УДК 519.61

БЛОЧНИЙ АЛГОРИТМ ПЕРЕТВОРЕНЬ ХАУСХОЛДЕРА ДЛЯ КОМП'ЮТЕРІВ ГІБРИДНОЇ АРХІТЕКТУРИ

О.В. Попов, О.В. Рудич

Інститут кібернетики імені В.М. Глушкова НАН України,
03680, Київ, проспект Академіка Глушкова, 40.
Тел.: (044) 526 6088,
alex50popov@gmail.com, olgar2006@ukr.net

В роботі для комп'ютерів гібридної архітектури з багатоядерними та графічними процесорами запропоновано паралельний блочно-циклічний алгоритм двосторонніх перетворень Хаусхолдера. Наведено результати тестування та дослідження як (масштабованості, прискорення тощо) алгоритм в залежності від його параметрів та параметрів матриці, що перетворюються.

A parallel block cyclic algorithm for two-sided Householder transformations for hybrid architecture computers with multi-core and graphic processors is dealt with in the paper. Results of testing are presented together with investigation of algorithm's characteristics (scalability, acceleration) depending both on its parameters and parameters of matrix being transformed.

Вступ

Перетворення Хаусхолдера (ортогональні перетворення віддзеркалення) досить широко використовуються при розв'язуванні багатьох задач лінійної алгебри: на власні значення матриць, сингулярного розвинення матриць, знаходження узагальнених розв'язків систем лінійних алгебраїчних рівнянь методом найменших квадратів тощо. Такі перетворення дозволяють замінити вихідні задачі більш простими. Наприклад, обчислення власних значень довільної матриці зводиться до обчислення власних значень тридіагональної матриці або сингулярне розвинення довільної прямокутної матриці – до сингулярного розвинення дводіагональної матриці. Тому першим етапом знаходження розв'язків таких задач може бути зведення вихідної матриці задачі до дво- або тридіагональної матриці, до верхньої форми Хесенберга (для квадратних матриць порядку n), до верхньої дводіагональної форми $J \equiv \begin{pmatrix} J^{(0)} \\ 0 \end{pmatrix}$ (для прямокутних $m \times n$ матриць, $m \geq n$), де $J(0)$ – квадратна верхня дводіагональна матриця порядку n, в якій відмінні від нуля лише елементи $j^{(0)}_{k,k}$, $j^{(0)}_{k,k+1}$.

Для виконання вказаних вище зведень необхідно $n-2$ двосторонніх (або односторонніх) елементарних перетворень віддзеркалення. А загальна кількість арифметичних операцій складає [1] $O(n3)$ для квадратних матриць порядку n або $O(mn2)$ для прямокутних $m \times n$ матриць. Тому, починаючи з $n = O(103)$, виконання цих зведень на персональному комп'ютері потребує досить багато часу.

Скорочення часу розрахунків досягається за рахунок зростання продуктивності комп'ютерів, яке базується на використанні комп'ютерів з багатьма процесорними пристроями, зокрема з багатоядерними процесорами та розпаралелюванні обчислень. В цих комп'ютерах реалізується, як правило, MIMD-архітектура (архітектура з множинним потоком команд і даних) з розподіленою пам'яттю. Прискорення обчислень можна досягти також шляхом використання графічних процесорів (GPU), на яких реалізується SIMD-архітектура паралельних обчислень, для виконання великих обсягів однорідних арифметичних операцій. Майбутнє обчислювальної техніки в найближчій перспективі, на наш погляд, – це комп'ютери гібридної архітектури, які поєднують MIMD- і SIMD-архітектури, тобто обчислення на багатоядерних комп'ютерах з використанням GPU для прискоренням обчислень (див. [2]).

В цій роботі на прикладі задачі зведення щільної симетричної матриці до тридіагональної симетричної матриці розглядається реалізація перетворень Хаусхолдера на комп'ютерах гібридної архітектури.

Постановка та метод розв'язування задачі

Розглянемо задачу зведення щільної симетричної матриці A порядку n до тридіагональної симетричної матриці T, тобто задачу обчислення розвинення

$$A = HTH^T, \tag{1}$$

де матриця перетворень H така, що $H^T = H^{-1}$.

Введемо позначення: $C^{(k)} = \{c^{(k)}_{i,j}\}_{i,j=1}^{n}$, $A^{(0)} \equiv A$, $A^{(n-2)} \equiv T$, $H^{(n-2)} \equiv H$, $H^{(0)} \equiv I_n$, I_n – одинична матриця, $d_i \equiv t_{i,i}$, $e_{i+1} \equiv t_{i+1,i}$, $e_1 \equiv 0$, $a^{(i)}_j = \left(0,\ldots,0,a^{(i)}_{j,i+2},\ldots,a^{(i)}_{j,n}\right)^{\mathsf{T}}$ $(j > i)$, $u^{(i)} = \left(0,\ldots,0,u^{(i)}_{i+1},u^{(i)}_{i+2},\ldots,u^{(i)}_{n}\right)^{\mathsf{T}}$, $v^{(i)} = \left(0,\ldots,0,-1,v^{(i)}_{i+1},v^{(i)}_{i+2},\ldots,v^{(i)}_{n}\right)^{\mathsf{T}}$, $h^{(i)} = \left(0,\ldots,0,e^{-1}_{i+1},h^{(i)}_{i+2},\ldots,h^{(i)}_{n}\right)^{\mathsf{T}}$.

