УДК 519.6 https://doi.org/10.15407/pp2020.02-03.208

ГІБРИДНИЙ АЛГОРИТМ МЕТОДУ НЬЮТОНА ДЛЯ РОЗВ'ЯЗУВАННЯ СИСТЕМ НЕЛІНІЙНИХ РІВНЯНЬ З БЛОЧНИМИ МАТРИЦЯМИ ЯКОБІ

О.М. Хіміч, В.А. Сидорук, А.Н. Нестеренко

Системи нелінійних рівнянь часто виникають при моделюванні процесів різної природи. Це можуть бути як самостійні задачі, що описують фізичні процеси, так і задачі, що виникають на проміжному етапі вирішення складніших математичних задач. Зазвичай це задачі великих порядків з великою кількістю невідомих, які краще враховують локальні особливості процесу або явища, що моделюється. Крім того, більш точні дискретні моделі дозволяють отримати більш точні розв'язки. Зазвичай матриці таких задач мають розріджену структуру. Часто структура розріджених матриць є однією з наступних: стрічкова, профільна, блочно-діагональна з обрамленням і т. д. У багатьох випадках матриці дискретних задач є симетричними і додатно визначеними або напіввизначеною. Розв'язання систем нелінійних рівнянь здійснюється в основному ітераційними методами на основі методу Ньютона, який має високу швидкість збіжності (квадратичну) поблизу розв'язку за умови, що початкове наближення лежить в області гравітації розв'язку. В цьому випадку метод вимагає на кожній ітерації обчислення матриці Якобі і подальшого розв'язання систем лінійних алгебраїчних рівнянь. Як наслідок, складність однієї ітерації дорівнює $O(n^3)$. Використання паралельних обчислень на етапі розв'язання систем лінійних алгебраїчних рівнянь значно прискорює процес знаходження розв'язку систем нелінійних рівнянь. У роботі запропоновано новий метод розв'язання систем нелінійних рівнянь високого порядку з блочною матрицею Якобі. Основою нового методу є об'єднання класичного алгоритму методу Ньютона з ефективним дрібно-плитковим алгоритмом для розв'язання систем лінійних рівнянь з розрідженими матрицями. Наведено часи розв'язання систем нелінійних рівнянь різних порядків на вузлах суперкомп'ютера СКІТ.

Ключові слова: системи нелінійних рівнянь, гібридний алгоритм, розріджені матриці, системи лінійних алгебраїчних рівнянь, високопродуктивні обчислення.

Системы нелинейных уравнений часто возникают при моделировании процессов различной природы. Это могут быть как самостоятельные задачи, описывающие физические процессы, так и задачи, возникающие на промежуточном этапе решения более сложных математических задач. Обычно это задачи высокого порядка с большим количеством неизвестных, которые лучше учитывают локальные особенности процесса или моделируемого явления. Кроме того, более точные дискретные модели позволяют получить более точные решения. Обычно матрицы таких задач имеют разреженную структуру. Часто структура разреженных матриц является одной из следующих: ленточная, профильная, блочно-диагональная с обрамлением и т. д. Во многих случаях матрицы дискретных задач являются симметричными и положительно определеными или полуопределеными. Решение систем нелинейных уравнений осуществляется в основном итерационными методами на основе метода Ньютона, который имеет высокую скорость сходимости (квадратичную) вблизи решения при условии, что начальное приближение лежит в области гравитации решения. В этом случае метод требует на каждой итерации вычисления матрицы Якоби и дальнейшего решения систем линейных алгебраических уравнений. Как следствие, сложность одной итерации равна $O(n^3)$. Использование параллельных вычислений на этапе решения систем линейных алгебраических уравнений значительно ускоряет процесс нахождения решения систем нелинейных уравнений. В работе предложен новый метод решения систем нелинейных уравнений высокого порядка с блочной матрицей Якоби. Основой нового метода является объединение классического алгоритма метода Ньютона с эффективным мелко плиточным алгоритмом для решения систем линейных уравнений с разреженными матрицами. Приведены времена решения систем нелинейных уравнений разных порядков на узлах суперкомпьютера СКИТ.

Ключевые слова: системы нелинейных уравнений, гибридный алгоритм, разреженные матрицы, системы линейных алгебраических уравнений, высокопроизводительные вычисления.

