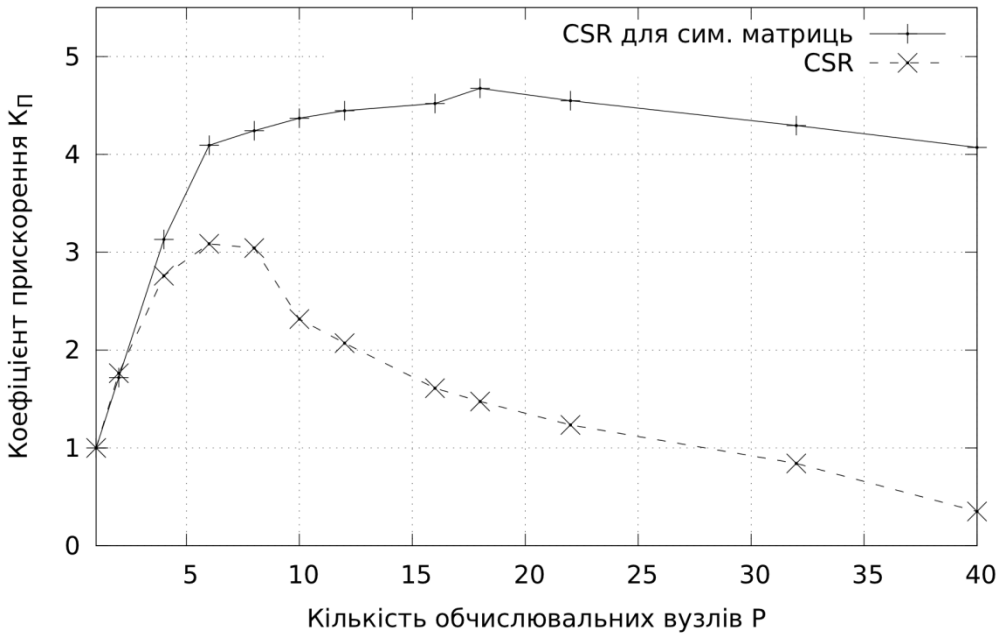

Рис. 9. Залежність коефіцієнту прискорення від кількості запущених задач, система «B»

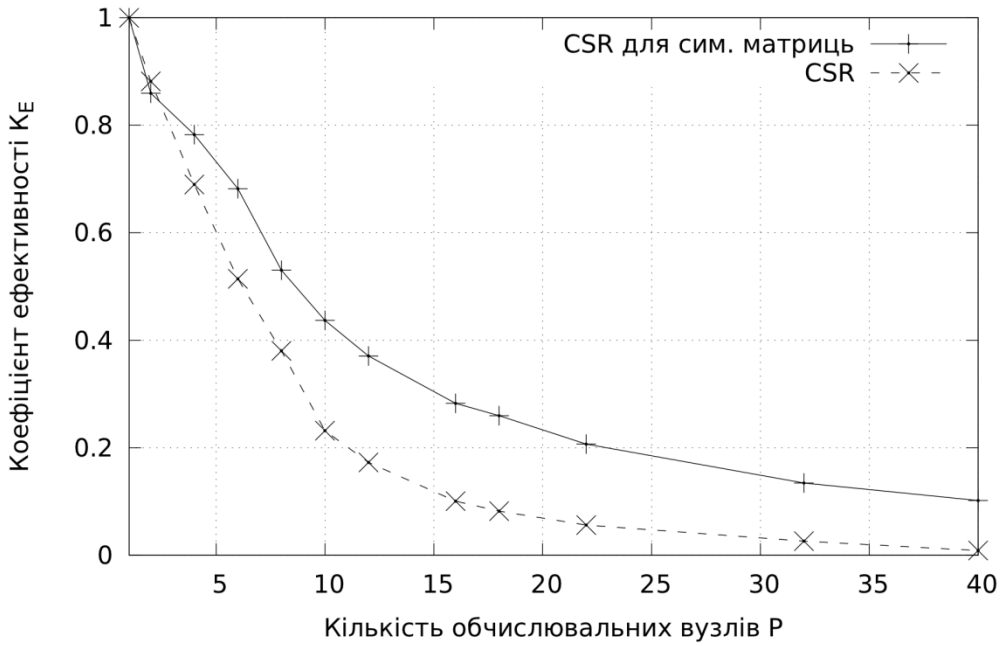

Рис. 10. Залежність коефіцієнту ефективності від кількості запущених задач, система «B»