Для розвинення (1) можна використати $n-2$ двосторонніх ортогональних перетворень віддзеркалення (Хаусхолдера)

$$A^{(i)} = P^{(i)} A^{(i-1)} P^{(i)}, \quad P^{(i)} = I - x_i x_i^T,\ x_i^T x_i = 2, \quad i = 1, 2, \ldots, n-2, \tag{2}$$

де вектори x_i вибираються з умов $a_{i,k}^{(i)} = a_{k,i}^{(i)} = 0$ $(k = i+2, \ldots, n)$, $d_i \equiv a_{i,i}^{(i-1)}$. Виходячи з цих умов, маємо

$$\begin{gathered} x_i = \frac{\sqrt{2}}{\left\| u^{(i)} \right\|} u^{(i)}, \quad u_k^{(i)} = a_{i,k}^{(i-1)}\ (k = i+2, \ldots, n), \quad u_{i+1}^{(i)} = a_{i,i+1}^{(i-1)} - e_{i+1}, \\ e_{i+1} \equiv a_{i,i+1}^{(i)} = -\operatorname{sign}(a_{i,i+1}^{(i-1)})\sigma_i, \quad \sigma_i^2 = \left\| a_i^{(i-1)} \right\|^2 = \sum_{k=i+1}^{n} \left(a_{i,k}^{(i-1)} \right)^2. \end{gathered} \tag{3}$$

Тоді, визначивши наступним чином вектор $v^{(i)}$

$$v^{(i)} = w^{(i)} + c_i u^{(i)}, \quad w^{(i)} = b_i g^{(i)}, \quad c_i = \frac{b_i}{2} \left(w^{(i)} \right)^{\mathsf{T}} u^{(i)}, \quad g^{(i)} = A^{(i-1)} u^{(i)}, \quad b_i = \left(e_{i+1} u_{i+1}^{(i)} \right)^{-1}, \tag{4}$$

двосторонні перетворення (2) можна записати у вигляді дворангової модифікації

$$A^{(i)} = A^{(i-1)} + u^{(i)} v^{(i)\mathsf{T}} + v^{(i)} u^{(i)\mathsf{T}} \quad (i = 1, 2, \ldots, n-2). \tag{5}$$

Отже, на кожному кроці (для кожного i) модифікується нижня права квадратна підматриця порядку $n-i$ матриці $A^{(i-1)}$. При виконанні перетворень (5) враховується симетричність вихідної матриці A і всіх матриць $A^{(i)}$.

Аналогічно використовуються двосторонні перетворення Хаусхолдера для зведення $A = PJQ$ прямокутної $m \times n$ $(m \geq n)$ матриці A до верхньої дводіагональної форми J. В цьому випадку кожне двостороне перетворення можна записати у вигляді такої дворангової модифікації

$$A^{(i)} = A^{(i-1)} + u^{(i)} y^{(i)\mathsf{T}} + z^{(i)} v^{(i)\mathsf{T}} \quad (i = 1, 2, \ldots, n-2). \tag{6}$$

Тут вектори $u^{(i)}$, $v^{(i)}$, $y^{(i)}$, $z^{(i)}$ з умов та формул аналогічних попередньому випадку (див. [3]).

Використовуючи односторонні перетворення Хаусхолдера, можна сформувати матрицю перетворень з H (1): $H^{(i)} = P^{(k)} H^{(i-1)}$ $(k = n-i-1,\ i = 1, 2, \ldots, n-2,\ H^{(0)} \equiv I_n,\ H^{(n-2)} \equiv H)$. Кожне таке перетворення можна записати у вигляді однорангової модифікації $H^{(i)} = H^{(i-1)} + u^{(k)} z^{(k)\mathsf{T}}$. За аналогічними формулами формуються матриці перетворень Хаусхолдера для зведення прямокутної матриці до верхньої дводіагональної форми тощо.

Блочний алгоритм зведення щільної симетричної матриці до тридіагональної симетричної матриці. На ефективність комп'ютерної реалізації алгоритмів зведення матриць, які використовують перетворення Хаусхолдера, істотний вплив має організація роботи з оперативною пам'яттю. В сучасних процесорах ця пам'ять має складну архітектуру, причому швидкість звернення (запису або читання) до оперативної пам'яті різних рівнів суттєво відрізняється. В той же час при одно- або дворангових модифікаціях матриць доводиться досить часто звертатися до найповільнішої основної оперативної пам'яті, що призводить до суттєвого збільшення часу обчислень. Цю проблему можна вирішити шляхом модифікації алгоритму – використання блочного виконання перетворень відзеркалення [3, 4].

