DOI 10.34185/1562-9945-3-146-2023-06
УДК 004.032.24:519.612.2

Г.Я. Вохмянін, О.О. Жульковський, І.І. Жульковська,
В.А. Катан, В.Ю. Клим, І.І. Кузнєцов

ОЦІНКА ЕФЕКТИВНОСТІ РЕАЛІЗАЦІЇ АСИНХРОННИХ ОБЧИСЮВАЛЬНИХ АЛГОРИТМІВ ЗАСОБАМИ КОРУТИН ТА ПОТОКІВ С++

Аннотація. Одним із механізмів підвищення продуктивності обчислень при комп'ютерному моделюванні є корутини – зручний засіб управління асинхронними операціями, введений у стандарт С++20. Метою роботи є підвищення швидкодії обчислювальних алгоритмів – основи комп'ютерних моделей – за рахунок використанням механізму корутин та потоків даних. В результаті застосування нового, більш ефективного і простого підходу вдалося досягти відносного збільшення швидкості обчислень у 1,94 рази при значній розмірності матриці.
Ключові слова: співпрограма, корутина, асинхронне програмування, багатопоточність, паралельні обчислення, алгоритм, швидкодія.

Постановка проблеми. Багатоядерні системи є найбільш ефективними при застосуванні у великих серверних центрах, а також для хмарних обчислень. Однак, не зважаючи на відому складність програмної реалізації, паралельні обчислення на мультипроцесорах все частіше застосовуються під час комп'ютерного моделювання.

Традиційне програмування часто використовує операції, що блокують основний потік програми до тих пір, поки відповідну операцію не буде завершено. Зазвичай це призводить до уповільнення або простою програмного застосунку замість пришвидшення виконання програми чи виконання іншого завдання.

Корутини (courutines) надають змогу уникати блокування основного потоку шляхом тимчасового призупинення і відновлення програми. Окрім блокуючих операцій використання корутин було б у нагоді при обробці великих масивів даних. Під час призупинення корутини зберігається її стан включно зі значеннями локальних змінних із переходом в очікування завершення відповідної операції. Після завершення операції виконання корутини поновлюється зі збереженого місця, що дає змогу продовжити роботу без блокування потоку вико-

ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)

нання [1].

Отже використання корутин поліпшує ефективність програми і особливо, коли остання повинна виконувати безліч блокувальних операцій або значних обчислень.

Аналіз останніх досліджень і публікацій. Можливість призупинення та поновлення виконання певних програмних конструкцій вперше описана М. Конвеєм (1963 р.) у публікації про розробку компіляторів [2]. Тоді ж започатковано термін «співпрограма». Д. Кнут (1968 р.) розробив концепцію кооперативних багатозадачних систем (Cooperative Multitasking Systems) [3], яка може бути реалізована саме за допомогою механізму корутин. Він описав, як саме багатозадачні системи можуть бути використані для ефективного розподілу процесорних ресурсів між різноманітними процесами.

Розвиток співпрограм та корутин триває й досі та реалізовано у сучасних системах та мовах програмування високого рівня.

Авторами даної роботи також були проведені дослідження у галузі паралельного та багатопоточного програмування із застосуванням методів розбиття виконання програмного коду між декількома потоками з метою скорочення часу обчислення СЛАР та підвищення ефективності комп'ютерного моделювання [4, 5].

Мета дослідження. Метою даної роботи є підвищення ефективності комп'ютерного моделювання за рахунок збільшення швидкодії обчислювального експерименту на прикладі реалізації сучасними засобами C++ алгоритму множення матриці на вектор з використанням корутин й потоків даних.

Для досягнення поставленої мети перед роботою сформульовані наступні завдання: реалізація послідовного й асинхронного (з розбиттям на два потоки) алгоритмів реалізації методу множення матриці на вектор; проведення порівняльної характеристики продуктивності реалізації послідовного й асинхронного алгоритмів при значній розмірності матриці; вироблення концепції для подальшого розвитку підходів до підвищення швидкодії комп'ютерного моделювання, що застосовує реалізацію асинхронних обчислювальних алгоритмів на мультипроцесорах.