Systems of nonlinear equations often arise when modeling processes of different nature. These can be both independent problems describing physical processes and also problems arising at the intermediate stage of solving more complex mathematical problems. Usually, these are high-order tasks with the big count of un-knows, that better take into account the local features of the process or the things that are modeled. In addition, more accurate discrete models allow for more accurate solutions. Usually, the matrices of such problems have a sparse structure. Often the structure of sparse matrices is one of next: band, profile, block-diagonal with bordering, etc. In many cases, the matrices of the discrete problems are symmetric and positively defined or half-defined. The solution of systems of nonlinear equations is performed mainly by iterative methods based on the Newton method, which has a high convergence rate (quadratic) near the solution, provided that the initial approximation lies in the area of gravity of the solution. In this case, the method requires, at each iteration, to calculates the Jacobi matrix and to further solving systems of linear algebraic equations. As a consequence, the complexity of one iteration is $O(n^3)$. Using the parallel computations in the step of the solving of systems of linear algebraic equations greatly accelerates the process of finding the solution of systems of nonlinear equations. In the paper, a new method for solving systems of nonlinear high-order equations with the Jacobi block matrix is proposed. The basis of the new method is to combine the classical algorithm of the Newton method with an efficient small-tile algorithm for solving systems of linear equations with sparse matrices. The times of solving the systems of nonlinear equations of different orders on the nodes of the SKIT supercomputer are given.

Key words: systems of nonlinear equations, hybrid algorithm, sparse matrices, systems of linear algebraic equations, high-performance computing.

Вступ

Системи нелінійних рівнянь (СНР) часто виникають при моделюванні процесів різної природи. Це можуть бути як самостійні задачі, що описують фізичні процеси, так і задачі, що виникають на проміжному етапі розв'язування більш складних математичних задач. Як правило, це задачі надвисоких порядків, які дозволяють краще враховувати локальні особливості процесу або явища, що моделюється. Окрім того, більш точні дискретні моделі дозволяють отримувати точніші наближені розв'язки.

З іншого боку матриці таких задач мають розріджену структуру. Найчастіше це стрічкова, профільна, блочно-діагональна з обрамленням тощо. У багатьох випадках матриці дискретних задач симетричні і додатно визначені або напіввизначені.

Розв'язування СНР здійснюється в основному ітераційними методами, що базуються на методі Ньютона, який має високу швидкість збіжності (квадратичну) поблизу розв'язку за умови, що початкове наближення лежить в області тяжіння розв'язку. При цьому метод вимагає на кожній ітерації обчислення матриці Якобі та подальшого розв'язування систем лінійних алгебраїчних рівнянь (СЛАР). Як наслідок складність однієї ітерації становить $O(n^3)$.

Розпаралелювання обчислення розв'язку СЛАР істотно прискорює процес знаходження розв'язку СНУ.

Постановка задачі з наближеними вихідними даними

Нехай дана система *n* нелінійних рівнянь

$$f(x)=0, \tag{1}$$

де $x=(x_1,x_2,\ldots,x_n)^T$, $f(x)=(f_1(x),f_2(x),\ldots,f_n(x))^T$ – *n*-вимірний вектор шуканого розв'язку та *n*-вимірна вектор-функція відповідно.

Задача (1) є деяким наближенням до точної системи нелінійних рівнянь $\varphi(x)=0$, і для цих вектор-функцій виконується нерівність:

$$\|f(u)-\varphi(u)\|\le\Delta$$

для будь-якого *n*-вимірного вектора *u*.

Для розв'язування задачі (1) задаються початкове наближення $x^{(0)}$, необхідна точність ε отримання наближення до розв'язку системи та визначається область, в якій шукається розв'язок, $D=\{a_i\le x_i\le b_i,\ \ i=1,2,\ldots,n\}$. При цьому початкове наближення належить визначеній області $x^{(0)}\in D$. Нижнім індексом у формулах позначаються номери компонент векторів, а верхнім індексом будуть позначатися номери ітерацій.

Якщо $A=\left\{\frac{\partial f_i}{\partial x_j}\right\}_{i,j=1}^{n}$ – матриця Якобі системи (1) (або деяке наближення до неї), то ітераційний процес методу Ньютона знаходження розв'язку при заданому початковому наближенні записується у вигляді

$$A^{(k)}w^{(k)}=-f\left(x^{(k)}\right), \tag{2}$$

де $w^{(k)}=x^{(k+1)}-x^{(k)}$ – поправка, *k* = 0, 1, ... – номер ітерації. Наступне наближення до розв'язку обчислюється за формулою:

$$x^{(k+1)}=x^{(k)}+w^{(k)}. \tag{3}$$

Як видно з формули (2), на кожній ітерації необхідно розв'язувати систему лінійних алгебраїчних рівнянь, обчислюючи значення вектор-функції і матрицю Якобі.