В блочній версії алгоритму зведення щільної симетричної матриці до тридіагональної форми перетворення матриці виконуються так:

$$A^{(Is)} = A^{(Is-s)} + U_I^{(s)} V_I^{(s)^T} + V_I^{(s)} U_I^{(s)^T} \quad (I = 1, 2, \ldots, N-1), \tag{7}$$

де $N = \lceil (n+s-3)/s \rceil$ (значення $\lceil a \rceil$ дорівнює цілій частині числа a), s – розмір блоку, останній N-й блок має розмір s_N+2 (s_N-1 – дорівнює залишку від ділення $n-3$ на s) і модифікується за формулою

$$A^{(n-2)} = A^{(Ns-s)} + U_N^{(s_N)} V_N^{(s_N)^T} + V_N^{(s_N)} U_N^{(s_N)^T}, \tag{8}$$

а прямокутні матриці $U_K^{(q)}$, $V_K^{(q)}$ формуються наступним чином:

$$\begin{gathered} U_K^{(1)} = u^{(Kq-q+1)}, \quad U_K^{(r)} = \left(U_K^{(r-1)}, u^{(Kq-q+r)} \right), \\ V_K^{(1)} = v^{(Kq-q+1)}, \quad V_K^{(r)} = \left(V_K^{(r-1)}, v^{(Kq-q+r)} \right); \end{gathered} \tag{9}$$

тут $r = 2, \ldots, q$, $q = s$, якщо $K = 1, 2, \ldots, N-1$, і $q = s_N$, якщо $K = N$.

Вектор віддзеркалення $u^{(i)}$ і вектор $v^{(i)}$ ($i = Is–s+r$) визначаються за формулами (3), (4). Причому добуток $A^{(i-1)}u^{(i)}$ (тобто вектор $v^{(i)}$) обчислюється за формулами ($r > 1$)

$$A^{(i-1)}u^{(i)} = A^{(Is-s)}u^{(i)} + U_I^{(r-1)}y_r + V_I^{(r-1)}x_r, \quad x_r = \left(U_I^{(r-1)^T}u^{(i)}\right), \quad y_r = \left(V_I^{(r-1)^T}u^{(i)}\right), \tag{10}$$

а елементи i-х рядка та стовпчика матриці $A^{(i-1)}$ за наступними формулами ($k = i+1, \ldots, n$)

$$a_{i,i}^{(i-1)} = a_{i,i}^{(Is-s)} + 2U_{i,I}^{(r-1)}V_{i,I}^{(r-1)\mathsf{T}}, \quad a_{i,k}^{(i-1)} \equiv a_{k,i}^{(i-1)} = a_{i,k}^{(Is-s)} + U_{k,I}^{(r-1)}V_{i,I}^{(r-1)\mathsf{T}} + V_{k,I}^{(r-1)}U_{i,I}^{(r-1)\mathsf{T}}, \tag{11}$$

де $U_{j,I}^{(r-1)}, V_{j,I}^{(r-1)}$ – j-й рядок матриць $U_I^{(r-1)}, V_I^{(r-1)}$ відповідно. Причому в цих обчисленнях використовується наступне представлення матриці $A^{(Is-s+r)}$ ($r = 1, 2, \ldots, s–1$):

$$A^{(Is-s+r)} = A^{(Is-s)} + U_I^{(r)}V_I^{(r)^T} + V_I^{(r)}U_I^{(r)^T}. \tag{12}$$

Так само дворангову модифікацію (6) можна замінити 2s-ранговою

$$A^{(Is)} = A^{(Is-s)} + U_I^{(s)}Y_I^{(s)^T} + Z_I^{(s)}V_I^{(s)^T}, \tag{13}$$

де прямокутні матриці $U_K^{(q)}, V_K^{(q)}, Y_K^{(q)}, Z_K^{(q)}$ формуються за формулами аналогічними формулам (9)–(11) (див. [3]). Головним тут є використання представлення виду (12), формули аналогічні (10) для обчислення добутків матриці $A^{(i-1)}$ або матриці $A^{(i-1/2)}$ на вектор та формули аналогічні (11) для обчислення елементів i-го рядка матриці $A^{(i-1)}$ та i-го стовпчика матриці $A^{(i-1/2)}$. Аналогічним чином однорангові модифікації можна замінити s-ранговими.

Паралельні алгоритми

При зведенні щільної симетричної матриці до тридіагональної необхідно взяти до уваги, що обчислення за формулами (7), (8) та (11), враховуючи симетричність, сумарно вимагають виконання по $\frac{2}{3}n^3 + O(sn^2)$ арифметичних операцій з плаваючою комою, водночас як для всіх інших обчислень необхідно $O(sn^2)$ арифметичних операцій. Для несиметричних матриць або, якщо не враховувати симетричність матриці, то кількість арифметичних операцій збільшується майже вдвічі. Тому для обчислень за формулами (7), (8) та (10) доцільно використати графічні процесорні пристрої (GPU) та враховувати симетричність матриці. Але при використанні багатопроцесорного комп'ютера елементи симетричної матриці розподіляються між процесорними пристроями так, що в пам'яті кожного з них зберігається тільки підматриця цієї матриці, яка (як правило) буде несиметричною. Тому для використання симетричності матриці обчислення за формулами (7) та (10) необхідно організувати спеціальним чином, врахувавши розбиття на блоки. Далі на рис. 1 показано розбиття на блоки симетричної матриць $A^{(Is)}$, матриць $U_I^{(s)}$ (ідентично розбиваються матриці $V_I^{(s)}$) та векторів $u^{(i)}$ (ідентично розбиваються інші n-вимірні вектори).

