Теорія оптимальних рішень

УДК 519.6

А.Ю. БАРАНОВ

ГІБРИДНИЙ АЛГОРИТМ ФАКТОРИЗАЦІЇ СТРІЧКОВИХ НЕСИМЕТРИЧНИХ МАТРИЦЬ

Запропоновано алгоритм факторизації стрічкових несиметричних матриць на комп'ютерах гібридної архітектури. Проведено апробацію алгоритму на суперкомп'ютерах СКІТ-4 та Інпарком.

Вступ. При чисельному розв'язанні задач у багатьох випадках виникає необхідність розв'язувати систему лінійних алгебраїчних рівнянь (СЛАР). Наприклад, задачі лінійної алгебри виникають при дискретизації крайових задач чи задач на власні значення проекційно-різницевим методом (скінченних елементів, скінченних різниць). Також, при використанні методів математичного програмування, ітераційних методів розв'язання нелінійних задач часто на кожній ітерації розв'язується СЛАР.

Важливою особливістю задач лінійної алгебри, які виникають при дискретизації, являється те, що кількість ненульових елементів матриць таких задач складає kn, де $k << n$, а n – порядок матриці, тобто матриці є розрідженими [1]. Структура розрідженої матриці визначається нумерацією невідомих задач і часто є стрічковою, блочно-діагональною з обрамленням, профільною і тому подібне. В даній статті розглядаються несиметричні матриці стрічкової структури.

Іншою важливою особливістю є великий порядок матриці СЛАР – до десятків мільйонів. Це зумовлюється бажанням використовувати більш точні дискретні моделі, що дає можливість отримувати наближені розв'язки більш близькі до розв'язків вихідних задач, враховувати локальні особливості даного процесу чи явища.

Актуальним на сучасному етапі є створення ефективних паралельних алгоритмів для комп'ютерів гібридної (MIMD, SIMD) архітектури.

Постановка задачі. Розглянемо систему лінійних алгебраїчних рівнянь:

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

де матриця A – стрічкова несиметрична, n – порядок матриці A, $k1-$ кількість нижніх діагоналей, $k2$ – кількість верхніх діагоналей.

Найбільш ефективним прямим методом розв'язання такої задачі є, як відомо, метод Гаусса [2]. Розв'язання системи (1) полягає у розв'язанні підзадач: трикутне розвинення матриці системи, розв'язання двох СЛАР з трикутними матрицями:

$$A = PLU, \tag{2}$$

$$Ly = b, \tag{3}$$

$$Ux = y. \tag{4}$$

Послідовний алгоритм факторизації. У роботі [3] показно, що кількість верхніх діагоналей матриці U дорівнює $k1+k2$. Тому, для опису алгоритму будемо розглядати матрицю A з $k1+k2$ верхніми діагоналями. Нехай порядок стрічкової несиметричної матриці A дорівнює $n = pb,\quad k1 = lb, k1+k2 = ub,$ де b це деяке наперед задане натуральне число. Розіб'ємо матрицю A на квадратні блоки:

$$\begin{pmatrix}
A_{11} & A_{12} & A_{13} & \cdots & A_{1(1+u)} & 0 & 0 & \cdots & 0 \\
A_{21} & A_{22} & A_{23} & \cdots & A_{2(1+u)} & A_{2(2+u)} & 0 & \cdots & 0 \\
A_{31} & A_{32} & \ddots & & & & & & \\
\vdots & \vdots & & & & & & & \\
A_{(l+1)1} & A_{(l+1)2} & & & & & & & \\
0 & A_{(l+2)2} & & & & & & & \\
0 & 0 & & & & & \vdots & & \vdots \\
\vdots & \vdots & & & & & \cdots & A_{(p-1)(p-1)} & A_{(p-1)p} \\
0 & 0 & & & & & \cdots & A_{p(p-1)} & A_{pp}
\end{pmatrix}$$

