КОМПЬЮТЕРНАЯ МАТЕМАТИКА

УДК 519.6

О.М. ХІМІЧ, О.А. НІКОЛАЄВСЬКА, Т.В. ЧИСТЯКОВА

ПРО АЛГОРИТМ МЕТОДУ НАЙМЕНШИХ КВАДРАТІВ ДЛЯ РОЗВ'ЯЗУВАННЯ ЛІНІЙНИХ СИСТЕМ НА ГІБРИДНИХ КОМП'ЮТЕРАХ

Розглядається гібридний алгоритм методу найменших квадратів на основі сингулярного (SVD) розвинення матриці для розв'язування СЛАР з матрицями довільного рангу на багатоядерних комп'ютерах з графічними процесорами.

Вступ. Математичне моделювання процесів та явищ в різних предметних областях часто зводиться до розв'язування систем лінійних алгебраїчних рівнянь (СЛАР) великих розмірів. На сьогоднішній день ця проблема вирішується використанням гібридних комп'ютерів, які поєднують MIMD- і SIMD-архітектури, тобто реалізуються обчислення на багатоядерних комп'ютерах (CPU) з прискоренням на графічних процесорах (GPU) [1]. Проте величезний розрив між високою продуктивністю графічних процесорів і повільними комунікаційними зв'язками між CPU і GPU, а також принциповими архітектурними відмінностями цих обчислювальних пристроїв, значно ускладнює розробку ефективних гібридних алгоритмів. Тільки за умови правильного використання обчислювальних ресурсів можна створити ефективні гібридні алгоритми розв'язування СЛАР. Під ефективним алгоритмом розв'язування задачі розуміється алгоритм, який дозволяє отримати достовірний розв'язок з мінімальним використанням ресурсів комп'ютера – процесорів, пам'яті, часу.

Як відомо, комп'ютерна модель задачі має завжди наближений щодо вихідної задачі характер або через похибку вихідних даних, або через похибку отримання (вводу) числових даних про задачу в комп'ютері [2]. Тому властивості лінійної системи, яку доводиться розв'язувати на комп'ютері, можуть відрізнятися від властивостей математичної задачі. Виникає необхідність створення алгоритму розв'язування СЛАР з апріорі невизначеними властивостями та матрицями великих розмірів.

В роботі розглядається алгоритм методу найменших квадратів на основі сингулярного (*SVD*) розвинення матриць для розв'язування СЛАР з матрицями довільного рангу на комп'ютерах гібридної архітектури.

1. Архітектурні особливості CPU і GPU, які впливають на ефективність гібридного алгоритму розв'язування СЛАР [3]. Комп'ютер MIMD-архітектури, який входить до складу гібридної архітектури, складається з обчислювальних вузлів, кожен з яких може мати декілька багатоядерних процесорів та свою локальну пам'ять, що розподіляється між цими процесорами. На кожному ядрі процесора виконується один потік послідовних інструкцій. Кожен процесор має доступ тільки до локальної пам'яті свого вузла. Обчислювальні вузли об'єднуються між собою деяким комунікаційним середовищем, тому міжпроцесорні операції за тривалістю виконання значно перевершують арифметичні операції. Отже, необхідно передбачити найбільш ефективну топологію міжпроцесорних зв'язків та рівномірне завантаження процесорів.

Графічний процесор влаштований принципово інакше. Множина великої кількості потоків команд виконується одночасно, забезпечуючи масовий паралелізм обчислень та їх високу швидкодію на GPU. Причому ніяких накладних витрат у графічному процесорі немає. Використати обчислювальні можливості GPU у повній мірі можна лише в тому випадку, коли задача розпаралелюється на сотні виконуючих потоків, тобто коли одна і та ж послідовність математичних операцій застосовується до великого обсягу даних. Наприклад, виконання матрично-матрично та матрично-векторних операції доцільно виконувати на GPU.

GPU відрізняється від CPU також за принципами доступу до пам'яті. На GPU є 6 видів пам'яті, кожна з яких має своє призначення. Доцільно враховувати призначення кожного виду пам'яті при здійсненні комунікаційних зв'язків між CPU і GPU, які є надто повільними у порівнянні з виконанням обчислень.