Використовуючи позначення рис. 1 та наступні

$$U_{I,K}^* = \begin{pmatrix} U_{I,K+1} \\ \vdots \\ U_{I,N-1} \\ U_{I,N} \end{pmatrix}, \quad V_{I,K}^* = \begin{pmatrix} V_{I,K+1} \\ \vdots \\ V_{I,N-1} \\ V_{I,N} \end{pmatrix}, \quad u_{i,K}^* = \begin{pmatrix} u_{i,K+1} \\ \vdots \\ u_{i,N-1} \\ u_{i,N} \end{pmatrix}, \quad g_i^{(K)} = \begin{pmatrix} g_{i,K+1}^{(K)} \\ \vdots \\ g_{i,N-1}^{(K)} \\ g_{i,N}^{(K)} \end{pmatrix},$$

можна формулу (7) розписати так: ($K = I, I+1, \ldots, N$)

$$\begin{gathered} A_{K,K}^{(Is)} = A_{K,K}^{(Is-s)} + U_{I,K}V_{I,K}^T + V_{I,K}U_{I,K}^T, \\ A_{K,K+1}^{(Is)} = A_{K,K+1}^{(Is-s)} + U_{I,K}\left(V_{I,K}^*\right)^T + V_{I,K}\left(U_{I,K}^*\right)^T \quad \text{або} \quad A_{K+1,K}^{(Is)} = A_{K+1,K}^{(Is-s)} + U_{I,K}^*V_{I,K}^T + V_{I,K}^*U_{I,K}^T \equiv \left(A_{K,K+1}^{(Is)}\right)^T. \end{gathered} \tag{14}$$

А формули (10) для $g_i = A^{(i-1)}u_i$ тоді розписуються так: ($K = I, I+1, \ldots, N$)

$$g_{i,K}^{(K)} = A_{K,K}^{(Is-s)}u_{i,K} + A_{K,K+1}^{(Is-s)}u_{i,K}^*, \quad g_i^{(K)} = \left(A_{K,K+1}^{(Is-s)}\right)^T u_{i,K}, \quad g_{i,K} = \sum_{J=I}^{K} g_{i,K}^{(J)} + U_{I,K}^{(r-1)}y_r + V_{I,K}^{(r-1)}x_r. \tag{15}$$

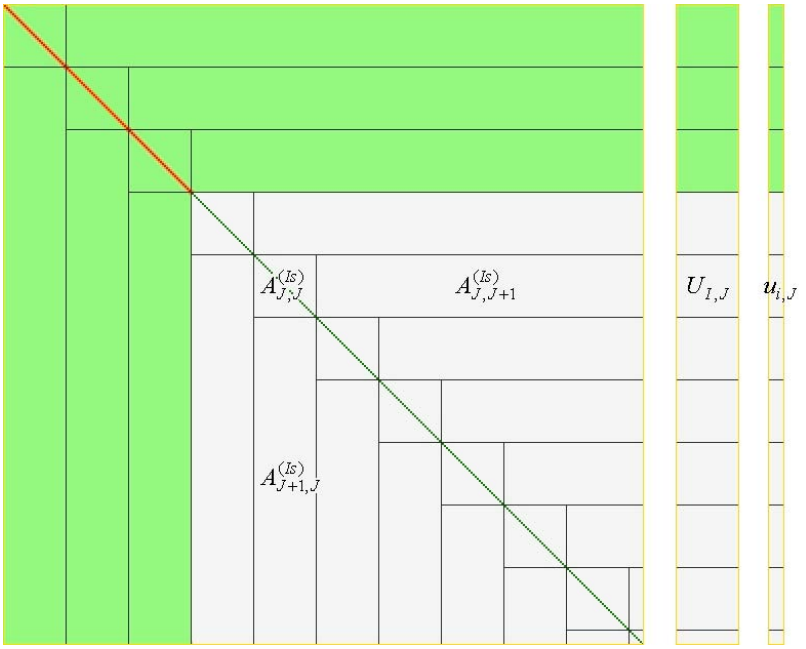

Рис. 1. Розбиття на блоки симетричної матриці

У випадку двосторонніх перетворень прямокутної матриці або односторонніх перетворень матриця $A^{(Is)}$ розбивається на прямокутні блоки $A_K^{(Is)}$ $(K = I{+}1, \ldots, N)$ розміру $s\times(n\text{-}Is)$. Тоді 2s-рангова модифікація або s-рангова модифікація таких блоків виконуються відповідно за формулами

$$A_K^{(Is)} = A_K^{(Is-s)} + U_{I,K}\left(Y_I^*\right)^T + Z_{I,K}\left(V_I^*\right)^T, \quad H_K^{(Is)} = H_K^{(Is-s)} + U_{I,K}\left(Z_I^*\right)^T, \tag{16}$$

де використано позначення $(X_J^*)^T = (X_{J,J+1}^T, \ldots, X_{J,N}^T)$. Відповідно по аналогії з (15) та (16) записуються формули обчислення добутку матриці на вектор.