Для отримання розв'язку системи нелінійних рівнянь (1) з даною точністю $\|x^{(k)}-x\|\le\varepsilon$ ітераційний процес необхідно закінчувати при виконанні умови

$$\left\|f\left(x^{(k+1)}\right)\right\|\le\frac{\varepsilon}{\left\|\left(A^{(k+1)}\right)^{-1}\right\|}, \tag{4}$$

де $A^{-1^{(k)}}$ – матриця, обернена по відношенню до матриці Якобі, обчисленої на *k*-ій ітерації, а відношення

$$\frac{\varepsilon}{\left\| A^{-1(k)} \right\|}$$

в подальшому будемо позначати як Δ [1]. Оскільки для перевірки умови (4) на кожній ітерації необхідно обертати матрицю Якобі, що вимагає значної кількості арифметичних операцій, то на перших ітераціях перевіряється більш економна умова закінчення ітерацій $\left\| f\left(x^{(k)}\right) \right\| \le \varepsilon$, а після її виконання здійснюється перехід до перевірки умови (4).

Після завершення ітераційного процесу обчислюється похибка отриманого наближення до розв'язку задачі з наближеними даними відносно точного розв'язку системи з точними даними:

$$\| x_k - x \| \le \varepsilon + \left\| A_k^{-1} \right\| \Delta ,$$

де *x* – точний розв'язок точної СНР.

Звідси випливає, що при розв'язуванні СНР значна частина арифметичних операцій припадає на обчислення значень вектор-функції, розв'язування відповідної СЛАР та обчислення оберненої матриці для знаходження оцінки розв'язку. В загальному випадку складність ітераційного методу Ньютона становить $O(n^3)$. З метою скорочення часу виконання однієї ітерації для розв'язування СЛАР застосуємо дрібно-плитковий гібридний алгоритм факторизації, де правою частиною СЛАР виступає $-f\left(x^{(k)}\right)$.

Дрібно-плитковий гібридний алгоритм факторизації розрідженої блочно-діагональної матриці з обрамленням

Розглянемо задачу розв'язання системи лінійних алгебраїчних рівнянь

$$\tilde{A}\tilde{x} = \tilde{b} \tag{5}$$

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

Ідеологічною передумовою використання гібридних обчислень при обробці розріджених матриць довільної структури є попереднє перевпорядкування структури вихідної матриці методом паралельних перерізів, що приводить матрицю до блочно-діагонального вигляду з обрамленням [2, 3].

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

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

де P – матриця перестановок, а блоки $D_{i\,i}$, C_{pi}, C_{ip}, $i = \overline{1, p-1}$, D_{pp}, зберігають розріджену структуру, p – кількість діагональних блоків у матриці, $p \ge 3$.

Природнім для методу паралельних перерізів є те, що розмір останнього діагонального блоку є набагато меншим ніж порядки інших діагональних блоків. Також структура блочно-діагональної матриці з обрамленням дозволяє проводити паралельну й незалежну обробку блоків $D_{i\,i}$, C_{pi}, C_{ip}, $i = \overline{1, p-1}$.

З огляду на структуру отримуваної матриці для зберігання даних доцільно використовувати блочний розподіл. Причому, кількість блоків слід вибирати з урахуванням кількості процесорів та графічних прискорювачів, які задіяні в розрахунках.

На рис. 1 показано профіль наповненості ненульовими елементами матриці. В результаті застосування методу паралельних перерізів структура матриці приймає форму показану на рис. 2.

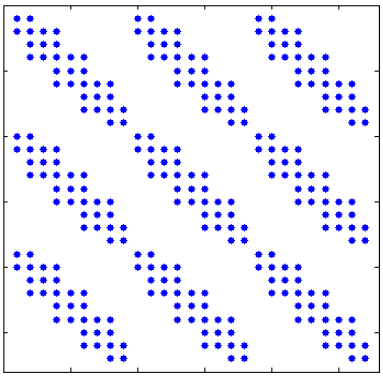

Рис. 1. Профіль оригінальної матриці

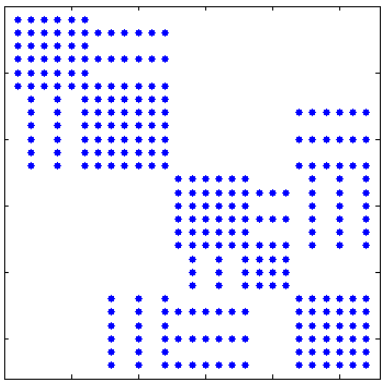

Рис. 2. Профіль блочно-діагональної матриці

Найбільш ефективним прямим методом розв'язання такої задачі є, як відомо, метод Холецького. Розв'язування системи полягає в розв'язуванні підзадач: трикутне розвинення матриці системи, розв'язування двох СЛАР з трикутними матрицями (7) та (8):

$$A = LL^T, \tag{6}$$

$$Ly = b, \tag{7}$$

$$L^T x = y. \tag{8}$$

Розглянемо дрібно-плитковий гібридний алгоритм факторизації розрідженої блочно-діагональної матриці з обрамленням A. Даний алгоритм краще враховує профільну, або розріджену структуру діагональних блоків, блоків обрамлення.

