УДК 004.8

В.В. Пірняк
В.О. Денисюк

ДОСЛІДЖЕННЯ ТА РОЗРОБКА ПРОГРАМИ МНОЖЕННЯ МАТРИЦЬ У РОЗПОДІЛЕНІЙ СИСТЕМІ ІЗ ЗАСТОСУВАННЯМ ТЕХНОЛОГІЇ RPC

Вінницький національний технічний університет

Анотація

Розглянуто розробку алгоритму та програмного забезпечення для розподіленого множення матриць з використанням технології віддаленого виклику процедур RPC. Проведено аналіз архітектури «клієнт-сервер» та протоколу XML-RPC для організації міжпроцесної взаємодії. Обґрунтовано вибір стрічкової декомпозиції як основного методу паралелізації для мінімізації комунікаційних затримок. Створено програмну реалізацію на мові Python з використанням бібліотеки NumPy. Проведено тестування продуктивності системи, проаналізовано вплив накладних витрат на серіалізацію даних та визначено межі ефективності використання RPC для обчислювально інтенсивних задач.

Ключові слова: *розподілені системи, RPC, XML-RPC, матричне множення, Python, продуктивність*.

Abstract

The development of an algorithm and software for distributed matrix multiplication using Remote Procedure Call technology is considered. An analysis of the client-server architecture and the XML-RPC protocol for organizing inter-process communication was carried out. The choice of strip decomposition as the main parallelization method to minimize communication delays is justified. A software implementation in Python using the NumPy library was created. Performance testing of the system was conducted, the impact of data serialization overhead was analyzed, and the efficiency limits of using RPC for computationally intensive tasks were determined.

Keywords: *distributed systems, RPC, XML-RPC, matrix multiplication, Python, performance.*

Вступ

Актуальність розробки програми множення матриць у розподіленій системі із застосуванням технології віддаленого виклику процедур (RPC - remote procedure call) зумовлена необхідністю обробки надвеликих обсягів даних у таких сферах, як машинне навчання, комп'ютерна графіка та наукове моделювання [1]. Операція множення матриць є обчислювально інтенсивною задачею, ефективність виконання якої безпосередньо залежить від архітектури системи [2]. Використання технології RPC дозволяє створити гнучку клієнт-серверу інфраструктуру, де складні математичні розрахунки розподіляються між незалежними обчислювальними вузлами, забезпечуючи прозорість мережевої взаємодії для розробника.

Розподілені системи на базі RPC дозволяють значно покращити швидкодію за рахунок паралельного виконання підзадач на різних серверах, що є критично важливим для сучасних високопродуктивних систем. Це забезпечує масштабованість рішення: можливість нарощувати обчислювальну потужність шляхом додавання нових вузлів без зміни основної логіки програми[3-7].

Метою роботи є розробка та програмна реалізація алгоритму розподіленого множення матриць за допомогою технології RPC для оптимізації часу виконання обчислень у гетерогенних мережевих середовищах.

Постановка задачі дослідження

Задачі дослідження полягають у вирішенні наступних питань:

- розробка архітектури розподіленої системи типу Master-Worker для матричного множення;
- математичне моделювання процесу стрічкової декомпозиції та агрегації результатів;

- програмна реалізація модулів клієнта та сервера з використанням XML-RPC;
- аналіз ефективності E_p та прискорення S_p алгоритму при різних величинах матриць.

Виклад основного матеріалу

Матричне множення є фундаментальною операцією, що використовується для розв'язання систем лінійних рівнянь, відображення геометричних трансформацій у комп'ютерній графіці, а також у багатьох алгоритмах машинного навчання для оптимізації ваг та здійснення прогнозів. Методи розв'язання включають: стандартний алгоритм з часовою складністю $\boldsymbol{O(N^3)}$; алгоритм Штрассена зі складністю приблизно $\boldsymbol{O(N^{2.807})}$; більш ефективні методи в розподілених та паралельних обчисленнях, що дозволяють значно скоротити час операцій над великими масивами даних.