Паралельний алгоритм зведення щільної симетричної матриці до тридіагональної симетричної матриці. На комп'ютерах гібридної архітектури для зведення симетричної матриці до тридіагональної симетричної матриці доцільно використати блочний алгоритм перетворень Хаусхолдера, оскільки в блочному варіанті наявна велика кількість матрично-матричних і матрично-векторних операцій, які можна виконувати на GPU за допомогою програмних модулів бібліотеки CUBLAS [5]. Блочний алгоритм також дозволяє зменшити кількість обмінів даними між процесорними ядрами.

Розпаралелення обчислень на багатоядерних комп'ютерах з графічними процесорами досягається як за рахунок паралельного виконання розрахунків на різних ядрах, так і за рахунок паралельним обчисленням на GPU. Крім того слід брати до уваги, що такі комп'ютери гібридної архітектури можуть мати в своєму складі декілька обчислювальних вузлів, кожен з яких складається з декількох багатоядерних процесорів з під'єднаними до них GPU. Тому при розробці та реалізації паралельних алгоритмів для цих комп'ютерів слід розрізняти різні архітектури, зокрема такі: 1 ядро + 1 GPU, m ядер + m GPU на одному вузлі, m ядер + m GPU на декількох вузлах, причому одне ядро зв'язано з одним GPU. В останніх двох випадках різниця полягає в тому, що на одному вузлі може використовуватися спільна пам'ять.

Розподіл даних і результатів. Хоча для виконання перетворень (7)-(11) достатньо елементів головної діагоналі матриці і, наприклад, її верхнього трикутника, все-ж доцільно задавати всі елементи матриці, а модифікувати тільки елементи верхнього трикутника. Для організації паралельних обчислень використовується одновимірна блочна циклічна схема [3] розподілу між процесорними ядрами та між GPU елементів матриць $A^{(Is)}$, у тому числі і вихідної матриці $A^{(0)}$. А саме, рядки блоків матриці циклічно розподіляються і зберігаються в пам'яті ядра та їх копії в глобальній пам'яті відповідного GPU. У випадку використання архітектури 1 ядро + 1 GPU всі елементи матриць $A^{(Is)}$ зберігаються в пам'яті ядра та в пам'яті GPU.

Результати зведення – діагональні $t_{j,j}$ $(j = 1,2,\ldots,n)$ та позадіагональні $t_{j,j+1} = t_{j+1,j}$ $(j = 1,2,\ldots,n-1)$ елементи тридіагональної матриці $T_1 \equiv A^{(n-2)}$ – обчислюються кожним ядром CPU. Якщо необхідно обчислити матрицю перетворень H з (1), то на місці елементів верхнього трикутника вихідної матриці у відповідності з їх розподілом між процесорними пристроями в пам'яті ядер зберігаються ненульові елементи векторів u_i.

Для виконання проміжних обчислень, обмінів даними між процесорними ядрами та між ядрами і GPU кожне ядро використовує робочий масив загальним обсягом (2s+3)n+s+2 слів, а GPU 2 робочі масиви по ns слів кожний і один з n+2s слів.

Алгоритм. З формул (7)–(11) та рис. 1 випливає, що на I-у кроці алгоритму ($I = 1, 2, \ldots, N\text{-}1, N$) можна виділити наступні підзадачі:

- мультирозсилка всім процесорним ядрам та всім GPU I-го (провідного) рядка блоків матриці $A^{(Is-s)}$ (у випадку m ядер + m GPU);
- формування кожним ядром, використовуючи GPU, прямокутних матриць $U_I^{(s)}$, $V_I^{(s)}$;
- $2s$-рангова модифікація (7) елементів підматриці порядку $n - Is$ матриці $A^{(Is)}$ відповідно до розподілу між процесорними ядрами; виконується на GPU.

