УДК 532.7:512.644(043.2)

*О.М. Глазок, к.т.н., М.М. Квач*
*(Національний авіаційний університет, Україна, Київ)*

**Декомпозиція СЛАР на розподілених обчислювальних системах при розв'язанні задач гідродинаміки**

*Розглядаються методи та алгоритми розподілу між вузлами розподіленої обчислювальної системи обчислювальних підзадач задачі розв'язання систем лінійних алгебраїчних рівнянь при математичному моделюванні течії рідини, з урахуванням вимог оптимізації використання обчислювальних ресурсів.*

На сьогодні відомо багато актуальних задач прикладної математики, які зводяться до розв'язання розріджених систем лінійних алгебраїчних рівнянь (СЛАР) високих порядків. До цієї категорії, зокрема, належать задачі обчислювальної динаміки рідин для площинних та просторових областей складних геометричних форм. Велика розмірність задач приводить до необхідності мати значні обчислювальні потужності для їх розв'язання. Сучасними напрямками розв'язання цієї проблеми є виконання розподілених обчислень на апаратній основі багатоядерних комп'ютерних систем, використання графічних процесорів, або набору комп'ютерів, об'єднаних у розподілену обчислювальну систему. Актуальною науковою задачею є організація ефективного використання цих обчислювальних ресурсів [1].

Для вирішення задачі на системах з розподіленою пам'яттю потрібно обрати деякий підхід до розподілу даних між вузлами системи. Одним з цільових показників при цьому є обсяг даних комунікації між вузлами в процесі розрахунку, а метою оптимізації – його зменшення. Основна проблема при вирішенні цієї задачі – пошук алгоритмів, що генерують розбиття СЛАР високої якості, але при цьому потребують мінімальних обчислювальних ресурсів. Таким чином, розв'язання даної задачі вимагає пошуку компромісних алгоритмів, які дозволяють отримати досить хороше розбиття і при цьому вкластися в допустимий час.

При розподілі даних можна використати два підходи до декомпозиції: геометричний та алгебраїчний.

У геометричному методі елементи сітки, яка апроксимує розрахункову область, розподіляються між процесорами за геометричними ознаками. Під час розрахунку кожен з процесорів обробляє свою частину сітки. При цьому ефективність обчислювальної системи залежить від того, наскільки рівномірно було розподілено навантаження між процесорами системи, від того, як швидко обмінюються повідомленнями процесори, а також від кількості наявних ліній комунікаційних зв'язків.

Алгоритм геометричної декомпозиції розрахункової області можна розглядати як спрощений варіант побудови BSP дерева (двійкове розбиття простору). Метод двійкового розбиття оснований на рекурсивному розбитті простору гіперплощинами на випуклі множини. Побудова оптимального BSP

дерева – надзвичайно складна задача, тому в більшості випадків використовують субоптимальні дерева [2].

Диференціальні рівняння, що описують гідродинамічну задачу, зводяться до різницевих співвідношень. Ці співвідношення записуються на групах вершин, які утворюють просторові скінченні елементи – наприклад, паралелепіпеди або тетраедри. Побудуємо бінарне дерево, проводячи січні площини ортогонально осям координат, при цьому поставимо ціль розділити кожну множину скінченних елементів на дві частини, приблизно рівні за кількістю елементів. Алгоритм побудови розбиття буде заснований на методі обробки «подій». (Під подією будемо розуміти перетин січною площиною, що рухається, однієї з вершин скінченних елементів задачі. Таким чином, вершини набувають впорядкованості.)

Підготовча частина алгоритму:

1. Спроектувати усі тетраедри на кожну з осей Ox, Oy і Oz. В результаті для кожної з осей Ox, Oy і Oz отримаємо відрізки, які відповідають «початку» та «кінцю» проеції кожного тетраедра.

2. Для кожної з осей Ox, Oy і Oz вісортувати «події» (початок і кінець проекції кожного тетраедра) за відповідною координатою.

