УДК 004.8

І.І. Пущал
В.О. Денисюк

РОЗРОБКА АЛГОРИТМУ ПАРАЛЕЛЬНОГО ПРОГРАМУВАННЯ З ВИКОРИСТАННЯМ MPI

Вінницький національний технічний університет

Анотація

У роботі розглянуто особливості розробки паралельного алгоритму з використанням технології MPI на прикладі розробки паралельного алгоритму для обчислення суми елементів великого масиву. Також створено програмну реалізацію паралельного алгоритму з використанням MPI та проведено тестування його продуктивності на різних наборах даних. Результати можуть бути використані для покращення швидкодії і продуктивності у різноманітних програмах, які потребують обробки великих обсягів даних.

Ключові слова: *паралельне програмування, MPI, розподілені обчислення, продуктивність.*

Abstract

The paper considers the features of developing a parallel algorithm using MPI technology using the example of developing a parallel algorithm for calculating the sum of elements of a large array. A software implementation of a parallel algorithm using MPI was also created and its performance was tested on various data sets. The results can be used to improve performance and productivity in various programs that require processing large amounts of data.

Keywords: *parallel programming, MPI, distributed computing, performance.*

Вступ

Актуальність теми полягає у тому, що в сучасному світі, де обсяги даних зростають експоненціально, паралельне програмування стає надзвичайно важливим для ефективної обробки інформації та виконання складних обчислювальних задач.

Використання багатоядерних процесорів і кластерних систем вимагає від програмістів знань про паралельні алгоритми та технології, такі як MPI (Message Passing Interface), які ідозволяють розподіляти навантаження між кількома процесами.

Метою роботи є дослідження можливостей паралельного програмування з використанням MPI для реалізації ефективних алгоритмів обробки даних, а також порівняння продуктивності паралельних та послідовних реалізацій.

Об'єктом дослідження є паралельне програмування з використанням MPI для обробки великих обсягів даних.

Розглянуто теоретичні основи паралельного програмування та основи MPI; розроблено оптимальні алгоритми, які реалізують паралельне обчислення з урахуванням обраних критеріїв ефективності; проведено тестування програмної реалізації розроблених алгоритмів.

Постановка задачі дослідження

Задачі дослідження полягають у вирішенні наступних питань:

- проаналізувати існуючі підходи до паралельного програмування і обґрунтувати переваги використання MPI в порівнянні з іншими технологіями;
- розробити оптимальні алгоритми для обчислення на основі mpi, з урахуванням часу виконання та використання пам'яті;
- розробити програму для обчислення суми елементів масиву з використанням паралельного програмування, провести тестування програми та провести аналіз отриманих результатів.

Виклад основного матеріалу

Паралельний алгоритм - це опис процесу обробки інформації, орієнтований на реалізацію в колективі обчислювачів. Такий алгоритм, у якому декілька операцій або підзадач виконуються одночасно, на відміну від послідовного, передбачає одночасне виконання множини операцій в межах одного кроку обчислень і як послідовний алгоритм зберігає залежність подальших етапів від результатів попередніх [1 – 4].

Паралельний алгоритм рішення задачі складає основу паралельної програми. Паралельна програма у свою чергу, впливає на алгоритм функціонування колективу обчислювачів. Запис паралельного алгоритму на мові програмування, доступній колективу обчислювачів, називають паралельною програмою. Вони можуть бути ефективно реалізовані в різних середовищах, таких як багатоядерні процесори, кластери або суперкомп'ютери, використовуючи технології типу OpenMP, MPI (Message Passing Interface), CUDA тощо.

Вважатимемо, що відомо традиційний (послідовний) спосіб розв’язання деякої задачі і далі необхідно організувати її виконання з використанням паралельної обробки даних. Загальна схема розробки паралельного алгоритму містить такі етапи [2]:

- виконання декомпозиції задачі на складові частини одним із відомих способів;
- виявлення інформаційних залежностей;
- масштабування складових частин задачі;
- розподіл складових частин задачі між процесорами (ядрами).

