КІБЕРНЕТИКА та КОМП'ЮТЕРНІ ТЕХНОЛОГІЇ

УДК 519.6 DOI:10.34229/2707-451X.20.2.6

О.М. ХІМІЧ, В.В. ПОЛЯНКО, Т.В. ЧИСТЯКОВА

ПАРАЛЕЛЬНІ АЛГОРИТМИ РОЗВ'ЯЗУВАННЯ ЛІНІЙНИХ СИСТЕМ НА ГІБРИДНИХ КОМП'ЮТЕРАХ

Розглядаються паралельні алгоритми для розв'язування лінійних систем з наближеними даними на суперкомп'ютерах з графічними процесорами. Обговорюються проблеми розв'язування задач з наближеними даними, а також особливості створення алгоритмів для гібридних комп'ютерів. Наводяться результати експериментальних досліджень.

Ключові слова: *системи лінійних алгебраїчних рівнянь, наближені дані, паралельні комп'ютери, графічні процесори, оцінки достовірності комп'ютерних результатів.*

© О.М. Хіміч, В.В. Полянко, Т.В. Чистякова, 2020

Вступ. У наш час в науці та інженерії постійно з'являються нові обчислювальні задачі з великими обсягами даних, для розв'язування яких необхідно використовувати потужні суперкомп'ютери. Більшість таких задач зводиться до розв'язування систем лінійних алгебраїчних рівнянь (СЛАР).

Головні проблеми розв'язування задач на комп'ютері – це отримання вірогідних розв'язків при мінімальних витратах обчислювальних ресурсів.

Задача, яка розв'язується на комп'ютері, завжди має наближені дані щодо вихідної задачі (через похибку вихідних даних, через похибку введення числових даних в комп'ютер тощо). Отже, властивості комп'ютерної задачі можуть істотно відрізнятися від властивостей вихідної задачі. Виникає необхідність розв'язувати задачі з урахуванням наближених даних та аналізом отримуваних комп'ютерних результатів.

Незважаючи на значні результати досліджень в області лінійної алгебри, роботи в напрямку подолання існуючих проблем комп'ютерного розв'язування задач з наближеними даними, які ще більше поглиблюються при використанні сучасних суперкомп'ютерів, не втрачають своєї значущості та потребують подальшого розвитку

На сьогодні одними з найпродуктивніших суперкомп'ютерів є багатоядерні MIMD-комп'ютери з графічними процесорами (гібридні комп'ютери), які протягом останніх років займають чільні позиції у світовому рейтингу Top500 [1]. Архітектурні та технологічні особливості цих комп'ютерів дають можливість значно підвищити ефективність розв'язування задач великих обсягів, у тому числі задач лінійної алгебри, при відносно невеликих енерговитратах.

У роботі розглядаються паралельні алгоритми методів Гаусса та Холецького для розв'язування СЛАР на гібридних комп'ютерах.

Стан проблеми. На сьогоднішній день створенню ефективного програмного забезпечення для розв'язування задач лінійної алгебри на різних паралельних архітектурах приділяється велика увага, адже більшість задач математичного моделювання зводиться до розв'язування саме цього класу задач.

Найвідоміші бібліотеки алгоритмів та програм розв'язування задач лінійної алгебри на паралельних комп'ютерах (у тому числі гібридної архітектури) розробляються під керівництвом професора Донгарри (Jack Dongarra, Electrical Engineering and Computer Science Department, University of Tennessee) в США. Цією групою дослідників на основі бібліотеки математичних операцій над матрицями та векторами BLAS, та бібліотеки програм блочних алгоритмів розв'язування задач лінійної алгебри LAPACK розробляється програмно-алгоритмічне забезпечення для різних комп'ютерних архітектур [2]. Зокрема створено для розв'язування СЛАР на комп'ютерах MIMD-архітектури − бібліотеку ScaLAPACK [2], для багатоядерних комп'ютерів − бібліотеки програм: Core Math Library AMD (оптимізована під процесори компанії AMD) [3], Intel Math Kernel Library (оптимізована під процесори компанії Intel) [4].

З появою у 2007 році першої версії програмно-апаратної архітектури CUDA (Compute Unified Device Architecture) [5] паралельних обчислень з використанням графічних процесорів компанією NVIDIA були адаптовані бібліотеки математичних операцій: cuBLAS [6] − для щільних матриць та cuSPARSE [7] для розріджених матриць. На основі цих бібліотек під керівництвом Донгарри створено бібліотеки паралельних алгоритмів та програм для розв'язування СЛАР із щільними матрицями: CULA [8] − з використанням одного графічного прискорювача, MAGMA [9] − з використанням декількох графічних прискорювачів. Високоефективна також бібліотека програм CUSP [10], яка реалізує розв'язування розріджених лінійних систем ітераційними методами на гібридних комп'ютерах та деякі інші.

Також активно використовуються для розв'язування задач обчислювальної математики на різних паралельних платформах, у тому числі СЛАР, високопродуктивні пакети програм комп'ютерної математики MathCAD [11], Maple [12], Mathematica [13], Matlab [14]. Ці пакети програм мають інтерфейси для спілкування з користувачами за допомогою спеціальних символьних мов, а також засоби графічної візуалізації даних та результатів обчислень.

Проте, як відзначають спеціалісти, великий набір існуючого програмного забезпечення не вирішує у достатній мірі проблем достовірності комп'ютерних результатів розв'язування задач з наближеними даними [15 – 20]. Реалізація чисельних алгоритмів базується на точно заданих даних, аналіз достовірності комп'ютерних результатів покладається на користувачів. Тому при використанні цих програмних продуктів можуть виникати проблеми достовірності комп'ютерних результатів, а для їх аналізу необхідно володіти відповідними знаннями з математики.

