УДК 681.3

Ефективна реалізація ієрархічного методу RBF на паралельних комп'ютерних системах

Бабков В.С.
Донецький національний технічний університет
victor@babkov.name

Abstract

Babkov V.S. Effective hierarchical RBF method implementation on parallel computer systems. *In article the reflections of method for 3D-models reconstruction of the real objects on parallel architectures SIMD, MIMD UMA, MIMD NC-NUMA is proposed. The theoretical estimations of numerical complication depending on the volume of information and amount of processor are got. The comparative analysis for method realization on SIMD and MIMD architectures is executed. Theoretical estimations for the degree of parallelism for SIMD and MIMD-realization are got.*

Вступ

Задача отримання ізоповерхні за дискретними проекційними точками виникає у багатьох задачах комп'ютерної графіки, зокрема при отриманні тривимірних моделей об'єктів за результатами 2D та 3D-сканування [1]. Математично ця задача зводиться до інтерполяції у тривимірному просторі.

Одним з відомих методів інтерполяції, що використовується для відновлення поверхонь та ізоповерхонь, є метод RBF (radial basis function).

Головним недоліком методу RBF є його часова та просторова складність, яка визначається необхідністю обробляти велику кількість проекційних точок, розв'язувати систему лінійних рівнянь великої розмірності та багаторазово обчислювати інтерполянт [2]. У багатьох роботах увага приділяється саме зменшенню обчислювальних витрат за рахунок:

- зменшення кількості центрів інтерполяції;
- використання „компактних" RBF та деревовидних структур даних;
- декомпозиції центрів інтерполяції.

Саме декомпозиційні методи мають таку значну перевагу, як здатність до розпаралелювання. Вперше декомпозиційний метод, так званий метод ієрархічної RBF, був запропонований Дж. Поудероксом у роботі [3]. Недоліком методу було те, що він був орієнтований на обчислення поверхні, тобто отримання інтерполянта для 2D-простору у вигляді $F(x, y) = 0$. У подальшому метод, заснований на декомпозиції центрів інтерполяції, був запропонований В. Кьянгом та ін. у роботі [4]. Даний метод міг застосовуватися у тривимірному просторі для обчислення ізоповерхні, але лише у випадках, коли проекційні точки розташовані у паралельних пласких площинах. Прикладом такого розташування є контури об'єктів на медичних томограмах.

У роботі [5] автором було запропоновано модифікований ієрархічний метод RBF, що може бути застосований у тривимірному просторі для обчислення ізоповерхні у вигляді $F(x, y, z) = 0$.

Метою даного дослідження є відображення запропонованого алгоритму реконструкції на паралельні комп'ютерні системи та теоретична оцінка ефективності для різних паралельних архітектур.

Відображення методу на архітектуру SIMD

Розглядаючи реалізацію запропонованого у [5] алгоритму на SIMD архітектурі, необхідно визначити множину типових операції кожного процесорного елементу (ПЕ), необхідних для реалізації алгоритму та ефективну структуру зв'язків між ПЕ, тобто топологію.

У загальному випадку алгоритм реконструкції складатиметься з наступних послідовних етапів:

- ієрархічний розподіл центрів інтерполяції між доменами;
- обчислення коефіцієнтів матриць для кожного домену найнижчого рівня;
- розв'язання систем рівнянь кожного домену;
- ієрархічне обчислення інтерполянта для шуканої точки простору.

У випадку SIMD-архітектури загальна структура обчислювальної системи матиме наступний вигляд: рис. 1.

На рис. 1 пам'ять використовується як для зберігання центрів інтерполяції до початку обчислювального процесу, так і для зберігання груп точок доменів, коефіцієнтів та інших даних

під час роботи. Для розподілу центрів інтерполяції за доменами пропонується завантажувати на процесорні елементи координати центрів та координати центру домену поточного рівня. Безпосередньо декомпозиція також буде здійснюватися за допомогою одночасного розрахунку на ПЕ параметрів доменів. Декомпозиція створює у пам'яті деревовидну структуру, приклад якої наведено на рис. 2. У даному випадку кожен вузол дерева відповідає одному домену і містить інформацію про його центр, габарити та покажчики на піддомени нижчого рівня, яких завжди вісім.

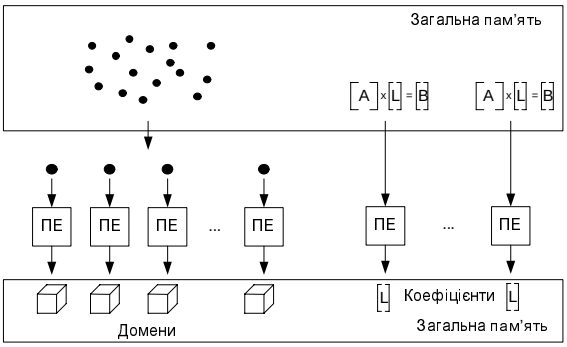

Рисунок 1 - Загальна схема організації обчислювального процесу на архітектурі SIMD (етапи декомпозиції та розв'язання систем)

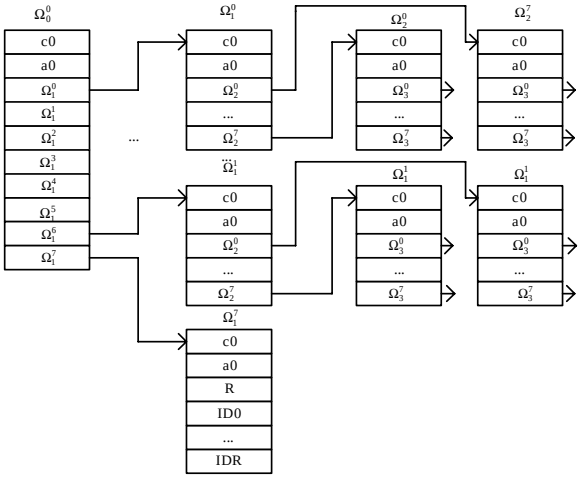

Рисунок 2 - Структура дерева доменів

У випадку домену найнижчого рівня, замість покажчиків на піддомени у вузлі міститься перелік ідентифікаторів тих центрів інтерполяції, які належать даному домену. Поле R - це кількість центрів інтерполяції у домені. Повинна виконуватися умова $R \le Thresh$, де $Thresh$ - параметр алгоритму (максимально допустима кількість точок у домені).