Викладення основного матеріалу дослідження. Як зазначалося, невеликі частини програмного коду, які можна призупинити і відновити через деякий проміжок часу, називаються співпрограмами. Їх характерною особливістю є можливість зберігання свого стану під час призупинення та відновлення програми з місця попередньої зупинки [6, 7]. Співпрограмами можуть бути кору-

ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)

тини, генератори числових, символьних та інших послідовностей, ітератори та асинхронні функції.

Корутини, в основі яких лежить концепція співпрограм, є новим механізмом синхронізації, доданим у стандарт C++20 мови програмування C++ [6, 7]. Вони дозволяють використовувати синхронний стиль написання коду в асинхронному середовищі – код, написаний асинхронним чином, виглядає так само, як і синхронний, проте виконується асинхронно. Такий код, у порівнянні з традиційним асинхронним, є більш лаконічним, легким для читання, підтримки та тестування, забезпечує зменшення навантаження на потоки, збільшення продуктивності, має простий та гнучкий синтаксис. Серед недоліків виділяють обмежену підтримку функціональності корутин різними компіляторами C++ [6]. Використання корутини неналежним чином може призвести до некоректного розподілу пам’яті та блокуванню потоку, наслідком чого стане уповільнення роботи програмного застосунку [7].

Корутини та програмні потоки є суттєво різними підходами до організації паралельних обчислень на ЕОМ із багатоядерною архітектурою. Механізм потоків відносять до багатопоточного програмування [8, 9]. При цьому кожен потік має незалежний власний набір даних, викликів, локальних змін тощо. Корутини ж відносять до одного з сучасних засобів асинхронного програмування. Незважаючи на те, що ці концепції реалізують різні підходи до паралельного програмування, їх можна ефективно використовувати разом для досягнення максимальної продуктивності. При цьому програмний код буде розбито на деяку кількість потоків [8, 9], а корутини будуть ефективними при роботі з асинхронною обробкою функцій введення-виведення даних (робота з файлами, мережевими протоколами, веб-сервісами) і з обробкою великих об’ємів даних. Отже використання корутин для реалізації багатопоточної обробки даних шляхом розподілення задач між різними корутинами, працюючими у різних потоках, є цілком допустимим [1, 7]. Необхідно пам’ятати, що для правильного використання потоків та корутин треба враховувати особливості конкретної задачі, що дасть можливість правильно розподіляти ресурси [9]. Інакше продуктивність може знизитись через додаткові витрати на контекстне перемикання, синхронізацію доступу до ресурсів тощо.

Структура для організації роботи з корутинами містить перелік відповідних функцій з бібліотеки <coroutine>, яка є стандартною у C++20 [6]. Попередні стандарти C++ містять експериментальну версію цієї бібліотеки, що підключається через імпорт <experimental /coroutine>. Використання експериментальної

версії може дуже відрізнятися від основної.

Найголовніші оператори бібліотеки <coroutine>, що використовуються для управління корутинами, представлені наступним чином [6]:

– ключове слово co_yeild – для передачі управління від поточної корутини до тієї, що зробила виклик;

– ключове слово co_await – для призупинення корутини, поки асинхронна операція, від якої корутина чекає відповіді, не буде завершена;

– ключове слово co_return – для повернення значення з кору тини (звичайний return не дозволяє корутині зберегти власний стан та поновити її виконання у майбутньому).

Для створення асинхронної операції й забезпечення ефективного використання ресурсів системи ключові слова co_yeild та co_return зазвичай використовуються разом з оператором co_await. До того ж, оператор co_yeild доцільно використовувати у тих випадках, коли мова йде про генерування послідовності значень певного типу, або організацію «лінивих» обчислень [1, 6].

Всі вищеописані оператори стандарту C++20 разом дозволяють створювати ефективні корутини, що значно спрощує розроблення асинхронних та паралельних застосунків, підвищуючи при цьому продуктивність їхнього виконання.

З метою проведення досліджень розроблено класичний алгоритм множення матриці на вектор та його асинхронну модифікацію з використанням корутин та розбиттям на програмні потоки. Для заміру часу виконання програми була використана стандартна бібліотека <chrono>, яка надає багато можливостей для роботи з часом.

