Паралельний поперемінно-трикутний метод розв'язання алгебраїчної проблеми власних значень для розріджених матриць на комп'ютерах гібридної архітектури

Хіміч О.М., Чистяков О.В.

Інститут кібернетики ім. В.М. Глушкова НАН України, пр. Глушкова 40, м. Київ, Україна

khimich_ic@mail.ru, alexej.chystyakov@gmail.com

Анотація. *Комп'ютери гібридної архітектури, які поєднують обчислення на багатоядерних комп'ютерах з прискоренням обчислень на графічних процесорах, дають можливість значно прискорити процес розв'язування алгебраїчної проблеми власних значень. Проте виникають проблеми ефективної реалізації паралельних алгоритмів з урахуванням особливостей гібридного комп'ютера, а саме: необхідність у плануванні обчислень на обчислювальних ресурсах CPU і GPU, оптимізації комунікаційних витрат між ядрами CPU і процесорами GPU і т. д. В роботі розглядаються деякі підходи створення ефективних алгоритмів та програм для розв'язування алгебраїчної проблеми власних значень розріджених матриць на комп'ютерах гібридної архітектури.*

Ключові слова

Комп'ютери гібридної архітектури, алгебраїчна проблема власних значень, розріджена матриця, технологія програмування CUDA.

1 Вступ

Велика кількість наукових та інженерних задач, при дослідженні стійкості конструкцій, розрахунку динаміки напружено-деформовного стану об'єктів різної природи та ін., зводяться до розв'язання часткової алгебраїчної проблеми власних значень з симетричними додатно-визначеними матрицями розрідженої структури великої розмірності [1]. Розв'язування таких задач на комп'ютерах потребує великих об'ємів обчислювальних ресурсів та часу їх виконання.

В статті запропоновано паралельні ітераційні алгоритми розв'язання алгебраїчної проблеми власних значень для симетричних додатно-визначених розріджених матриць на комп'ютерах гібридної архітектури. Ідеологічною передумовою запропонованого підходу є застосування до вихідної матриці ортогонально-подібних перетворень на основі методу паралельних перерізів, які приводять вихідну матрицю до блочно-діагонального вигляду з обрамленням, зберігаючи спектр вихідної матриці. Паралельні алгоритми базуються на сімействі однокрокових ітераційних алгоритмів знаходження найменшого власного значення та відповідного власного вектора симетричної додатно-визначеної блочно-діагональної матриці з обрамленням.

Останнім часом спостерігається певна тенденція в розвитку обчислювальних систем. З одного боку продовжується зростання продуктивності комп'ютерів за рахунок збільшення кількості процесорних ядер комп'ютера та їх тактової частоти, а з іншого боку появляються комп'ютери принципово нової архітектури, наприклад, гібридні, гетерогенні та різні технології паралельного програмування такі як MPI [2], OpenMP, CUDA [3], а також різні спеціалізовані високопродуктивні обчислювальні бібліотеки.

Таке різноманіття паралельних комп'ютерних архітектур та засобів програмування породжує проблеми стандартизації програмного забезпечення, сумісного використання відомих високопродуктивних бібліотек та технологій їх розробки. При розробці алгоритмів та програм розв'язання алгебраїчної проблеми власних значень для розріджених матриць ці проблеми пов'язані також з різноманітними форматами збереження

розріджених матриць в комп'ютері. Тому при розробці даного алгоритмічно-програмного забезпечення були розглянуті такі підходи, які забезпечують їх високу швидкодію на гібридних комп'ютерах та спрощують використання високопродуктивних паралельних технологій та бібліотек програм.

Дослідження показали, що найбільш ефективними виявляються блочні алгоритми для розв'язання задач даної проблематики. Вони забезпечують збалансовану обробку розріджених матриць на комп'ютерах гібридної архітектури та ефективне використання обчислювальних ресурсів. Крім того, реалізацію цих алгоритмів можна звести до обмеженої кількості базових операцій лінійної алгебри, які можна оптимізувати під задану архітектуру та використати для їх виконання стандартні бібліотеки програм у поєднанні з комбінованими форматами збереження розріджених матриць [4].