Для виконання формування дерева доменів необхідно, щоб поле ПЕ відпрацьовувало наступні команди:

- завантаження групи точок та параметрів центру домену поточного рівня на ПЕ;
- одночасне порівняння координат точок та координат центру домену;
- покрокове вивантаження індексів точок у відповідні вузли в пам'ять, у відповідності до результату аналізу.

Як результат декомпозиції, у піддоменах найнижчого рівня буде отримано сукупності точок, для яких необхідно обчислити значення функції RBF. Тобто на кожному кроці буде відпрацьовуватися команда обчислення функції за заданою формулою для пар точок у межах одного піддомену. Потім розрахунок повторюватиметься для всіх піддоменів найнижчого рівня.

Наступний крок – розв'язання системи рівнянь кожного піддомену найнижчого рівня, у яких функції, розраховані на попередньому кроці, виступають у якості коефіцієнтів.

Як було зазначено у [5], матриці систем є симетричними тому для їх розв'язання доцільно використовувати метод LDLT-розкладання [6], оскільки побудована матриця може бути не додатно визначеною і метод Холецького для симетричних матриць не матиме розв'язання.

На етапі обчислення інтерполянту система повинна дати відповідь для заданої точки, чи лежить вона на шуканій поверхні, чи ні. Для даного алгоритму неможливо реалізувати одночасну обробку кількох вхідних точок, тому що заздалегідь невідомо, у якому саме домені і якого рівня розташована точка. Відповідно ПЕ доведеться на одному кроці виконувати різні операції, що не відповідає архітектурі SIMD.

При цьому на деяких етапах даного алгоритму можливе прискорення за рахунок одночасної обробки різних даних:

- визначення множини значень функцій β_{int} та β_{ext} для кожного домену;
- визначення коефіцієнтів α;
- розрахунок інтерполянту для листа деревовидної структури доменів.

Кількість кроків, що витрачаються на декомпозицію, визначається виходячи з того, що у найгіршому випадку ми будемо мати збалансоване дерево, у якому кожний піддомен найнижчого рівня міститиме 1 точку, тобто всього таких піддоменів буде N. У цьому випадку кількість рівнів у дереві визначатиметься наступним співвідношенням: $q =]\log_8 N[$. Якщо на кожному рівні доводиться перебирати N точок, то у найгіршому випадку це вимагатиме $S = N]\log_8 N[$ кроків. Враховуючи кількість ПЕ, кількість кроків визначатиметься за формулою:

$$S1' = \frac{(P+2)N\log_8 N}{P}, \qquad (1)$$

де

N - кількість центрів інтерполяції;

P - кількість процесорних елементів.

Кількість кроків, витрачених на утворення нових піддоменів ($S1''$), буде визначатися як кількість піддоменів, утворених у найгіршому випадку. Якщо у найгіршому випадку кількість рівнів дерева дорівнює q, то послідовну зміну кількості піддоменів при переході від рівня до рівня можна визначити наступним чином: $8^0, 8^1, 8^2, ..., 8^q$. Тобто фактично ми маємо геометричну прогресію. Сума членів прогресії визначатиметься за формулою

$$S = \frac{1-8^q 8}{1-8}$$

або після перетворення

$$S = \frac{8^{\lfloor \log_8 N \rfloor + 1} - 1}{7}$$

Враховуючи вищенаведену формулу, можна записати:

$$S1'' = 24\frac{8^{\lfloor \log_8 N \rfloor + 1} - 1}{7}.$$

Тоді загальна кількість кроків на етапі декомпозиції $S1 = S1' + S1''$:

$$S1 = \frac{(P+2)N \log_8 N}{P} + 24\frac{8^{\lfloor \log_8 N \rfloor + 1} - 1}{7},$$

де

N - кількість центрів інтерполяції;

P - кількість процесорних елементів.

Етап обчислення коефіцієнтів перед розв'язанням потребуватиме $S2$ кроків. Параметр $S2$ обчислюватиметься за формулою:

$$S2 = 3K\left\lceil \frac{R^2}{2P} \right\rceil,$$

де

R - максимальна кількість невідомих у кожній системі ($R = \max(r_1, r_2, ... r_N)$);

P - кількість процесорних елементів;

коефіцієнт 3 – означає три кроки на операцію (завантаження, обчислення функції, вивантаження результату);

K - кількість систем, які доведеться розв'язувати.

Скорочення числа обчислюваних коефіцієнтів для кожної системи в два рази обумовлене симетрією матриці коефіцієнтів.

За обраним варіантом схеми обчислювального процесу на етапі розв'язання систем, кожний ПЕ обробляє почергово елемент однієї системи, при цьому P систем розв'язується одночасно.

У цьому випадку кількість кроків визначатиметься наступними формулами:

$$S3' = 6\left\lceil \frac{K}{P} \right\rceil$$

$$S3'' = 10\left\lceil \frac{K}{P} \right\rceil$$

$$S3''' = 1 + 2\left\lceil \frac{K}{P} \right\rceil$$

При цьому загальна кількість кроків визначатиметься співвідношенням:

$$S3 = R[\left\lceil \frac{K}{P} \right\rceil (6 + 12R) + R]$$

Етап обчислення інтерполянту не має властивостей, які дозволяли б реалізувати одночасний пошук значення інтерполянту для кількох точок [7]. Тому розглянемо процес пошуку значення для однієї точки. Згідно з алгоритмом, процес перевірки приналежності точки шуканій поверхні складатиметься з двох проходів по дереву. Перший прохід складатиметься для кожного піддомену з наступних кроків:

- визначення приналежності точки домену (кількість кроків $S4' = 3$: завантаження, операція, вивантаження);

- розрахунку функцій β_{ext} (кількість кроків $S4'' = 3\left\lceil \frac{18}{P} \right\rceil$), параметр 18 визначається найгіршим випадком, коли домен має 18 сусідів, а 3 – це завантаження, операція, вивантаження;

- розрахунку функції β_{int}, яка завжди одна і потребуватиме 5 кроків для обчислення ($S4''' = 5$);

- обчислення вагових коефіцієнтів α, ця операція потребуватиме $S4'''' = 3\left\lceil \frac{19}{P} \right\rceil$, параметр 19 визначається максимальною кількістю сусідніх доменів у сукупності з поточним доменом.