Обчислювальна система «A» має наступне апаратне забезпечення:

Дана обчислювальна система використовувалась для тестування послідовних варіантів програм у розділах, представлених вище.

Обчислювальна система «Б» Для тестування паралельної реалізації методу Ланцоша використовувалась кластерна обчислювальна система Центру суперкомп'ютерних обчислень НТУУ «КПІ» з наступним апаратним забезпеченням вузла:

- процесори: два Intel Xeon E5345 (2,33 ГГц, 4 ядра, 8 Мб кеш-пам'яті другого рівня);
- оперативна пам'ять: DDR2 1333 МГц, 8192 Мб.

В якості програмного забезпечення використовувались:

- операційна система: CentOS release 5.5 x86_64;
- компілятор C++: g++ 4.6.0;
- реалізація MPI: OpenMPI 1.3.3.

Обчислювальна система «В» Обчислювальна система для пакетного запуску задач Intel Multicore Testing Lab має наступне апаратне забезпечення:

- процесори: чотири Intel Xeon E7-4860 (2,26 ГГц, 10 ядер, 24 Мб кеш-пам'яті другого рівня);
- оперативна пам'ять: DDR2 1333 МГц, 262144 Мб.

В якості програмного забезпечення використовувались:

- операційна система: RedHat Linux RHEL 5.4 x86_64;
- компілятор C++: g++ 4.5.1.

Результати тестування програмного забезпечення

Результати тестування на одному вузлі кластеру КПІ представлені на рис. 7–8. Максимальний коефіцієнт прискорення дорівнює 3,4 при запуску на 7 процесорах для формату CSR. Необхідно відмітити, що оптимальним можна вважати запуск на 4 процессорах, так як він показав $K_П \approx 3$ та високий коефіцієнт ефективності.

Результати тестування на Intel Manycore Testing Lab представлені на рис. 9–10. Максимальний коефіцієнт прискорення дорівнює 4,5 при запуску на 18 процесорах для формату CSR. Оптимальним можна вважати застосування 6 процесорів, коли спостерігається $K_П \approx 4$ та високий коефіцієнт ефективності.

За допомогою інструменту Intel VTune було встановлено, що вузьким місцем програми є швидкість каналу пам'ять-процесор. Це і є головним фактором, що обмежує можливість використання великої кількості процесорів даною програми. Це спостереження підтверджується наступним: в системі «Б» два процесори, кожен з яких має свій контролер доступу до пам'яті, а в системі «В» – чотири. Тому максимальні коефіцієнти прискорення при довільному доступі до пам'яті, що виникають у великій кількості при застосуванні формату CSR для симетричних матриць, дорівнюють приблизно двом та чотирьом відповідно. При застосуванні звичайного формату CSR довільних доступів до пам'яті значно менше, ніж послідовних, що дозволяє використати на свою користь попереднє завантаження даних в кеш, що відбувається автоматично при послідовному доступі до пам’яті.

Висновки

В даній роботі розглянуто не тільки реалізацію власне методу Ланцоша, але всі суміжні питання, що виникають під час створення реалізації для кластерних обчислювальних систем.

