УДК 004.75+519.612

Д. Федасюк, П. Сердюк, Ю. Семчишин
Національний університет "Львівська політехніка",
кафедра програмного забезпечення

ІЄРАРХІЧНЕ РОЗПОДІЛЕННЯ РОЗВ'ЯЗУВАННЯ БЛОКОВО-СТРІЧКОВИХ СИСТЕМ ЛІНІЙНИХ РІВНЯНЬ ВЕЛИКОЇ РОЗМІРНОСТІ

Запропоновано застосування розподілення обчислень з використанням ієрархічного підходу для розв'язування систем лінійних рівнянь великої розмірності.

An application of distributed computing using hierarchical approach to solving high dimensional systems of linear equations is suggested.

Вступ

Розвиток обчислювальної техніки спричинив перехід до нових, складніших математичних моделей у вигляді систем диференційних рівнянь. Їх розв'язування потребує значних обчислювальних ресурсів для роботи з розрідженими системами лінійних алгебраїчних рівнянь (СЛАР) [1]. Зокрема, вирішення задач теплового проектування електронних пристроїв методами скінченних елементів (Finite Element Method, FEM) та скінченних об'ємів (Finite Volume Method, FVM) зводиться до розв'язування СЛАР з розрідженими матрицями надвеликих розмірностей, як правило, асиметричних [2].

Огляд

Для розв'язування СЛАР з розрідженими матрицями великих розмірностей розроблено значну кількість алгоритмів факторизації, зокрема лівосторонній, правосторонній, двосторонній, фронтальний, мультифронтальний. Саме мультифронтальний алгоритм факторизації таких матриць, вперше описаний в [3] та надалі багато разів модифікований та вдосконалений, як, наприклад, в [4], є сьогодні найвживанішим та в загальному випадку найефективнішим.

Набір інструментів для роботи з розрідженими матрицями `SPARSKIT` (http://www-users.cs.umn.edu/~saad/software/SPARSKIT/sparskit.html) розроблений Ю. Саадом (Y. Saad) [5]. Вихідні коди системи `SPARSKIT` розповсюджуються вільно згідно з LGPL (GNU Lesser General Public License). Проте варто зауважити, що останню версію `SPARSKIT` розроблено у 1994 році мовою програмування `ForTran` без можливості організації паралельних чи розподілених обчислень засобами самого продукту, що робить його незастосовним у сучасних умовах.

Інший продукт, об'єктно-орієнтований розв'язувач розріджених лінійних рівнянь `SPOOLES` (SParse Object Oriented Linear Equations Solver, http://www.netlib.org/linalg/spooles/spooles.2.2.html), розроблений К. Ешкрафтом (C. Ashcraft) та Р. Ґраймзом (R. Grimes) в 1999 році мовою програмування `C` [6]. У цьому продукті реалізовано паралельний лівосторонній алгоритм факторизації симетричних матриць та матриць із симетричним шаблоном. Саме таке обмеження на симетричність шаблону матриці в поєднанні з неможливістю організації розподілених обчислень засобами самого розв'язувача робить сумнівною доцільність його подальшого використання в сучасних умовах.

Ще один засіб розв'язування СЛАР з розрідженими матрицями `PARDISO` (http://www.pardiso-project.org/) розроблений О. Шенком (O. Schenk) та К. Ґьортнером (K. Gärtner) у 2005 році мовами програмування `ForTran` та `C` [7]. Продукт містить реалізацію паралельного двостороннього алгоритму факторизації матриць із симетричним шаблоном. Незважаючи на обмеження на симетричність шаблону матриці, цей інструмент увійшов до складу `MKL` (Math Kernel Library, http://www.intel.com/software/products/mkl) – комерційного продукту для кластерних обчислень від `Intel`, що, вочевидь, свідчить про високу його якість [8]. Проте за відсутності вартісного

кластера використання PARDISO стає недоцільним, оскільки цей інструмент не підтримує розподілених обчислень.