Крім того, математичні моделі фізичних процесів можуть мати природні похибки в результаті вимірювань, спостережень, припущень, гіпотез і т. п. Надалі при дискретизації математичної моделі ці похибки трансформуються в похибки коефіцієнтів рівнянь, які підлягають розв'язуванню. Вихідні дані математичних моделей можуть бути задані точно в числовому вигляді або представлені математичними формулами. Але заокруглення при введенні чисел у комп'ютер та обчисленні формул також призводять до машинних моделей з наближеними даними.

Оригінальними для гібридних алгоритмів, що пропонуються, є комп'ютерне дослідження математичних властивостей СЛАР з урахуванням наближеного характеру даних та розв'язування з аналізом достовірності отримуваних комп'ютерних розв'язків при ефективному використанні обчислювальних пристроїв гібридного комп'ютера.

Деякі особливості створення паралельних алгоритмів для гібридних комп'ютерів. Гібридні комп'ютери поєднують MIMD- і SIMD-архітектури тобто поєднуються багатоядерні центральні процесори (Central Processor Unit – CPU) з розподіленою пам'яттю та масивно-паралельні прис-

корювачі обчислень − графічні процесори (Graphics Processing Unit – GPU). Кожна архітектура дає можливість здійснювати різного рівня розпаралелення, проте без урахування архітектурних відмінностей CPU і GPU, а також різних систем розпаралелення обчислень, які на них використовуються, неможливо забезпечити високу швидкодію гібридних алгоритмів та програм.

Для забезпечення ефективного розв'язування задачі на гібридному комп'ютері при створенні алгоритмів необхідно передбачити автоматичне виконання таких кроків:

− побудова ефективної віртуальної топології з оптимальної кількості обчислювальних елементів CPU для конкретної задачі;

− розділення задачі на окремі частини алгоритму (підзадачі) та їх розпаралелення між обчислювальними елементами CPU при використанні графічних процесорів для виконання однотипних математичних операцій з великими обсягами даних;

− дослідження математичних властивостей комп'ютерної задачі та її розв'язування з аналізом достовірності результатів.

Розділення задачі на окремі підзадачі здійснюється з метою визначення: пріоритетів підзадач, на яких обчислювальних ресурсах їх краще реалізовувати за часом виконання, які підзадачі можна виконати паралельно, в якій послідовності їх треба виконувати тощо. При цьому враховуються архітектурні особливості процесорів CPU та GPU, відмінності в організації пам'яті, системи розпаралелення.

Розпаралелення підзадач, які не потребують великого часу, краще виконувати на CPU за допомогою MPI (Message Passing Interface) – інтерфейсу передачі повідомлень [21]. В цьому випадку підзадачі розпаралелюються між процесорами CPU на окремі MPI-процеси, що виконуються незалежно. При цьому швидкодію виконання алгоритмів на CPU можна значно підвищити, якщо розробляти їх з урахуванням наявної досить великої кеш-пам'яті. З цією метою розроблено блочно-циклічні гібридні алгоритми, що дає можливість узгоджувати розмір блоку матриці з розміром кеш-пам'яті.

На ефективність реалізації паралельного алгоритму впливає використання різних способів обміну даними між процесами. Тому важливо передбачити в алгоритмах побудову топології з оптимальної кількості MPI-процесів, яка забезпечуватиме виконання взаємозв'язків між процесами при мінімальних затратах обчислювальних ресурсів. На основі теоретичних та експериментальних досліджень в представлених алгоритмах це здійснюється автоматично.

На відміну від CPU, на графічних процесорах система CUDA організує 6 видів пам'яті, кожна з яких має своє призначення. В розпаралеленні обчислень на GPU здебільшого використовуються глобальна пам'ять (global-memory) та розподілена пам'ять (shared-memory). Global-пам'ять призначена для встановлення зв'язків між процесорами CPU та GPU, а також для збереження даних на GPU. Вона досить велика, повільна і не кешується.

Shared-пам'ять – це швидка пам'ять GPU, яка використовується для паралельного виконання обчислень. Проте ця пам'ять дуже мала у порівнянні з кеш-пам'яттю CPU. На один GPU-процесор доступно всього 16 кбайт розподіленої пам'яті. На shared-пам'яті за допомогою CUDA може виконуватися паралельно величезна кількість обчислювальних потоків. При цьому кожному потоку ставиться у відповідність один елемент матриці або компонента вектора. Тому в гібридних алгоритмах на процесорах GPU доцільно виконувати математичні операції над векторами та матрицями (скалярні добутки, перемноження вектора на скаляр, матричні операції), які потребують найбільших затрат комп'ютерного часу.

Копіювання даних з пам'яті CPU в global-пам'ять GPU і назад відноситься до найбільших обчислювальних витрат, що суттєво збільшують час виконання алгоритму. Тому в гібридних алгори-

тмах передбачено синхронне виконання копіювання даних між CPU та GPU та деякі обчислення на GPU, які можна виконувати незалежно, наприклад, матрично-векторні операції.

Головна частина паралельної програми, що реалізує гібридний алгоритм, виконується на CPU і керує обчисленнями на GPU таким чином:

- виділяється пам'ять на GPU для збереження матриці та вектора;
- копіюються дані з пам'яті CPU у виділену пам'ять GPU;
- запускається ядро (kernel–програма) на GPU;
- по закінченню обчислень вектор розв'язку копіюється з пам'яті GPU у пам'ять CPU;
- звільняється виділена пам'ять GPU.

Схема зберігання та розподіл даних між процесорними пристроями гібридного комп'ютера. Дослідження гібридних алгоритмів розв'язування СЛАР із щільними матрицями показали, що найбільш ефективними є блочні алгоритми як з погляду ефективного використання кеш-пам'яті на CPU, так і з погляду швидкодії виконання арифметичних операцій на shared-пам'яті. На процесах CPU розмір блоку матриці узгоджується з розміром кеш-пам'яті, а на процесорах GPU вихідний масив вектора або матриці ділиться на блоки так, щоб його довжина відповідала розміру блоку shared-пам'яті. Як правило, розмір блоку матриці на GPU є кратним 16.

