КОМПЬЮТЕРНАЯ МАТЕМАТИКА

УДК 519.6

О.М. ХІМІЧ, В.А. СИДОРУК

ГІБРИДНИЙ АЛГОРИТМ РОЗВ'ЯЗУВАННЯ ЛІНІЙНИХ СИСТЕМ З РОЗРІДЖЕНИМИ МАТРИЦЯМИ НА ОСНОВІ БЛОЧНОГО LL^T МЕТОДУ

Розглядається новий 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.

Вступ. При чисельному моделюванні процесів та явищ різної природи у багатьох випадках виникає необхідність розв'язувати задачу (або декілька задач) лінійної алгебри. Наприклад, задачі лінійної алгебри виникають при дискретизації крайових задач або задач на власні значення проекційно-різницевим методом (скінченних різниць, скінченних елементів). Причому, як правило, розв'язування лінійних систем займає значну частину (50 % і більше) часу розв'язування всієї задачі загалом.

Важливою особливістю задач лінійної алгебри, що виникають при дискретизації, є високий порядок їх матриць – до десятків мільйонів. Це спричинено бажанням оперувати більш точними дискретними моделями, що дозволяють отримувати наближені розв'язки більш близькі до розв'язків вихідних задач, краще враховувати локальні особливості даного процесу або явища. Також характерною 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 задачі і часто є стрічковою, блочно-діагональною з обрамленням, профільною тощо. В роботі розвивається актуальний на даний час напрямок, пов'язаний з прискоренням обчислень за рахунок використання графічних прискорювачів.

Постановка задачі

Розглянемо задачу

$$Ax = b \qquad (1)$$

з симетричною додатно-визначеною розрідженою матрицею порядку *n*.

Теоретичною передумовою методу розв'язання задачі (1) для розріджених матриць на комп'ютерах гібридної архітектури є попереднє застосування до вихідної матриці методу паралельних перерізів, який приводить вихідну матрицю до блочно-діагонального вигляду з обрамленням [1, 2]

$$\tilde{A} = P^T A P = \begin{pmatrix} A_{11} & 0 & 0 & \cdots & 0 & A_{1p} \\ 0 & A_{22} & 0 & \cdots & 0 & A_{2p} \\ 0 & 0 & A_{33} & \cdots & 0 & A_{3p} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & A_{p-1,p-1} & A_{p-1,p} \\ A_{p1} & A_{p2} & A_{p3} & \cdots & A_{p,p-1} & A_{pp} \end{pmatrix},$$

де P – матриця перестановок, а блоки A_{ii}, A_{ip}, A_{pi} зберігають розріджену структуру, p – кількість діагональних блоків у матриці.

Таким чином, задача розв'язування вихідної задачі (1) зводиться до розв'язування еквівалентної задачі

$$\tilde{A}\tilde{x} = \tilde{b}, \qquad (2)$$

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

Найбільш ефективним прямим методом розв'язання такої задачі є, як відомо, метод Холецького [1 – 3]. Розв'язання системи (2) полягає у розв'язанні підзадач: трикутне розвинення матриці системи (3), розв'язання двох СЛАР з трикутними матрицями (4) та (5):

$$\tilde{A} = \tilde{L} * \tilde{L}^T. \qquad (3)$$

$$\tilde{L}y = \tilde{b}. \qquad (4)$$

$$\tilde{L}^T \tilde{x} = y. \qquad (5)$$

Блочний алгоритм прямого методу. Враховуючи те, що матриця є блочно-діагональною з обрамленням, розглянемо блочний варіант алгоритму прямого методу на основі LL^T факторизації. Його можна представити у такій формі.

Факторизація.

Для *i*, $i = \overline{0, p-1}$ послідовно виконуємо:

- факторизацію блоку A_{ii};

$$A_{ii} = L_{ii} L_{ii}^T; \qquad (6)$$

- модифікацію блоку обрамлення

$$L_{ip} = L_{ii}^{-1} A_{ip}; \qquad (7)$$

- знаходження добутку $L_{ip}L_{ip}^T$ та модифікацію A_{pp}

$$\tilde{A}_{pp} = A_{pp} - \sum_{i=0}^{p-1} L_{ip}L_{ip}^T. \quad (8)$$

На останньому кроці виконується факторизація $\tilde{A}_{pp}$

$$\tilde{A}_{pp} = L_{pp}L_{pp}^T. \quad (9)$$

В результаті

$$\tilde{L} = \begin{pmatrix} L_{11} & & & & & 0 \\ 0 & L_{22} & & & & \\ 0 & 0 & L_{33} & & & \\ \vdots & \vdots & & \ddots & & \\ 0 & 0 & 0 & & L_{p-1} & \\ L_{p1} & L_{p2} & L_{p3} & \cdots & L_{pp-1} & L_{pp} \end{pmatrix},$$

де $L_{pi} = L_{ip}^T$, $i = \overline{0, p-1}$.