Виникає необхідність розподілу задачі на окремі підзадачі з метою визначення: пріоритетів підзадач, на CPU чи GPU їх краще реалізовувати за часом виконання, які підзадачі можна виконати паралельно, в якій послідовності їх треба виконувати тощо.

2. Постановка задачі. Розглянемо задачу розв'язування СЛАР з виродженою квадратною або прямокутною матрицею у загальному випадку довільного рангу за допомогою гібридного паралельного алгоритму методу найменших квадратів на основі *SVD* розвинення матриці [2].

При введенні несумісної у загальному випадку СЛАР з точно заданими вихідними даними (математична модель з точно заданими вихідними даними):

$$Ax = b,$$
$$A \in R^{m \times n},\ x \in R^n,\ b \in R^m \tag{1}$$

в комп'ютері отримуємо систему з наближеними даними (машинна модель з наближеними даними):

$$\overline{A}\overline{x} = \overline{b}\ , \qquad \Delta A = A - \overline{A}, \qquad \Delta b = b - \overline{b}\ . \tag{2}$$

Будемо вважати, що для похибок елементів матриці та правої частини виконуються наступні співвідношення:

$$\|\Delta A\| \le \varepsilon_A \|A\|, \qquad \|\Delta b\| \le \varepsilon_b \|b\|.$$

Тут і в подальшому, якщо не обумовлено додатково, використовуються евклідова векторна та підпорядкована їй спектральна матрична норми.

Властивості машинної моделі задачі можуть відрізнятися від властивостей вихідної задачі, оскільки елементи матриці та правих частин в комп'ютері мають наближений характер. Тобто задача, яку необхідно розв'язати в комп'ютері, – це задача з апріорі невідомими властивостями, отже необхідно спочатку дослідити її властивості. Фундаментальну роль при дослідженні задачі найменших квадратів відіграє визначення повноти рангу в межах машинної точності та повноти рангу в межах завдання вихідних даних, а також визначення ефективного рангу матриці (δ-рангу).

Комп'ютерний алгоритм дослідження повноти рангу зводиться до перевірки двох співвідношень:

$$1.0 + \frac{1}{h(\bar{A})} \neq 1.0, \tag{3}$$

$$\varepsilon_{\bar{A}} h(\bar{A}) < 1, \tag{4}$$

де $h(\bar{A}) = \|\bar{A}\| \, \|\bar{A}^{+}\|$ – число обумовленості матриці $\bar{A}$.

Перша умова (3), яка виконується в арифметиці з плаваючою комою, означає, що матриця має повний ранг у межах машинної точності, а друга (4) – що вона є повного рангу також у межах точності задання вихідних даних.

Таку машинну задачу слід розглядати як коректно поставлену в межах точності задання вихідних даних.

Ефективний ранг матриці визначає номер стійкої проекції нормального псевдорозв'язку задачі найменших квадратів з наближеними вихідними даними. Ця проекція апроксимує або нормальний псевдорозв'язок або його проекцію. Отже, для знаходження δ-рангу необхідно знайти величину r, яка дорівнює найбільшому значенню i, для якого виконується нерівність:

$$\frac{\delta}{\mu_i} < 1, \quad \mu_i \neq 0, \; i = 1, 2...,$$

де μ_i – сингулярні числа матриці A.

3. Алгоритм методу найменших квадратів на основі сингулярного розвинення матриці. В загальному випадку задача розв'язування СЛАР з матрицею довільного рангу є некоректною, а система в багатьох випадках – несумісною. Для таких СЛАР можна отримати розв'язок методом найменших квадратів (псевдорозв'язок), який для сумісної і коректної задачі збігається з класичним розв'язком [2]:

$$A = U\text{р}V^{\mathrm{T}}, \tag{5}$$

яке виконується за допомогою послідовності двосторонніх ортогональних перетворень Хаусхолдера та QR-алгоритма з неявним зсувом.

Узагальнений розв'язок отримується за формулами:

$$x^{\#} = V(\Sigma^{\#} c), \quad c = U^{T} b, \tag{6}$$

де $\Sigma^{\#}$ – псевдообернена матриця сингулярних чисел.