Розіб'ємо матрицю A на блоки розмірністю $c \times c$.

Далі для факторизації блочно-діагональної матриці застосуємо алгоритм запропонований в [4] для щільних матриць.

Для факторизації матриці на $к$-ому кроці використаємо наступне співвідношення:

$$A^к = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} = \begin{pmatrix} L_{11} & 0 \\ L_{21} & L_{22} \end{pmatrix} \begin{pmatrix} L_{11}^T & L_{21}^T \\ 0 & L_{22}^T \end{pmatrix},$$

де розміри блоків $A_{11} - c \times c$, $A_{21} - (n\text{-}kc)$c, $A_{22} - (n\text{-}kc)(n\text{-}kc)$, блоки A_{21} та A_{22} враховують структуру блоків $D_{i\,i}$, C_{pi}, D_{pp}.

Звідси отримаємо алгоритм, за яким проводиться розвинення на $к$ кроці:

$$A_{11} = L_{11} * L_{11}^T, \tag{9}$$

$$L_{21} = A_{21} * \left(L_{11}^T \right)^{-1}, \tag{10}$$

$$\tilde{A}_{22} = A_{22} - L_{21} * L_{21}^T. \tag{11}$$

Зазначимо, що реалізація (9)–(10) на кожному кроці модифікує тільки блоки D_{ii}, C_{pi}, $i = \overline{1, p-1}$, D_{pp}.

Для реалізації алгоритму будемо використовувати наступний розподіл даних: в GPU(i) $i = \overline{1, p-1}$ містяться блоки $D_{i\,i}$, C_{pi} та блок $A_{pp}^{(i)}$ того ж розміру, що й D_{pp}; в GPU(p) зберігається блок D_{pp}.

На рис. 3 показано блочний розподіл даних на k-му кроці факторизації блочно-діагональної матриці з обрамленням, враховуючи вище запропоновану декомпозицію.

Паралелізація обчислень трикутної факторизації полягає у тому, що розвинення блоків A_{11} та модифікація A_{21} та A_{22} може здійснюватися незалежно у кожному CPU(i) та GPU(i) $i = \overline{1, p-1}$.

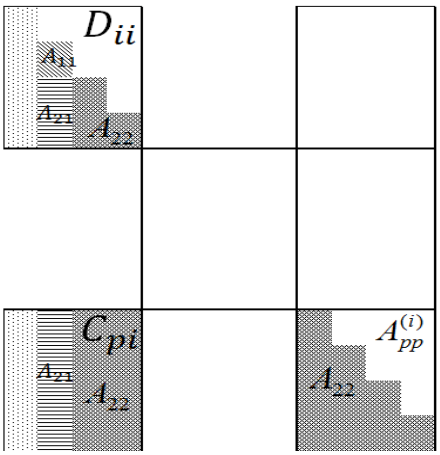

Рис. 3. Декомпозиція даних на GPU(i)

На кожному кроці у всіх парах CPU(i) та GPU(i) $i = \overline{1, p-1}$ виконуємо:

- у CPU(i), $i = \overline{1, p-1}$ факторизуємо A_{11} із $D_{i\,i}$:

$$A_{11} = L_{11} * L_{11}^T;$$

- у GPU(i), $i = \overline{1, p-1}$ модифікуємо стовпчик блоків L_{21}:

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

- у GPU(i), $i = \overline{1, p-1}$ модифікуємо блоки матриці A_{22} з $A_{pp}^{(i)}$ за формулою:

$$\tilde{A}_{22} = A_{22} - L_{21} L_{21}^T.$$

- У CPU(p), використовуючи мультизбирання, модифікуємо D_{pp}:

$$D_{pp}^* = D_{pp} - \sum_{i=1}^{p-1} A_{pp}^{(i)}.$$

Факторизуємо блок D_{pp}^*, тим самим завершуючи процес факторизації матриці.

Будемо вважати, що порядки всіх діагональних блоків приблизно рівні

$$q_i \approx q = \frac{n-s}{p-1},$$

де s – порядок останнього діагонального блоку. Тоді справедливі наступні теореми.