Формування прямокутних матриць $U_I^{(s)}$, $V_I^{(s)}$ відповідно до (8)–(11) полягає у тому, що для кожного $i = Is\text{-}s+1, \ldots, ks$ обчислюються вектори u_i, g_i, w_i, v_i та значення ряду скалярних величин. Тут доцільно виділити наступні операції або групи операцій:

1) формування вектора u_i: якщо використовується архітектура 1 ядро + 1 GPU, то на GPU, а якщо m ядер + m GPU, то можна виконати на кожному процесорному ядрі з подальшим копіюванням на GPU;

2) формування кожною парою ядро + GPU векторів g_i, w_i, v_i згідно (10), (11), (15). При реалізації цих операцій доцільно використати GPU, зокрема, застосувавши необхідні програмні модулі CUBLAS, та процесорні ядра при використанні архітектури m ядер + m GPU для мультизбирання вектора $g_i = A^{(i-1)}u_i$ з обчислених на GPU згідно (15) часткових сум; при використанні архітектури 1 ядро + 1 GPU всі операції цієї групи доцільно виконувати на GPU;

3) при $i < Is$ $2(i\text{-}Is+s)$-рангова модифікація $(i\text{-}Is+s+1)$-го рядка провідного рядка блоків матриці за формулою (11), де використовуються вже сформовані частини відповідних прямокутних матриць. Цю підзадачу можна розпаралелити на GPU з використанням необхідних функцій CUBLAS.

Підзадача з $2s$-рангової модифікації (7) за формулами (14) верхнього трикутника підматриці $A^{(Is-s)}$ поділяється на $2s$-рангові модифікації окремих симетричних діагональних блоків $A_{K,K}$ та 2s-рангові модифікації окремих наддіагональних прямокутних блоків $A_{K,K+1}$, що складають підматрицю $A^{(Is-s)}$: Ці $2s$-рангові модифікації окремих блоків виконуються на GPU, використовуючи відповідні функції CUBLAS, згідно розподілу елементів матриці між GPU.

При програмній реалізації цього алгоритму можна використовувати можливості, які надаються останніми версіями технології CUDA [6], асинхронних пересилок масивів даних між різними GPU та асинхронного виконання програмних модулів на GPU, що дозволяє виконувати паралельно обчислення на процесорних ядрах та на GPU. Ці можливості дозволяють на архітектурі m ядер + m GPU використовувати тільки GPU для формування векторів u_i, g_i, w_i, v_i та підвищити ефективність мультирозсилки I-го (провідного) рядка блоків матриці. Але необхідно брати до уваги, що на деяких гібридних комп'ютерах реалізація програмних засобів останніх версій CUDA може призводити до суттєвого збільшення часу розв'язування задач.

Для задачі зведення прямокутної матриці до верхньої дводіагональної форми можна запропонувати аналогічний паралельний блочно-циклічний алгоритм двосторонніх перетворень Хаусхолдера (13). В цьому алгоритмі виділяються три аналогічні підзадачі. При цьому в блочних операціях з під матрицями матриць $A^{(j)}$ не треба враховувати симетричність і тому використовуються тільки прямокутні блоки.

Паралельний алгоритм накопичення елементарних перетворень віддзеркалення. Аналогічні паралельні алгоритми для комп'ютерів гібридної архітектури можна запропонувати для задач, які згадувались вище – формування матриць перетворень H, P, Q, зведення прямокутної матриці до верхньої дводіагональної форми тощо, на основі формул (16). Розглянемо наприклад, задачу накопичення елементарних перетворень віддзеркалення, формування матриці перетворень H.

Вихідними даними цієї задачі є результати зведення симетричної матриці до тридіагональної – вектори віддзеркалення $u^{(k)}$ та скалярні множники b_k, $k = 1, 2, \ldots, n\text{-}2$. Елементи векторів віддзеркалення $u^{(k)}$ можуть зберігатися на місці елементів верхнього трикутника вихідної матриці в пам'яті GPU та за необхідності в пам'яті багатоядерних процесорів відповідно до їх одновимірного блочного циклічного розподілу. Для збереження скалярних множників b_k можна використати тимчасовий (робочий) масив. Результат – матриця H – формується на GPU, її елементи розподілено між GPU за тією ж одновимірною блочною циклічною схемою. Для проміжних обчислень і обмінів даними можна використати ті ж самі робочі масиви, що і в алгоритмі зведення.

Як і в попередньому паралельному алгоритмі зведення симетричної матриці до тридіагональної на I-му кроці алгоритму ($I = 2, \ldots, N\text{-}1, N$) можна виділити наступні підзадачі:

- мультирозсилка всім ядрам та всім GPU елементів прямокутної матриці $U_{N-I+1}^{(s)}$ (у випадку використання m ядер + m GPU);
- формування кожним ядром, використовуючи GPU, прямокутної матриці $Z_I^{(s)}$ (можливе виконання всіх обчислень на GPU);
- s-рангова модифікація (16) підматриці $H^{(Is)}$ відповідно до розподілу між GPU, виконується на GPU.

Апробація паралельних алгоритмів

Для досягнення найвищої продуктивності при використанні запропонованих паралельних алгоритмів необхідно, враховуючи параметри задачі, вибрати відповідну архітектуру гібридного комп'ютера, тобто кількість ядер та GPU, а також розмір блоку. На ці фактори впливають доволі багато чинників, пов'язаних з характеристиками використовуваних програмно-технічних засобів. Тому теоретичне дослідження ефективності паралельних алгоритмів для комп'ютерів гібридної архітектури складає великі труднощі і не дає достатньо достовірних результатів.