Для проведення експериментів використана наступна інфраструктура ЕОМ: Intel Core i5-10300H (6 cores, 2.5 GHz/ 4.5 GHz); Goodram DDR4 (4 GB) × 4 = 16 GB; Microsoft Windows 11; IDE Microsoft Visual Studio C++; ISO/IEC C++20; <coroutine>.

Тестування проводилось із використанням однопоточного та асинхронного, з розбиттям на два потоки, алгоритмів множення генерованої матриці випадкових значень змінної розмірності на вектор.

З’ясовано, що при невеликій розмірності матриці розроблений асинхронний алгоритм з використанням корутин та розбиттям на два потоки менш ефективний, ніж однопоточний. Це пов’язано з тим, що компілятору необхідний деякий час, аби створити потоки та запустити виконання одночасно. Вже з великою розмірністю помітне зростання швидкодії використання саме асинхрон-

ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)

ного алгоритму. При розмірності n>1200 використання асинхронного алгоритму, розбитого на два потоки, гарантовано доцільніше за однопоточний. На рис. 1 зображені порівняльні результати проведених експериментів.

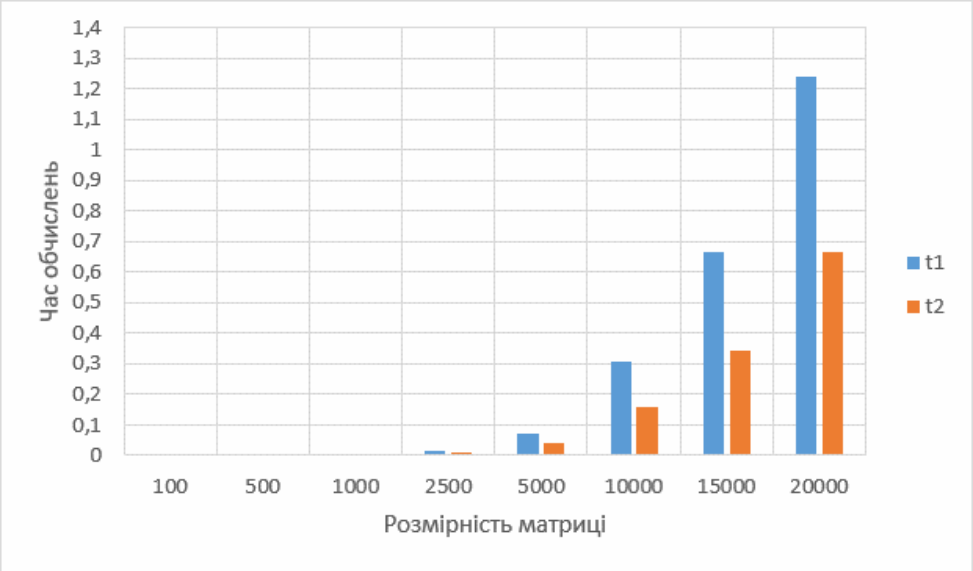

Рисунок 1 – Залежність часу обчислень від розмірності матриці: t1 – однопоточний; t2 – багатопоточний

Висновки. В результаті роботи розроблено алгоритм множення матриці на вектор та його модифікований асинхронний варіант із використанням механізму корутин та розбиттям на два потоки даних, за рахунок якого вдалося досягти збільшення швидкості обчислень у 1,94°рази при розмірності матриці 15000 ($2{,}25\times10^6$ елементів). Отримані дані кореспондуються з результатами подібних досліджень проблеми підвищення ефективності комп'ютерного моделювання з використанням альтернативних програмних та апаратних засобів. Таким чином доведено, що новий спосіб вирішення проблем асинхронного програмування надає більш ефективний та простий механізм управління асинхронними операціями. Результати роботи успішно використано при розробці інформаційно-моделюючої системи [10], створеної на основі авторської методики математичного моделювання процесів комбінованого теплообміну [11].

ЛІТЕРАТУРА

1. Andrist B. C++ High Performance: Master the art of optimizing the functioning of your C++ code / B. Andrist, V. Sehr, B.°Garney. – 2nd Edition: textbook. – 2020. – 540 p.

2. Conway Melvin E. Design of a separable transition-diagram compiler/ Melvin E. Conway // Communications of the ACM. – 1963. – Vol. 6. – № 1. – P.°396 – 408.

