7. Иванов В. В. Методы вычислений на ЭВМ (справочное пособие). Киев: Наукова думка, 1986. 584 с.
8. Задірака В. К., Олексюк О. С. Комп'ютерна арифметика багаторозрядних чисел: наук. вид. К., 2003. 264 с.
9. Сергиенко И. В., Задирака В. К., Бабич М. Д., Березовский А. И., Бесараб П. Н., Людвиченко В. А. Компьютерные технологии решения задач прикладной и вычислительной математики с заданными значениями характеристик качества. *Кибернетика и системный анализ*. 2006. № 5. С. 33–41.
10. Бабич М. Д., Задирака В. К., Сергиенко И. В. Вычислительный эксперимент в проблеме оптимизации вычислений. *Кибернетика и системный анализ*. 1999. Ч.1. № 1. С. 51–63; Ч.2. № 2. С. 59–79.

The main stages of topics development on issues of calculations optimization are considered.

Key words: *error theory, optimal algorithms, information operations, information given a priory, calculations optimization.*

Одержано 28.02.2017

УДК 519.6

В. А. Сидорук, канд. фіз.-мат. наук

Інститут кібернетики імені В. М. Глушкова НАН України, м. Київ

ОДНОВУЗЛОВИЙ ГІБРИДНИЙ АЛГОРИТМ ФАКТОРИЗАЦІЇ РОЗРІДЖЕНИХ МАТРИЦЬ

Розглядається гібридний алгоритм розв'язування систем лінійних алгебраїчних рівнянь з розрідженими симетричними додатно визначеними матрицями на комп'ютерах з графічними прискорювачами. Подано результати апробації алгоритму на багатоядерному комп'ютері з графічними прискорювачами Інпарком.

Ключові слова: *плитковий алгоритм, гібридна архітектура, CUDA, Openmp.*

Вступ. При чисельному розв'язанні задач у багатьох випадках доводиться розв'язувати задачу (або декілька підзадач) лінійної алгебри — систему лінійних алгебраїчних рівнянь (СЛАР). Наприклад, задачі лінійної алгебри виникають при дискретизації крайових задач проекційно-різницевим методом (скінченних різниць, скінченних елементів).

Важливою особливістю задач лінійної алгебри, які виникають при дискретизації є невелика кількість ненульових елементів матриці, тобто матриці є розрідженими [1]. Кількість ненульових елементів у таких матрицях складає kn, де $k << n$, а n — порядок матриці. Структура розрідженої матриці визначається нумерацією невідомих

задачі і часто буває стрічковою, профільною, блочно-діагональною з обрамленням, або просто довільної розрідженої структури. Ще однією важливою особливістю СЛАР з розрідженими матрицями є їх великий порядок — до десятків мільйонів.

У роботі розглядаються додатно-визначені розріджені матриці нерегулярної структури.

Постановка задачі. Розглянемо задачу

$$Ax = b \tag{1}$$

з симетричною додатно-визначеною розрідженою матрицею порядка n.

Однією з передумов розв'язання задачі (1) на комп'ютерах гібридної архітектури з багатоядерними процесорами (CPU) і графічними прискорювачами (GPU) є приведення матриці до такого виду, який дозволяє працювати з більш щільними групами ненульових елементів. Перетворення матриці зумовлене особливостями роботи GPU, які дозволяють отримати найкращі показники продуктивності при обробці великих щільних масивів даних.

Нині існує велика кількість алгоритмів перевпорядкування елементів матриць: метод мінімальної степені, метод вкладених перерізів, метод паралельних перерізів і т. д. Кожен з цих алгоритмів дозволяє привести довільну розріджену структуру до більш регулярного вигляду. Найбільш зручну для паралельної обробки структуру матриці отримується після застосування до матриці методу вкладених, або паралельних, перерізів.

$$\tilde{A} = P^T AP = \begin{pmatrix} A_{11} & 0 & 0 & A_{1p} \\ 0 & A_{22} & 0 & A_{2p} \\ 0 & 0 & \ddots & \vdots \\ A_{p1} & A_{p2} & \dots & A_{pp} \end{pmatrix},$$

де P — матриця перестановок, p — кількість діагональних блоків у матриці, блоки A_{pp}, A_{ip}, A_{pi}, A_{ii} $i = \overline{1, p-1}$ зберігають розріджену структуру.

Таким чином, задача розв'язання (1) зводиться до розв'язування еквівалентної системи

$$\tilde{A}\tilde{x} = \tilde{b}, \tag{2}$$

де $\tilde{x} = P^T x$, $\tilde{b} = P^T b$.

Найбільш ефективним прямим методом розв'язання (2) є метод Холецького [1–3]. В статті буде висвітлено гібридний алгоритм який відповідає саме етапу факторизації матриці.

Гібридний алгоритм. Розіб'ємо матрицю A на блоки розмірністю $s \times s$. Далі для факторизації блочно-діагональної матриці застосуємо алгоритм запропонований в [4] для щільних матриць.