Теорема 1. Кількість операцій, що виконуються на одному GPU для знаходження факторизації розрідженої блочно-діагональної матриці з обрамленням оцінюється величиною

$$N_p \approx \frac{q^3}{3} + sq^2 = \frac{q^2}{3}(q+3s).$$

Доведення. Введемо наступне позначення $l = \frac{q}{c}$ кількість рядків плиток в діагональному блоці.

Оскільки (9)-(11) виконуються паралельно і незалежно у всіх p-1 парах CPU та GPU і максимальна кількість операцій припадає на етап (11), то оцінка кількості операцій виконуваних на одному GPU визначається саме складністю етапу (11).

Кількість операцій необхідних для виконання (11) можна оцінити величиною

$$N_p \approx 2c\left(\sum_{k=1}^{l}\frac{(q-k)^2}{2} + \sum_{k=1}^{l}(q-kc)s + l\frac{s^2}{2}\right). \tag{12}$$

Розпишемо перший доданок з (12)

$$2c\sum_{k=1}^{l}\frac{(q-k)^2}{2} = c\left(\sum_{k=1}^{l}(q-k)^2\right) = c\left(\sum_{k=1}^{l}q^2 - \sum_{k=1}^{l}2qkc + \sum_{k=1}^{l}k^2c^2\right). \tag{13}$$

Тут

$$\sum_{k=1}^{l}q^2 = lq^2, \quad \sum_{k=1}^{l}2qkc = 2qc\sum_{k=1}^{l}k = qcl^2, \quad \sum_{k=1}^{l}k^2c^2 = c^2\sum_{k=1}^{l}k^2 = \frac{c^2l^3}{3}.$$

Підставимо значення l у формули й отримаємо

$$\sum_{k=1}^{l}q^2 = \frac{q^3}{c}, \quad \sum_{k=1}^{l}2qkc = \frac{q^3}{c}, \quad \sum_{k=1}^{l}k^2c^2 = \frac{q^3}{3c}.$$

Звідси

$$2c\sum_{k=1}^{l}\frac{(q-k)^2}{2} = \frac{q^3c}{3c} = \frac{q^3}{3}. \tag{14}$$

Розпишемо другий доданок з (12).

$$2c\sum_{k=1}^{l}s(q-kc) = 2cs\left(\sum_{k=1}^{l}q + \sum_{k=1}^{l}kc\right) = 2cs\left(ql - \frac{l^2c}{2}\right) = 2cs\left(\frac{q^2}{c} + \frac{q^2c}{2c^2}\right) = 2cs\left(\frac{q^2}{2c}\right) = sq^2. \tag{15}$$

Останнім доданком у (12) можна знехтувати, оскільки порядок останнього діагонального блоку невеликий.

Підставивши (14) та (15) у (13) отримуємо

$$N_p \approx \frac{q^3}{3} + sq^2 = \frac{q^2}{3}(q+3s).$$

Теорема доведена.

Теорема 2. Прискорення дрібно-плиткового гібридного алгоритму LL^{T}-розвинення розрідженої блочно-діагональної матриці з обрамленням A, становить

$$S_p \approx (p-1)\left(1 + \frac{3(p-1)s^2}{2q^2(q+3s)}\left(\tau_{opp} + \left(\frac{2p}{(p-1)} + \frac{4qc}{s^2} + \frac{4c}{(p-1)s}\right)\tau_{opg}\right)\right)^{-1},$$

де c – розмір плитки.

Доведення. Для доведення теореми будемо використовувати методологію оцінки ефективності та прискорення гібридних алгоритмів, приведену в [5].

Знайдемо об’єм даних, що передаються між процесорами підчас виконання алгоритму. Оскільки використовується топологія комунікаційних зв’язків «кільце» і процесори обмінюються s^2 машинними

словами, сумарний об'єм даних, яким обмінюються процесори становить $\frac{(p-1)s^2}{2}$.

Перед виконанням операції мультизбирання необхідно у всіх парах CPU та GPU крім останньої виконати обмін даними об'ємом s^2. Також аналогічний обмін здійснюється і в останній парі CPU та GPU.

Також у процесі факторизації діагональних блоків D_{ii} всі пари CPU і GPU крім останньої обмінюються даними об'ємом $2qc$. Остання пара CPU і GPU обмінюється об'ємом даних рівним $2sc$.

Сумарний об'єм даних, яким обмінюються усі пари CPU та GPU, дорівнює $2qc(p-1)+ps^2+2sc$.