При досягненні домену найнижчого рівня одноразово обчислюється значення інтерполянту. Обчислення складається з виконання операції множення та складання. Кількість кроків у даному випадку:

$$S4''''' = 2 + \left\lceil \frac{N}{P} \right\rceil + \left\lceil \frac{N-2P}{P} \right\rceil \lfloor \log_2 P \rfloor.$$

При другому проході на кожному кроці здійснюється обчислення функції безперервності, що потребуватиме

$$S5' = 3\left\lceil \frac{18}{P} \right\rceil$$

кроків. На останньому етапі також будуть присутніми три кроки для порівняння отриманого значення з одиницею ($S5'' = 3$).

Перший та другий проходи деревом доменів у гіршому випадку будуть виконані за q кроків, де $q = \lceil \log_8 N \rceil$, тоді:

$$S4 = q(S4' + S4'' + S4''' + S4'''') + S4''''$$

або після перетворення:

$$S4 = q(8 + 3\left\lceil \frac{18}{P} \right\rceil + 3\left\lceil \frac{19}{P} \right\rceil) + 2 + \left\lceil \frac{N}{P} \right\rceil + \left\lceil \frac{N - 2P}{P} \right\rceil \lceil \log_2 P \rceil,$$

$$S5 = 3q\left\lceil \frac{18}{P} \right\rceil + 1.$$

Таким чином, якщо записати сумарну кількість кроків, що буде витрачена для обробки множини з N центрів інтерполяції на P процесорних елементах, то будуть отримані наступні вирази:

$$\tau_1 = S1 + S2 + S3,$$

$$\tau_2 = S4 + S5$$

Дослідимо залежність часових витрат на процес реконструкції за допомогою запропонованого алгоритму при його реалізації на векторному процесорі SIMD-архітектури. Розглянемо залежність кількості кроків роботи системи від кількості ПЕ та N на етапі побудови дерева доменів та розв'язання систем. Цей етап доцільно розглядати окремо, тому що він виконується для заданого набору точок один раз, на відмінність від етапу обчислення інтерполянта, який виконується для кожної шуканої точки поверхні.

На рис. 3 наведено залежність кількості кроків τ_1 від кількості ПЕ та кількості центрів інтерполяції. Оскільки вплив параметру R полягає у пропорційному збільшенні величин $S2$ та $S3$, то загальний графік будемо будувати для одного фіксованого значення R .

Розглянемо етап визначення приналежності довільної точки шуканій поверхні. На рис. 4 наведено залежність кількості кроків при прямому та зворотному проході від кількості ПЕ та кількості центрів інтерполяції.

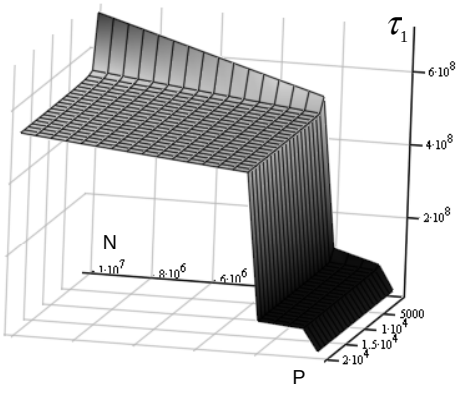

Рисунок 3 - Залежність $\tau_1(N, P)$ (SIMD)

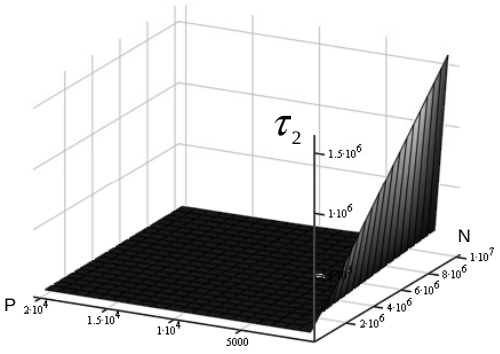

Рисунок 4 - Залежність $\tau_2(N, P)$ (SIMD)

Відображення методу на архітектуру MIMD UMA

Розглянемо реалізацію модифікованого алгоритму на основі мультипроцесорів MIMD. Згідно з класифікацією [8] існує велика кількість різновидів MIMD-архітектури:

- з однорідним доступом до пам'яті;
- з неоднорідним доступом до пам'яті;
- з доступом лише до кеш-пам'яті і т.д.

У даному випадку розглядається реалізація на архітектурі MIMD з однорідним доступом до пам'яті та шинною організацією. В такому випадку загальна схема системи матиме наступний вигляд: див. рис. 5.

Структура вхідних даних та дерева доменів у пам'яті така ж, як і у випадку SIMD-архітектури (див. рис. 2).

Розглянемо почергово кожний етап реконструкції.

На етапі декомпозиції пропонується задачу формування дерева розбити на множину підзадач з формування гілок одного вузла, і цю

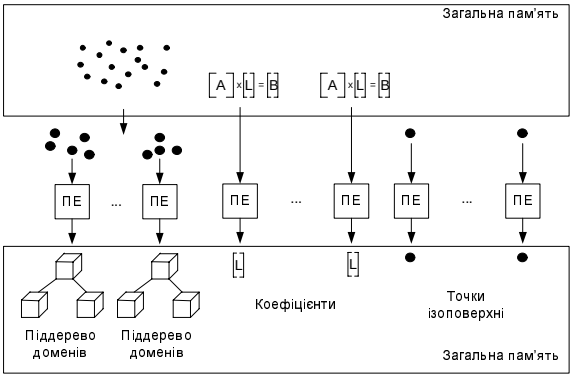

Рисунок 5 - Загальна схема організації обчислювального процесу на архітектурі MIMD UMA (етапи декомпозиції, розв'язання систем та обчислення ізоповерхні)

задачу покласти на кожний ПЕ. У такому випадку кожний ПЕ буде відпрацьовувати свою копію алгоритму декомпозиції [5].

Етап розв'язання систем зводиться до розв'язання кожної системи на окремому ПЕ. При цьому етапу розв'язання передуватиме етап обчислення матриць коефіцієнтів.

Етап пошуку значення інтерполянта для шуканої точки на основі MIMD-архітектури (на відміну від SIMD) може бути реалізований одночасно для кількох точок простору (не більше за P) [7]. У цьому випадку визначення значення інтерполянта для шуканої точки буде також (як і у SIMD реалізації) виконуватися за два проходи.