Для факторизації матриці на k-му кроці використаємо наступне співвідношення:

$$A^k = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} = \begin{pmatrix} L_{11} & 0 \\ L_{21} & L_{22} \end{pmatrix} \begin{pmatrix} L_{11}^T & L_{21}^T \\ 0 & L_{22}^T \end{pmatrix}, \quad (3)$$

де розмірності блоків A_{11} — $s \times s$, A_{12} — $(n-ks)s$, A_{22} – $(n-ks)(n-ks)$, блоки A_{12} та A_{22} враховують структуру діагональних блоків та блоків обрамлення.

Звідси отримаємо алгоритм, за яким проводиться розвинення на k кроці:

$$A_{11} = L_{11} * L_{11}^T; \quad (4)$$

$$L_{21} = A_{21} * \left(L_{11}^T\right)^{-1}; \quad (5)$$

$$\tilde{A}_{22} = A_{22} - L_{21} * L_{21}^T. \quad (6)$$

Зазначимо, що реалізація (4)–(6) на кожному кроці модифікує тільки блоки D_{ii}, C_{pi}, $i = \overline{1, p-1}$, D_{pp}.

Нехай для розв'язування задачі на комп'ютері гібридної архітектури маємо CPU з p процесорними ядрами і 1 GPU. Для роботи алгоритму на k кроці реалізовується наступна декомпозиція даних: у пам'яті CPU і GPU зберігається плитка A_{11}; у пам'яті GPU зберігаються плитки необхідні для модифікації підматриці A_{22}. На рис. 1. показано блочний розподіл даних на k-му кроці факторизації блочно-діагональної матриці з обрамленням, враховуючи вище запропоновану декомпозицію.

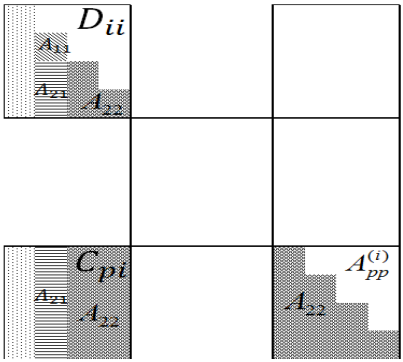

Рис. 1. *Декомпозиція даних в GPU на k-му кроці факторизації*

Враховуючи декомпозицію даних, приведену вище, плитковий гібридний алгоритм факторизації записується у наступній формі:

1) над всіма діагональними блоками крім D_{pp} та відповідними блоками обрамлення послідовно виконуються такі операції:
   - на CPU факторизуємо A_{11} $A_{11} = L_{11} * L_{11}^T$;
   - на GPU паралельно в різних потоках модифікуємо стовпчик блоків L_{21}:
   $$L_{21} = A_{21}\left(L_{11}^T\right)^{-1};$$
   - у GPU незалежно в кількох потоках модифікуємо блоки матриці A22 за формулою:
   $$\tilde{A}_{22} = A_{22} - L_{21}L_{21}^T;$$
2) факторизуємо блок $A_{pp}^{(i)}$, тим самим завершуючи процес факторизації матриці A.

Оцінка прискорення. Для оцінки якості гібридних алгоритмів будемо використовувати коефіцієнт прискорення S_p що обчислюється за формулою

$$S_p = T_1 / T_p,$$

де T_1 — час розв'язування задачі на гібридному комп'ютері з одним CPU та одним GPU, T_p — час розв'язування тієї ж задачі на гібридному комп'ютері з використанням один CPU і один GPU в паралельному режимі.

Введемо наступні позначення: N_1 — кількість операцій, виконуваних алгоритмом на гібридному комп'ютері з використанням 1 CPU і 1 GPU; N_p – кількість операцій, виконуваних алгоритмом на гібридному комп'ютері з використанням один CPU і один GPU з використанням кількох паралельних ниток; t_g — середній час виконання однієї арифметичної операції на GPU; t_{opg} – час обміну між одним CPU та GPU.

Обчислимо кількість операцій виконуваних алгоритмом факторизації. Введемо наступні позначення $m = \frac{n}{p}$ — порядок діагонального блоку матриці A та $l = \frac{m}{s}$ — кількість плиток у стовпці діагонально блоку. При підрахунку кількості операцій будемо вважати, що матриця A має діагональні блоки однакових порядків.

Оскільки в алгоритмі на парі CPU та GPU $p-1$ раз етапи (4)-(6) і один раз факторизується останній діагональний блок, то кількість арифметичних операцій можна обчислити за наступною формулою:

$$N_1 \approx (p-1)\alpha + \beta .$$

Для обчислення кількості операції необхідних для факторизації останнього діагонального блоку будемо вважати, що останній діагональний блок — щільна матриця. Тоді $\beta = \frac{m^3}{3}$.

Розглянемо етапи (4)–(6). Тут найбільше арифметичних операцій припадає на виконання етапу (6). Обчислимо їх кількість.

На k-му кроці факторизації, як видно з рис. 1., нам потрібно модифікувати $\left\lceil \frac{(m-ks)^2}{2} \right\rceil$ елементів з D_{ii}, $(m-ks)m$ — з C_{pi} та $\left\lceil \frac{m^2}{2} \right\rceil$ — з $A_{pp}^{(i)}$.