В гібридних алгоритмах, що розглядаються, використовується двовимірний блочно-циклічний розподіл вихідної щільної матриці A порядку n між p-процесами у вигляді [17]. Будемо вважати, що матриця A порядку n розділена на квадратні блоки розміру $s \times s$. Не втрачаючи загальності міркувань, вважатимемо, що n/s – ціле число. Крім того, введемо наступні блочні представлення матриці $A = (A_{ij})$, $i, j = 1,...l$, $l = n/s$:

$$A = \begin{pmatrix} A_{11}^* & A_{12}^* & \dots & A_{1p}^* \\ A_{21}^* & A_{22}^* & \dots & A_{2p}^* \\ \dots & \dots & \dots & \dots \\ A_{q1}^* & A_{q2}^* & \dots & A_{qp}^* \end{pmatrix},$$

де

$$A_{ij}^* = \begin{pmatrix} A_{ij} & A_{ij+p} & \dots & A_{ij+\left(p^*-1\right)p} \\ A_{i+pj} & A_{i+pj+p} & \dots & A_{i+pj+\left(p^*-1\right)p} \\ \dots & \dots & \dots & \dots \\ A_{i+\left(q^*-1\right)pj} & A_{i+\left(q^*-1\right)pj+p} & \dots & A_{i+\left(q^*-1\right)pj+\left(p^*-1\right)p} \end{pmatrix}.$$

Тут A_{ij} – блок елементів матриці A розміру $s \times s$, $q^* = n \backslash sq$, $p^* = n \backslash sp$.

Нехай, $n = 8$, $s = 2$, $p = 2$, $q = 2$. Тоді для матриці

$$A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{18} \\ a_{21} & a_{22} & \dots & a_{28} \\ \dots & \dots & \dots & \dots \\ a_{81} & a_{82} & \dots & a_{88} \end{pmatrix}$$

блочне представлення має вигляд

$$A=\begin{pmatrix} A_{11} & A_{12} & A_{13} & A_{14} \\ A_{21} & A_{22} & A_{23} & A_{24} \\ A_{31} & A_{32} & A_{33} & A_{34} \\ A_{41} & A_{42} & A_{43} & A_{44} \end{pmatrix}, \quad \text{де} \quad A_{ij}=\begin{pmatrix} a_{2i-1,2j-1} & a_{2i-1,2j} \\ a_{2i,2j-1} & a_{2i,2j} \end{pmatrix},$$

В цьому випадку блочно-циклічний розподіл матриці A матиме вигляд $A=\begin{pmatrix} A_{11}^{*} & A_{12}^{*} \\ A_{21}^{*} & A_{22}^{*} \end{pmatrix}$, де

$$A_{11}^{*}=\begin{pmatrix} A_{11} & A_{13} \\ A_{31`} & A_{33} \end{pmatrix}, \quad A_{12}^{*}=\begin{pmatrix} A_{12} & A_{14} \\ A_{32`} & A_{34} \end{pmatrix}, \quad A_{21}^{*}=\begin{pmatrix} A_{21} & A_{23} \\ A_{41`} & A_{43} \end{pmatrix}, \quad A_{22}^{*}=\begin{pmatrix} A_{22} & A_{24} \\ A_{42`} & A_{44} \end{pmatrix}.$$

Отже, в кожному процесі P_{ij} містяться елементи матриці A_{ij}^{*}, а також відповідні блоки матриці правих частин. Кожний вектор правих частин розподіляється циклічно блоками розміру $p\times q$ між процесами першого стовпчика «процесорної решітки». Отже, P_{k1} містить елементи з номерами ks, $ks+1$, $ks+2$,..., $ks+s-1$, $(k+p)s$, $(k+p)s+1$, $(k+p)s+2$, ..., $(k+p)s+s-1$,..., де s – розмір блоку, що дорівнює відповідній величині в розподілі матриці.

Комп'ютерне дослідження математичних властивостей СЛАР з наближеними даними та аналіз достовірності результатів розв'язування. Розглядаємо розв'язування системи лінійних рівнянь із щільною невиродженою матрицею та наближеними даними

$$Ax=b. \tag{1}$$

Тобто елементи матриці A та вектора правої частини b мають деякі збурення щодо вихідної задачі

$$\bar{A}\bar{x}=\bar{b}, \tag{2}$$

що визначаються формулами:

$$\left\|\bar{A}-A\right\|=\left\|\Delta A\right\|\le E_A\left\|A\right\|, \qquad \left\|\bar{b}-b\right\|=\left\|\Delta b\right\|\le E_b\left\|b\right\|. \tag{3}$$

При цьому передбачається, що структура матриці вихідної задачі (2) і збуреної задачі (1), (3) не змінюється, тобто, якщо вихідна матриця є симетричною, то й збурена залишається симетричною, якщо вихідна – стрічкова, то й збурена – стрічкова і т. д.

Отже, при розв'язуванні СЛАР з наближеними вихідними даними необхідно розглядати цілий клас систем рівнянь (1), (3), що мають досить широку множину формально допустимих розв'язків.

Отримуючи результати розв'язування задачі (1), необхідно оцінити збурення розв'язку в залежності від збурення вихідних даних (3). Не завжди близькість елементів матриць A і $\bar{A}$ та правих частин b і $\bar{b}$ забезпечує достатню близькість розв'язків. Наприклад, при деякому збуренні в межах точності елементів матриці і/або правої частини несумісної точної системи (2) збурена система (1), яку отримано в комп'ютері, може виявитися сумісною і навпаки – сумісна СЛАР може перетворитися на несумісну [16, 17, 22 – 25].

Похибку розв'язку $\Delta x=\bar{x}-x$, пов'язаною з наближеним наданням вихідних даних, називають спадковою. Її значення залежить як від похибок вихідних даних (3), так і від властивостей матриць вихідної (2) та збуреної (1) систем.