Програми, що реалізують паралельні ітераційні алгоритми розв'язання алгебраїчної проблеми власних значень для симетричних додатно-визначених розріджених матриць на комп'ютерах гібридної архітектури, складаються з обчислювальних кодів для CPU, написаних C++ з використанням бібліотеки програм Intel MKL [5], та кодів для GPU, написаних з використанням технології CUDA та бібліотек програм cuSPARSE [6] та cuBLAS [7]. Апробація розроблених алгоритмів та програм проводилась на інтелектуальній робочій станції гібридної архітектури Інпарком-G, використовуючи один CPU та один GPU.

2 Постановка задачі

Розглянемо задачу на власні значення:

$$Ax = \lambda Dx, \qquad (1)$$

де *A*, *D* розріджені додатно - визначені матриці порядку *n*, що діють в n-вимірному евклідовому просторі H зі скалярним добутком (,), λ та x – відповідне власне значення та власний вектор. Розглянемо наступну канонічну ітераційну однокрокову схему знаходження λ_1 та x_1:

$$B(y_{k+1} - y_k) + \tau_{k+1} r_k = 0 \text{, для } k = 0,1,2,3,\ldots,$$

де y_0 − довільне початкове значення, $r_k = Ay_k - \mu_k Dy_k$ − нев'язка; $\mu_k = (Ay_k, y_k)(Dy_k, y_k)^{-1}$ − наближення до власного значення; y_k − нормоване наближення до власного вектора, $\tau_к$ − ітераційний параметр; *B* − оператор (регуляризатор), що покращує швидкість ітераційного процесу та для якого легко знаходиться обернений.

В залежності від обраного оператору *B* та наборів параметрів τ можуть бути отримані різноманітні схеми ітераційних методів, наприклад:

1. Метод найшвидшого спуску для λ_1, $x_1(\tau^+)$ та λ_N, λ_N, $x_N(\tau^-)$, B = E

$$\tau_{k+1}^{\pm} = 2\left[\theta_k \pm \sqrt{\theta_k^2 + 4(w_k, w_k) - 4\theta_k (w_k, w_k)}\right]^{-1},$$

$$\theta_k = \left[(Aw_k, w_k) - \mu_k (w_k, w_k)\right](Bw_k, w_k)^{-1},$$

$$w_k = B^{-1} r_k.$$

2. Поперемінно-трикутний метод $B = (E + \omega\hat{R})(E + \omega\hat{R}^T)$, $A = \hat{R} + \hat{R}^T$, де R_1 та R_2 − відповідно верхня та нижня трикутні матриці.

$$(E + \omega_{(k)} R_1)(E + \omega_{(k)} R_2) w_{(k)} = r^{(k)}$$

3 Паралельний алгоритм поперемінно-трикутного методу

Ідеологічною передумовою підходу до вибору передобумовлювача, що пропонується, є застосування до вихідної матриці попередньо методу паралельних перерізів, який приводить вихідну матрицю до блочно-діагонального вигляду з обрамленням [8]:

$$\hat{A} = P^T AP = \begin{pmatrix} D_1 & 0 & 0 & ... & 0 & B_1 \\ 0 & D_2 & 0 & ... & 0 & B_2 \\ 0 & 0 & D_3 & & 0 & B_3 \\ \vdots & \vdots & & \ddots & & \vdots \\ 0 & 0 & 0 & & D_{p-1} & B_{p-1} \\ B_1 & B_2 & B_3 & ... & B_{p-1} & D_p \end{pmatrix}$$

де P – матриця перестановок, а блоки D_i і B_i зберігають розрідженість.

Таким чином, початкова задача зводиться до наступної:

$$\hat{A}y = \lambda \hat{D}y,$$

де $\hat{A}$ - блочно-діагональна з обрамленням, $\hat{D} = P^T DP$ - розріджена додатньо-визначена матриця.