Тобто $\alpha \approx 2s\left(\sum_1^l \frac{(m-ks)^2}{2} + \sum_1^l (m-ks)m + \sum_1^l \frac{m^2}{2} \right)$.

Розкриємо дужки та розпишемо суми

$$\alpha \approx \left(s\sum_1^l \left(m^2 - 2mks + 2k^2 s^2 \right) + 2s\sum_1^l (m-ks)m + s\sum_1^l m^2 \right).$$

Після проведення спрощень і підстановки значень отримаємо наступну формулу для обчислення: $\alpha \approx \frac{8m^3}{3}$.

Звідси $N_1 \approx (p-1)\frac{8m^3}{3} + \frac{m^3}{3}$.

Кількість операцій виконуваних при реалізації паралельного варіанта гібридного алгоритму обчислюється за формулою

$$N_p \approx N_1 / th ,$$

де th — кількість незалежних потоків.

Прискорення гібридного алгоритму LL^T — розвинення розрідженої блочно-діагональної матриці з обрамленням A, становить

$$S_p \approx \frac{th N_1 t_g}{N_1 t_g + th\mathrm{T}} ,$$

де $\mathrm{T} = (p-1)\left(4ms + m^2\right)t_g$.

Результати чисельних експериментів. Розрахунки проводились на вузлах кластера Інпарком-G [5], які мають наступні характеристики:

- процесори: 2 Xeon 5606 (8 ядер) з частотою 2.13 ГГц;

- графічні прискорювачі: 2 Tesla M2090;
- обсяг оперативної пам'яті: 24 Гб;
- комунікаційне середовище: InfiniBand 40 Гбіт/с (з підтримкою GPUDirect), Gigabit Ethernet.

Чисельні експерименти проводились на розріджених матрицях, що наведені в таблиці. Також в таблиці наведені такі характеристики матриці як порядок матриці, кількість ненульових елементів.

Для програмної реалізації етапів (5), (6) використовувались функції бібліотеки CUBLAS та функції бібліотеки Openmp. Для плиток з порядками 64-1024 використовувались виклики в циклі функції cublasDtrsm та cublasDgemm, виконання кожної з функцій проходило в окремому потоці cudaStream. Нитки Openmp відповідають потокам cudaStream.

Копіювання даних і обчислення проводились в асинхронному режимі.

Таблиця

Набір тестових матриць з Флоридської колекції розріджених матриць

Назва	**Проблемна область**	**Порядок**	**Кількість ненульових елементів**
G3_circuit	circuit simulation problem	1 585 478	7 660 826
G2_circuit	circuit simulation problem	150 102	726 624
parabolic_fem	computational fluid dynamics problem	525 825	3 674 625
apache2	structural problem	715 176	4 817 870

На рис. 2. показано графік залежності продуктивності від використовуваної кількості ниток Openmp на прикладі виконання факторизації матриці G2_circuit на гібридній архітектурі 1CPU та 1 GPU з розміром плитки 128.

Висновки. Алгоритм добре враховує профільну, або розріджену структуру діагональних блоків і матриці в цілому. Можна регулювати розмірність блоку з яким проводяться обчислення на кожному кроці алгоритму, за рахунок цього може досягатись ефект кешизації обчислень, коли блоки поміщаються в швидкій пам'яті GPU. Також така блочна структура дозволяє працювати з нерозривними масивами даних на GPU, що зменшує кількість індексних операцій і перевірок які на графічному прискорювачі є досить затратними. Також регулювання кількості використовуваних потоків дозволяє підвищити ефективність алгоритму.

Рис. 2. *Продуктивність алгоритму*

Список використаних джерел:

1. Джордж А., Лю Дж. Численное решение больших разреженных систем уравнений. М.: Мир, 1984. 334 с.
2. Химич А. Н., Попов А. В., Полянко В. В. Алгоритмы параллельных вычислений для задач линейной алгебры с матрицами нерегулярной структуры. *Кибернетика и системный анализ*. 2011. 47, № 6. С. 159–174.
3. Хімич О. М., Сидорук В. А. Гібридний алгоритм розв'язування лінійних систем з розрідженими матрицями на основі блочного LL^T методу. *Комп'ютерна математика*. 2015. Вип. 1. С. 67–74.
4. Buttari Alfredo, Langou Julien, Kurzak Jakub, and Dongarra Jack. A Class of Parallel Tiled Linear Algebra Algorithms for Multicore Architectures. *Parallel Computing*. 2009. Vol. 35, Is. 1. P. 38–53.
5. Химич А. Н., Молчанов И. Н., Мова В. И. и др. Численное программное обеспечение MIMD-компьютера Инпарком. Киев: Наук. думка, 2007. 222 с.

A hybrid algorithm for solving systems of linear algebraic equations with sparse symmetric positive definite matrix on computers with GPU is considered. The results of testing of the algorithm on multicore computer Inparcom are presented.

Key words: *tiled algorithm, hybrid architecture, CUDA, Openmp.*

Одержано 02.03.2017