Багатофронтальний паралельний розв'язувач MUMPS (A MUltifrontal Massively Parallel Solver, http://www.enseeiht.fr/apo/MUMPS/) розроблений П. Емстоєм (P. Amestoy), І. С. Даффом (I. S. Duff) та їхньою командою у 2003 році мовою програмування ForTran [9]. У MUMPS реалізовано паралельний багатофронтальний алгоритм факторизації симетричних матриць та матриць із симетричним шаблоном. Незважаючи на те, що MUMPS реалізовано мовою програмування ForTran, доступний також інтерфейс програміста мовою C. На відміну від усіх перелічених вище, цей продукт може працювати на обчислювальних машинах з розподіленою пам'яттю, проте лише за наявності на них середовища MPI (Message Passing Interface), що, на жаль, робить неможливою адаптацію MUMPS до розподілених обчислень.

Загалом, жоден з існуючих сьогодні засобів розв'язування СЛАР з розрідженими матрицями не може бути застосовним для розподілених обчислень в умовах гетерогенних обчислювальних ресурсів, об'єднаних у денормалізовані мережі.

Постановка задачі

Розглянемо лінійну систему рівнянь

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

де A — блоково-стрічкова матриця розміру $n \times n$, b — стовпець змінних, x — вектор невідомих, який необхідно знайти.

Для розв'язування матричних рівнянь запропоновано нижче описаний алгоритм, що ґрунтується на багатофронтальному підході та ієрархічному розподіленні даних матриці. Розглянемо запропоновані особливості алгоритму розподілення для блокових матриць.

Алгоритм має прямий та зворотний хід, який може виконуватись одночасно на різних обчислювальних ресурсах для різних блоків матриці.

Для спрощення розглянемо матрицю A, яка має наведену нижче просту блокову структуру:

$$A = \begin{pmatrix} A_{11} & A_{12} & O \\ A_{21} & A_{22} & A_{23} \\ O & A_{32} & A_{33} \end{pmatrix}, \tag{2}$$

де A_{11} — матриця розміром $n_1 \times n_1$, A_{12} — матриця розміром $n_1 \times n_2$ тощо, O — матриці, всі елементи яких нулі. Як правило, $n_2 << n_1, n_3$ для матриць, отриманих за допомогою числових обчислень за методом скінченних чи граничних елементів або різниць. Числа n_1, n_2, n_3, … називатимемо розмірами блоків матриці.

1. Розподілення ресурсів та визначення взаємозалежностей. Розбиваємо матрицю A на блоки так:

$$A = \begin{pmatrix} A_{11} & A_{12} & O \\ A_{21} & A_{22} & A_{23} \\ O & A_{32} & A_{33} \end{pmatrix} = \begin{pmatrix} A_{11} & A_{12} & O \\ O & O & O \\ O & O & O \end{pmatrix} + \begin{pmatrix} O & O & O \\ A_{21} & A_{22} & A_{23} \\ O & O & O \end{pmatrix} + \begin{pmatrix} O & O & O \\ O & O & O \\ O & A_{32} & A_{33} \end{pmatrix}. \tag{3}$$

Переформуємо цю систему у три окремі блоки рівнянь,

$$\begin{pmatrix} A_{11} & A_{12} & O \end{pmatrix} \begin{pmatrix} x_1 & x_2 & x_3 \end{pmatrix} = b_1, \tag{3.1}$$

$$\begin{pmatrix} A_{21} & A_{22} & A_{23} \end{pmatrix} \begin{pmatrix} x_1 & x_2 & x_3 \end{pmatrix} = b_2, \tag{3.2}$$

$$\begin{pmatrix} O & A_{32} & A_{33} \end{pmatrix} \begin{pmatrix} x_1 & x_2 & x_3 \end{pmatrix} = b_3, \tag{3.3}$$

кожне з яких розв'язуватиметься на окремому обчислювальному ресурсі. При цьому рівняння (3.2) очікуватиме на розв'язування рівнянь (3.1) і (3.3).

У разі, якщо матриця A_{11} чи A_{33} є надто великої розмірності, для покращання швидкодії крок 1 можна повторити, розбиваючи матриці на більшу кількість матриць A_{ii}.

У такому випадку розподілення ресурсів буде ієрархічним, тобто робота з розподілення обчислень та збирання результатів також буде розподіленою та виконуватиметься виконавцями разом з виконанням частин самого завдання.