Для поперемінно-трикутного методу передобумовлювач має вигляд [9]:

$$B = (E + \omega\hat{R})(E + \omega\hat{R}^{\mathrm{T}}), \quad \hat{A} = \hat{R} + \hat{R}^{\mathrm{T}}.$$

При цьому $\hat{R}$ зберігає блочно-трикутну структуру, успадковану від матриці $\hat{A}$:

$$\hat{R} = \begin{pmatrix} \hat{R}_1 \\ \hat{R}_2 \\ \hat{R}_3 \\ \vdots \\ \hat{R}_{p-1} \\ \hat{R}p \end{pmatrix} = \begin{pmatrix} \tilde{D}_1 & 0 & 0 & \dots & 0 & 0 \\ 0 & \tilde{D}_2 & 0 & \dots & 0 & 0 \\ 0 & 0 & \tilde{D}_3 & \dots & 0 & 0 \\ \vdots & \vdots & & \ddots & & \dots \\ 0 & 0 & 0 & & \tilde{D}_{p-1} & 0 \\ C_1 & C_2 & C_3 & \dots & C_{p-1} & \tilde{D}_p \end{pmatrix} \qquad (2)$$

де блоки $\tilde{D}_i$ $(1 \le i \le p)$ є нижні трикутні матриці блоків D_i.

За умов $\|\hat{x}\|_{\hat{A}}^2 \ge \delta\|\hat{x}\|^2$, $\left\|\hat{R}^{\mathrm{T}}\hat{x}\right\| \le \Delta\|\hat{x}\|_{\hat{R}}^2 / 4$, параметр $\omega = \dfrac{2}{\sqrt{\delta\Delta}}$ забезпечує збіжність ітераційного процесу зі швидкістю геометричної прогресії [4].

Розподіл вхідної матриці по процесах. При реалізації поперемінно-трикутного методу на гібридних комп'ютерах для розподілу підматриць матриці $\hat{R}$ по процесах CPU було використано блочну схему (2). З огляду на структуру $\hat{R}$ це означає, що процеси з номерами $0 \le i < p$ зберігають блоки $\tilde{D}_i$ та C_i, а процес з номером p–1 зберігає блок $\tilde{D}_p$. Тут p – загальна кількість процесів.

Паралельна реалізація алгоритму визначається блочно-трикутною структурою матриць $\hat{R}_i$ при розв'язанні системи $Bw = r$,

$$B = (E + \omega\hat{R})(E + \omega\hat{R}^T), \quad \hat{A} = \hat{R} + \hat{R}^T.$$

Паралельний алгоритм розв'язання нижньої трикутної системи:

$$(E + \omega\hat{R})y = r$$

зводиться до одночасного та незалежного виконання на окремих процесах розв'язування трикутних систем:

$$(E + \omega\tilde{D}_q)y_q = r_q,\ 1 \le q < p,$$

та наступного обчислення $\tilde{y}_q$:

$$\tilde{y}_q = C_q y_q, \quad 0 \le q < p$$

де q – номер GPU, після чого останній процес сумує значення $\tilde{y}_q$, надіслані від інших GPU і знаходить y_p, розв'язуючи систему $(E + \omega\tilde{D}p)y_p = r_p - \sum_{q=1}^{p-1} \tilde{y}_q$.

Аналогічно для знаходження розв'язку системи

$$(E + \omega R^T)w = y$$

p-й процес розв'язує систему

$$(E + \omega\tilde{D}_p^T)w_p = y_p$$

і розсилає компоненти w_p іншим процесам, вони незалежно розв'язують системи:

$$(E + \omega\tilde{D}_q^T)w_q = y_q - C_q^T w_p,\ 1 \le q < p$$

4 Комбіновані формати збереження розріджених матриць

Складність ефективної обробки розріджених матриць привела до створення великої кількості форматів їх збереження в комп'ютері, наприклад, CSR (compressed sparse row), CSC (compressed sparse column), COO (coordinate) та інші [10]. На рис.1 представлено різні варіанти збереження розрідженої матриці у CSR форматі у трьох та чотирьох векторному представленні, з нульовою та одиничною індексацією.