Розв'язання системи (4).

Для i, $i = \overline{0, p-1}$ послідовно виконуємо:

- розв'язання системи $L_{ii}y_i = b_i$;
- знаходження добутків $L_{pi}y_i$ і модифікацію p-ї частини вектора b в нульовому процесорі $\tilde{b}_p = b_p - L_{pi}y_i$.

На останньому кроці розв'язуємо систему $L_{pp}y_p = \tilde{b}_p$.

Розв'язання системи (5).

Виконуємо розв'язання системи $L_{pp}{}^T x_p = y_p$.

Для i, $i = \overline{0, p-1}$ послідовно виконуємо наступні операції:

- знаходження добутків $L_{ip}x_p$ і модифікацію i-ої частини вектора y $y_i = y_i - L_{ip}x_p$;
- розв'язання системи $x_i = L_{ii}^{-T} y_i$.

Гібридний алгоритм. Розглянемо декомпозицію даних для комп'ютерів гібридної архітектури з багатоядерними (CPU) процесорами і графічними (GPU) прискорювачами. Нехай для розв'язування задачі маємо p – 1 процесорне ядро і p – 1 GPU. Будемо використовувати наступний розподіл даних:

- на GPU, що відповідають процесам з номерами i, $i = \overline{0, p-1}$ зберігаються відповідні блоки A_{ii}, A_{ip} і відповідні частини векторів x, y, b;

- на GPU, що відповідають процесу з номером 0, додатково буде зберігатись блок A_{pp}.

Тоді гібридний алгоритм прямого методу на основі LL^T-факторизації можна представити наступним чином.

Факторизація.

У всіх процесах з номерами i, $i=\overline{0, p-1}$ одночасно реалізуємо гібридний алгоритм факторизації блоку A_{ii} : $A_{ii} = L_{ii} L_{ii}^T$, використовуючи, наприклад, підхід, запропонований в [4, 5], активно використовуючи GPU в розрахунках.

Одночасно на відповідних GPU виконуємо:

- модифікацію блоку обрамлення $L_{ip} = L_{ii}^{-1^T} A_{ip}$;
- знаходження добутку $L_{ip} L_{ip}^T$.

Копіюємо на відповідні CPU результати виконання $L_{ip} L_{ip}^T$.

Виконуємо модифікацію A_{pp} через мультизбирання в процесі з номером 0 $\tilde{A}_{pp} = A_{pp} - \sum_{i=0}^{p-1} L_{ip} L_{ip}^T$.

У нульовому процесі копіюємо результат у відповідний GPU і виконуємо факторизацію $\tilde{A}_{pp}$: $\tilde{A}_{pp} = L_{pp} L_{pp}^T$.

Розв'язання системи (4).

На GPU, що відповідають процесам з номерами i, $i=\overline{0, p-1}$, виконуємо:

- розв'язання системи $L_{ii} y_i = b_i$;
- знаходження добутків $L_{pi} y_i$.

Копіюємо результати виконання $L_{pi} y_i$ на відповідний процесор і виконуємо мультизбирання та модифікацію p-ї частини вектора b в нульовому процесі $\tilde{b}_p = b_p - L_{pi} y_i$.

На GPU, що відповідає процесу з номером 0, копіюємо $\tilde{b}_p$ і розв'язуємо систему $L_{pp} y_p = \tilde{b}_p$.

Розв'язання системи (5).

На GPU, що відповідає процесу з номером 0, розв'язуємо систему $L_{pp}{}^T x_p = y_p$.

Копіюємо результат виконання на процесор і виконуємо розсилку x_p у інші процеси.

На відповідних GPU виконуються наступні операції:

- знаходимо добутки $L_{ip} x_p$ і модифікуємо i-у частину вектора y $y_i = y_i - L_{ip} x_p$;
- розв'язуємо систему $L_{ii}{}^T x_i = y_i$.

Прискорення та ефективність алгоритму. Оскільки в загальному випадку складність наведеного алгоритму визначається в основному складністю трикутного розвинення матриці, наступні оцінки будуть стосуватись саме етапу факторизації.