2. Нехай відокремлений блок, що має бути розв'язаним таким ресурсом, має вигляд:

$$(A_{i0}A_{i1...}A_{ii}...A_{in})x = b_i\,. \tag{4}$$

За допомогою LU-факторизації матриці A_{ii} зводимо (4) до вигляду:

$$(\tilde{A}_{i0}\tilde{A}_{i1...}I...\tilde{A}_{in})x = \tilde{b}_i\,, \tag{5}$$

де I — одинична матриця, $\tilde{A}_{ij}$, $\tilde{b}$ — відповідно матриці рівняння та вектор-стовпець, отримані у результаті лінійних перетворень під час LU-факторизації матриці A_{ii}. Очевидно, що з матрицями A_{ij}, які складаються з виключно нульових елементів, жодних операцій не проводиться.

Після закінчення цього кроку обчислювальний ресурс очікує закінчення кроку 4 — передачі невідомих у ненульових блоках $\tilde{A}_{ij}$.

3. Після кроку 2 обчислювальний ресурс передає дані іншому обчислювальному ресурсу, який очікує розв'язання, оскільки має спільні невідомі із вхідною матрицею.

Нехай маємо дві системи рівнянь вигляду (5), які мають спільні невідомі у *i*-му блоці:

$$\begin{matrix}(\tilde{A}_{i0}\ \tilde{A}_{i1...}\tilde{A}_{i,i-1}\ \ I\ \ \tilde{A}_{i,i+1}...\tilde{A}_{in})x = \tilde{b}_i \\ (A_{j0}\ A_{j1...}A_{j,i-1}\ A_{j,i}\ A_{j,i+1}..A_{jn})x = b_j\end{matrix}\,. \tag{6}$$

Приводимо матрицю до вигляду, в якому всі елементи у стовпцях *i*-го блоку дорівнюють нулю, за допомогою лінійних перетворень:

$$\begin{matrix}(\tilde{\mathrm{A}}_{\mathrm{i0}}\ \tilde{\mathrm{A}}_{\mathrm{i1...}}\tilde{\mathrm{A}}_{\mathrm{i,i-1}}\ \ \mathrm{I}\ \tilde{\mathrm{A}}_{\mathrm{i,i+1}}...\tilde{\mathrm{A}}_{\mathrm{in}})\mathrm{x} = \tilde{\mathrm{b}}_{\mathrm{i}} \\ (\tilde{\mathrm{A}}_{\mathrm{j0}}\ \tilde{\mathrm{A}}_{\mathrm{j1...}}\tilde{\mathrm{A}}_{\mathrm{j,i-1}}\ \mathrm{O}\ \tilde{\mathrm{A}}_{\mathrm{j,i+1}}..\tilde{\mathrm{A}}_{\mathrm{jn}})\mathrm{x} = \mathrm{b}_{\mathrm{j}}\end{matrix}\,, \tag{7}$$

де $\tilde{A}_{j,k} = A_{jk} - A_{j,i}\cdot\tilde{A}_{i,k}$. У випадку, якщо така матриця не очікує більше жодних обчислень, переходимо для неї до кроку 2. У випадку отримання одиничної матриці переходимо до кроку 4.

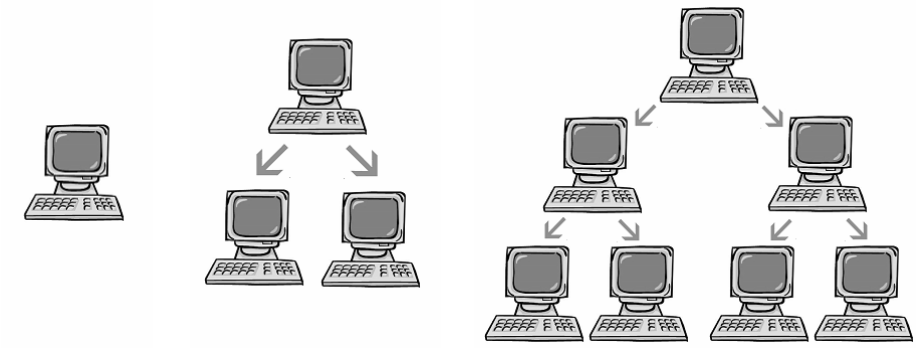
4. Отримавши матрицю вигляду

$$(\mathrm{O}\,\mathrm{O}....\mathrm{O}\ \ I\,\mathrm{O}...\mathrm{O})x_i = \tilde{b}_i\,,$$

передаємо значення невідомих x_i, що знаходяться у векторі $\tilde{b}_i$, на основний центр розподілення для їх подальшої обробки користувачем, та на інші обчислювальні ресурси, які містять у блоках ці невідомі.