На наш погляд більш доцільним є експериментальне дослідження запропонованих алгоритмів на розв'язуванні ряду тестових задач на конкретному комп'ютері. Таке досліджено для запропонованих алгоритмів було проведено на комп'ютері гібридної архітектури Інпарком-G – спільній розробці Інституту кібернетики імені В.М. Глушкова НАН України та ДНВП "Електронмаш". Цей комп'ютер має в своєму складі 4 обчислювальні вузли. Кожен вузол складається з двох чотириядерних процесорів та двох графічних процесорів.

Чисельні експерименти проводились на обчисленні розвинення (1) щільних симетричних матриць різних порядків, тобто на обчисленні як тридіагональної матриці T, так і матриці перетворень H, використовуючи перетворення Хаусхолдера.

Проведені експерименти підтвердили зроблений раніше висновок про доцільність використання в розрахунках рівної кількості процесорних ядер та GPU. Це пов'язано з тим, що більшість розрахунків проводиться на графічних процесорах, а ядра переважно використовуються для обмінів даними. Водночас використання на кожному вузлі одного ядра та двох GPU не дало відчутних переваг, особливо якщо для обчислень використовувалось декілька вузлів.

Деякі результати дослідження впливу розмірів блоків на продуктивність комп'ютерів гібридної архітектури на задачах зведення щільних симетричних матриць до тридіагональних, використовуючи перетворення Хаусхолдера, та обчислення матриць перетвореннь H наведено в табл. 1. Дослідження проводились на однову вузлі.

Таблиця 1. Час (сек.) виконання перетворень Хаусхолдера для матриць різних порядків в залежності від розмірів блоків

Розмір блоку s	Час (сек.)				
	n = 6 144		n = 7 000		n = 10 000
	1 ядро + + 1 GPU	2 ядра + + 2 GPU	1 ядро + + 1 GPU	2 ядра + + 2 GPU	2 ядра + + 2 GPU
32	84,090	55,980	119,690	72,419	–
64	58,770	44,760	82,810	61,009	138,306
96	52,800	43,670	73,760	58,212	117,313
112	–	–	–	58,593	116,671
128	**50,830**	**39,270**	70,580	**53,610**	118,648
144	–	–	–	60,214	131,105
160	50,930	45,280	**70,210**	54,179	**116,355**
176	–	–	–	61,568	–
192	51,510	46,670	76,110	55,823	130,939
256	53,780	45,560	72,490	66,680	–
320	54,500	50,890	73,220	67,450	–
512	–	–	–	82,299	–

Наведені результати та результати інших експериментів свідчать, що найвища продуктивність (найменший час) досягається при розмірі блоку в межах від 96 до 192. Також ці експерименти показали, що менше часу витрачається при розмірах блоків кратних 16.

Також було перевірено масштабованість запропонованих паралельних алгоритмів, досліджено продуктивність та прискорення.

В табл. 2 показано показники продуктивності обчислень на різних архітектурах, різних середовищах паралельного програмування та двох різних розмірах блоків при виконанні перетворень Хаусхолдера для зведення щільних симетричних матриць різних порядків n до тридіагональних симетричних матриць. Для порівняння наведено також продуктивність обчислень без використання графічних процесорів (зауважимо, що в цьому випадку найвища продуктивність досягається при розмірі блоку від 8 до 16).

На діаграмах на рис. 2 показано прискорення запропонованих алгоритмів отримані при зведенні матриць 10 000-го та 20 000-го порядків при розмірі блоків 96 та 128 на різній кількості ядер та GPU порівняно з 1 ядром та 1 GPU.

Як наведені результати, так і результати інших експериментів демонструють ефективність використання графічних прискорювачів при виконанні блочних перетворень Хаусхолдера. Продуктивність обчислень зростає із зростанням порядку матриць. Те ж саме стосується прискорення. Це пов'язано з тим, що для менших порядків присутнє недостатнє завантаження використовуваного обчислювального ресурсу. Тому продуктивність обчислень незначно збільшується або навіть зменшується (за рахунок накладних витрат) починаючи з деякої кількості ядер та GPU: для n = 4 096 це 2, для n = 7 000 це 4, для n = 10 000 це 6. Тільки при n > 14 000 наявна масштабованість для всього діапазону використовуваних процесорних ядер та GPU. Зауважимо також, що при використанні одного вузла дещо вищу продуктивність отримано при розмірі блоків s = 128, а при використанні більшої кількості вузлів при s = 96.