Схема організації обчислювального процесу для етапу декомпозиції може бути поділена на три частини:

- $S1' = 3$ (завантаження, обчислення, вивантаження);
- $S1'' = 3$ (завантаження, визначення приналежності, вивантаження);
- $S1''' = 3$ (відправка запиту на наявність вільного ПЕ, отримання відповіді, відправка даних до вільного ПЕ або відправка запиту на наявність вільного ПЕ, отримання відповіді та рекурсивний виклик такого ж алгоритму декомпозиції.

Кроки $S1', S1'', S1'''$ будуть виконуватися 1, n та 8 разів, відповідно, де n - кількість точок поточного домену. У результаті загальна кількість кроків на виконання ПЕ одного екземпляру алгоритму визначатиметься наступною формулою:

$$S1 = 3(9 + n), \qquad (2)$$

де

n - кількість точок, що потрапляє на обробку у поточному екземплярі алгоритму;

У загальному випадку кількість точок, що потраплятиме на обробку в кожному екземплярі алгоритму, залежить від багатьох чинників, зокрема:

- просторового розташування точок у габаритному домені;
- параметру *Thresh*, що визначає максимальну кількість точок у домені найнижчого рівня.

Якщо розглядати ідеалізований випадок, у якому точки рівномірно розподіляються доменами, то утворюється збалансоване дерево, що на найнижчому рівні містить $8^{]\log_8 N[}$ доменів, кожен з яких місить лише одну точку.

У такому випадку на кожному рівні дерева будуть присутні 8^i вузлів, де $i = \overline{0,]\log_8 N[}$.

Для того, щоб записати кількість кроків, що витрачається на декомпозицію розглянемо дві ситуації:

вільні ПЕ присутні;

вільні ПЕ відсутні.

Перша ситуація спостерігатиметься тоді, коли кількість працюючих екземплярів алгоритму менша за кількість ПЕ. Тобто $8^i < P$, відповідно $i <]\log_8 P[$.

У цьому випадку чекання відсутнє, і кількість кроків для кожної копії визначається формулою 2.

Оскільки параметр n на довільному рівні дерева визначається формулою:

$$n_i = \left\rceil \frac{N}{8^i} \right\lceil,$$

де

i - рівень дерева.

У результаті отримуємо, що для $i <]\log_8 P[$ справедлива наступна формула:

$$S1 = \sum_{i=0}^{]\log_8 P[-1} 3\left(\left\rceil \frac{N}{8^i} \right\lceil + 9\right),$$

де

P - кількість ПЕ;

N - загальна кількість центрів інтерполяції.

У випадку, якщо $i \geq]\log_8 P[$, спостерігатиметься друга ситуація, тобто вільних ПЕ не буде. Тоді кожний ПЕ буде послідовно відпрацьовувати рекурсивний алгоритм декомпозиції для всіх восьми піддоменів кожного вузла на кожному рівні.

У цьому випадку буде справедлива наступна формула:

$$S1 = \sum_{i=]\log_8 P[}^{]\log_8 N[} \frac{8^i}{P} 3\left(\left\rceil \frac{N}{8^i} \right\lceil + 9\right).$$

У результаті загальна кількість кроків,

витрачена на етап декомпозиції, визначатиметься за формулою:

$$S1 = \sum_{i=0}^{\log_8 P[-1} 3\left(\left\lfloor \frac{N}{8^i} \right\rfloor + 9\right) + \sum_{i=\log_8 P[}^{\log_8 N[} \frac{8^i}{P} 3\left(\left\lfloor \frac{N}{8^i} \right\rfloor + 9\right).$$

Слід зазначити, що даний випадок не є найгіршим, він є усередненим. Теоретично оцінити кількість кроків у найгіршому та найкращому випадку неможливо, оскільки ці випадки визначаються просторовим розташуванням центрів інтерполяції, яке у випадку довільних об'єктів заздалегідь передбачити неможливо. Теоретично можна висунути гіпотезу, що найгіршим випадком буде ситуація, за якої центри інтерполяції скупчені у деякій частині простору таким чином, що декомпозиція цієї ділянки простору (на відміну від інших) відбувається досить довго (скінченність цього процесу гарантується тим, що домен найнижчого рівня завжди містить не більше за *Thresh* точок і $Thresh << N$). Відповідно, кращим випадком буде розташування центрів інтерполяції таким чином, що вони рівномірно розподілятимуться частиною доменів (зокрема периферійних). У цьому випадку загальна кількість екземплярів алгоритму декомпозиції знизиться. Підтвердити вищенаведену гіпотезу та отримати дані про те, наскільки реальні об'єкти відповідають ідеалізованому усередненому випадку, можна за допомогою експерименту.

Для етапу розв'язання систем кількість кроків визначатиметься наступною формулою (відповідно до обраного методу):

$$S2 = \left\lceil \frac{K}{P}\left(3\left\lfloor \frac{R^2}{2} \right\rfloor + 6R + 12R^2 + 1\right)\right\rceil,$$

де

P - кількість ПЕ;

K - кількість систем, які необхідно розв'язати;

R - розмірність системи.

При цьому враховується, що у гіршому випадку система має розмірність $R = \max\{r_0,..,r_i\}$, де $R \le Thresh$.

Враховуючи, що на етапі визначення приналежності точки поверхні в архітектурі MIMD можлива одночасна обробка кількох точок, можна отримати вираз для кількості кроків на прямий та зворотний прохід для однієї точки.

У відповідності з алгоритмом [5], визначення приналежності виконуватиметься 1 раз, обчислення β_{ext} - 18 раз у гіршому випадку, обчислення β_{int} - 1 раз, обчислення коефіцієнту α - 19 разів у гіршому випадку, обчислення суми інтерполянту - R разів. З урахуванням вищенаведених значень та того, що процес у гіршому випадку повторюватиметься $q = \lfloor \log_8 N \lceil$ разів, формула для розрахунку загальної кількості кроків для обчислення інтерполянту (прямий прохід) матиме вигляд:

$$S3 = q(118 + 2R) + 1,$$

де

P - максимальна кількість змінних у системі;

$q = \lfloor \log_8 N \lceil$ - кількість рівнів у дереві доменів.