При обчисленні за допомогою сингулярного розвинення (5) псевдорозв'язку (6) СЛАР з прямокутною матрицею довільного рангу розміру $m \times n$ ($m \ge n$) можна виділити п'ять підзадач:

1) приведення вихідної матриці до верхньої дводіагональної форми $A^{(n-1)} \equiv \begin{pmatrix} J^{(0)} \\ \hline 0 \end{pmatrix}$, де $J^{(0)}$ – квадратна верхня дводіагональна матриця порядку n, в якій відмінні від нуля лише елементи $j_{k,k}^{(0)},\ j_{k,k+1}^{(0)}$;

2) сингулярне розвинення верхньої дводіагональної матриці $J^{(0)}$, тобто приведення $J^{(0)}$ до діагональної матриці Σ;

3) формування матриці правих сингулярних векторів V;

4) формування матриці (вектора) c з (6);

5) обчислення псевдорозв'язку (6).

Для приведення прямокутної матриці $A^{(0)}$ розміру $m_1 \times n_1$ ($m_1 = \max\{m, n\}$, $n_1 = \min\{m, n\}$, $A^{(0)} \equiv A$, якщо $m \ge n$, або $A^{(0)} \equiv A^T$, якщо $m < n$), до верхньої дводіагональної форми можна використати алгоритм Хаусхолдера. При цьому необхідно виконати $n_1 - 1$ двосторонніх елементарних перетворень відображення:

$$A^{(i)} = P^{(i)} A^{(i-1)} Q^{(i)} \quad (i = 1, 2,..., n_1-1), \tag{7}$$

де ортогональні матриці $P^{(i)} = I + s_i u_i u_i^T$, $Q^{(i)} = I + t_i v_i v_i^T$, а вектори u_i, v_i та множники s_i, t_i визначаються так, щоб для кожного $i = 1, 2,...,\ n_1 - 1$ виконувались умови:

$$s_i u_i^T u_i = t_i v_i^T v_i = 2, \tag{8}$$

$$a_{k,j}^{(i)} = a_{j,k}^{(i)} = 0 \quad (k = i+1,\dots, m_1; \quad j = i+2,\dots, n_1). \tag{9}$$

Двосторонні перетворення (7) з умовами (8), (9) можна записати наступним чином:

$$A^{(i)} = A^{(i-1)} + u_i y_i^T + z_i v_i^T, \quad (i = 1, 2,\dots, n-1), \tag{10}$$

де

$$u_i = \left(0,\dots,0, a_{i,i}^{(i-1)} - d_i, a_{i,i+1}^{(i-1)},\dots, a_{i,m}^{(i-1)}\right)^T,$$

$$v_i = g_i - f_{i+1} e_{i+1}, \quad y_i = w_i + c_i v_i, \quad z_i = t_i x_i - c_i u_i,$$

$$g_i^T = \frac{1}{d_i s_i} w_i^T + e_i^T A^{(i-1)}, \quad w_i^T = s_i u_i^T A^{(i-1)}, \quad x_i = A^{(i-1)} v_i,$$

$$d_i = -\mathrm{sign}(a_{i,i}^{(i-1)})\ {}_i, \quad f_{i+1} = -\mathrm{sign}(g_{i,i+1}) \cdot {}_i \tag{11}$$

$$c_i = 0{,}5 t_i w_i^T v_i, \quad s_i = \left(d_i a_{i,i}^{(i-1)} - \mathrm{i}^2\right)^{-1}, \quad t_i = \left(f_{i+1} g_{i,i+1} - \cdot\ {}_i^2\right)^{-1},$$

$$\mathrm{i}^2 = \sum_{j=i}^{m} \left(a_{i,j}^{(i-1)}\right)^2, \quad \psi_i^2 = \sum_{j=i+1}^{n} \left(g_{i,j}\right)^2,$$

e_i – одиничний орт.

Таким чином, на кожному кроці (для кожного i) модифікується розташована в правому нижньому кутку підматриця розміру (mJ i + 1) × (n J i + 1), за виконанням умов (8), (9), а елементи $d_i \equiv a_{i,i}^{(i)}$ та $f_{i+1} \equiv a_{i,i+1}^{(i)}$ є відповідно діагональним та позадіагональним елементами дводіагональної матриці $J^{(0)}$.