Розв'язок $\tilde{x}$ системи рівнянь (1), що отримується на комп'ютері, будемо називати комп'ютерним розв'язком задачі, який (через похибки заокруглень даних, методу розв'язування і

комп'ютерних обчислень) відрізняється від точного (математичного) її розв'язку. Різниця цих розв'язків $\Delta\tilde{x} = x - \tilde{x}$ – обчислювальна похибка.

Таким чином, розв'язування СЛАР з наближеними вихідними даними на комп'ютері полягає в дослідженні математичних властивостей задачі (1), (3), визначенні одного з допустимих розв'язків цієї задачі та в оцінці похибок (спадкової і обчислювальної) отриманого розв'язку [16, 17, 22].

Дослідження властивостей СЛАР загалом базується на оцінці числа обумовленості матриці, значення якого обчислюється в процесі факторизації матриці одним з прямих методів, наприклад, LU -факторизації методом Гаусса за такою схемою [17]:

$$A \cong LU, \tag{4}$$

$$\|A\|_1 = \max_{\mathrm{j}} \sum_{i=1}^{n} |a_{ij}|, \tag{5}$$

$$U^T w = e, \tag{6}$$

$$L^T y = w, \tag{7}$$

$$Lv = y, \tag{8}$$

$$Uz = v, \tag{9}$$

$$\mathrm{cond}A = \|A\|_1 \|z\|_1 / \|y\|_1, \tag{10}$$

$$\|A\|_1 = \max_{\mathrm{j}} \sum_{i=1}^{n} |a_{ij}|, \quad \|z\|_1 = \sum_{i=1}^{n} |z_i|, \quad \|y\|_1 = \sum_{i=1}^{n} |y_i|. \tag{11}$$

Обчислення відповідних норм можна реалізувати на CPU. В кожному MPI-процесі на CPU вибирається максимальна з наявних у ньому векторних норм, після цього за допомогою відповідних MPI-функцій вибирається їх максимальна величина з усіх процесів.

Для розв'язування системи (6) існує спеціальний алгоритм, який враховує стратегію вибору компонент вектора e при розв'язуванні системи [17]. Наведемо його.

1. Покладемо
$$e_1 = 1, \quad w_1 = 1/u_{11}.$$
2. Для $k = 2,...,n$ обчислюємо
$$t_j = \sum_{i=1}^{k-1} u_{ij} \times w_i, \quad j = k,...,n.$$
3. Для двох можливостей вибору e_k покладаємо
$$e_k^+ = \mathrm{sign}(-t_k), \quad e_k^- = -e_k^+.$$
4. Обчислюємо два розв'язки:
$$w_k^+ = \left(e_k^+ - t_k\right)/u_{kk}, \quad w_k^- = \left(e_k^- - t_k\right)/u_{kk}.$$
5. Для вибору одного розв'язку із w_k^+ та w_k^- обчислюємо:
$$t_k^+ = e_k^+ - t_k, \quad t_k^- = e_k^- - t_k, \quad t_j^+ = t_j + u_{kj} \times w_k^+, \quad t_j^- = 0 + u_{kj} \times w_k^-, \quad j = k+1,...,n.$$
6. Обчислюємо величини $\sum_{j=k}^{n} \left|t_j^+\right|$ та $\sum_{j=k}^{n} \left|t_j^-\right|$ і порівнюємо їх. Якщо перша величина більша за другу, то покладаємо $w_k = w_k^+$, у протилежному випадку $w_k = w_k^-$.

При рядковій схемі зберігання матриці U^T обчислення величин t_j, t_j^+, t_j^- легко розпаралелюються між процесами CPU. Якщо ж U^T зберігається у відповідності зі стовпчиковою схемою, то реалізація описаного алгоритму породжує послідовну програму. Ситуація дещо покращується, якщо обчислення величин t_j, t_j^+, t_j^-, $(j = k+1,...,n)$ виконувати одночасно на двох процесах CPU.

Розв'язування систем (7) – (9), з огляду на спосіб зберігання матриці, також ефективніше можна реалізувати на CPU.

Після визначення числа обумовленості за формулами (10), (11) в процесі реалізації того чи іншого прямого методу здійснюється перевірка коректності постановки задачі та обчислюється оцінка спадкової похибки. Якщо значення $\mathrm{cond}A$ в комп'ютері задовольняє умові:

$$1.0 + 1/\mathrm{cond}\,A = 1.0, \tag{12}$$

то матриця вважається виродженою у межах машинної точності. Якщо матриця не класифікується за (12) як вироджена, але

$$E_A \times \mathrm{cond}A \geq 1,$$

то задача вважається некоректно поставленою при заданій точності елементів матриці, тому не можна гарантувати достовірність обчисленого розв'язку.

Верхня межа "відносної" спадкової похибки розв'язку визначається за формулою:

$$\frac{\|x - \bar{x}\|}{\|\bar{x}\|} = E \leq \mathrm{cond}A \times \frac{E_A + E_b}{1 - E_b}, \qquad E_b < 1,$$

де $\bar{x}$ – точний розв'язок системи (2), x – точний розв'язок системи (1), (3), E_A, E_b – максимальні відносні похибки елементів матриці та правої частини відповідно. Як правило, їх вказує користувач. В іншому випадку $E_A = E_b = macheps$, де $macheps$ – найменше в комп'ютері число з плаваючою комою, для якого виконується умова: $1 + macheps > 1$.

Оцінка обчислювальної похибки виконується за ідеєю ітераційного уточнення отриманого розв'язку [17], один крок якого реалізується способом накопичення скалярного добутку за схемою:

$$x^{(0)} = x,$$
$$r^{(s)} = b - A \times x^{(s)},$$
$$A \times \Delta x^{(s)} = r^{(s)},$$
$$x^{(s+1)} = x^{(s)} + \Delta x^{(s)}, \; s = 0, 1, 2, \ldots.$$

Оцінка обчислювальної похибки розв'язку визначається за формулою

$$E_1 < \left\| A^{-1} \times r^{(0)} \right\| / \left\| x^{(1)} \right\|,$$

де $x^{(1)}$ – наближення точного розв'язку, яке отримано за один крок ітераційного уточнення.