Переходимо до опису алгоритму. Для кожного $i = \overline{1, p}$ треба виконати такі кроки:

1) виконати факторизацію прямокутної матриці, елементами якої є блоки $A_{ii}, A_{(i+1)i}, A_{(i+2)i}, \ldots, A_{q,i},$ де $q = \min(i+l, p),$ використовуючи алгоритм Гаусса для щільних матриць:

$$\begin{pmatrix} A_{ii} \\ \vdots \\ A_{qi} \end{pmatrix} = P_i \begin{pmatrix} L_{ii} \\ \vdots \\ L_{qi} \end{pmatrix} U_{ii}. \tag{5}$$

2) перестановка рядків у тих підматрицях, де це необхідно, використовуючи матрицю P_i;

3) знаходження матриць $U_{i(i+1)}, U_{i(i+2)}, \ldots, U_{ir}$, де $r = \min(i+u, p)$ за формулою:

$$\begin{pmatrix} U_{i(i+1)} & \cdots & U_{ir} \end{pmatrix} = L_{ii}^{-1} \begin{pmatrix} A_{i(i+1)} & \cdots & A_{ir} \end{pmatrix}; \tag{6}$$

4) оновлення матриць A_{jt}, де $j = \overline{i+1, q}$ та $t = \overline{i+1, r}$ за формулою:

$$A_{jt} \leftarrow A_{jt} - L_{ji} U_{it}. \tag{7}$$

Після виконання всіх кроків алгоритму отримуємо трикутне розвинення вихідної матриці.

Гібридний алгоритм факторизації. Розглянемо алгоритм факторизації несиметричної стрічкової матриці для гібридних комп'ютерів з графічними прискорювачами. Такі алгоритми потребують копіювання даних з пам'яті CPU до пам'яті GPU та навпаки. Сучасні GPU мають можливість одночасно виконувати копіювання даних та операції обчислення. Також, обсяг глобальної пам'яті GPU є меншим за пам'ять CPU. Тому, доцільним є створення гібридних алгоритмів, які будуть оптимально використовувати пам'ять GPU, використовуватимуть можливість одночасно виконувати обчислення та копіювання даних.

Нехай кількість доступних графічних прискорювачів дорівнює g. На i-му кроці послідовного алгоритму факторизації нам потрібні тільки блоки, які знаходяться в i-му стовпчику, та в u стовпчиках, що слідують за ним. При цьому, в кожному стовпчику знаходиться найбільше $u+l+1$ блоків. Таким чином, на кожному кроці алгоритму, достатньо зберігати в GPU тільки ці блоки. Переходимо до опису гібридного алгоритму факторизації. Блоки матриці A, що знаходяться на GPU, будемо позначати використовуючи верхній індекс d. Будемо виконувати операцію з п. 1 послідовного алгоритму на CPU, всі інші операції доцільно виконувати на GPU.

Перед початком роботи алгоритму, виконується копіювання блоків матриці A, що знаходяться в перших $u+1$ стовпчиках, у пам'ять GPU. При цьому, блоки які знаходяться в стовпчику з номером t копіюються в GPU з номером $(t-1) \bmod g$.

Для кожного $i = \overline{1, p}$ треба виконати наступні кроки:

1) копіювання діагонального блоку A_{ii}^d та блоків що знаходяться нижче його в стовпчику i: $A_{(i+1)i}^d, A_{(i+2)i}^d, \ldots, A_{q,i}^d$ у пам'ять CPU;

2) факторизації прямокутної матриці, елементами якої є блоки $A_{ii}, A_{(i+1)i}, A_{(i+2)i}, \ldots, A_{q,i}$ за формулою (5);

3) копіювання отриманих на попередньому кроці блоків $L_{ii}, \ldots, L_{qi}$, U_{ii}, P_i у пам'ять всіх GPU;

4) у кожному GPU виконується перестановка рядків у тих блоках, де це необхідно, використовуючи матриці P_i^d;