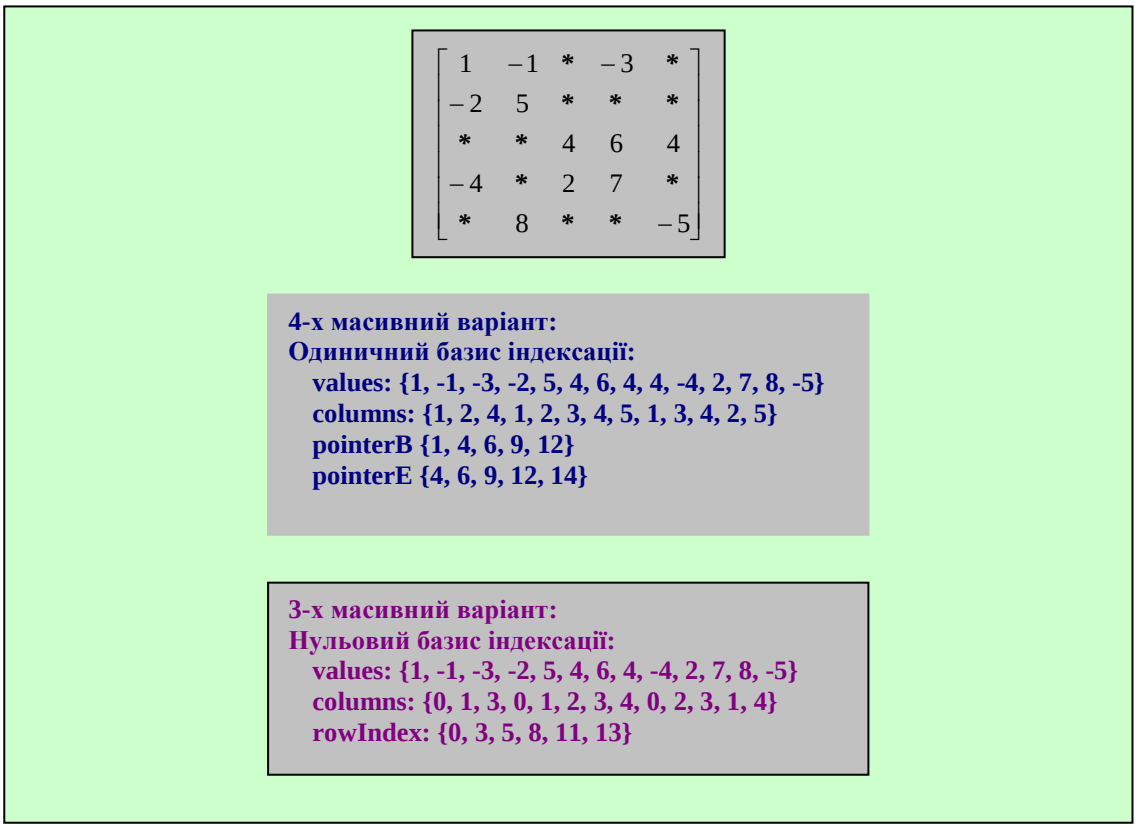

Рис. 1. Представлення розрідженої матриці у форматі CSR

Оскільки розріджені матриці великої розмірності потребують великих об'ємів обчислювальних ресурсів для їх збереження в комп'ютері та витрат часу на їх обробку, виникає потреба у виконанні великої кількості додаткових операцій при їх переформатуванні або переіндексації для забезпечення масштабованості обчислювального процесу. Тому був розроблений комбінований формат збереження матриць, що базується на CSR форматі та враховує особливості представлення блочно-діагональної розрідженої матриці з обрамленням. Схематично комбінований формат блочно-діагональної розрідженої матриці з обрамленням представлений на рис. 2.

Рис. 2. Комбінований формат збереження блочно-діагональної розрідженої матриці з обрамленням

З рис.2 видно, що блочно-діагональна матриця з обрамленням зберігається у class CSR::BD::Matrix як масиви вказівників на діагональні блоки та блоки з обрамлення. Тобто кожен блок є окремою підматрицею, збереженою у модифікованому форматі CSR (`class CSR::Matrix`).

Такий підхід дозволяє маніпулювати кожним блоком незалежно та забезпечувати можливість розпаралелення обчислень, як на MIMD-комп'ютерах, так і на комп'ютерах гібридної архітектури. `Class CSR::BD::Matrix` зберігає інформацію про вхідну матрицю `class MatrixInfo`, а саме:

- розмірність матриці, кількість ненульових елементів та її розрідженість;
- кількість заданих перерізів при перетворенні матриці;
- кількість отриманих блоків у діагоналі та в обрамленні;
- характеристики матриці такі як симетричність, представлення матриці у трикутному вигляді, та місце розташування трикутника (над чи під головною діагоналлю);
- базис матриці (нульовий чи одиничний).