Припустимо, що кількість CPU і GPU, які використовуються однаково – p. Тоді для знаходження величини прискорення та ефективності паралельного алгоритму скористуємось наступними формулами:

$$S_p = T_1 / T_p,\ E_p = S_p / p,$$

де T_p – час розв'язування задачі на гібридному комп'ютері з використанням p CPU і p GPU, T_1 – час розв'язування тієї ж задачі на гібридному комп'ютері з одним CPU та одним GPU.

Враховуючи, що тільки незначна кількість арифметичних операцій виконується на CPU, а основна частина обчислень в алгоритмі реалізовуються на GPU, для обчислення часових характеристик можна використовувати формули.

$$T_1 = Nt_g,\ T_g = Nt_g + M_1 t_{opg} + M_2 t_{opp} + Q_1 t_{cpg} + Q_1 t_{cpg},$$

де N – кількість арифметичних операцій, t_g – середній час виконання однієї арифметичної операції на GPU, t_{opp} – час, необхідний для обміну одним машинним словом між двома процесами, t_{opg} – час обміну між CPU і GPU, t_{cpp} – час, який потрібен для встановлення зв'язку між двома процесами, t_{cpp} – час, який потрібен для встановлення зв'язку між CPU і GPU, M_i, Q_i – кількість відповідних обмінів та синхронізацій.

Враховуючи (6) – (9), а також відповідну декомпозицію даних по процесах, наведемо оцінки для коефіцієнтів прискорення та ефективності у випадку, коли кількість діагональних блоків у матриці на одиницю більше кількості процесів ($p \geq 2$). Коефіцієнти отримуються для топології «Кільце».

Обчислення прискорення алгоритму розглянемо у випадку, коли діагональні блоки крім останнього зберігаються в стрічковому форматі. Вважається, що кількість наддіагоналей k у всіх діагональних блоках крім останнього однакова.

Слід зазначити, що складність гібридного алгоритму факторизації одного діагонального блоку з [5] визначається в основному графічною складовою обробки матриці на GPU.

Тоді головні члени для визначення кількості арифметичних операцій обчислюються наступним чином: кількість операцій для факторизації діагонального блоку за (6) – $1/2qk^2$; кількість операцій для виконання одного кроку (7) – $2qsk$; кількість операцій для виконання одного кроку (8) – $2q^2s$; факторизація останнього діагонального блоку реалізується за $\frac{1}{3}s^3$ арифметичних операцій, де s-порядок останнього діагонального блоку, q-порядок всіх діагональних блоків крім останнього.

Тоді

$$T_1 = \left(\left(p-1\right)\alpha + \beta\right)t_g,$$

де $\alpha = (\frac{1}{2}qk^2 + 2qsk + 2q^2s)$, $\beta = \frac{1}{3}s^3$.

Для паралельного алгоритму отримаємо наступне значення для кількості арифметичних операцій:

$$N = \alpha + \beta .$$

Тоді

$$T_p = \left(\alpha + \beta\right) t_g + \frac{s^2\left(p-1\right)}{2}t_{opp} + s^2 pt_{opg} + \frac{\left(p-1\right)}{2}t_{cpp} + pt_{cgg},$$

а значення S_p визначається формулою

$$S_p \approx \frac{\left(\left(p-1\right)\alpha + \beta\right)t_g}{\left(\alpha + \beta\right) t_g + \frac{s^2\left(p-1\right)}{2}t_{opp} + s^2 pt_{opg} + \frac{\left(p-1\right)}{2}t_{cpp} + pt_{cgg}}.$$

Оскільки, практично s набагато менше q, приходимо до

$$S_p \approx \frac{\left(p-1\right)}{1 + \frac{s^2\left(p-1\right)}{2}t_{opp} + s^2 pt_{opg} + \frac{\left(p-1\right)}{2}t_{cpp} + pt_{cgg}}, \quad E_p = \frac{S_p}{p}.$$

Результати чисельних експериментів. Для реалізації стандартних обчислювальних процедур (множення матриць, розв'язування трикутних систем) можна використовувати функції відомих бібліотек, наприклад, CUSPARSE, CUSP, Paralution.

Розрахунки проводились на вузлах кластера Інпарком-G [6], які мають такі 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;
- також на вузлах встановлена бібліотека MKL 10.2.6 та CUDA, починаючи з версії 3.2.

Чисельні експерименти здійснювались на розріджених матрицях, що наведені в таблиці. Також у таблиці наведені характеристики матриці такі як порядок матриці, кількість ненульових елементів. На рис. 1. показано графіки, що відображають залежність часу реалізації алгоритму для матриць на архітектурах 1 CPU, 8 CPU, 1 CPU + 1 GPU та n CPU + n GPU (n = 8). На рис. 2. показано залежність часу виконання алгоритму від кількості використовуваних GPU.