Для сингулярного розвинення верхньої дводіагональної матриці $J^{(0)}$ за сингулярними числами використовується QR-алгоритм з неявним зсувом. Цей алгоритм полягає у побудові послідовності верхніх дводіагональних матриць $J^{(0)}, J^{(1)}, \ldots, J^{(N)}, \ldots$, які збігаються до діагональної матриці Σ. Перехід від $J^{(k-1)}$ до $J^{(k)}$ виконується за допомогою серії двосторонніх плоских обертань Гівенса:

$$J^{(k)} = S^{(k)T} J^{(k-1)} T^{(k)}, \tag{12}$$

де $S^{(k)} = S_2^{(k)} S_3^{(k)} \cdots S_n^{(k)}$, $T^{(k)} = T_2^{(k)} T_3^{(k)} \cdots T_n^{(k)}$, а $S_j^{(k)}, T_j^{(k)}$ – елементарні матриці плоских обертань. В результаті отримуємо сингулярне розвинення дводіагональної матриці:

$$J^{(0)} = G\tilde{\tilde{\Sigma}} H^T, \qquad \tilde{\tilde{\Sigma}} \equiv J^{(N)}, \tag{13}$$

$$G = S^{(1)} S^{(2)} \cdots S^{(N)}, \quad H = T^{(1)} T^{(2)} \cdots T^{(N)}. \tag{14}$$

Формування матриці правих (або лівих) сингулярних векторів проводиться шляхом накопичення елементарних перетворень відображення:

$$Q = Q^{(1)} Q^{(2)} \cdots Q^{(n-1)} \text{ або } P = P^{(1)} P^{(2)} \cdots P^{(n-1)} \tag{15}$$

та обертань (14). Тоді наближеними правими сингулярними векторами вихідної матриці є стовпчики матриці $V = QH$, а лівими – стовпчики матриці $U = PG$.

Аналогічним чином проводиться формування матриці (вектора) c з (4) – спочатку у відповідності з формулами:

$$b^{(i)} = P^{(i)} b^{(i-1)} \ (m \geq n) \text{ або } b^{(i)} = Q^{(i)} b^{(i-1)} \ (m < n), \tag{16}$$

$i = 1, 2, \ldots, n_1 - 1$, формується $b^{(n-1)}$ або $b^{(m-1)}$, а потім обчислюється:

$$c = G^T b^{(n-1)} \text{ або } c = H^T b^{(m-1)}. \tag{17}$$

Одночасно з обчисленням матриці $J^{(0)}$ за формулою (14) формується прямокутна матриця (вектор) $b^{(n-1)}$ (або $b^{(m-1)}$, якщо $m < n$). Формулу (16) для $i = 1, 2, \ldots, \min(m, n) - 1$ можна записати наступним чином:

$$b^{(i)} = b^{(i-1)} + u_i h_i^T, \quad h_i^T = s_i u_i^T b^{(i-1)} \quad (m \geq n) \tag{18}$$

або

$$b^{(i)} = b^{(i-1)} + v_i h_i^T, \quad h_i^T = t_i v_i^T b^{(i-1)} \quad (m < n). \tag{19}$$

Тобто, для кожного i модифікується нижня підматриця розміру $(m\text{J}i+1)\times q$.

Формування матриці елементарних перетворень відображень Q з (15) можна виконати так:

$$Q_{(i)} = Q^{(k)} Q_{(i-1)} \equiv Q_{(i-1)} + v_k x_k^T \quad (i = 1, 2, \ldots, n-1), \tag{20}$$

де $Q_{(0)} \equiv I$, $Q \equiv Q_{(n-1)}$,

$$x_k = t_k Q_{(i-1)}^T v_k, \quad t_k = \left(f_{k+1} v_{k,k+1}\right)^{-1} \tag{21}$$

вектор відображення v_k визначено в (11), $f_{k+1} \equiv j_{k,k+1}^{(0)}$ – наддіагональний елемент дводіагональної матриці $J^{(0)}$ ($k = n - i$).

Формування матриці елементарних перетворень відображення P з (15), яка обчислюється при $m < n$, проводиться аналогічно.