Враховуючи, що обчислення функції безперервності при зворотному проході виконується q разів, а порівняння один раз, формула для розрахунку загальної кількості кроків на зворотному проході матиме вигляд:

$$S4 = 3(q + 1).$$

Таким чином, якщо записати сумарну кількість кроків, що буде витрачена системою для обробки множини з N центрів інтерполяції на P процесорних елементах, то будуть отримані наступні вирази:

$$\tau_1 = S1 + S2,$$
$$\tau_2 = S3 + S4$$

Дослідимо залежність часових витрат на процес реконструкції за допомогою запропонованого алгоритму при його реалізації на мультипроцесорі MIMD-архітектури з однорідним доступом до пам'яті.

На рис. 6 наведено залежність кількості кроків $\tau_1(N,P)$ від кількості ПЕ та кількості центрів інтерполяції.

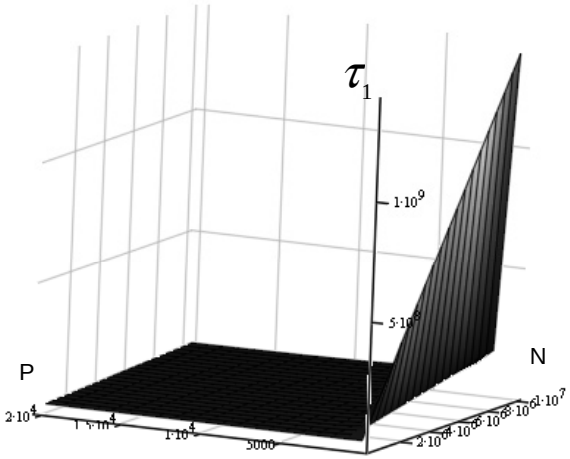

Рисунок 6 - Залежність $\tau_1(N,P)$ (MIMD UMA)

Залежність $\tau_2(N)$ наведено на рис. 7.

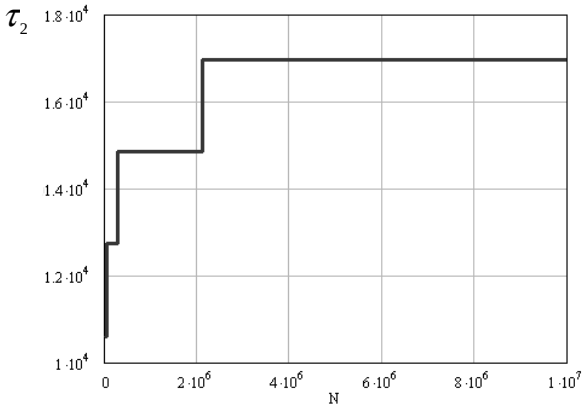

Рисунок 7 - Залежність $\tau_2(N)$ (MIMD UMA)

Слід зазначити, що отримані оцінки є суто теоретичними. На практиці реалізація MIMD-системи з однорідним доступом до пам'яті з достатньо великою кількістю ПЕ неможлива. Обумовлено це наявністю вузького місця – загальної шини доступу до пам'яті, через яку інтенсивний обмін здійснюється між ПЕ та пам'яттю.

За даними досліджень [9], теоретично кількість ПЕ в такій архітектурі обмежується величиною 64. Відомі реалізації з 256 ПЕ, але їх недолік – великі апаратурні витрати [10].

Відображення методу на архітектуру MIMD NC-NUMA

Аналізуючи реалізацію алгоритму на SIMD та MIMD-архітектурах, можна зробити висновок, що всі операції, які виконується в системі можна поділити на два класи: операції з обробки даних, операції пересилання (читання або запису).

Виконавши аналіз схем обчислювального процесу на різних етапах, було отримано наступне часткове співвідношення типів операцій для архітектури MIMD (див. рис. 8).

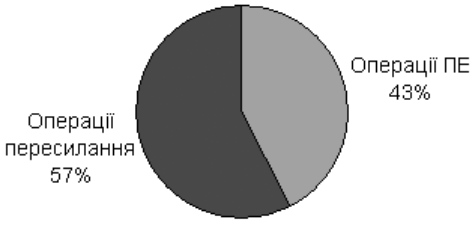

Рисунок 8 - Часткове співвідношення операцій для архітектури MIMD UMA

Операції пересилання є досить ресурсоємними, тому гіпотетично зменшення часткового внеску таких операції у загальний час роботи алгоритму, дозволило б покращити часові характеристики. Зменшення часткового внеску операцій пересилання можливе за рахунок використання локальної пам'яті процесора, яка використовується для зберігання оброблюваних даних. Така архітектура відповідає типу NC-NUMA, тобто неоднорідний доступ до пам'яті без кешу.

При використанні неоднорідного доступу до пам'яті передбачається, що існує два часи доступу $T1$ - час доступу до загальної (віддаленої) пам'яті та $T2$ - час доступу до локальної пам'яті. При цьому виконується умова $T1 >> T2$.

Розглянемо почергово етапи роботи алгоритму і визначимо, які дані можна зберігати у локальній пам'яті і яких відповідних операцій пересилання позбутися. Етап декомпозиції при запуску кожного екземпляру алгоритму обробляє таку кількість точок, скільки міститься в кожному вузлі дерева. У гіршому випадку ця кількість дорівнює N. Оскільки за умовами задачі величина N лежить у діапазоні $10^6..10^9$, то створення відповідного обсягу локальної пам'яті недоцільне. Крім того процес декомпозиції вимагає постійного звертання до загальної структури даних – дерева доменів з боку всіх ПЕ. Ґрунтуючись на вищесказаному, можна зробити висновок, що використання локальної пам'яті на етапі декомпозиції недоцільне.

В протилежність, виправданим є використання локальної пам'яті на етапі обчислення коефіцієнтів та розв'язання системи. Оскільки даний етап пов'язаний з відокремленою обробкою на ПЕ певного обсягу даних (матриця коефіцієнтів, допоміжні матриці і т.п.), то доцільно ці дані зберігати у локальній пам'яті.

На етапі прямого проходу деревом доменів для оцінювання приналежності точки поверхні можливе використання локальної пам'яті для зберігання розрахованих значень β, α і т.д. Через рекурсивність алгоритму для запобігання звертанню до глобальної пам'яті необхідно розраховувати обсяг локальної пам'яті так, щоб зберегти всі параметри, розраховані на всіх рівнях дерева.