3. Knuth Donald E. Fundamental algorithms, The art of computer programming /

Donald E. Knuth. – Publisher: Addison-Wesley, 1968. – Vol.1.°– 634 p.
4. Zhulkovskyi O.O. Evaluating the effectiveness of the implementation of computational algorithms using the OpenMP standard for parallelizing programs / O. O. Zhulkovskyi, I. I. Zhulkovska, V. V. Shevchenko // Informatics and Mathematical Methods in Simulation. – 2021. – Vol.11. – №°4. – P. 268 – 277.
DOI: 10.15276/imms.v11.no4.268
5. Zhulkovskyi O. O. Evaluation of the efficiency of the implementation of parallel computational algorithms using the <thread> library in C++ / O.°O.°Zhulkovskyi, I. I. Zhulkovska, V. V. Shevchenko, H. Ya. Vokhmianin // Computer Systems and Information Technologies. – 2022. – № 3. –P. 49°–°55. DOI: 10.31891/csit-2022-3-6
6. Fertig A. Programming with C++20: Concepts, Coroutines, Ranges, and more/ A. Fertig, F. Panter. – Textbook. – 2021. – 333 p.
7. Browning J. Burton. C++20 Recipes: A Problem-Solution Approach / J.°Burton Browning, B. Sutherland. – 2nd Edition: textbook. – 2020. – 657 p.
8. Williams A. C++ Concurrency in Action: Practical Multithreading/ A.°Williams. – Textbook. – 2012. – 528 p.
9. Mastering Posch M. C++ Multithreading: Write robust, concurrent, and parallel applications / Posch M. Mastering. – Textbook. – 2017. – 246 p.
10. Zhulkovskii O.A. Information-Modeling Forecasting System for Thermal Mode of Top Converter Lance / O.°A. Zhulkovskii, S.°P. Panteikov, I.°I.°Zhulkovskaya // Steel in Translation. – 2022. – Vol. 52. – № 5. –P. 495°–°502.
DOI: 10.3103/S0967091222050138
11. Zhulkovskyi O.O. Features of Mathematical simulation of the processes of combined heat transfer in waveguides / O.O. Zhulkovskyi, Iu.V. Savchenko, I.°I. Zhulkovska et al. // Proceedings of the 2022 IEEE 4th International Conference on Modern Electrical and Energy System, MEES 2022. –P.°452°–°456.
DOI: 10.1109/MEES58014.2022.10005676

REFERENCES

1. Andrist B. C++ High Performance: Master the art of optimizing the functioning of your C++ code / B. Andrist, V. Sehr, B.°Garney. – 2nd Edition: textbook. – 2020. – 540 p.
2. Conway Melvin E. Design of a separable transition-diagram compiler/ Melvin E. Conway // Communications of the ACM. – 1963. – Vol. 6. – № 1. – P.°396 – 408.
3. Knuth Donald E. Fundamental algorithms, The art of computer programming / Donald E. Knuth. – Publisher: Addison-Wesley, 1968. – Vol.1.°– 634 p.
4. Zhulkovskyi O.O. Evaluating the effectiveness of the implementation of computational algorithms using the OpenMP standard for parallelizing programs / O. O. Zhulkovskyi, I. I. Zhulkovska, V. V. Shevchenko // Informatics and Mathematical Methods in Simulation. – 2021. – Vol.11. – №°4. – P. 268 – 277.

ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)