При обчисленні T_1 та T_p будемо використовувати величину N_p з теореми 1.

$$T_1 \approx \frac{(p-1)N_p}{n_o}t_g, \quad T_p \approx \frac{N_p}{n_o}t_g + \frac{(p-1)s^2}{2}t_{opp} + \left(2qc(p-1)+ps^2+2sc\right)t_{opg}.$$

Підставивши всі знайдені величини у формулу для обчислення коефіцієнта прискорення S_p, отримаємо

$$S_p \approx \frac{(p-1)\frac{N_p}{n_o}t_g}{\frac{N_p}{n_o}t_g + \frac{(p-1)s^2}{2}t_{opp} + \left(2qc(p-1)+ps^2+2sc\right)t_{opg}}.$$

Розділивши чисельник та знаменник на $\frac{N_p}{n_o}t_g$ отримуємо

$$S_p \approx \frac{(p-1)}{1+\frac{1}{N_p}\left(\frac{(p-1)s^2}{2}\tau_{opp} + \left(2qc(p-1)+ps^2+2sc\right)\tau_{opg}\right)}.$$

Винесемо за дужки $\frac{(p-1)s^2}{2}$

$$S_p \approx \frac{(p-1)}{1+\frac{3(p-1)s^2}{2q^2(q+3s)}\left(\tau_{opp} + \left(\frac{2p}{(p-1)}+\frac{4qc}{s^2}+\frac{4c}{(p-1)s}\right)\tau_{opg}\right)}.$$

Теорема доведена.

Згідно з вищенаведеними алгоритмами розроблено програми та проведено чисельні експерименти.

Чисельні експерименти

Обчислювальні експерименти з розв'язування СНР проводилися на вузлах суперкомп'ютера СКІТ [6] з наступними характеристиками:

- 2 центральні процесори Intel Xeon E5-2600 з частотою 2.6 ГГц;
- інтегрований із загальним сховищем даних кластерного комплексу обсягом 200 ТБ;
- мережа передачі даних між вузлами Infiniband FDR 56 Гбіт/с;
- 128 ГБ оперативної пам'яті;
- 3 прискорювачі NVidia Tesla M2075.

При заданому початковому наближенні x^0 =0,5 в області $D=\{-1000 \le x_i \le 1000\}$, $i=0,1,2,\ldots,n-1$ методом Ньютона розв'язувалась наступна система нелінійних рівнянь з блочною матрицею Якобі:

– перший блок містить такі *m* рівнянь:

$$4x_1^2 - x_2^2 - x_{m+1}^2 + x_{m+2} - 3 = 0,$$

$$-2x_{j-1}x_j + 5x_j^2 - x_{j+1}^2 + x_{m+j-1} - x_{m+j}^2 + x_{m+j+1} - 3 = 0, \quad j = 2, \ldots, m\text{-}1$$

$$-2x_{m-1}x_m + 5x_m^2 + x_{2m-1} - x_{2m}^2 - 3 = 0.$$

– наступні N-2 (K=2,…, N-1) блоки – по m рівнянь:

$$-2x_{Km-2m+j}x_{Km-m+j}+x_{Km-2m+2}+5x^2_{Km-m+j}-x^2_{Km-m+j+1}-x^2_{Km+j}+x_{Km+j+1}-3=0\,,\quad j=1,$$

$$x_{Km-2m+j-1}-2x_{Km-2m+j}x_{Km-m+j}+x_{Km-2m+j+1}-$$

$$-2x_{Km-2m+j}x_{Km-m+j}+6x^2_{Km-m+j}-x^2_{Km-m+j+1}+x_{Km+j-1}-x^2_{Km+j}+x_{Km+j+1}-4=0\,,\qquad j=2,\ldots,m-1,$$

$$x_{Km-m-1}-2x_{Km-m}x_{Km}-2x_{Km-1}x_{Km}+6x^2_{Km}+x_{Km+m-1}-x^2_{Km+m}-3=0\,;$$

– останні m рівнянь:

$$-2x_{(N-2)m+1}x_{(N-1)m+1}+x_{(N-2)m+2}+5x^2_{(N-1)m+1}-x^2_{(N-1)m+2}-3=0\,,$$

$$x_{(N-2)m+j-1}-2x_{(N-2)m+j}x_{(N-1)m+j}+x_{(N-2)m+j+1}-$$

$$-2x_{(N-1)m+j}x_{(N-1)m+j}+6x^2_{(N-1)m+j}-x^2_{(N-1)m+j+1}-3=0\,,j=2,\ldots,m-1$$

$$x_{(N-1)m-1}-2x_{(N-1)m}x_{Nm}-2x_{Nm-1}x_{Nm}+6x^2_{Nm}-3=0$$

де m – кількість рядків у блоці, N – кількість блоків. Тоді $N=\lceil(n-1)/m+1\rceil$ або n=Nm.

При програмній реалізації алгоритмів на комп'ютерах гібридної архітектури доцільно використовувати функції оптимізованих програмних бібліотек. Зокрема, до таких бібліотек відносяться Intel MKL [77], CUBLAS [8].