Гібридний алгоритм *LU*-факторизації методом Гаусса. Алгоритм LU -факторизації матриці системи виду (1) приводить матрицю A до вигляду $A = PLU$, де P – вектор перестановок, L – нижня трикутна матриця (з одиницями на головній діагоналі), U – верхня трикутна матриця.

Коротко опишемо звичайний блочний алгоритм LU -факторизації з вибором максимального елемента (для однопроцесорного комп'ютера) [17]. Будемо вважати, що матриця A порядку n розділена на квадратні блоки розміру $s \times s$. Не втрачаючи загальності міркувань, вважатимемо, що n/s – ціле число.

На k-ому кроці алгоритму ($k = 1,2,...$) підматрицю $A^{(k)}$ (діагональний блок матриці A) порядку $r = n - (k-1)s$, яка містить останні r рядків і r стовпчиків матриці A, представимо у вигляді:

$$\begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} = P \begin{pmatrix} L_{11} & 0 \\ L_{21} & L_{22} \end{pmatrix} \begin{pmatrix} U_{11} & U_{12} \\ 0 & U_{22} \end{pmatrix} = P \begin{pmatrix} L_{11}U_{11} & L_{11}U_{12} \\ L_{21}U_{11} & L_{21}U_{12} + L_{22}U_{22} \end{pmatrix},$$

де блок A_{11} має розмір $s \times s$, блок A_{12} – розмір $s \times (r-s)$, блок A_{21} – розмір $(r-s) \times s$, блок A_{22} – $(r-s) \times (r-s)$.

Спочатку проводимо послідовність гауссових перетворень [22] над частиною матриці (Рис. 1), яка складається з блоків A_{11} та A_{21}, за формулами:

$$l_{im} = \frac{a_{im}}{a_{mm}}, \qquad l_{ij} = a_{ij} - l_{im} a_{mj}, \tag{13}$$

де $i = \overline{1,s}$, $m = \overline{1,s}$, $j = \overline{m+1,r}$.

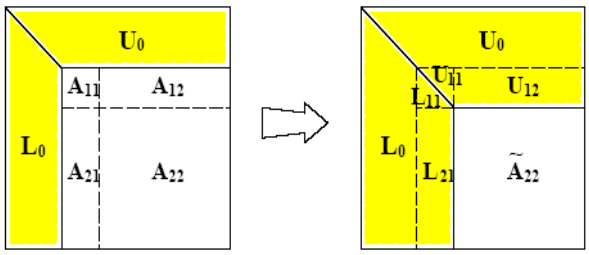

РИС. 1. Схема реалізації блочного алгоритму LU -факторизації

Таким чином, отримаємо дві підматриці L_{11} та L_{21}. Далі обчислюємо підматриці U_{11} та $\tilde{A}_{22}$ (остання записується на місце A_{22}):

$$U_{12} = (L_{11})^{-1} A_{12}, \tag{14}$$

$$\tilde{A}_{22} = A_{22} - L_{21}U_{12} = L_{22}U_{22}. \tag{15}$$

Обчислення повторюються, збільшуючи значення k на одиницю та розглядаючи блок $\tilde{A}_{22}$ в якості підматриці $A^{(k)}$.

Розглянемо реалізацію гібридного алгоритму LU -факторизації на архітектурі g CPU + g GPU (g – кількість обчислювальних пристроїв). Розташуємо матрицю у CPU гібридного комп'ютера таким чином, щоб $A_{ij}^{*} \in P_{ij}$, де i та j – декартові координати на решітці комп'ютерної топології, розміру $p \times q$. Не втрачаючи загальності, припускатимемо, що $m = n/(spq)$ – ціле число ($s \times s$ –розмір блоку матриці, n – кількість MPI-процесів).

Двовимірний блочний алгоритм LU -факторизації матриці для гібридного комп'ютера виконується в такому порядку:

1) в MPI-процесі на CPU провідного стовпчика процесорів решітки виконується модифікація блоків A_{11} та A_{21} (рис. 1) за формулою (13). Проводиться пошук максимального елемента (обмін рядків у випадку, якщо максимальний елемент виявився не у провідному процесі);

2) провідним процесом на CPU обчислюється обернена підматриця L_{11}. Оскільки блок A_{11} повністю розташований в одному процесі, обмінів з іншими процесами не відбувається;

3) обернена підматриця L_{11} розсилається всім процесам провідного рядка процесорів CPU;

4) у процесорах GPU провідного рядка паралельно виконуються операції над блоками матриць L_{11}^{-1} та A_{12} за формулою (14), отримуючи блок U_{12} (Рис. 1);

5) процеси на CPU провідного рядка розсилають процесам вертикально (вниз і вверх) блоки U_{12}, а процеси провідного стовпчика – горизонтально (вправо і вліво) блоки L_{21} (Рис. 1);

6) у процесорах GPU проводиться модифікація частин A_{22} згідно формули (15). Обчислення проводяться після того, як кожен процес CPU скопіював на GPU відповідні частини масивів A_{22}, L_{21}, U_{12};

7) після виконання обчислень порції $\tilde{A}_{22}$ копіюються з GPU у відповідні процеси на CPU.

Для ефективного виконання операцій над матрицями та векторами на CPU використано відповідні програми бібліотеки програм Intel MKL [4], а на GPU – бібліотеки програм cuBLAS [6].

Гібридний алгоритм LL^T-факторизації методом Холецького. При розв'язуванні СЛАР виду (1) з симетричною додатно означеною матрицею алгоритм LL^T-факторизації зводить матрицю A до виду $A = LL^T$, де L – нижня трикутна матриця, L^T – верхня трикутна матриця.