Етап декомпозиції є першим етапом розробки паралельного алгоритму. Існує декілька моделей даних – паралельна модель даних, модель графового завдання, модель робочого басейну, модель майстра-підданого, модель трубопроводу та гібридна модель. Проблема може полягати у виборі того чи іншого способу. Перелічимо відомі способи декомпозиції [5]:

- паралельна модель даних (The Data-Parallel Model),
- модель графового завдання (The Task Graph Model),
- модель робочого пулу (Work Pool Model),
- модель трубопроводу (The Pipeline Model),
- гібридна модель (Hybrid Model).

Після розбиття початкової задачі на складові частини виконується аналіз зв’язків між ними, тобто здійснюється етап виявлення інформаційних залежностей.

Етап масштабування складових частин задачі виконується в тому випадку, коли кількість наявних підзадач відрізняється від кількості наявних процесорів (ядер).

Існує навіть спеціальний термін – «зернистість», – який характеризує рівень декомпозиції початкової задачі на окремі підзадачі. Дрібнозернистий паралелізм може оптимально завантажити всі наявні процесори, однак важко аналізувати паралельну програму. При цьому є межа складності підпрограм, коли загальний час виконання програм вже не буде зменшуватись внаслідок зростання числа допоміжних операцій зі створення нових процесів та потоків

Таким чином, в багатьох випадках необхідний ще один етап розробки паралельних алгоритмів – етап розподілу підзадач між процесорами [3, 4].

MPI (Message Passing Interface) є стандартом для обміну повідомленнями між кількома комп'ютерами або процесорними ядрами, які виконують паралельну програму в розподіленій пам'яті.

MPI підтримує програмування за принципом MIMD (Multiple Instruction Multiple Data), що дозволяє об'єднувати процеси з різним вихідним кодом. Однак розробка таких програм є складною, тому на практиці часто використовується концепція SIMD (Single Instruction Multiple Data), коли всі процеси виконують один і той самий код [6].

Оптимізація паралельного алгоритму з використанням MPI виконано за такими напрямками:

- балансування навантаження;
- зменшення кількості комунікацій;

- агрегація даних;
- оптимізація використання пам'яті;
- використання асинхронних комунікацій;
- адаптація до архітектури%
- тестування масштабованості.

Аналіз результатів тестування програми виконано для різної кількості N елементів масиву (табл.1).

Таблиця 1 – Порівняння послідовного та паралельного алгоритмів при різній кількості N елементів масиву

N	Послідовний	Паралельний
10	All items sum: 422 Time execution: 0 seconds All items sum: 471 Time execution: 0 seconds All items sum: 330 Time execution: 0 seconds Середній час виконання: 0 с.	Total sum of the array: 527 Time execution: 0.0002281 seconds Total sum of the array: 545 Time execution: 0.0002252 seconds Total sum of the array: 475 Time execution: 0.0002669 seconds Середній час виконання: 0.00024 с
1.000	All items sum: 50456 Time execution: 0 seconds All items sum: 49829 Time execution: 0 seconds All items sum: 48154 Time execution: 0 seconds Середній час виконання: 0 с.	Total sum of the array: 48553 Time execution: 0.0002784 seconds Total sum of the array: 50184 Time execution: 0.0002541 seconds Total sum of the array: 50224 Time execution: 0.0003053 seconds Середній час виконання: 0.00028 с
1.000.000	All items sum: 49455131 Time execution: 0.044 seconds All items sum: 49462328 Time execution: 0.041 seconds All items sum: 49440179 Time execution: 0.04 seconds Середній час виконання: 0.42 с.	Total sum of the array: 49484802 Time execution: 0.0244812 seconds Total sum of the array: 49482207 Time execution: 0.0277303 seconds Total sum of the array: 49467513 Time execution: 0.023824 seconds Середній час виконання: 0.025 с
100.000.000	All items sum: 651714673 Time execution: 3.856 seconds All items sum: 652213743 Time execution: 3.78 seconds All items sum: 651351908 Time execution: 3.751 seconds Середній час виконання: 3.79 с.	Total sum of the array: 651516736 Time execution: 2.42357 seconds Total sum of the array: 651684244 Time execution: 2.30868 seconds Total sum of the array: 652009601 Time execution: 2.34366 seconds Середній час виконання: 2.36 с