В свою чергу `class CSR::Matrix` зберігає підматриці у модифікованому форматі CSR, тобто поєднує у собі 3-х та 4-х векторний формат збереження. В результаті зберігаються такі масиви:

- `vector<double> Value` – ненульові елементи матриці;
- `vector<int> Columns` – позиція ненульового елементу у рядку;
- `vector<int> RowIndex` – кількість ненульових елементів у кожному рядку;
- `vector<int> PointerB` – кількість ненульових елементів у кожному попередньому рядку;
- `vector<int> PointerE` – кількість ненульових елементів у кожному наступньому рядку;
- `MatrixInfo csr_bd_matrix_info` – містить інформацію про кожну окрему підматрицю та її розташування у блочно-діагональній матриці з обрамленням.

В результаті отримано уніфікований 5-ти рядковий формат збереження розрідженої матриці, що дозволяє використовувати більшість високопродуктивних бібліотек обчислювальної математики. Крім того, якщо в процесі обчислень додаткові масиви не потрібні, то вони видаляться автоматично, тим самим досягається більш оптимальне використання оперативної пам'яті.

Такий підхід до збереження розріджених матриць дозволяє ефективно використовувати концепцію «функцій-обгорток» [11] при розробці паралельних програм та спрощує процес реалізації паралельних алгоритмів. «Обгортка» (англ. Wrapper) – це функція, яка є проміжною ланкою між прикладними функціями або програмами та іншою бібліотекою чи програмним інтерфейсом, а також може модифікувати або

узагальнювати інтерфейси прикладних програм. До «обгорток» винесені службові функції для роботи з графічним прискорювачем, наприклад, `cusparseCreate(…)`, `cudaMemcpy(…)` та інші.

5 Апробація паралельних алгоритмів

Для експериментальної апробації розроблених паралельних алгоритмів та програм було використано розріджені матриці з колекції Флоридського університету [12]. В табл. 1 подано характеристику кожної тестової матриці, а саме: назва задачі, предметна область, з якої були отримані вхідні дані, порядок матриці, кількість ненульових елементів. Експерименти проводились для випаду, коли D – одинична матриця, тобто $D=E$.

Таблиця 1. Набір тестових розріджених матриць з Флоридської колекції

Назва задачі	***Проблемна область***	***Порядок матриці***	***Кількість ненульових елементів***
Dubcova2	2D/3D problem	65 025	1 030 225
Dubcova3	2D/3D problem	146 689	3 636 643
Bmwcra_1	Structural problem	148 770	10 641 602
Bone010	Model reduction problem	986 703	47 851 783
Emila_923	Structural problem	923 136	40 373 538

Технічні характеристики Інпарком-G [13]: чотири вузли з двома 4-х-ядерними Intel Xeon E5606 процесорами, оперативна пам’ять: 3 Гб на одне фізичне обчислювальне ядро, графічний прискорювач – Nvidia Tesla M2090 з 6-ма Гб пам’яті, два графічні прискорювачі на один вузол.

В табл. 2 приведено часові характеристики знаходження найменшого власного значення поперемінно-трикутним методом та методом найшвидшого спуску послідовною програмою з використанням бібліотеки Intel MKL та паралельною програмою на основі запропонованого гібридного алгоритму поперемінно-трикутного методу (1 CPU + 1 GPU) з використанням бібліотек cuBLAS та cuSPARSE для розріджених матриць.

Таблиця 2. Часові характеристики знаходження власного значення поперемінно-трикутним алгоритмом та алгоритмом найшвидшого спуску, сек

Назва задачі	***Поперемінно-трикутний метод***			***Метод найшвидшого спуску***		
	1 CPU	***1CPU+1GPU***	**Прискорення**	***1 CPU***	***1CPU+1GPU***	**Прискорення**
Dubcova2	0,76	1,688	0,45	1,553	3,304	0,47
Dubcova3	1,85	2,5	0,74	4,412	5,19	0,85
bmwcra_1	820,74	177,649	4,62	3756,55	789,19	4,76
Emilia_923	1185,52	246,469	4,81	4730,43	973,339	4,86
Bone	423,8	72,943	5,81	1345,18	243,251	5,53