Для того, щоб правильно визначити ідею використання розподілених систем, у роботі виконано дослідження архітектури RPC, яка є ключовою технологією для побудови сучасних обчислювальних мереж. RPC – це технологія міжпроцесної взаємодії, яка дозволяє програмі викликати процедури, що знаходяться в іншому адресному просторі (зазвичай на віддаленому комп'ютері), використовуючи той самий синтаксис, що й для локального виклику. Вузли у такій системі пов’язані між собою через мережевий протокол (у нашому випадку XML-RPC поверх HTTP), що дозволяє інформації переміщатися від клієнта до сервера без необхідності ручного керування сокетами. RPC-системи є «прозорими» для розробника, оскільки новий обчислювальний вузол може бути легко інтегрований у загальну структуру системи. Розподілена система на базі RPC може об’єднувати кілька пристроїв у єдиний кластер без шкоди для якості обчислень. Основний вузол (Master) підключається до обчислювальних вузлів (Workers) через порти глобальної або локальної мережі. Навіть якщо один вузол виходить з ладу, решта системи може продовжувати функціонувати (за умови коректної обробки винятків). На відміну від традиційних монолітних програм, розподілена RPC-система не потребує спільної пам'яті для всіх процесів. Кожен блок обчислень повторює логіку на кожному вузлі незалежно. Оскільки RPC базується на мережевих протоколах, вона має набагато ширший діапазон масштабування, ніж локальні багатопотокові рішення.

Технології RPC можуть бути різних типів залежно від формату даних та транспорту:

- XML-RPC - текстовий формат на базі XML та HTTP (використано в роботі);
- JSON-RPC - компактний формат на базі JSON;
- gRPC - високопродуктивний бінарний протокол від Google на базі HTTP/2;
- SOAP - важковаговий протокол зі строгими стандартами безпеки.

Основні переваги та ідеї застосування RPC у матричних обчисленнях включають:

- локальний обмін даними, кожен сервер спілкується тільки з клієнтом, що зменшує складність топології мережі;
- асинхронність, завдяки пулу потоків на стороні клієнта, можливо паралельне виконання обчислень на різних вузлах одночасно;
- масштабованість, структура легко розширюється шляхом додавання нових IP-адрес серверів у конфігурацію клієнта;
- абстракція, програміст оперує викликами функцій, не заглиблюючись у мережеву реалізацію;
- гнучкість, можливість розподіляти завдання між вузлами з різною апаратною архітектурою.

У роботі досліджено та використано чотири основні способи оптимізації алгоритму за допомогою RPC.

1. Паралелізація через ThreadPoolExecutor: розподіл даних між вузлами мережі та використання потоків на стороні клієнта дозволило паралельно ініціювати обчислення на всіх вузлах.
2. Стрічкова декомпозиція (1D): матриця розділяється на рядки, що мінімізує обсяг комунікацій порівняно з передачею повних матриць на кожному етапі.
3. Використання NumPy та маршалінгу: для підвищення кеш-ефективності та швидкості локальних обчислень дані на серверах обробляються оптимізованими бібліотеками NumPy, а передача здійснюється через конвертацію у списки Python.

4. Синхронізація через Futures: обмін даними відбувається асинхронно, а агрегація результатів здійснюється після завершення всіх віддалених викликів, що забезпечує правильність обчислень.

Програмно реалізовано задану задачу та описано всі компоненти коду.

Аналіз результатів тестування програми виконано для різної кількості елементів та вузлів (табл. 1-3).

Таблиця 1 – Час виконання програми на різних вхідних даних (с.)

Кількість процесів, ***P***	***N = 500*** (елементів)	***N = 1000*** (елементів)	***N = 2000*** (елементів)
Локально (1 потік)	0.15	1.80	15.20
2 Сервери (RPC)	0.22	1.10	7.80
4 Сервери (RPC)	0.35	0.85	4.10

Таблиця 2 – Коефіцієнти прискорення S_p

Загальна розмірність матриці, ***N***			2
Коефіцієнт прискорення, S_p	0.43	2.12	3.70

Таблиця 3 – Коефіцієнти ефективності E_p

Кількість процесів, ***P***	2	4
Коефіцієнт ефективності, E_p	0.97	0.925

Проаналізувавши обчислені коефіцієнти прискорення та ефективності (табл. 1-3), можливо зробити висновки:

- при невеликих вхідних даних ***(N=500)*** значення коефіцієнта прискорення є меншим за одиницю, оскільки накладні витрати на маршалінг XML-RPC та мережеву латентність перевищують виграш від паралелізації;
- збільшення кількості процесів при роботі з великими матрицями ***(N=2 000)*** призводить до значного збільшення прискорення ***(S_4 =3.7)***, оскільки частка обчислень ***($O(N^3)$)*** стає домінуючою над комунікаціями ***($O(N^2)$)***;
- ефективність системи ***(E_p)*** залишається високою (понад 90%) для великих задач, що підтверджує масштабованість обраної 1D-декомпозиції в архітектурі Master-Worker.

Отже, основними чинниками, які впливають на зміну показників, є розмір вхідних даних, кількість обчислювальних вузлів та накладні витрати на серіалізацію даних у протоколі RPC. Використання розподілених систем на базі RPC є економічно та технічно доцільним лише для задач із грубою гранулярністю (великим обсягом обчислень на одиницю переданих даних).

Висновки

Досліджено матричне множення як одну із ключових операцій в лінійній алгебрі, визначено основні сфери її застосування. Розглянуто три різні методи виконання операції матричного множення: «наївний» метод, метод Штрассена та метод Коппермана-Віноградa, описано алгоритми їх вирішення та сфери застосування. На основі літературних джерел досліджено технологію RPC як альтернативу Mesh-мережам у розподілених системах, розглянуто принципи роботи XML-RPC та сформульовано ідею використання архітектури Master-Worker для паралельних обчислень.

Програмно реалізовано задану задачу мовою Python та описано всі компоненти коду [8]. Також наведено UML-діаграми класів та активності [9]. Проведено тестування розробленої програми на різних розмірностях матриць, проаналізовано її результати, досліджено коефіцієнти прискорення та ефективності.

Визначено, що збільшення кількості процесів призводить до значного збільшення коефіцієнта прискорення, проте, при малих обсягах даних спостерігається зменшення ефективності через високі накладні витрати на комунікацію в RPC-системі.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Remote procedure call. URL: https://en.wikipedia.org/wiki/Remote_procedure_call
2. Множення матриць URL: https://uk.wikipedia.org/wiki/Множення_матриць.
3. Глущенко В. М. Технології розподілених систем і паралельних обчислень: Навчальний посібник. Вінниця: ВНТУ, 2018. 214 с.
4. Олійник А. М. Розподілені обчислювальні системи та мережі: Підручник. Львів: Видавництво Львівської політехніки, 2021. 410 с.
5. Степаненко В. Я. Основи теорії паралельних обчислень. К.: Техніка, 2016. 320 с.
6. Ковальчук Л. С. Чисельні методи в інформатиці: Навчальний посібник. К.: Вища освіта, 2017. 380 с.
7. Минайленко Р.М. Паралельні та розподілені обчислення: Навч. посібник. Кропивницький: Видавець Лисенко В. Ф., 2021. 153 с. URL: https://dspace.kntu.kr.ua/server/api/core/bitstreams/396e02d2-725b-47b5-a1c0-ae07a9bec326/content .
8. Tkinter — Python Interface to Tcl/Tk. Python.org. URL: https://docs.python.org/3/library/tkinter.html
9. The Unified Modeling Language. URL: https://www.uml-diagrams.org/.

Пірняк Владислав Віталійович – студент кафедри комп`ютерних наук, факультет інтелектуальних інформаційних технологій та автоматизації, Вінницький національний технічний університет, м.Вінниця, e-mail: pirniak2005@gmail.com;

Денисюк Валерій Олександрович – канд. техн. наук, доцент, доцент кафедри комп'ютерних наук, Вінницький національний технічний університет, м.Вінниця, e-mail: vad64@i.ua.

Pirniak Vladyslav Vitaliyovich – student of Computer Science Department, Faculty of Intelligent Information Technologies and Automation, Vinnytsia National Technical University, Vinnytsia, e-mail: pirniak2005@gmail.com;

Denysiuk Valerii Olexandrovich – Ph.D., Assistant Professor, Assistant Professor of the Chair of Computer Science, Vinnytsia National Technical University, Vinnytsia, e-mail: vad64@i.ua