4. Гібридний паралельний алгоритм Хаусхолдера зведення прямокутної матриці до верхньої дводіагональної форми на комп'ютерній архітектурі з декількома CPU та GPU. При реалізації гібридного паралельного алгоритму Хаусхолдера для зведення прямокутної матриці $A^{(0)}$ розміру $m \times n$ до верхньої дводіагональної форми $A^{(n-1)}$ застосовується стрічково-циклічний розподіл матриці СЛАР між процесами CPU, за яким процесу з логічним номером $k-1$ послідовно розподіляються елементи рядків з номерами $k, k+p, k+2p, \ldots$, де $k = 1, 2, \ldots, p$, p – кількість процесів, що використовується. Розпаралелювання на CPU виконується за допомогою MPI-функцій. При такому розташуванні матриці на процесах простій процесорів (ядер) CPU буде мінімальний, адже обчислення в кожному з них відбуватимуться майже до кінця модифікації матриці ^. Отже, такий підхід забезпечить рівномірне завантаження (балансування) процесів. Зрозуміло, що необхідно передбачити виконання обчислень на процесах таким чином, щоб наявність інформаційних зв'язків (комунікаційних взаємодій) між ними була мінімальною.

4.1. Алгоритм Хаусхолдера на гібридних комп'ютерах з декількома CPU та GPU складається з виконання для $i = 1, 2, \ldots, n-1$ послідовності таких підзадач:

1) для знаходження значення φ_i та вектора w_i з (11) дані пересилаються з CPU на GPU і кожним процесом обчислюються масиви останніх nJ i+1 та q елементів векторів часткових сум добутків транспонованого i-го вектор-стовпчика матриці $A^{(i-1)}$ на прямокутні матриці (відповідно) $A^{(i-1)}$ та $b^{(i-1)}$, елементи яких розподілені по процесах за стрічково-циклічною схемою;

2) мультизбірка реалізується провідним процесом на CPU (в якому розташовані i-ті рядки прямокутних матриць) векторів добутків транспонованого i-го вектор-стопчика матриці $A^{(i-1)}$ на прямокутні матриці $A^{(i-1)}$ та $b^{(i-1)}$ з обчислених кожним процесом векторів часткових сум;

3) за допомогою операції мультирозсилки масив елементів $a_{i,i}^{(i-1)}, a_{i,i+1}^{(i-1)}, \ldots,$ $\ldots, a_{i,n}^{(i-1)}$ i-го рядка матриці, останні nJ i + 1 та q елементів векторів добутків транспонованого i-го вектор-стовпчика матриці $A^{(i-1)}$ на прямокутні матриці $A^{(i-1)}$ та $b^{(i-1)}$ розсилаються всім процесам для обчислення на відповідних GPU;

4) одночасно всіма процесами обчислюються значення φ_i та $d_i \equiv a_{i,i}^{(i)}$ з (11), формується лівий вектор відображення u_i та після цього обчислюється значення s_i з (11), а потім формуються останні nJ i елементів векторів w_i та g_i у відповідності з (11) та вектор h_i при $m \geq n$;

5) одночасно всіма процесами обчислюються значення ψ_i та $f_{i+1} \equiv a_{i,i+1}^{(i)}$ з (11), формується правий вектор відображення v_i, а після цього обчислюється значення t_i та c_i і потім формуються останні nJ i елементів вектора y_i;

6) кожним процесом у відповідності зі стрічково-циклічною схемою розподілення прямокутних матриць та з формулами (11) обчислюються останні mJ i елементів векторів x_i та z_i, а також вектор h_i з (19) при $m < n$;

7) кожним процесом, у відповідності зі стрічково-циклічною схемою розподілення прямокутних матриць, виконується модифікація (10) розташованої в правому нижньому кутку підматриці розміру $(m - i) \times (n - i)$ матриці $A^{(i-1)}$, а також модифікація (18) або (19) нижньої підматриці розміру $(n - i + 1) \times q$ матриці $b^{(i-1)}$.

Після виконання всіх цих операцій на GPU результати обчислень пересилаються на CPU у провідний процес і для всіх $i = 1, 2,\ldots, n - 1$ усім процесам має бути розісланий елемент $a_{n,n}^{(n-1)}$.