На етапі зворотного проходу деревом доменів також можна розраховувати функцію безперервності, спираючись на величини, що вже є у локальній пам'яті. Враховуючи розташування частини даних в локальній пам'яті, запишемо формули для визначення кількості кроків у випадку реалізації на архітектурі MIMD NC-NUMA:

$$S1 = \sum_{i=0}^{]\log_8 P[-1} 3\left(\left\lfloor \frac{N}{8^i} \right\rfloor + 9\right) + \sum_{i=]\log_8 P[}^{]\log_8 N[} \frac{8^i}{P} 3\left(\left\lfloor \frac{N}{8^i} \right\rfloor + 9\right),$$

$$S2 = \left\lceil \frac{K}{P}(2\left\rceil \frac{R^2}{2}\right[+ 4R + 4R^2 + 1 + \alpha(\left\rceil \frac{R^2}{2}\right[+ 2R + 8R^2))\right[,$$

$$S3 = q(78 + R) + 1) + \alpha q(40 + R) ,$$

$$S4 = 2(q + 1) + \alpha(q + 1) ,$$

де

$K = \frac{N}{R}$ - кількість систем, які доводиться розв'язувати;

$q = \left]\log_8 N\right[$ - кількість рівнів у дереві доменів;

N - кількість центрів інтерполяції;

R - максимальна кількість змінних в одній системі;

α - коефіцієнт, що показує відношення часу доступу до локальної пам'яті до часу доступу до глобальної пам'яті ($0 < \alpha < 1$). Часткове співвідношення операцій для архітектури MIMD NC-NUMA наведено на рис. 9.

Рисунок 9 - Часткове співвідношення операцій для архітектури MIMD NC-NUMA

Порівняльний аналіз реалізацій методу на паралельних архітектурах

Порівняння реалізацій модифікованого алгоритму на паралельних архітектурах будемо проводити за наступними характеристиками: часткове співвідношення операцій ПЕ та операцій пересилання, кількість шуканих точок, які можуть оброблятися одночасно і т.д.

Результати аналізу зведено у таблицю 2.

Для наочного порівняння часових характеристик в абсолютних величинах побудуємо залежності для загального часу роботи алгоритму для трьох типів архітектур (SIMD, MIMD UMA, MIMD NC-NUMA): див. рис. 10–11. Залежності окремо будуються для двох діапазонів: $P \in [1..256]$ та $P \in [256..2 \cdot 10^4]$.

Таблиця 2. Порівняльні характеристики реалізації методу на паралельних архітектурах

Параметр	Тип архітектури		
	SIMD	MIMD UMA	MIMD NC-NUMA
Співвідношення операцій пересилання та операцій ПЕ	54% 46%	57% (глобал.) 43%	23% (глобал.) 45% (локал.) 43%
Максимальна кількість шуканих точок	1	P	
Обсяг локальної пам'яті	-	$O(R^2)$	
Обсяг глобальної пам'яті	$O(N\log_8 N + + R^2)$	$O(N\log_8 N)$	

Така розбивка на діапазони обумовлена тим, що за даними досліджень [10], 256 – це фактично межа кількості ПЕ у випадку MIMD систем з шинною організацією. Більші значення P мають суто теоретичний характер, на відмінність від SIMD, де такої межі немає.

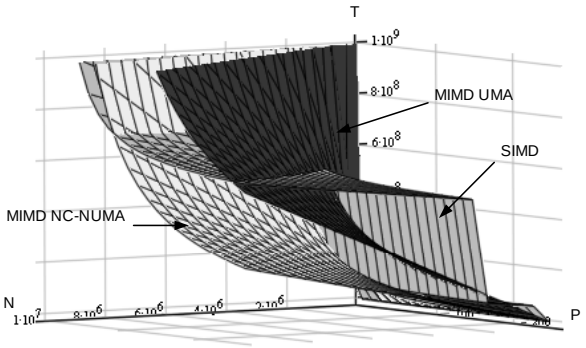

Рисунок 10 - Загальний час роботи алгоритму для трьох типів архітектур ($R = 1000$, $\alpha = 0.1$, $P \in [1..256]$)

Аналізуючи графіки на рис. 10-11, можна зробити наступні висновки:

- у діапазоні малих значень P (1..256) найменші часові витрати має реалізація на

архітектурі MIMD NC-NUMA, виняток складають діапазони $P \to 1$, $N \to 0$ та $P \to 1$, $N \to \infty$, у яких перевагу має SIMD архітектура;

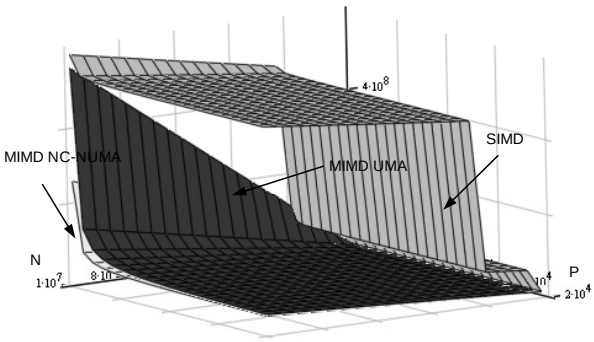

Рисунок 11 - Загальний час роботи алгоритму для трьох типів архітектур ($R = 1000$, $\alpha = 0.1$, $P \in [256..2 \cdot 10^4]$)

- відмова від використання локальної пам'яті ($\alpha \to 1$) та перехід до архітектури MIMD UMA приводить до того, що MIMD архітектура має перевагу у вузькому діапазоні $P \to 256$, $N \to \infty$, а при $P \to 1$ перевагу має SIMD архітектура;

- у діапазоні великих значень P $(256..\infty)$ архітектури MIMD UMA та MIMD NC-NUMA мають практично однакові часові характеристики, причому виграш у даному випадку у порівнянні з SIMD архітектурою носить суто теоретичний характер, оскільки відповідна кількість ПЕ у MIMD архітектурах з шинною організацією практично не реалізується. Це дозволяє зробити припущення, що використання MIMD архітектур з комутаційною сіткою [8] гіпотетично дозволить розширити використовуваний на практиці діапазон P, в якому MIMD матиме перевагу над SIMD.

Визначимо числові характеристики паралельної реалізації запропонованого методу: ступінь паралелізму, прискорення та ефективність при реалізації на SIMD та MIMD-архітектурах.