Для реалізації основних обчислювальних етапів алгоритмів використовуються наступні бібліотечні функції:

- dpotrf – знаходження LLT розвинення щільної матриці. Функція виконується на CPU;
- cublasDsyrk – проводить s-рангову модифікацію щільної матриці. Функція виконується на GPU;
- cublasDgemm – знаходить множення двох щільних матриць. Функція виконується на GPU;
- cublasDtrsm – розв'язання трикутної системи з багатьма правими частинами. Функція виконується на GPU.

Для виконання комунікацій в алгоритмі використовуються функції з бібліотек MPI [9] та CUDA [10]:

- Mpi_reduce – функція глобальної редукції зі збереженням результату у вказаному процесорі;
- cudaMemcpyAsync – функція, що виконує асинхронне копіювання даних між CPU та GPU. Запуск функції у кількох потоках cudaStream, дозволяє скоротити час виконання програми, оскільки копіювання виконуються на фоні обчислень і не викликає зупинки в обчисленнях.

Обчислення на GPU також одночасно виконуються в різних потоках cudaStream.

Наведемо деякі з отриманих результатів.

В таблиці наведено часи (сек.) виконання однієї ітерації методу Ньютона з використанням паралельного варіанту (8 CPU) дрібно-плиткового алгоритму факторизації блочно-діагональної матриці з обрамленням. Розмір плитки – 128.

Таблиця

Порядок матриці / кількість потоків	15000	20000	25000
1	0.184481	0.2459	0.3074
2	0.16771	0.2236	0.2795
3	0.0819	0.1108	0.1336
4	0.0631	0.0842	0.0988
5	0.0519	0.0630	0.0772
6	0.0445	0.0539	0.0637
7	0.0393	0.0474	0.0540
8	0.0354	0.0427	0.0468

З таблиці можна зробити висновок, що зростання порядку системи, яка розв'язується та перетворення матриці для різної кількості процесорів сприяє рівномірній завантаженості процесорів обчисленнями. Ще одним фактором, який сприяє такому скороченню часу розрахунків є регулювання розміру плитки. При правильному виборі розміру плитки може досягатись ефект кешизації обчислень, що може проявлятись як суперприскорення на певних конфігураціях параметрів.

На рис. 4. показано залежність часу виконання розрахунків гібридного варіанта (8 CPU + 1 GPU) дрібно-плиткового алгоритму від кількості потоків та розміру плитки. Порядок матриці – 10050.

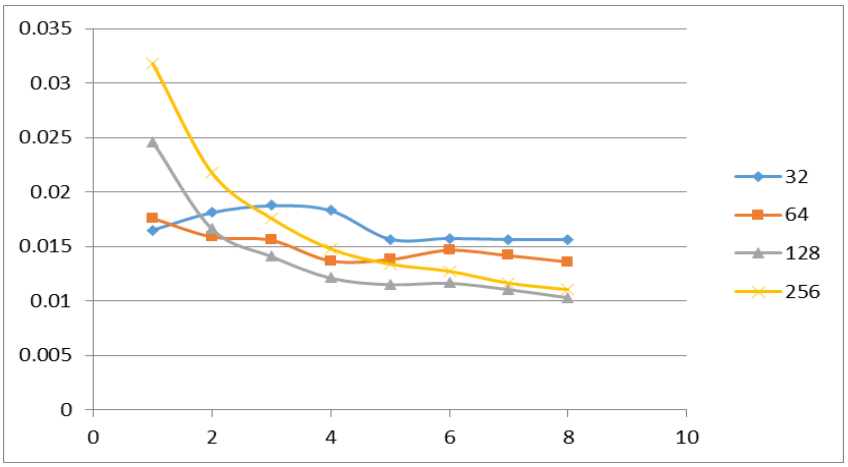

Рис. 4. Час розрахунків на GPU, залежно від розміру плитки та кількості потоків

Висновки

У роботі розглянуто новий метод розв'язання систем нелінійних рівнянь великих порядків з блочними матрицями Якобі. Його ідеологічною основою є поєднання класичного алгоритму методу Ньютона з ефективним дрібно-плитковим алгоритмом розв'язання систем лінійних рівнянь з розрідженими матрицями. Приведено часи розв'язування СНУ різних порядків на вузлах суперкомп'ютера СКІТ. Отримано суттєве скорочення часу розв'язування СНУ. Цей алгоритм дозволяє регулювати розмірність блоку з яким проводяться обчислення на кожному кроці алгоритму, за рахунок цього може досягатись ефект кешезації обчислень, коли блоки повністю розміщуються у 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.