ТАБЛИЦЯ. Набір тестових матриць

Назва	Тип задачі	Порядок матриці	Кількість ненульових елементів
Minsurfo	Задача оптимізації	40806	203622
Dubcova3	2D/3D problem	146689	3636643
G2_circuit	Circuit simulation problem	150102	726674
A22bd_200k	Задача теплопровідності	200000	100473400
A22bd_400k	Задача теплопровідності	400000	81172000
A22bd_750k	Задача теплопровідності	750000	377119000

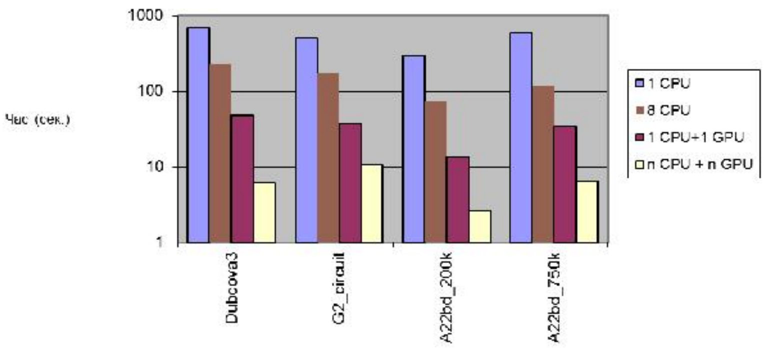

РИС. 1. Часи розв'язання систем з відповідними матрицями на різних архітектурах

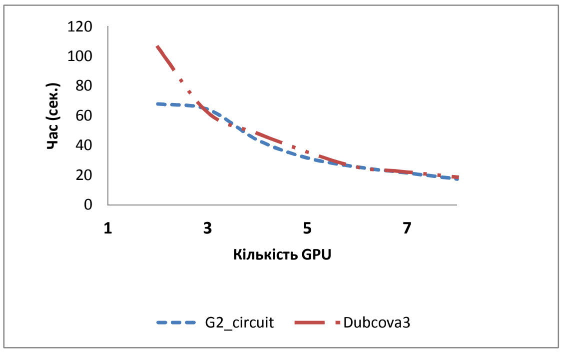

РИС. 2. Залежність часу виконання алгоритму від кількості використовуваних GPU

Висновки. Для СЛАР із розрідженими симетричними додатно визначеними матрицями, що попередньо приводяться до блочно-діагонального виду з обрамленням запропоновано алгоритм розв’язання, який забезпечує високу ефективність розпаралелювання на GPU, враховує структуру розрідженої матриці, оптимізує використання пам’яті GPU. Також наведено оцінки прискорення запропонованого паралельного алгоритму факторизації матриці.

А.Н. Химич, В.А. Сидорук

ГИБРИДНЫЙ АЛГОРИТМ РЕШЕНИЯ ЛИНЕЙНЫХ СИСТЕМ С РАЗРЕЖЕНЫМИ МАТРИЦАМИ НА ОСНОВАНИИ БЛОЧНОГО LL^T МЕТОДА

Рассматривается новый гибридный алгоритм решения систем линейных алгебраических уравнений с разрежеными симметричными положительно определенными матрицами на компьютерах с графическими ускорителями. Представлены результаты апробации алгоритма на многоядерном компьютере с графическими ускорителями Инпарком.

O.M. Khimich, V.A. Sydoruk

HYBRID ALGORITHM FOR SOLVING LINEAR SYSTEMS WITH SPARSE MATRICES BASED ON BLOCK LL^T-METHOD

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

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. – № 6. –С. 159 – 174.
3. *Уилкинсон Дж. Х., Райнш К.* Справочник алгоритмов на языке Алгол. Линейная алгебра. – М.: Машиностроение, 1976. – 389 с.
4. *Хіміч О.М., Баранов А.Ю.* Гібридний алгоритм розв’язування лінійних систем із стрічковими матрицями прямими методами // Компьютерная математика. – 2013. – № 2. – С. 80 – 88.
5. https://developer.nvidia.com/cholmod
6. *Химич А.Н., Молчанов И.Н., Мова В.И. и др.* Численное программное обеспечение MIMD-компьютера Инпарком. – Киев: Наукова думка, 2007. – 222 с.

Одержано 10.02.2015

Про авторів:

Хіміч Олександр Миколайович,

доктор фізико-математичних наук, професор, завідувач відділу
Інституту кібернетики імені В.М. Глушкова НАН України,

Сидорук Володимир Антонович,

молодший науковий співробітник
Інституту кібернетики імені В.М. Глушкова НАН України.