Як результат, отримаємо три відсортовані списки подій для трьох осей координат.

Далі до цих списків ми будемо рекурсивно виконувати наступні дії. Для кожної з осей рухаємо відповідну січну площину в напрямку збільшення відповідної координати. Оскільки площина проходить через вершини (або через їх проекції), то кількість тетраедрів в кожному з напівпросторів при цьому змінюється. Тетраедри, що дотикаються до січної площини або перетинаються нею, будемо відносити до обох напівпросторів одночасно. Схема алгоритму наступна:

1. Для кожної координати l_number ← 0, r_number ← T, де T число тетраедрів.

2. Для кожної координати послідовно обробити події:

2.1. Обробка початку тетраедрів: для кожного l_number←l_number + 1.

2.2. Перевірити l_number і r_number на оптимальність.

2.3. Обробити кінці тетраедрів: r_number ← r_number - 1.

3. Запам'ятати оптимальне розбиття для осі.

4. Вибрати найкращу вісь для розбиття.

5. Побудувати розбиття:

6. Для кожної координати лінійним проходом розбити список подій для відповідної осі на дві частини відповідно до розбиття. При цьому впорядкованість збережеться.

7. Рекурсивно запустити алгоритм для двох отриманих частин.

Алгоритм зупиняє процес рекурсивного розбиття тоді, коли кількість тетраедрів в оброблюваній частині стає досить малою, або коли виявляється, що в результаті ділення кількість тетраедрів в одній з частин дорівнює вихідному їх числу. Перевірка на оптимальність для кожної з осей може бути наступною: обирається положення площини, при якому розміри лівої і правої частин максимально близькі. Найкраща з осей вибирається по мінімуму min (l_number, r_number) для оптимального положення площини, що дозволяє

обрати мінімальний з розрізів підпростору і зменшити комунікації між підобластями.

Можна модифікувати алгоритм, змінивши товщину шару перетину. Для цього потрібно виконати прохід по подіях, використовуючи два вказівники, один з яких буде відповідати початку перетину підобластей, другий кінцю. При цьому легко підтримувати геометричну товщину перетину підобластей і мінімальне число тетраедрів в перетині.

Асимптотична складність алгоритму дорівнює O (T log T), де T – вихідне число тетраедрів, за умови, що кожен раз в перетині виявляється невелика кількість тетраедрів і частини діляться приблизно порівну. Якщо кількість тетраедрів зростає несуттєво порівняно з величиною T, то на кожному рівні дерева потрібний лінійний прохід по всіх подіях для кожного з вузлів дерева цього рівня, сумарна кількість яких є O (T). При цьому число підобластей при переході на рівень нижче збільшується в 2 рази, тому загальне число рівнів не більше за O (log T). Звідси отримуємо заявлену складність.

Алгоритм здійснює розбиття сітки на підобласті з перетинами, причому ніякі дві підобласті не мають протяжної спільної межі, крім, можливо, частини зовнішньої межі розрахункової області. Цей факт можна довести на основі наступного тривіального спостереження: два тетраедра із загальною гранню неможливо розбити площиною так, щоб один тетраедр лежав по одну сторону від неї, а другий по іншу, і при цьому площина не торкалася б хоча б одного з тетраедрів. При цьому, як випливає з алгоритму, тетраедри, що дотикаються до площини перетину або пересікаються з нею, поміщаються в перетин підобластей. Це необхідно буде враховувати для подальшого розв'язання задачі методом декомпозиції на підобласті. Імовірно, ці тетраедри будуть породжувати переважну більшість комунікаційного трафіку.