Згідно із визначенням [11], ступінь паралелізму визначається як кількість одночасно виконуваних операцій алгоритму або усереднено, як відношення загальної кількості операцій до кількості етапів.

Враховуючи, що загальна кількість операцій (для SIMD-реалізації) на етапі декомпозиції: $N \log_8 N$, етапі розрахунку коефіцієнтів - $\frac{NR}{2}$, етапі розв'язання систем NR^2, а кількість етапів паралельного методу у відповідних випадках - $\frac{8N-1}{7}$, 1, R, отримуємо наступний вираз для ступеня паралелізму:

$$Q = 3.5N \frac{2\log_8 N + R + 2R^2}{8N + 6 + 7R}, \qquad (3)$$

де

N - кількість центрів інтерполяції загалом;

R - кількість центрів інтерполяції в одному піддомені найнижчого рівня.

Верхню межу для прискорення визначимо за Амдалом [11]. Для цього визначимо частку послідовних операцій алгоритму по відношенню до загального числа операцій:

$$\alpha(N,P) = \frac{\tau_1(N,P)}{2N\log_8 N + 24\frac{N}{7} + 3\frac{NR}{2} + 6R + 12R^2 + 1 + PN\log_8 N}$$

де

N - кількість центрів інтерполяції загалом;

R - кількість центрів інтерполяції в одному піддомені найнижчого рівня;

P - кількість ПЕ;

τ_1 - кількість кроків послідовної роботи.

Прискорення в ідеальному випадку визначатиметься виразом:

$$S(P) = \frac{1}{\alpha + \frac{1-\alpha}{P}}$$

Графічно залежність прискорення від P при $R = 1000$ та $N \to \infty$ наведено на рис. 12. На рис. 13 наведено залежність ефективності від P.

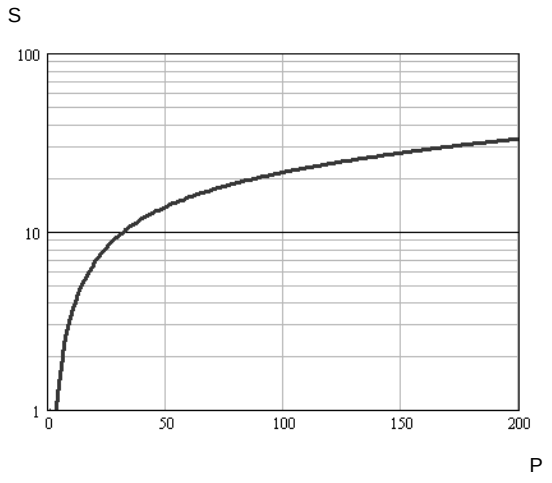

Рисунок 12 - Прискорення паралельної реалізації методу (SIMD) ($N \to 10^{10}$)

За рис. 12–13 можна зробити наступні висновки. Верхня межа прискорення в ідеальному випадку при SIMD-реалізації ($P \to 200$) наближається до 40. Ефективність при цьому має

максимум у межах $P \approx 10$, а при $P \to \infty$ ефективність не перевищує 0.1.

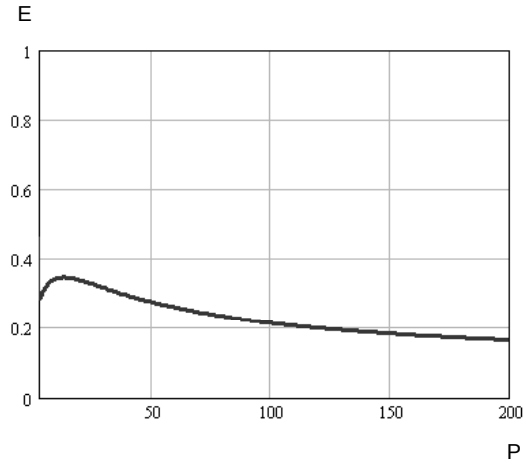

Рисунок 13 - Ефективність паралельної реалізації методу (SIMD) ($N \to 10^{10}$)

Для MIMD-реалізації ступінь паралелізму визначатиметься наступними виразами:

$N \log_8 N$ - загальна кількість операцій на етапі декомпозиції;

$\frac{NR}{2}$ - загальна кількість кроків на етапі обчислення коефіцієнтів;

NR^2 - загальна кількість кроків на етапі розв'язання систем;

$\frac{8N-1}{7}$ - кількість етапів для паралельної декомпозиції;

$\frac{N}{R}$ - кількість етапів для паралельного обчислення коефіцієнтів та розвя'зання систем.

За аналогією до формули 3 отримуємо:

$$Q = 3.5NR\frac{2\log_8 N + R + 2R^2}{R(8N-1)+14N}, \qquad (4)$$

де

N - кількість центрів інтерполяції загалом;

R - кількість центрів інтерполяції в одному піддомені найнижчого рівня.

Порівнюючи вирази (3) та (4), можна зробити висновок, що ступінь паралелізму у випадку MIMD-реалізації, загалом вищий, ніж при SIMD-реалізації.

Кількісно це можна виразити як:

$$\frac{Q_{mimd}}{Q_{simd}} = R\frac{8NR+6R+7R^2}{8NR-R+14N}.$$

Розраховуючи вираз як ліміт за $N \to \infty$, визначимо, що ступінь паралелізму MIMD-реалізації перевищує ступінь паралелізму SIMD-реалізації у:

$$\lim_{N\to\infty} R\frac{8NR+6R+7R^2}{8NR-R+14N} = \frac{4R^2}{4R+7}$$

разів.

Верхню межу для прискорення визначимо аналогічно SIMD-реалізації.

Графічно залежність прискорення від P при $R = 1000$ та $N \to \infty$ наведено на рис. 14. На рис. 15 наведено залежність ефективності від P.