Експериментальне порівняння форматів зберігання з точки зору часу виконання операції множення матриці на вектор в послідовній системі показало:

- Правильний вибір формату зберігання розрідженої матриці дозволяє зменшити час обчислень до кількох разів.
- Для розріджених симетричних матриць без характерних особливостей портрету (матриці загального вигляду) з точки зору часу обчислень за алгоритмом множення матриці на вектор в послідовній системі оптимальним є формат CSR.
- Як наслідок попереднього: за замовчанням доцільно використовувати формат CSR.
- Застосування модифікації формату CSR для симетричних матриць дозволяє зменшити вимоги до пам'яті за рахунок збільшення часу обчислень.
- Для розріджених блочно-щільних матриць з точки зору часу обчислень за алгоритмом множення матриці на вектор в послідовній системі оптимальним є формат BCSR.

Для підвищення продуктивності обчислень було застосовано SIMD команди сучасних процесорів, що дало зменшення часу обчислень в окремих випадках до 1,7 разів.

Було встановлено, що головним фактором, що обмежує можливість використання систем з великою кількістю процесорів є швидкість каналу пам'ять–процесор.

Автор виражає вдячність Центру суперкомп'ютерних обчислень НТУУ «КПІ», [12] а також Intel Manycore Testing Lab [13] за наданий машинний час для розробки і тестування програмного забезпечення.

Перелік посилань

1. Sagan Hans. Boundary and Eigenvalue Problems in Mathematical Physics [Text]. – Dover Publications, 1989. – ISBN 0-486-66132-6.
2. Tore Opsahl Filip Agneessens John Skvoretz. Node centrality in weighted networks: Generalizing degree and shortest paths [Text]. – 2010. – Vol. 32. – Pp. 245–251.
3. We knew the web was big [Electronic resource]. – Access mode: http://googleblog. blogspot.com/2008/07/we-knew-web-was-big.html. – Last access: 15.04.2011. – Title from the screen.
4. Lanczos C. An iteration method for the solution of the eigenvalue problem of linear differential and integral operators [Text] // Journal of the Research of the National Bureau of Standards. – 1950. – Vol. 45. – Pp. 255--282.
5. Paige C. C. Computational variants of the lanczos method for the eigenproblem [Text] // Journal of the Institute of Mathematics. – 1972. – Vol. 10. – Pp. 373--381.
6. Saad Y. Iterative Methods for Sparse Linear Systems, 2nd edition [Text]. – Philadelpha, PA: SIAM, 2003.
7. Saad Y. Theoretical error bounds and general analysis of a few Lanczos-type algorithms [Text] // Proceedings of the Cornelius Lanczos International Centenary Conference / Ed. by J. D. Brown, M. T. Chu, D. C. Ellison, R. J. Plemmons. – Philadelphia, PA: SIAM, 1994. – Pp. 123--134.
8. The matrix market exchange formats: Initial design [Electronic resource]. – Access mode: http://math.nist.gov/MatrixMarket/reports/MMformat.ps.gz. – Last access: 15.04.2011. – Title from the screen.
9. Duff Iain S., Erisman A. M., Reid J. K. Direct Methods for Sparse Matrices [Text]. – 1986. – Pp. xiii + 341. – ISBN 0-19-853408-6 (hardcover).
10. Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods, 2nd Edition [Text] / R. Barrett, M. Berry, T. F. Chan et al. – Philadelphia, PA: SIAM, 1994.
11. Karakasis Vasileios, Goumas Georgios, Koziris Nectarios. Perfomance models for blocked sparse matrix-vector multiplication kernels [Text] // Proceedings of the 2009 International Conference on Parallel Processing. – ICPP '09. – Washington, DC, USA: IEEE Computer Society, 2009. – Pp. 356--364. http://dx.doi.org/10.1109/ICPP.2009.21.
12. Центр суперкомп'ютерних обчислень НТУУ «КПІ» [Електронний ресурс]. – Режим доступу: http://hpcc.org.ua/. – Останнє звернення: 15.04.2011. – Назва з екрану.
13. Intel Manycore Testing Lab [Electronic resource]. – Access mode: http://software.intel.com/ en-us/articles/intel-many-core-testing-lab/. – Last access: 15.04.2011. – Title from the screen.