Таким чином, в результаті виконання всіх операцій алгоритму кожним процесом будуть сформовані два n-мірних масиви діагональних та наддіагональних елементів дводіагональної матриці $J^{(0)}$.

В даному алгоритмі реалізація обчислень на GPU здійснюється за допомогою технології CUDA. Через те, що на гібридних комп'ютерах дуже повільні зв'язки між CPU та GPU, матриці та вектори в алгоритмі розподілені таким чином, щоб за один раз переслати всі необхідні масиви на глобальну пам'ять GPU, виконати з ними необхідні матрично-векторні операції, розпаралелюючи на GPU, і отримані результати переслати на CPU. Оскільки на GPU обчислення розпаралелюються на дуже велику кількість потоків, це дає можливість ефективно виконати матрично-векторні операції з великою кількістю даних, забезпечуючи при цьому високу швидкодію. Крім того, швидкодію операцій можна підвищити застосовуючи відповідні функції бібліотеки CUBLAS [4].

Пересилка блоків матриць між CPU виконується за допомогою MPI-функцій. Матрично-векторні операції виконуються на GPU при використанні відповідних функцій бібліотеки CUBLAS.

4.2. Паралельний *QR*-алгоритм. Всі діагональні та наддіагональні елементи верхньої дводіагональної матриці $J^{(0)}$ зберігаються в кожному процесі. Також в кожному процесі зберігаються всі обчислені наближені сингулярні числа.

Оскільки окреме (елементарне) плоске обертання, яке використовується при обчисленні матриць сингулярних векторів (14), складається з попарних перетворень лише двох елементів (належать стовпчикам, що модифікуються) кожного рядка, а параметри плоского обертання обчислюються кожним процесом, то елементи матриці сингулярних векторів, що формується, та матриці c^T з (17) розподіляються за стрічково-циклічною схемою. В цьому випадку потреби в обмінах даними між процесами при формуванні цих матриць немає.

Для виконання паралельного *QR*-алгоритму на кожній ітерації $s = 1, 2, \ldots$ виконуються такі операції:

- одночасно кожним процесом, без обмінів даними між ними, обчислюється значення зрушення k_s, послідовно для $j = 1, 2, \ldots, n_s < n$ формуються матриці плоских обертань $S_j^{(s)}, T_j^{(s)}$ та модифікується дводіагональна матриця $J^{(s-1)}$ у відповідності з (12);

- одночасно кожним процесом, без обмінів даними, у відповідності зі схемою розподілення модифікуються елементи матриці сингулярних векторів, що формується, та матриці c^T з (17);
- одночасно кожним процесом, без обмінів даними, перевіряється умова досягнення заданої точності обчислення чергового наближеного сингулярного значення.

Після цього результати обчислень збираються в одному процесі на CPU.

5. Експериментальне дослідження гібридного алгоритму. Проведено дослідження ефективності гібридного алгоритму для розв'язування СЛАР з виродженими матрицями різного порядку та при використанні різної кількості CPU та GPU на комп'ютері Інпарком_g гібридної архітектури [5] (рисунок).

На графіку видно, що розроблений гібридний алгоритм добре масштабований, тобто зі збільшенням кількості процесів прискорення обчислень зростає, для великих порядків матриці найкраще прискорення досягається при використанні 8 процесів. У перспективі можливе покращення алгоритму завдяки динамічному розпаралелюванню обчислень між процесорами GPU, яке дає можливість пересилати дані безпосередньо від одного GPU до іншого GPU, оминаючи CPU. Така можливість реалізується при використанні версій CUDA 5 та вище на гібридних комп'ютерах з графічними процесорами на базі архітектури NVIDIA Kepler.