Розглянемо блочний алгоритм LL^T-факторизації для однопроцесорного комп'ютера. При цьому будемо вважати, що матриці A та L порядку n розділені на квадратні блоки розміром $s \times s$.

На k-ому кроці алгоритму $(k = 1,2,...)$ підматрицю $A^{(k)}$ (діагональний блок матриці A) порядку $r = n-(k-1)\times s$, яка містить останні r рядків та r стовпчиків матриці A, запишемо у вигляді

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

де блок A_{11} має розмір $s \times s$, блок A_{12} – $s \times (r-s)$, блок A_{21} – $(r-s) \times s$, блок A_{22} – $(r-s) \times (r-s)$.

Для реалізації звичайного блочного алгоритму на кожному k-кроці необхідно виконати наступні дії [22]:

– LL^T-факторизація підматриці A_{11}, отримуючи підматрицю L_{11}:

$$A_{11} = L_{11}L_{11}^T; \tag{16}$$

– модифікація підматриці L_{21}:

$$\tilde{L}_{21} = L_{21}\left(L_{11}^T\right)^{-1}; \tag{17}$$

– обчислення підматриці A_{22}:

$$\tilde{A}_{22} = A_{22} - \tilde{L}_{21}\tilde{L}_{21}^T = L_{22}L_{22}^T. \tag{18}$$

Таким чином, на k-ому кроці $(k = 1, 2, ...)$ отримуємо модифіковану частину матриці L. На наступний крок переходимо з підматрицею $\tilde{A}_{22}$ (Рис. 2).

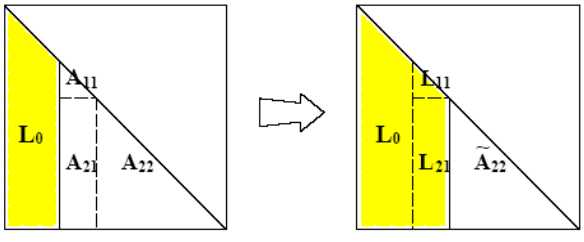

РИС. 2. Схема реалізації блочного алгоритму LL^T-факторизації

Розглянемо реалізацію гібридного алгоритму LL^T -факторизації на архітектурі g CPU + g GPU після блочно-циклічного розподілу матриці системи між процесами на CPU таким чином, щоб $A_{ij}^* \in P_{ij}$, де i та j − декартові координати на решітці комп'ютерної топології розміру $p \times q$.

На кожному *k*-ому кроці алгоритму $(k = 1, 2, ...)$ виконується така послідовність дій:

1) провідний MPI-процес на CPU, що містить підматрицю (блок) A_{11}, виконує її факторизацію за формулою (16), одержуючи підматрицю L_{11};

2) провідний процес розсилає обернену підматрицю $\left(L_{11}^T\right)^{-1}$ всім процесам по вертикалі;

3) всі процесори GPU провідного стовпчика незалежно один від одного модифікують відповідні частини матриці L_{21} (панель L_{21} на Рис. 2) за формулою (17). Обчислення проводяться після того, як кожен процес на CPU скопіював відповідну частину L_{21} на GPU. Одержані в результаті обчислень порції L_{21} копіюються з процесорів GPU назад до процесів CPU;

4) процеси на CPU провідного стовпчика розсилають всім процесам по горизонталі блоки матриці $\tilde{L}_{21}$ та незалежно виконують їх транспонування, одержуючи $\tilde{L}_{21}{}^T$;

5) кожен процесор GPU незалежно один від одного виконує модифікацію своїх порцій підматриці A_{22} за формулою (18), отримуючи $\tilde{A}_{22}$. Обчислення проводяться після того, як кожен процес CPU скопіював на процесор GPU відповідні частини масивів A_{22}, L_{21}, L_{22}. Отримані в результаті обчислень порції $\tilde{A}_{22}$ копіюються з процесорів GPU у відповідні процеси CPU.

У програмах, що реалізують гібридний алгоритм LL^T -факторизації, для виконання матрично-векторних операцій використані програми Intel MKL та cuBLAS.

Алгоритми розв'язування трикутних систем. Для розв'язування СЛАР виду (1) прямим методом, після проведення факторизації вихідної матриці, необхідно розв'язати дві системи з трикутними матрицями.

Наприклад, після LU -факторизації матриці розв'язуються такі системи:

− $Ly = b$ з нижньою трикутною матрицею L (цю задачу часто називають прямою підстановкою або прямим ходом);

− $Ux = y$ з верхньою трикутною матрицею U (зворотній хід).

Кількість операцій, що затрачається для розв'язування трикутних систем значно менша, ніж для факторизації матриці, тому ці задачі ефективно можна реалізувати, розпаралелюючи обчислення лише на CPU гібридного комп'ютера. В програмах, що реалізують дані гібридні алгоритми, розв'язування трикутних систем виконується за допомогою відповідних програм бібліотеки Intel MKL.

Експериментальне дослідження гібридних алгоритмів. Апробацію запропонованих гібридних алгоритмів проведено на суперкомп'ютері гібридної архітектури СКІТ-4, використовуючи різну кількість процесорних вузлів CPU та графічних прискорювачів з такими технічними характеристиками [26]: CPU – Intel(R) Xeon(R) CPU E5-26700, тактова частота 2.60 GHz, швидкість 8 GT/s, кеш-пам'ять 20 MB, у вузлі: 2 CPU по 8 ядер + Hyperthrefding = 32 ядра, Max Memory Size 384 GB; GPU – Nvidia Tesla M2050, пам'ять 3 GB, пікова продуктивність 515 Gflops.

На графіках (Рис. 3) показано прискорення розробленого гібридного алгоритму *LU*-факторизації при розв'язуванні на різній гібридній архітектурі g CPU + g GPU (g – кількість обчислювальних елементів) лінійних систем із щільними матрицями різного порядку (8192, 10240, 16374).