5) обчислення блоків $U^d_{i(i+1)}, U^d_{i(i+2)}, \ldots, U^d_{ir}$, де $r = \min(i+u, p)$ за формулою:

$$\left(U^d_{i(i+1)} \quad \cdots \quad U^d_{ir}\right) = \left(L^d_{ii}\right)^{-1} \left(A^d_{i(i+1)} \quad \cdots \quad A^d_{ir}\right). \tag{8}$$

Оскільки кожний GPU має свою копію блоку L^d_{ii}, а всі інші члени формули (8) знаходяться в одному стовпчику, то обчислення на цьому кроці можуть здійснюватися одночасно всіма GPU;

6) модифікація блоків A^d_{jt}, де $j = \overline{i+1, q}$ та $t = \overline{i+1, r}$ за формулою:

$$A^d_{jt} \leftarrow A^d_{jt} - L^d_{jt}\, U^d_{it}. \tag{9}$$

Кожний GPU має свою копію блоку L^d_{ji}, а блоки A^d_{jt} та U^d_{it} знаходяться в одній колонці, а отже і в пам'яті одного GPU. Тому обчислення на цьому кроці можуть здійснюватися одночасно всіма GPU;

7) якщо $i+u+2 < p$, то копіюємо блоки матриці в стовпчику $i+u+2$ у пам'ять GPU з номером $(i+u+1) \bmod g$.

Реалізація гібридного алгоритму. При реалізації гібридного алгоритму, доцільно використовувати функції з бібліотеки cuBLAS [4]. Для обрахунків за формулою (8) доцільно використовувати функцію cublasDtrsm, а для формули (9) – cublasDgemm. Для факторизації щільної прямокутної матриці на CPU доцільно використовувати високопродуктивну функцію з бібліотеки Intel MKL [5].

На i-му кроці, оновлення блоків із стовпчика $i+1$, за формулою (9), слід виконувати в спеціально виділеному потоці cudaStream_*t*. Оскільки CUDA має можливість асинхронного запуску функцій на GPU, то використання спеціального потоку для оновлення блоків у стовпчику $i+1$, дає можливість почати факторизацію за формулою (5) на $i+1$-му кроці одночасно з виконанням обчислень за формулою (9) на i-му кроці. Таким чином, одночасно виконуються обчислення на CPU та GPU.

Також слід зазначити, що даний алгортим може бути реалізований як на одновузлових комп'ютерах з декількома GPU (використовуючи POSIX Threads, OpenMP тощо), так і на багатовузлових комп'ютерах, використовуючи MPI.

Експериментальне дослідження ефективності гібридного алгоритму. Запропонований паралельний алгоритм реалізовано на комп'ютерах з гібридною (MIMD, SIMD) архітектурою: комп'ютері з графічними прискорювачами Інпарком (Tesla M2090, 2 CPU Intel(R) Xeon(R) E5606 @ 2.13GHz) [6], та СКІТ-4 (Tesla M2050, 4 CPU Intel(R) Xeon(R) X5675 @ 3.07GHz) [7]. В чисельних експериментах використовувалися матриці порядку 100000.

На рис. 1 показано залежність часу факторизації матриці від напівширини плитки для різних значень ширини блоку b, експерименти проведені на комп'ютері СКІТ-4. На рис. 1, а використовується 2 GPU, а на рис. 1, б використовується 3 GPU.