DOI: 10.15276/imms.v11.no4.268
5. Zhulkovskyi O. O. Evaluation of the efficiency of the implementation of parallel computational algorithms using the <thread> library in C++ / O.°O.°Zhulkovskyi, I. I. Zhulkovska, V. V. Shevchenko, H. Ya. Vokhmianin // Computer Systems and Information Technologies. – 2022. – № 3. – P. 49°–°55. DOI: 10.31891/csit-2022-3-6
6. Fertig A. Programming with C++20: Concepts, Coroutines, Ranges, and more/ A. Fertig, F. Panter. – Textbook. – 2021. – 333 p.
7. Browning J. Burton. C++20 Recipes: A Problem-Solution Approach / J.°Burton Browning, B. Sutherland. – 2nd Edition: textbook. – 2020. – 657 p.
8. Williams A. C++ Concurrency in Action: Practical Multithreading/ A.°Williams. – Textbook. – 2012. – 528 p.
9. Mastering Posch M. C++ Multithreading: Write robust, concurrent, and parallel applications / Posch M. Mastering. – Textbook. – 2017. – 246 p.
10. Zhulkovskii O. A. Information-Modeling Forecasting System for Thermal Mode of Top Converter Lance / O.°A. Zhulkovskii, S.°P. Panteikov, I.°I.°Zhulkovskaya // Steel in Translation. – 2022. – Vol. 52. – № 5. – P. 495°–°502.
DOI: 10.3103/S0967091222050138
11. Zhulkovskyi O.O. Features of Mathematical simulation of the processes of combined heat transfer in waveguides / O.O. Zhulkovskyi, Iu.V. Savchenko, I.°I. Zhulkovska et al. // Proceedings of the 2022 IEEE 4th International Conference on Modern Electrical and Energy System, MEES 2022. – P.°452°–°456.
DOI: 10.1109/MEES58014.2022.10005676

Received 31.03.2023.
Accepted 05.04.2023.

Evaluation of the efficiency of implementation of asynchronous computing algorithms using coroutines and threads in C++

Modern multi-core systems are most effective when used in large server centers and for cloud computing. However, despite the known complexity of software implementation, parallel computing on multiprocessors is increasingly used in computer modelling.

Advanced mechanisms of synchronous and multithreaded programming are increasingly used to improve the productivity of numerical studies, reducing the time of computer models implementation. One such mechanism is coroutines, a convenient tool for managing asynchronous operations introduced in the C++20 standard. A special feature of coroutines is the ability to suspend a function at a certain stage, saving its state, and after some time resume its execution from the previous stop.

The aim of this research is to improve the performance of computer modelling by using coroutines and data threads.

As a result of the work, a test algorithm for multiplying a matrix by a vector and its modified asynchronous version using the coroutine mechanism and splitting into two data threads was developed, which allowed to achieve 1.94 times increase in the comput-

ing speed when the matrix dimension is 15000 (2.25×10^6 elements). It has been found that at a small matrix dimension, the developed asynchronous algorithm using coroutines and splitting into two threads is less efficient than the single thread algorithm. This is due to the fact that the compiler needs some time to create threads and start execution simultaneously. With a large dimensionality, the performance of the asynchronous algorithm increases significantly. With a matrix dimension of more than 1200, the use of an asynchronous algorithm divided into two threads is guaranteed to be more efficient than a single-threaded.

The data obtained are consistent with the results of similar studies of the problem of increasing the efficiency of computer modelling using alternative software and hardware.

The new method of solving the problems of asynchronous programming provides a more efficient and simple mechanism for managing asynchronous operations.

Вохмянін Гліб Ярославович – здобувач вищої освіти бакалаврського рівня, Дніпровський державний технічний університет.
Жульковський Олег Олександрович – доцент кафедри програмного забезпечення систем, Дніпровський державний технічний університет.
Жульковська Інна Іванівна – доцент кафедри кібербезпеки та інформаційних технологій, Університет митної справи та фінансів.
Катан Володимир Олександрович – доцент кафедри економічного моделювання, обліку та статистики, Дніпровський національний університет ім. Олеся Гончара.
Клим Вікторія Юріївна – доцент кафедри кібербезпеки та інформаційних технологій, Університет митної справи та фінансів.
Кузнєцов Ілля Ігоревич – магістр, Дніпровський державний технічний університет.

Vokhmianin Hlib – Student, Dniprovsky State Technical University.
Zhulkovskyi Oleg – Associate Professor of Department of Software Systems, Dniprovsky State Technical University.
Zhulkovska Inna – Associate Professor of Department of Cybersecurity and Information Technologies, University of Customs and Finance.
Katan Volodymyr – Associate Professor of Department of Economic Modeling, Accounting and Statistics, Oles Honchar Dnipro National University.
Klym Viktoriia – Associate Professor of Department of Cybersecurity and Information Technologies, University of Customs and Finance.
Kuznetsov Illia – Master, Dniprovsky State Technical University.

ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)