Аналізуючи таблицю 1, можна побачити чітку різницю в часі виконання між послідовним і паралельним алгоритмом для різних розмірів масивів. Для найменшого масиву з 10 елементів як послідовний, так і паралельний алгоритми демонструють практично однаковий результат, де час

виконання настільки малий, що він округляється до нуля. Однак, при цьому паралельний алгоритм все ж таки показує деяку різницю в часі на рівні мікросекунд, що можна пояснити витратами на розподіл і збирання даних між процесами.

Зі збільшенням розміру масиву до 1 000 000 елементів, можна спостерігати, як час виконання послідовного алгоритму починає значно зростати — до 0.42 секунди, тоді як паралельний алгоритм продовжує демонструвати значно менші часи виконання (0.025 с). Для масиву на 100 мільйонів елементів різниця стає ще більш очевидною: послідовний алгоритм працює майже 4 секунди, тоді як паралельний завершує роботу приблизно за 2.36 секунди. Це демонструє перевагу паралельного підходу на великих обсягах даних, оскільки він дозволяє розподілити обчислення між кількома процесами і суттєво зменшити загальний час виконання.

Висновки

Отже, розроблено паралельний алгоритм для обчислення суми елементів великого масиву з використанням MPI. Завдання розподілялися між процесами, що значно скоротило час виконання порівняно з послідовним підходом. Для спрощення управління даними застосовано об'єктно-орієнтований підхід, зокрема створено класи MPIManager для роботи з MPI та ArrayProcessor для обробки масиву. Це дозволило покращити структуру програми, зробити код читабельним і легким для модифікації. Також розроблено UML-діаграми класів і активностей, які візуалізували структуру програми та логіку алгоритму, забезпечуючи глибше розуміння принципів паралельного програмування. Підтверджено перевагу паралельного підходу на великих обсягах даних.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Жолубак Л.І., Буряк Н.Є. Етапи розробки паралельних алгоритмів. URL: https://sci.ldubgd.edu.ua/bitstream/123456789/9280/1/Zholubak.pdf .
2. Гомон, К. О. Паралельний алгоритм розв'язування задач теорії пружності/ К. О. Гомон, І. І. Дияк, М. Ф. Копитко // Вісник Національного університету "Львівська політехніка". Серія: Інформаційні системи та мережі: збірник наукових праць / МОН України. Львів : Львівська політехніка, 2017. № 872. С. 101-110. URL: https://science.lpnu.ua/sites/default/files/journal-paper/2018/jun/13007/ilovepdfcom-101-110.pdf .
3. Паралельний алгоритм. URL: https://uk.wikipedia.org/wiki/Паралельний_алгоритм .
4. Parallel Algorithm Models in Parallel Computing. URL: https://www.geeksforgeeks.org/parallel-algorithm-models-in-parallel-computing .
5. Декомпозиція в програмуванні. URL: https://foxminded.ua/dekompozytsiia-v-prohramuvanni .
6. Message Passing Interface (MPI). URL: https://www.techtarget.com/searchenterprisedesktop/definition/message-passing-interface-MPI .

Пущал Ігор Ігорович – студент кафедри комп`ютерних наук, факультет інтелектуальних інформаційних технологій та автоматизації, Вінницький національний технічний університет, м.Вінниця, e-mail: igorek12102004@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.

Pushchal Igor Igorovich – student of Computer Science Department, Faculty of Intelligent Information Technologies and Automation, Vinnytsia National Technical University, Vinnytsia, e-mail: igorek12102004@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