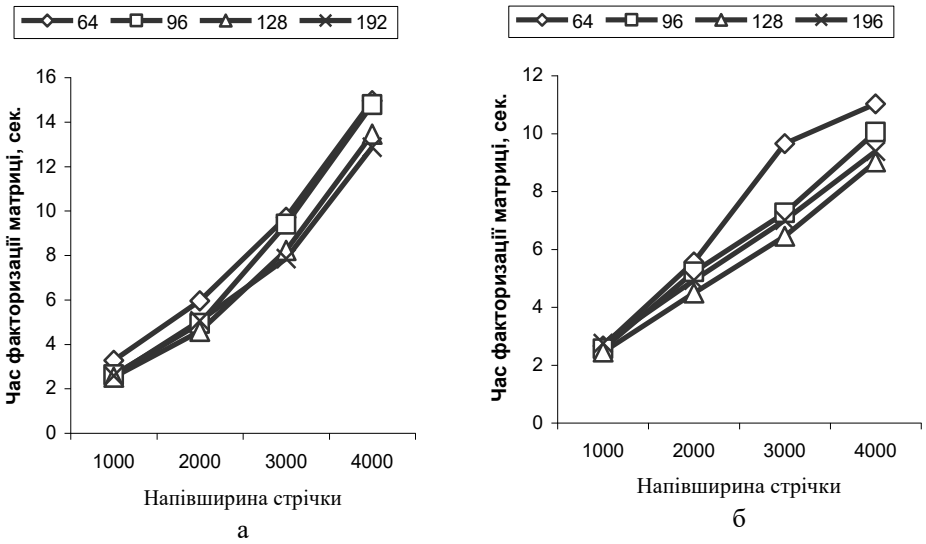

РИС. 1. Залежність часу факторизації матриці від напівширини стрічки для різних розмірів блоку на комп'ютері СКІТ-4

На рис. 2 показано залежність часу факторизації матриці від напівширини плитки для різних значень ширини блоку b, експерименти проведені на комп'ютері Інпарком. На рис. 2, а використовується 1 GPU, а на рис. 2, б використовується 2 GPU.

На рис. 3, а показано прискорення алгоритму факторизації матриці в залежності від напівширини стрічки для двох GPU; експерименти проводились на комп'ютері СКІТ-4. На рис. 3, б показано прискорення алгоритму факторизації матриці для двох та трьох GPU; експерименти проводились на комп'ютері Інпарком. У цих експериментах вибирався оптимальний порядок блоку b в залежності від напівширини стрічки матриці.

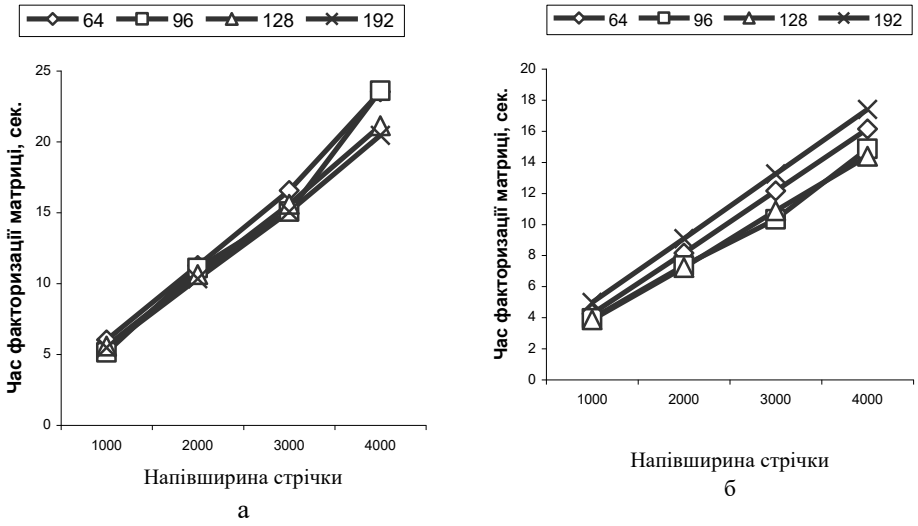

РИС. 2. Залежність часу факторизації матриці від напівширини стрічки для різних розмірів блоку на комп'ютері Інпарком