Як видно з графіків, для задачі невеликого порядку (8192) прискорення обчислень дещо спадає при збільшенні кількості процесорів до 4 та 8. Це пояснюється недостатнім заповненням GPU-процесорів при виконанні матрично-векторних операцій та збільшенням комунікаційних зв'язків між CPU та GPU. Але з ростом порядку матриці прискорення суттєво зростає при збільшенні кількості процесорів**.** При розв'язуванні СЛАР порядку 16374 на архітектурі $8\text{CPU}+8\text{GPU}$ отримано прискорення приблизно в 6 раз у порівнянні з прискоренням на архітектурі $1\text{CPU}+1\text{GPU}$.

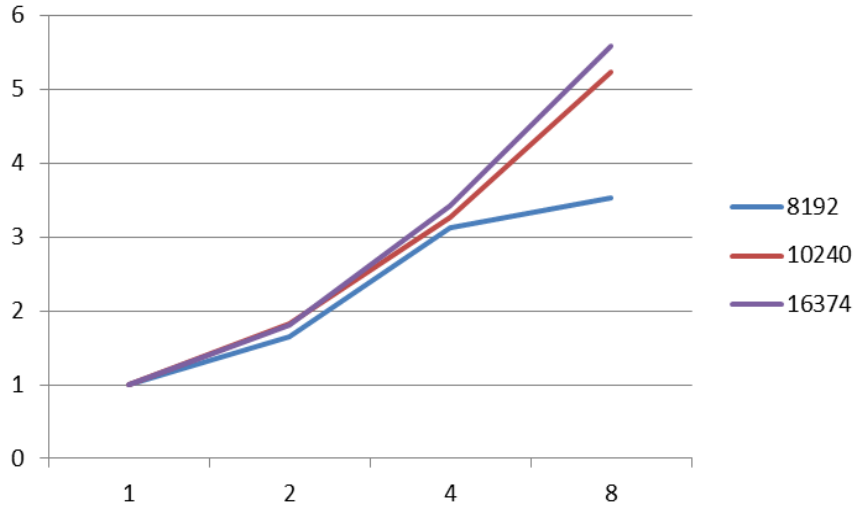

РИС 3. Прискорення гібридного блочного алгоритму LU -факторизації на СКІТ-4

Аналогічні результати отримуємо і при розв'язуванні СЛАР гібридним алгоритмом LL^T -факторизації на СКІТ-4.

Висновки. У роботі запропоновано нові гібридні двовимірні блочно-циклічні алгоритми для розв'язування СЛАР із щільними невиродженими матрицями на багатоядерних комп'ютерах MIMD-архітектури з графічними прискорювачами, які передбачають комп'ютерне дослідження математичних властивостей задачі з наближеними даними та оцінку достовірності результатів.

Розроблені гібридні алгоритми увійшли до складу бібліотеки програм з обчислювальної математики Inparlib_G, що функціонує на суперкомп'ютері СКІТ-4 та використовується при розв'язуванні прикладних задач.

У подальшому передбачається створення інтелектуальної системи комп'ютерної математики для автоматичного вибору ефективного алгоритму, топології паралельного комп'ютера, а також розв'язування лінійних систем з різною структурою матриць на гібридних комп'ютерах, використовуючи програми з бібліотеки програм Inparlib_G.

Список літератури

1. TOP 500 supercomputers 2019. http://www.top500.org/lists/2019/11/.
2. NetLib. https://www.netlib.org/.
3. AMD math LibM. http://developer.amd.com/amd-aocl/amd-math-library-libm/.
4. Math Kernel Library. https://software.intel.com./en-us/mkl/.
5. Боресков А.В., Харламов А.А. Основы работы с технологией CUDA. М.: Пресс, 2010. 232 с.
6. cuBLAS. https://developer.nvidia.com/cublas/.
7. cuSparse Library. https://developer.nvidia.com/cusparse/.

8. CULA. https://developer.nvidia.com/em-photonics-cula-tools
9. MAGMA Software. https://icl.cs.utk.edu/magma/.
10. CUSP. https://developer.nvidia.com/cusp/.
11. MathCAD. https://www.ptc.com/ru/products/mathcad/.
12. Maple. https://www.maplesoft.com/products/Maple/.
13. Mathematica. https://www.wolfram.com/mathematica/.
14. Matlab. https://www.mathworks.com/help/matlab/.
15. Воеводин В.В., Воеводин Вл.В. Параллельные вычисления. СПб.: БХВ-Петербург, 2002. 608 с.
16. Химич А.Н., Молчанов И.Н, Мова В.И. и др. Численное программное обеспечение MIMD–компьютера Инпарком. Киев: Наукова думка, 2007. 222 с.
17. Химич А.Н., Молчанов И.Н., Попов А.В., Чистякова Т.В., Яковлев М.Ф. Параллельные алгоритмы решения задач вычислительной математики. Киев: Наукова думка, 2008. 247 с.
18. Сергиенко И.В., Молчанов И.Н., Химич А.Н. Интеллектуальные технологии высокопроизводительных вычислений. *Кибернетика и системный анализ*. 2010. Т. 46, № 5. С. 164–176.
19. Ильин В.П. О некоторых проблемах «облачного» математического моделирования. *Вестник Южно-Уральского государственного университета. Серия вычислительная математика и информатика*. 2014. Т. 3, Вып. 1. С. 68–79.
20. Тарнавский Г.А., Алиев А.В. Математическое моделирование: Основные сегменты, их особенности и проблемы. *Вычислительные методы и программирование*. 2007. Т. 8. С. 297–310.
21. Немнюгин С.А., Стесик О.Л. Параллельное программирование для многопроцессорных вычислительных систем. СПб.: БХВ-Петербург, 2002. 400 с.
22. Молчанов И.Н. Машинные методы решения прикладных задач. Алгебра, приближение функций. Киев: Наукова думка, 1987. 288 с.
23. Wilkinson J.H. Rounding Errors in Algebraic Processes. London: H.W. Staat. Off, 1963. 161 p.
24. Уилкинсон Дж.Х., К Райнш. Справочник алгоритмов на языке Алгол. Линейная алгебра. М.: Машиностроение, 1976. 389 с.
25. Воеводин В.В. Ошибки округлений и устойчивость в прямых методах линейной алгебры. М.: Изд. ВЦ МГУ, 1969. 153 с.
26. Головинський А.Л., Маленко А.Л., Сергієнко І.В., Тульчинський В.Г. Енергоефективний суперкомп'ютер СКІТ-4. *Вісник НАН України*. 2013. № 2. С. 50–59.