Таблиця 2. Продуктивність обчислень при виконанні перетворень Хаусхолдера в залежності від порядку n, розмірів блоків s та використаних програмно-технічних засобів

Кіл-ть вузлів	Кіл-ть ядер	Кіл-ть GPU	s	n = 4 096	n = 7 000	n = 10 000	n = 16 000	n = 20 000	Середовище розпаралелювання
				Продуктивність (Gflops.)					
1	1	1	96	8,01	11,54	14,42	17,76	18,68	CUDA
	1	1	128	8,14	12,06	15,09	19,22	21,09	
	1	–	128	0,31	0,57				–
	1	–	12	0,71	0,87				
1	2	2	96	10,30	14,58	18,62	28,05	31,38	MPI+CUDA
	2	2	128	8,90	15,48	20,87	28,51	32,45	
	8	–	128	1,16	1,79				MPI
	8	–	12	4,39	5,34	5,02		5,56	
2	4	4	96	10,10	16,59	23,06	33,05	37,62	MPI+CUDA
	4	4	128	9,34	15,76	23,13	32,81	36,56	
	16	–	12	8,18	8,25	8,45			MPI
3	6	6	96	10,43	17,83	**25,92**	**42,72**	45,04	MPI+CUDA
	6	6	128	9,48	16,38	21,96	36,21	39,61	
	24	–	12	11,53	11,71	11,96			MPI
4	8	8	96	**10,69**	**18,38**	25,89	39,79	**47,11**	MPI+CUDA
	8	8	128	9,56	17,04	22,91	36,13	43,04	
	32	–	12	14,01	14,97	14,41			MPI

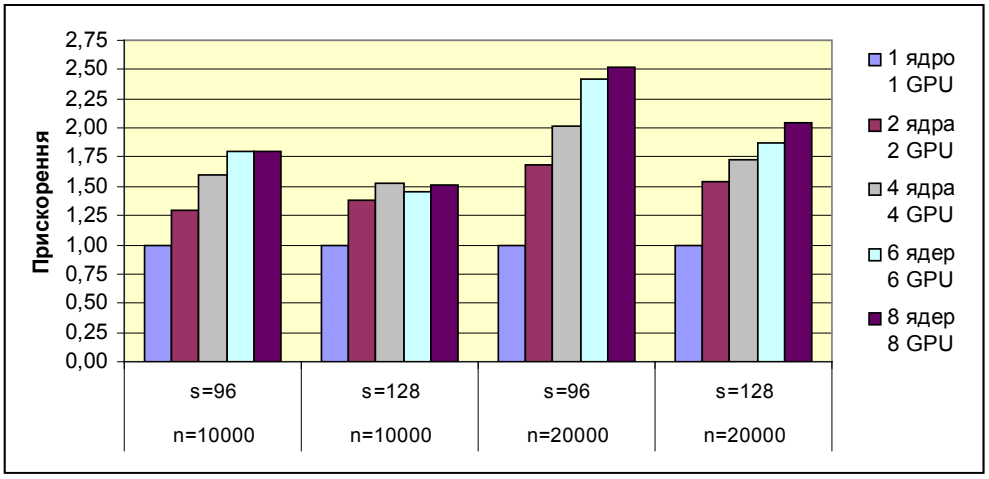

Рис. 2. Прискорення алгоритмів

Проведені дослідження засвідчили, що прискорення, а отже і зростання продуктивності наявне при кількості, яка перевищує 10^7, елементів підматриці, що обробляється однією парою ядро+GPU. Також, як зазначалось вище, при використанні ресурсу більше одного вузла доцільно використовувати блоки, прядок яких $s = 96$.

Висновки

В роботі для комп'ютерів з багатоядерними та графічними процесорами (гібридної архітектури) запропоновано паралельні блочно-циклічні алгоритми перетворень Хаусхолдера, зокрема, двосторонніх перетворень для зведення щільної симетричної матриці до тридіагональної. Проведене тестування запропонованих алгоритмів засвідчило ефективність використання графічних процесорів для виконання перетворень Хаусхолдера та правильність вибраних підходів до розробки алгоритмів високопродуктивних обчислень для комп'ютерів гібридної архітектури. В тому числі:

- перехід до блочних версій відповідних методів та алгоритмів розв'язування задач з метою виділення підзадач з великим обсягом матрично-векторних та матрично-матричних операцій;
- виконання матрично-векторних та матрично-матричних операцій на графічних процесорах, використовуючи програмні засоби від розробників технічних засобів;
- проведення експериментального дослідження якостей розробленого алгоритму.

Проведені експериментальні дослідження запропонованих алгоритмів на задачах розвинення (1) симетричних матриць різних порядків дозволили визначити обчислювальний ресурс та розміри блоків, достатні для ефективного розв'язування конкретної задачі на конкретному комп'ютері. Використана для цього методика може застосовуватися для тестування та дослідження інших алгоритмів високопродуктивних обчислень для комп'ютерів гібридної архітектури.

1. *Уилкинсон Дж.Х., Райнш К.* Справочник алгоритмов на языке Алгол. Линейная алгебра. – М.: Машиностроение, 1976. – 389 с.
2. http://www.top500.org
3. *Химич А.Н., Молчанов И.Н., Попов А.В., Чистякова Т.В., Яковлев М.Ф.* Параллельные алгоритмы решения задач вычислительной математики. – К.: Наук. думка, 2008. – 248 с.
4. http://www.netlib.org/ lapack/
5. http://developer.download.nvidia.com/CUBLAS.pdf [Електронний ресурс] CUBLAS
6. http://developer.nvidia.com/cuda-toolkit-4.0 [Електронний ресурс] CUDA TOOLKIT 4.0