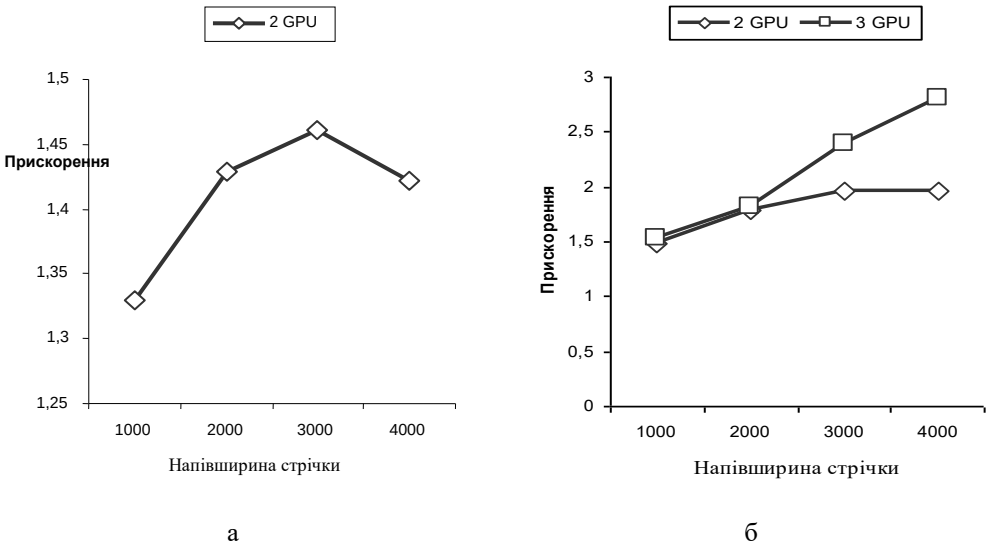

РИС. 3. Прискорення алгоритму факторизації матриці в залежності від напівширини стрічки на комп'ютерах: а – Інпарком та б – СКІТ-4

Висновки. Запропоновано алгоритм факторизації стрічкових несиметричних матриць, який забезпечує високу ефективність розпаралелювання на GPU, враховує структуру стрічкової матриці, оптимізує використання пам'яті GPU. Експериментально проведено дослідження вибору оптимального розміру ширини плиток, на які розбивається вихідна матриця.

Подальші дослідження доцільно направити на розробку алгоритмів для графічних прискорювачів архітектури Kepler [8], зокрема використання можливості динамічного паралелізму.

А.Ю. Баранов

ГИБРИДНЫЙ АЛГОРИТМ ФАКТОРИЗАЦИИ ЛЕНТОЧНЫХ НЕССИМЕТРИЧЕСКИХ МАТРИЦ

Предложен алгоритм факторизации ленточных несимметричных матриц на компьютерах гибридной архитектуры. Алгоритм апробирован на суперкомпьютерах СКИТ-4 и Инпарком.

A.Y. Baranov

HYBRID ALGORITHM FOR FACTORIZATION OF BAND NONSYMMETRIC MATRICES

An algorithm for factorization of band nonsymmetric matrices using hybrid computers is considered. The results of testing of the algorithm on supercomputers Inparcom and SCIT-4 are presented.

1. *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.
2. *Уилкинсон Дж.Х., Райнш К.* Справочник алгоритмов на языке Алгол // Линейная алгебра. – М.: Машиностроение, 1976. – 389 с.
3. *Gene H., Golub and Charles F. Van Loan.* 1996. Matrix Computations (3rd Ed.). Johns Hopkins University Press, Baltimore, MD, USA.
4. https://developer.nvidia.com/cuBLAS
5. http://software.intel.com/en-us/intel-mkl.
6. *Химич А.Н., Молчанов И.Н., Мова В.И. и др*. Численное программное обеспечение MIMD-компьютера Инпарком. – Киев: Наук. думка, 2007. – 222 с.
7. http://icybcluster.org.ua
8. http://www.nvidia.com/object/nvidia-kepler.html

Одержано 27.01.2015