Одержано 19.03.2020

Хіміч Олександр Миколайович,
доктор фізико-математичних наук, професор, заступник директора
Інституту кібернетики імені В.М. Глушкова НАН України, Київ,
https://orcid.org/0000-0002-8103-4223

Полянко Віктор Володимирович,
кандидат фізико-математичних наук, науковий співробітник
Інституту кібернетики імені В.М. Глушкова НАН України, Київ,

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

УДК 519.6

А.Н. Химич [1], В.В. Полянко [1], Т.В. Чистякова [1 *]

Параллельные алгоритмы решения линейных систем на гибридных компьютерах

[1] Институт кибернетики имени В.М. Глушкова НАН Украины, Киев
[*] Переписка: tamara.chistjakova@gmail.com

Введение. В настоящее время в науке и инженерии постоянно появляются новые вычислительные задачи с большими объемами данных, для решения которых необходимо использовать мощные суперкомпьютеры. Большинство таких задач сводится к решению систем линейных алгебраических уравнений (СЛАУ). Главными проблемами решения задач на компьютере является получение достоверных решений при минимальных затратах вычислительных ресурсов. Однако задача, которая решается на компьютере, всегда имеет приближенные данные по отношению к исходной задачи (из-за погрешности исходных данных, из-за погрешности ввода числовых данных в компьютер и т. п.). Таким образом, математические свойства компьютерной задачи могут существенно отличаться от свойств исходной задачи. Возникает необходимость решать задачи с учетом приближенных данных, а также анализировать получаемые компьютерные результаты. Несмотря на значительные результаты исследований в области линейной алгебры, работы в направлении преодоления существующих проблем компьютерного решения задач с приближенными данными, еще больше усугубляются при использовании современных суперкомпьютеров, не теряют своей значимости и требуют дальнейшего развития. На сегодня самые высокопроизводительные суперкомпьютеры – это параллельные компьютеры с графическими процессорами. Архитектурные и технологические особенности этих компьютеров дают возможность значительно повысить эффективность решения задач больших объемов при относительно небольших энергозатратах.

Цель работы. Разработать новые параллельные алгоритмы решения систем линейных алгебраических уравнений с приближенными данными на суперкомпьютерах с графическими процессорами, для автоматической настройки алгоритма на эффективную архитектуру компьютера и выявленные в компьютере математические свойства задачи, а также ее решение с оценками достоверности полученных результатов.

Результаты. Описана методология создания параллельных алгоритмов для суперкомпьютеров с графическими процессорами, реализующих исследование математических свойств линейных систем с приближенными данными и решение с анализом достоверности полученных результатов. Приведены результаты вычислительных экспериментов на суперкомпьютере СКИТ-4.

Выводы. Созданы параллельные алгоритмы для исследования и решения линейных систем с приближенными данными на суперкомпьютерах с графическими процессорами. Численные эксперименты при использовании новых алгоритмов показали существенное ускорение вычислений с гарантией достоверности получаемых результатов.

Ключевые слова: системы линейных алгебраических уравнений, гибридный алгоритм, приближенные данные, достоверность результатов, компьютеры с графическими процессорами.

MSC 28A12, 90C25

Alexander Khimich [1], Victor Polyanko [1], Tamara Chistyakova [1 *]

Parallel Algorithms for Solving Linear Systems on Hybrid Computers

[1] *V.M. Glushkov Institute of Cybernetics of the NAS of Ukraine, Kyiv*

[*] *Correspondence: tamara.chistjakova@gmail.com*

Introduction. At present, in science and technology, new computational problems constantly arise with large volumes of data, the solution of which requires the use of powerful supercomputers. Most of these problems come down to solving systems of linear algebraic equations (SLAE). The main problem of solving problems on a computer is to obtain reliable solutions with minimal computing resources. However, the problem that is solved on a computer always contains approximate data regarding the original task (due to errors in the initial data, errors when entering numerical data into the computer, etc.). Thus, the mathematical properties of a computer problem can differ significantly from the properties of the original problem. It is necessary to solve problems taking into account approximate data and analyze computer results. Despite the significant results of research in the field of linear algebra, work in the direction of overcoming the existing problems of computer solving problems with approximate data is further aggravated by the use of contemporary supercomputers, do not lose their significance and require further development. Today, the most high-performance supercomputers are parallel ones with graphic processors. The architectural and technological features of these computers make

it possible to significantly increase the efficiency of solving problems of large volumes at relatively low energy costs.

The purpose of the article is to develop new parallel algorithms for solving systems of linear algebraic equations with approximate data on supercomputers with graphic processors that implement the automatic adjustment of the algorithms to the effective computer architecture and the mathematical properties of the problem, identified in the computer, as well with estimates of the reliability of the results.

Results. A methodology for creating parallel algorithms for supercomputers with graphic processors that implement the study of the mathematical properties of linear systems with approximate data and the algorithms with the analysis of the reliability of the results are described. The results of computational experiments on the SKIT-4 supercomputer are presented.

Conclusions. Parallel algorithms have been created for investigating and solving linear systems with approximate data on supercomputers with graphic processors. Numerical experiments with the new algorithms showed a significant acceleration of calculations with a guarantee of the reliability of the results.

Keywords: systems of linear algebraic equations, hybrid algorithm, approximate data, reliability of the results, GPU computers.