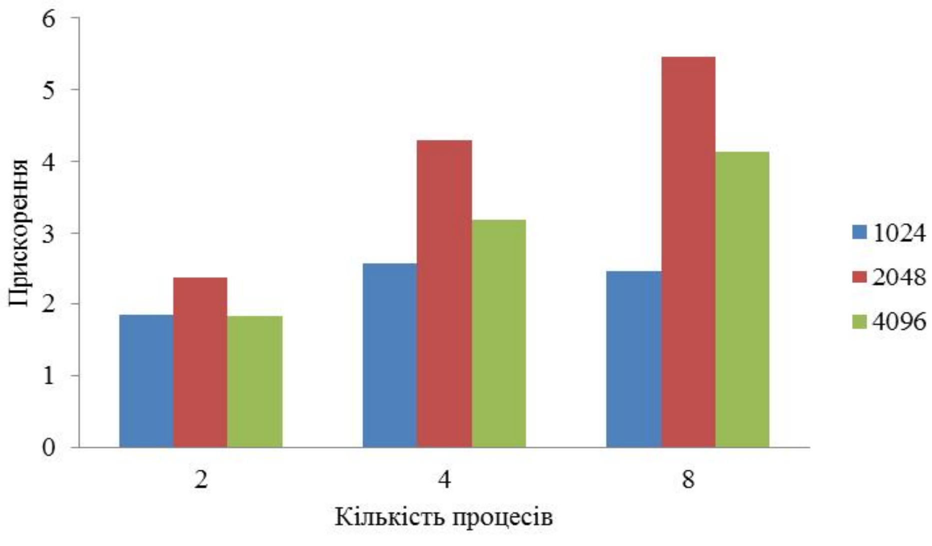

РИСУНОК. Залежність прискорення від кількості процесів та порядку матриці розв'язування задачі гібридним алгоритмом *SVD* на Інпарком_g

А.Н. Химич, Е.А. Николаевская, Т.В. Чистякова

ОБ АЛГОРИТМЕ МЕТОДА НАИМЕНЬШИХ КВАДРАТОВ
ДЛЯ РЕШЕНИЯ ЛИНЕЙНЫХ СИСТЕМ НА ГИБРИДНЫХ КОМПЬЮТЕРАХ

Рассматривается гибридный алгоритм метода наименьших квадратов на основе сингулярного (SVD) разложения матрицы для решения СЛАУ с матрицами произвольного ранга на многоядерных компьютерах с графическими процессорами.

A.N. Khimich, E.A. Nikolaevskaya, T.V. Chistyakova

ABOUT THE METHOD OF LEAST SQUARES ALGORITHM FOR SOLVING LINEAR SYSTEMS ON HYBRID COMPUTERS

The hybrid algorithm of the least squares method based on SVD of a matrix for solving linear systems with matrices of arbitrary rank on multicore computers with GPUs is considered.

1. *Боресков А.В., Харламов А.А.* Основы работы с технологией CUDA. – М.: Пресс, 2010. С. 232.
2. *Молчанов И.Н.* Машинные методы решения прикладных задач. Алгебра, приближение функций. Київ: Наук. думка, 1987. 288 с.
3. *Химич А.Н., Молчанов И.Н., Попов А.В. и др.* Параллельные алгоритмы решения задач вычислительной математики. Київ: Наук. думка, 2008. 248 с.
4. *CUBLAS:* http://www.nvidia.com.ua/object/tesla-gpu-accelerated-libraries-cublas-ru.html
5. *Молчанов И.М., Мова В.И.* Интеллектуальные параллельные компьютеры на графических процессорах для решения научно-технических задач. *Праці Міжнародної молодіжної математичної школи «Питання оптимізації обчислень (ПОО-XXXVII).* Київ: Інститут кібернетики імені В.М. Глушкова НАН України, 2011. С. 121 – 122.

Одержано 11.10.2016

Про авторів:

Хіміч Олександр Миколайович,
доктор фізико-математичних наук, профессор, член-кор. НАН України,
завідуючий відділом Інституту кібернетики імені В.М. Глушкова НАН України,
E-mail: khimich505@gmail.com

Ніколаєвська Олена Анатоліївна,
кандидат фізико-математичних наук, науковий співробітник
Інституту кібернетики імені В.М. Глушкова НАН України,
E-mail: elena_nea@ukr.net

Чистякова Тамара Василівна,
кандидат фізико-математичних наук, старший науковий співробітник
Інституту кібернетики імені В.М. Глушкова НАН України.
E-mail: tamara.chistjakova@gmail.com