Результати

З метою проведення експериментів було розроблено програмний засіб `SparseMatrix 1.11` для розподіленого розв'язування СЛАР великої розмірності. Платформою розроблення було обрано `Microsoft .NET 2.0`, розроблення проводилась мовою `C#` у середовищі `Microsoft Visual Studio 2005`.



а *б* *в*

Рис. 1. Конфігурації обчислювальної мережі:
а — один комп'ютер; б — три комп'ютери; в — сім комп'ютерів

Час розв'язування тестових СЛАР

К-сть ПК	Розмір блоку	Розмір СЛАР							
		64	128	256	512	1024	2048	4096	8192
1	1	0,016	0,016	0,016	0,063	0,109	0,422	2,078	52,750
	4	0,016	0,016	0,031	0,063	0,250	0,938	3,203	14,047
	16	0,016	0,031	0,063	0,266	0,844	3,203	12,359	50,031
	64	0,000	0,031	0,203	0,813	3,234	12,875	48,031	200,734
	256			0,328	2,750	15,750	70,266	277,109	821,344
	1024					11,234	131,844	762,000	3357,313
3	1	0,016	0,000	0,016	0,031	0,063	0,156	0,656	34,234
	4	0,000	0,016	0,031	0,047	0,141	0,500	1,594	6,406
	16	0,016	0,016	0,047	0,188	0,547	2,234	8,516	33,938
	64	0,000	0,016	0,141	0,547	2,281	9,250	34,375	145,813
	256			0,328	2,031	11,031	49,188	192,922	571,203
	1024					11,234	99,641	543,453	2357,109
7	1	0,016	0,000	0,016	0,016	0,047	0,094	0,422	21,719
	4	0,000	0,016	0,031	0,047	0,141	0,406	1,156	4,672
	16	0,016	0,016	0,047	0,156	0,500	1,906	7,281	29,219
	64	0,000	0,016	0,125	0,484	1,953	7,984	29,641	129,172
	256			0,328	2,031	10,094	43,016	162,750	512,016
	1024					11,234	99,641	501,063	2069,609

Для експериментальних досліджень було згенеровано 54 різні блоково-стрічкові симетричні додатно визначені матриці. Розміри блоків змінювались від 1×1 до 1024×1024 елементів, розміри самих матриць — від 1×1 до 8192×8192 елементів. Для того, щоб стала можливою обробка ще більших за розміром матриць, у наступних версіях програмного засобу планується використання структур даних, оптимізованих для розріджених матриць.

Набір тестових СЛАР було розв'язано тричі при різних конфігураціях обчислювальної мережі (рис. 1): з одним (див. рис. 1, а), трьома (див. рис. 1, б) та сімома (див. рис. 1, в) комп'ютерами відповідно.

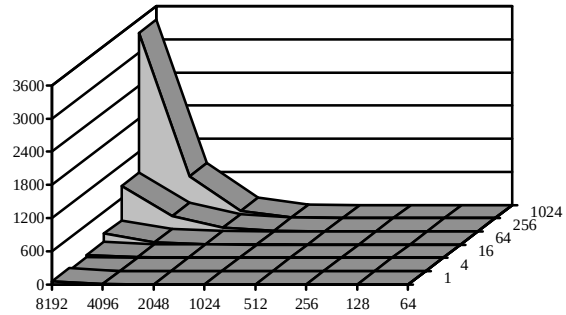
Під час розв'язування набору тестових СЛАР було отримано значення часу виконання для різних конфігурацій обчислювальної мережі, а також розмірів блоків та самих матриць, подані у таблиці.

Залежність часу розв'язування СЛАР від розмірів матриць та їх блоків зображено на рис. 2: для одного (див. рис. 2, а), трьох (див. рис. 2, б) та семи (див. рис. 2, в) комп'ютерів в обчислювальній мережі.