З таблиці видно, що час розв’язування задачі при використанні графічного прискорювача значно зменшується. Виявилося, що у разі знаходження власного значення для розріджених матриць невеликого порядку послідовний варіант програми працює ефективніше ніж паралельний варіант з використанням графічного прискорювача. З табл. 3 видно, що для матриць невеликого розміру (Dubcova2, Dubcova3) при використанні графічного прискорювача, прискорення менше одиниці, тобто використання графічного прискорювача недоцільне. В той час, як для матриць Bmwcra_1, Emila_923, Bone010, кількість ненульових елементів яких більше 10 000 000, використання GPU показує прискорення у 4 та більше разів.

6 Висновки

В даній роботі розглядається реалізація паралельних ітераційних алгоритмів розв'язання алгебраїчної проблеми власних значень для симетричних додатно-визначених розріджених матриць на гібридних комп'ютерах архітектури 1 CPU + 1 GPU.

Теоретичні дослідження і практичні експерименти показали, що використання графічних процесорів для розв'язування алгебраїчної проблеми власних значень з розрідженими матрицями великих розмірностей дає суттєве прискорення.

Застосування відомих високопродуктивних бібліотек програм cuBLAS, cuSPARSE, Intel MKL не тільки сприяє спрощенню створення паралельних програм для гібридних систем, але також значно зменшує час розв'язування задач. Таким чином, забезпечується висока продуктивність використання комп'ютерів гібридної архітектури.

Для ефективного виконання програм в розподіленому середовищі необхідно створювати такі алгоритми та програми, які добре масштабуються з урахуванням обмежень мережевих комунікацій: пропускні можливості мережі та затримки мають великий вплив на швидкодію алгоритмів та програм для гібридних архітектур. Для вирішення проблем пов'язаних з масштабованістю використовується приведення вхідної матриці до блочно-діагонального виду з обрамленням.

Використання комбінованих форматів збереження даних дозволило суттєво спростити процес розробки паралельного програмного забезпечення та використання різних прикладних програмних засобів.

У перспективі передбачається створення паралельних алгоритмів для комп'ютерів гібридної архітектури типу *n* CPU + *m* GPU з використанням технологій MPI та GPUDirect [14].

Посилання

[1] Приказчиков В. Г. Прототипы итерационных процессов в задаче на собственные значения. // Дифференциальные уравнения.– 1980. – том 16, №. 9. – С. 1688 − 1697.
[2] MPI. http://www.mpi-forum.org/
[3] CUDA 6.5. http://docs.nvidia.com/cuda/index.html#axzz3BRODbnj5.
[4] С. П. Копысов, А. К. Новиков Промежуточное программное обеспечение параллельных вычислений – Ижевск: «Удмуртский университет», 2012. – 140 с.
[5] Intel Maths Kernel Library. http://software.intel.com/en-us/intel-mkl.
[6] cuSPARSE. http://docs.nvidia.com/cuda/cusparse/#axzz3BRODbnj5.
[7] CUBLAS Linear Algebra. http://developer.download.nvidia.com/CUBLAS Library.pdf.
[8] Джордж А., Лю Дж. Численное решение больших разреженных систем уравнений – М.: Мир, 1984. – 334 с.
[9] Приказчиков В. Г., Химич А. Н. Итерационные методы решения задач устойчивости и колебания пластин и оболочек. // Прикладная механика.– 1984. – том 20, №. 1. – С. 88 − 94.
[10] Survey of Sparse Matrix Storage Formats. http://netlib.org/linalg/html_templates/node90.html.
[11] Э. Гамма, Р. Хелм, Р. Джонсон, Дж. Влиссидес Приемы объектно-ориентированного проектирования. Паттерны проектирования – СПб: «Питер», 2007. – 366 с.
[12] The University of Florida Sparse Matrix Collection. http://www.cise.ufl.edu/research/sparse/matrices.
[13] Химич А.Н., Молчанов И.Н., Мова В.И. и др. Численное программное обеспечение MIMD–компьютера Инпарком. – Киев: Наукова думка, 2007. – 222 с.
[14] GPUDirect. http://docs.nvidia.com/cuda/gpudirect-rdma/index.html#axzz3BRODbnj5.