Тепер розглянемо метод алгебраїчної декомпозиції СЛАР, який є модифікацією методу односпрямованої алгебраїчної декомпозиції задачі [3]. Виконаємо пошук псевдо-периферійної вершини v графа G, асоційованого з матрицею системи (СЛАР). Запускається обхід в ширину з довільної вершини графа матриці і знаходяться найбільш віддалені від неї вершини і відстань до них. Далі знову запускається обхід в ширину, починаючи вже з довільної вершини серед множини найбільш віддалених вершин, отриманих на попередньому кроці. Обходи в ширину повторюємо до тих пір, поки збільшується максимальна відстань від стартової вершини до інших. Як тільки відстань перестане збільшуватися, алгоритм зупиняється і в якості псевдо-периферійної вершини v повертає вершину, з якої починався останній обхід в ширину [4,5].

Після того, як обрана v в якості псевдо-периферійної деяка вершина, починаючи з неї, запускається обхід графа G в ширину. При цьому всі вершини поділяються на множини – «фронти» – згідно відстаней від них до вершини v. Тобто, у фронт $F_k$ (k = 0 ... d), де d – псевдо-діаметр графа G) потрапляють всі вершини, рівновіддалені від вершини v на відстань k. Тепер можемо виконати об'єднання послідовнио розташованих фронтів у алгебраїчні підобласті $\Omega_p$ : $\Omega_p = \bigcup_{k=l_p}^{r_p} F_k$, де $l_p$ і $r_p$ – межі підобласті $\Omega_p$ .

Час роботи алгоритму 1 – O (E·K), де E – кількість ребер в графі матриці. Для більшості кінцевих схем апроксимації рівнянь кількість ребер пропорційна кількості вершин сітки в розрахунковій області. Число K в оцінці дорівнює кількості ітерацій. Для довільних графів у гіршому випадку це число може бути досить велике, проте для графів практичних задач потрібно декілька ітерацій для отримання задовільного розбиття. За рахунок цього на практиці алгоритм пошуку псевдо-периферійних вершин виявляється досить ефективним. Більш того, у багатьох випадках може виявитися, що достатньо обрати довільну вершину графа, і взагалі не запускати алгоритм пошуку псевдо-периферійних вершин. Основні переваги такого методу – простота реалізації та незначний час виконання.

## Висновки

У доповіді розглянуто можливі підходи та відповідні алгоритми розв'язання задачі оптимального розподілу частин обчислювальних завдань з математичного моделювання течії рідини на вузли розподіленої обчислювальної системи. Метою оптимізації є скорочення часу розв'язання задачі, який складатиметься з часу обробки даних на окремих вузлах та часу обміну інформацією між вузлами. Доцільним напрямком подальших досліджень є конкретизація математичної моделі обчислень з урахуванням конкретної математичної задачі, конфігурації вузлів обчислювальної мережі, властивостей каналів обміну даними та алгоритмів координації групової роботи.

## Список літератури


1. *Глазок О.М.* Математична модель розв'язання гідродинамічної задачі у розподіленому обчислювальному середовищі //Проблеми інформатизації та управління: зб. наук. праць. – К.: НАУ, 2014. – Вип. 4(48). – С. 42-47.

2. *H.Fuchs, Z.M.Kedem, B.F.Naylor* On Visible Surface Generation by A Priori Tree Structures //ACM Computer Graphics. – 1980. – Vol. 14, №3. – P. 124-133.

3. Intel (R) Math Kernel Library Reference Manual. – Режим доступу: http://www.democritos.it/activities/IT-MC/mkl_lib-10.0.011/mklman.pdf

4. *Т.Кормен, Ч.Лейзерсон, Р.Ривест* Алгоритмы: построение и анализ – М., МЦНМО: БИНОМ. Лаборатория знаний, 2004. – 960 с.

5. *Д.С.Бутюгин, В.П.Ильин, Е.А Ицкович* Krylov: библиотека алгоритмов и программ для решения СЛАУ // Современные проблемы математического моделирования. Математическое моделирование, численные методы и комплексы программ. Сборник трудов Всероссийских научных молодёжных школ. Ростов-на-Дону: Изд-во Южного федерального университета, 2009. – С. 110-128.