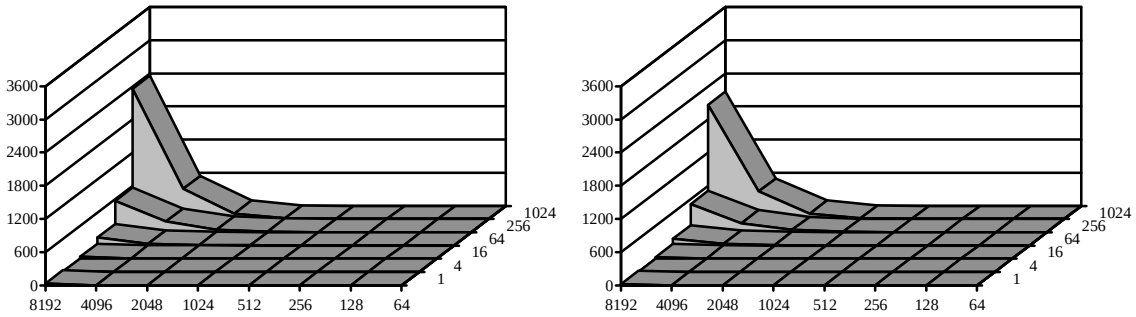
З отриманих результатів стає зрозумілим, що максимальне розподілення не завжди доцільне у зв'язку з високими накладними втратами часу. Очевидним вирішенням є встановлення розміру мінімальної підматриці, що все ще може бути розподіленою. На рис. 3 подано графічне представлення залежності часу розв'язування СЛАР розміром 8192×8192 елементів з розміром блоку 64×64 елементи від встановленої межі розподілення для одного, трьох та семи комп'ютерів в обчислювальній мережі. У цьому випадку оптимальним розміром неподільності підматриць є 128×128 елементів, проте всі значення в межах від 64×64 до 512×512 також задовільні.

Співвідношення значень часу розв'язування тестових СЛАР з розміром блоку 64×64 для різних конфігурацій обчислювальної мережі представлено на рис. 4. У середньому порівняно з обчислювальною мережею з одного комп'ютера використання трьох комп'ютерів дає виграш у часі на рівні 49,24 %, а семи — на рівні 66,92 %.

Проте зі зростанням кількості комп'ютерів в обчислювальній мережі відбувається також зростання сумарного часу простою, обумовлене неоднаковістю часу виконання підзавдань на одному рівні ієрархії. Значення втрат часу при розв'язуванні тестових СЛАР з розміром блоку 64×64 для різних конфігурацій обчислювальної мережі представлено на рис. 5.



а



б *в*

Рис. 2. Час розв'язування тестових СЛАР:
а — одним комп'ютером; б — трьома комп'ютерами; в — сімома комп'ютерами

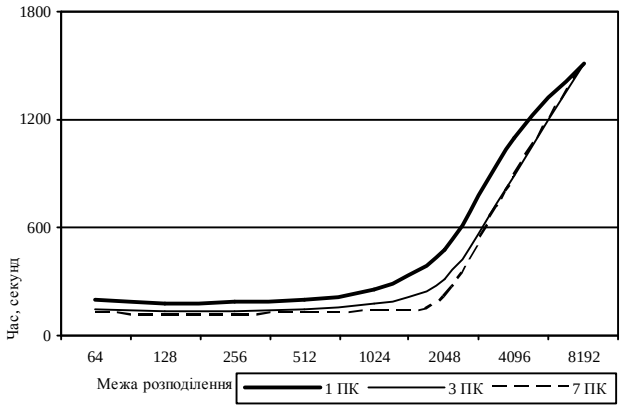


Рис. 3. Час розв'язування тестової СЛАР розміром 8192´8192
з розміром блоку 64´64