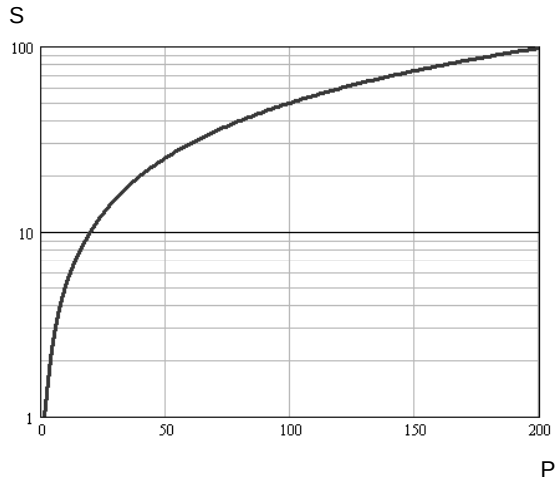

Рисунок 14 - Прискорення паралельної реалізації методу (MIMD) ($N \to 10^{10}$)

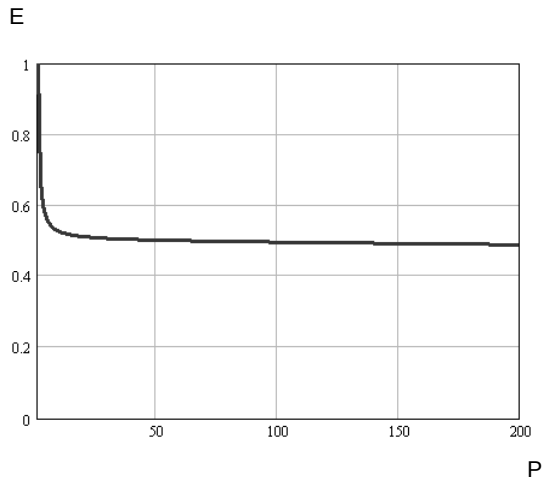

Рисунок 15 - Ефективність паралельної реалізації методу (MIMD) ($N \to 10^{10}$)

За рис. 14–15 можна зробити наступні висновки. Верхня межа прискорення в ідеальному випадку при MIMD-реалізації наближається до 100. Ефективність при цьому має максимум у межах $P \to 1$, а при $P \to \infty$ ефективність не перевищує 0.5.

Таким чином теоретичні оцінки дозволяють стверджувати, що більше прискорення (в 2.5 разів відносно SIMD) та більшу ефективність

(приблизно в 5 разів відносно SIMD) має MIMD-реалізація запропонованого методу.

Висновки

У результаті проведених досліджень отримані наступні результати:

- модифікований алгоритм реконструкції [5] відображено на паралельні архітектури типу SIMD, MIMD UMA, MIMD NC-NUMA та запропоновано схеми організації обчислювального процесу для вищевказаних архітектур;
- для відповідних архітектур отримано теоретичні оцінки часових витрат в залежності від обсягу вхідних даних та кількості процесорних елементів;
- визначено частковий внесок операцій пересилання та операцій ПЕ у часові витрати;
- виконано порівняльний аналіз характеристик для реалізацій методу на архітектурах SIMD та MIMD. SIMD-реалізація методу забезпечує прискорення ≤ 40, а MIMD-реалізація - ≤ 100. При цьому за великих P ефективність SIMD-реалізації спрямовується до 0.1, а MIMD-реалізації – до 0.5;
- отримано теоретичні оцінки для ступеню паралелізму SIMD та MIMD-реалізацій і визначено, що ступінь паралелізму для MIMD-реалізації теоретично перевищує ступінь паралелізму для SIMD-реалізації у $\frac{4R^2}{4R+7}$ разів.

Подальшим напрямком досліджень є експериментальна перевірка часових та просторових характеристик модифікованого методу на базі паралельних архітектур сучасних графічних процесорів.

Література

1. Бабков В.С. 3D-моделювання об’єктів на основі 2D та 3D-проекційних даних / В.С. Бабков // Матеріали IV науково-практичної конференції „Донбас-2020: наука і техніка – виробництву”, 27-28 травня 2008 р. – Донецьк: ДонНТУ Міністерства освіти і науки, 2008. – С. 383–387
2. Бабков В.С. Исследование возможностей применения RBF-алгоритма и его модификаций для построения поверхностных компьютерных моделей в медицинской практике / Е.А. Башков, В.С. Бабков // Сборник трудов международной конференции "Моделирование-2008", 14-16 мая 2008 г. – Киев: Институт проблем моделирования в энергетике им. Г.Е. Пухова, т. 1, 2008. - С. 166–171
3. Pouderox J. Adaptive hierarchical RBF interpolation for creating smooth digital elevation models / J. Pouderox [et al.] // Proc. 12-th ACM Int. Symp. Advances in Geographical information Systems, 2004. – ACP Press. – P. 232–240
4. Surface rendering for parallel slice of contours from medical imaging / W. Qiang, Z. Pan, C. Chun, B. Jiajun // Computing in science & engineering, 2007. – V.9, №1. – P. 32–37
5. Бабков В.С. Модифікація ієрархічного методу RBF для отримання 3D-моделей за результатами лазерного сканування / В.С. Бабков // IV Міжнародна науково-практична конференція „Сучасні проблеми і досягнення в галузі радіотехніки, телекомунікацій та інформаційних технологій”: тези доповіді, 24-26 вересня, 2008 р. – Запоріжжя, ЗНТУ. – С. 116–117
6. Богачев К.Ю. Практикум на ЭВМ. Методы решения линейных систем и нахождения собственных значений / Богачев К.Ю. – М.: Изд-во МГУ, 1998. – 79 с.
7. Бабков В.С. Реконструкція 3D-моделей реальних об’єктів методом RBF з використанням GPU / В.С. Бабков // Наукові праці Донецького національного технічного університету. Серія: “ Інформатика, кібернетика і обчислювальна техніка”, випуск 9 (132). – Донецьк: ДонНТУ. 2008. – С. 132–136
8. Таненбаум Э. Архитектура компьютера / Таненбаум Э.: 4-е изд. – СПб.: изд-во Питер, 2003. – 697 с.
9. A Two-Level Directory Architecture for Highly Scalable cc-NUMA Multiprocessors / M.E. Acacio, J. Gonzalez, J.M. Garcy, J. Duato // IEEE Transactions on Parallel and Distributed Systems, 2005. – V.16, №1. – P. 67–79
10. Monteiro J. Parallel and Distributed Computing. Uniform Memory Access (UMA) architecture / J. Monteiro // [Електронний ресурс], 2008. – Режим доступу: https://dspace.ist.utl.pt/bitstream/2295/203817/1/cpd-04.pdf. - Department of Computer Science and Engineering, Instituto Superior T´ecnico
11. Воеводин В.В. Вычислительная математика и структура алгоритмов / Воеводин В.В. – М.: Изд-во МГУ, 2006. – 112 с.

Надійшла до редколегії 10.03.2009