Література

1. Нестеренко А.Н., Химич А.Н., Яковлев М.Ф. Некоторые вопросы решения систем нелинейных уравнений на многопроцессорных вычислительных системах с распределенной памятью. *Вестник компьютерных и информационных технологий*. М.: 2006. № 10. С. 54–56.
2. Джордж А., Лю Дж. Численное решение больших разреженных систем уравнений. М.: Мир. 1984. 334 с.
3. Писанецки С. Технология разреженных матриц. М.: Мир. 1988. 410 с.
4. Alfredo Buttari, Julien Langou, Jakub Kurzak, and Jack Dongarra: A Class of Parallel Tiled Linear Algebra Algorithms for Multicore Architectures. Parallel Computing. 2009. Vol. 35. Issue 1. P. 38-53.
5. Попов О.В. Дослідження ефективності паралельних алгоритмів для комп'ютерів гібридної архітектури. Матеріали Міжнародної наукової школи-семінару «*Питання оптимізації обчислень* (*ПОО XLII*)», (Закарпатська обл., Мукачівський р.-н, смт. Чинадієво, 21–25 вересня 2015), Київ: 2015, С. 16.
6. Режим доступу: http://icybcluster.org.ua
7. Intel® Math Kernel Library (Intel® MKL) – Режим доступу: https://software.intel.com/en-us/intel-mkl
8. CUDA CUBLAS_Library – Santa Clara: Nvidia, 2010. – 254 с.
9. Gropp W., Lusk E., and Thakur R. Using MPI-2: Advanced Features of the Message-Passing Interface. Cambridge: MIT Press, 1999. 382 p.
10. Боресков А.В., Харламов А.А. Основы работы с технологией CUDA. М.: ДМК Пресс. 2010. 232 с.

References

1. Nesterenko, A.N. & Khimich, A.N. & Yakovlev, M.F. (2006) To the problem of solving of non-linear systems on multi-processor distributed memory computing system. *Gerald of computer and information technologies*. 10. pp. 54-56.
2. Dzhordzh A., Lyu Dzh. Chyslennoe reshenye bol′shykh razrezhennykh system uravnenyy. M.: Myr, 1984. 334 p.
3. Pysanetsky S. Tekhnolohyya razrezhennykh matryts. M.: Myr, 1988. 410 p.
4. Alfredo Buttari, Julien Langou, Jakub Kurzak, and Jack Dongarra: A Class of Parallel Tiled Linear Algebra Algorithms for Multicore Architectures. Parallel Computing. 2009. Vol. 35. Issue 1. P. 38-53.

5. Popov O.V. Doslidzhennya efektyvnosti paralel′nykh alhorytmiv dlya komp″yuteriv hibrydnoyi arkhitektury. Materialy Mizhnarodnoyi naukovoyi shkoly-seminaru «Pytannya optymizatsiyi obchyslen′ (POO XLII)», (Zakarpat·s′ka obl., Mukachivs′kyy r.-n, smt. Chynadiyevo, 21–25 veresnya 2015), Kyyiv: 2015, P. 16.
6. Rezhym dostupu: http://icybcluster.org.ua
7. Intel® Math Kernel Library (Intel® MKL) – Rezhym dostupu: https://software.intel.com/en-us/intel-mkl
8. CUDA CUBLAS_Library – Santa Clara: Nvidia, 2010. – 254 c.
9. Gropp W., Lusk E., and Thakur R. Using MPI-2: Advanced Features of the Message-Passing Interface. Cambridge: MIT Press, 1999. 382 p.
10. Boreskov A.V., Kharlamov A.A. Osnovy raboty s tekhnolohyey CUDA. M.: DMK Press, 2010. 232 p.

Одержано 06.03.2020

Про авторів:

Хіміч Олександр Миколайович,
доктор фізико-математичних наук, професор,
заступник директора.
Кількість наукових публікацій в українських виданнях – понад 150.
Кількість наукових публікацій в зарубіжних виданнях – понад 10.
https://orcid.org/0000-0001-9284-139X,

Сидорук Володимир Антонович,
кандидат фізико-математичних наук,
старший науковий співробітник.
Кількість наукових публікацій в українських виданнях – 30.
http://orcid.org/0000-0003-0210-6020,

Нестеренко Алла Никифорівна,
молодший науковий співробітник.
Кількість наукових публікацій в українських виданнях – 32.
https://orcid.org/0000-0001-6174-1812.

Місце роботи авторів:

Інститут кібернетики імені В.М. Глушкова НАН України,
проспект Академіка Глушкова, 40,
03187, Україна, Київ-187.
Тел. (066)-367-85-40.
E-mail: alla.nest1958@gmail.com