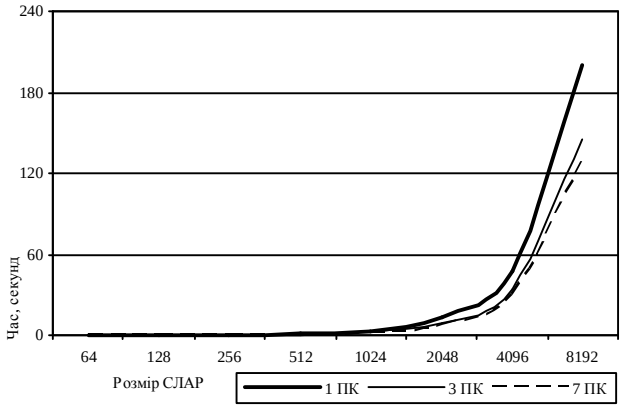


Рис. 4. Час розв'язування тестових СЛАР з розміром блоку 64´64

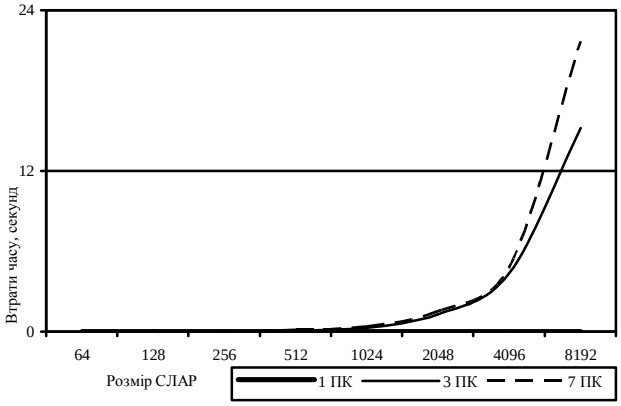


Рис. 5. Втрати часу при розв'язуванні тестових СЛАР з розміром блоку 64´64

Водночас такі втрати часу можуть бути мінімізовані при використанні адаптивного підходу до розподілення обчислень. Адаптивність системи розподілення обчислень при поділі завдання на підзавдання дасть змогу досягти одночасного завершення обчислень усіма комп'ютерами наступного рівня ієрархії і тим звести втрати часу практично до нуля. Реалізація адаптивного підходу до розподілення обчислень при розв'язуванні блоково-стрічкових СЛАР буде предметом подальших досліджень.

Висновки

У роботі розглянуто блоково-стрічкові системи лінійних рівнянь великої розмірності та запропоновано метод їх розподіленого розв'язування з використанням ієрархічного підходу.

Також здійснено огляд існуючих методів та засобів розподіленого та розпаралеленого розв'язування систем лінійних рівнянь з розрідженими матрицями.

Було розроблено програмний засіб для розподіленого розв'язування СЛАР великої розмірності та згенеровано 54 блоково-стрічкові матриці, що дало змогу провести низку експериментів.

Результати експериментів показали, що використання обчислювальної мережі з трьох комп'ютерів замість одного дає виграш у часі приблизно 49,24 %, а з семи — приблизно 66,92 %.

1. Баландин М.Ю., Шурина Э.П. Методы решения СЛАУ большой размерности. — Новосибирск: Изд-во НГТУ, 2000. — 70 с. 2. Natarajan R. Finite element applications on a shared-memory multiprocessor: algorithms and experimental results. Ibid., 94, 1991, pages 352–381. 3. I. S. Duff and J. K. Reid. The multifrontal solution of indefinite sparse symmetric linear equations. // ACM Trans. Math. Software, 9(3):302-325, 1983. 4. S. Chandrasekaran, M. Gu, X. S. Li, J. Xia. Superfast Multifrontal Method for Structured Linear Systems of Equations — 28 pages. 5. Youcef Saad. SPARSKIT: A basic tool kit for sparse computations, VERSION 2. Technical report, Computer Science Department, University of Minnesota, June 1994 — 27 pages. 6. C. Ashcraft, R. Grimes. SPOOLES: An Object-Oriented Sparse Matrix Library // Proceedings of the Ninth SIAM Conference on Parallel Processing for Scientific Computing, March 22–27, 1999 — 10 pages. 7. O. Schenk and K. Gärtner. Solving unsymmetric sparse systems of linear equations with PARDISO. Future Generation Comp. Sys., 20(3):475-487, 2001. 8. Math Kernel Library Reference Manual. — USA: Intel — 2001 — 1029 pages. 9. Patrick Amestoy, Iain Duff, Jacko Koster, Jean-Yves L'Excellent. MUMPS: A Multifrontal Massively Parallel Solver. — ERCIM News No.50, July 2002, 4 pages.