УДК 004.8

Д. Ю. Савлук
В.О. Денисюк

ДОСЛІДЖЕННЯ ТА РЕАЛІЗАЦІЯ ЗАВДАННЯ СИНХРОНІЗАЦІЇ КІЛЬКОХ ПОТОКІВ З ОДНИМ

Вінницький національний технічний університет

Анотація

Розглянуто задачу синхронізації кількох потоків у паралельній програмі за допомогою механізму події. Реалізовано модель, у якій один керуючий потік встановлює подію, а чотири робочі потоки очікують на її активацію та продовжують виконання після отримання сигналу. Проаналізовано основні принципи багатопотокового програмування, особливості механізмів синхронізації та обґрунтовано вибір інструментів реалізації. Побудовано UML-діаграми класів і діяльності, що формалізують структуру та логіку роботи системи. На основі діаграм розроблено програмну реалізацію мовою Python із використанням бібліотеки threading. Проведено тестування алгоритму, яке підтвердило коректність роботи, стабільність виконання та відсутність взаємоблокувань. Результати роботи демонструють ефективність застосування механізму події для координації потоків у паралельних системах.

Abstract

The problem of synchronizing multiple threads in a parallel program using an event mechanism was examined. A model was implemented in which one controller thread sets an event, while four worker threads wait for its activation and continue execution after receiving the signal. The fundamental principles of multithreaded programming and the characteristics of synchronization mechanisms were analyzed, and the choice of implementation tools was justified. UML class and activity diagrams were constructed to formalize the structure and logic of the system. Based on these diagrams, a software implementation in Python using the threading library was developed. Testing of the algorithm confirmed its correct operation, execution stability, and the absence of deadlocks. The obtained results demonstrate the effectiveness of using the event mechanism for thread coordination in parallel systems.

Вступ

Актуальність теми полягає в тому, що багатопоточність стала обов'язковим елементом сучасних програмних систем: від серверних застосунків і веб-платформ до систем реального часу, ігор, дата-центрів, моделювання та машинного навчання. Неправильна синхронізація потоків може призвести до критичних помилок - втрати даних, зависань, зниження продуктивності або непередбачуваної поведінки програми. Тому вміння правильно організовувати взаємодію між потоками та застосовувати механізми синхронізації є ключовою компетенцією майбутнього фахівця з комп'ютерних наук.

Метою роботи є розроблення та програмна реалізація багатопотокового алгоритму, у якому один керуючий потік генерує подію, а чотири робочі потоки очікують її настання та продовжують обчислення після синхронізації.

Постановка задачі дослідження

Задачі дослідження полягають у вирішенні наступних питань: дослідити механізми синхронізації потоків у паралельних системах та визначити доцільність використання подій (threading.Event) для моделі «керуючий - робочі потоки»; розробити програмну архітектуру, яка забезпечує: створення керуючого потоку та створення групи робочих потоків; розробити алгоритм обробки даних у потоках, що виконується після встановлення події (локальне сортування підмасивів); розробити окрему послідовну програму для виконання аналогічної задачі в одному потоці; провести експерементальне тестування.

Виклад основного матеріалу

Паралельні обчислення є ключовою складовою сучасних програмних систем, оскільки забезпечують можливість одночасного виконання декількох обчислювальних задач та підвищують пропускну здатність і продуктивність програмних комплексів. Розвиток багатоядерних процесорів та апаратних засобів сприяв широкому поширенню технологій багатопотоковості, які дозволяють розподіляти обчислення між кількома потоками виконання. У таких системах важливою є правильна організація синхронізації потоків, що запобігає конфліктам доступу до спільних ресурсів та забезпечує передбачувану взаємодію між ними [1–3].

У роботі реалізовано паралельну програмну модель, у якій один керуючий потік подає сигнал про початок виконання чотирьох робочих потоків. Керуючий потік ініціює подію, що виступає механізмом синхронізації, тоді як робочі потоки перебувають у стані очікування доти, доки подія не буде активована. Після отримання сигналу кожний робочий потік виконує обчислювальну задачу — локальне сортування частини масиву, що дозволяє моделювати паралельну обробку даних. Така модель відповідає класичній схемі «керуючий – робітники», у якій один потік визначає момент початку обробки, а група потоків виконує паралельні операції [4].

Математичне моделювання роботи системи базується на оцінці пропускної здатності кожного потоку та загального часу обробки даних. Пропускна здатність паралельної системи визначається найповільнішим потоком, а ефективність обчислень оцінюється за законом Амдала. При рівномірному розподілі навантаження між потоками час обробки скорочується пропорційно до кількості потоків, проте на швидкодію впливають накладні витрати на синхронізацію та розподіл даних [5, 6].

Для формалізації структури системи побудовано UML-діаграму класів, яка відображає компоненти програми: керуючий потік, робочі потоки та клас синхронізатора події. Додатково створено UML-діаграму діяльності, що демонструє логіку переходу потоків між станами очікування та виконання. Розроблено блок-схему алгоритму роботи програми, яка відображає основні етапи запуску, синхронізації та обробки даних.

Програмну реалізацію виконано мовою Python із використанням бібліотеки threading, що забезпечує підтримку багатопоточності та механізмів синхронізації подій. Робочі потоки виконують сортування частин масиву після отримання сигналу від керуючого потоку, що підтверджує коректність реагування на подію. Для проведення експериментального порівняння розроблено також послідовну програму, яка виконує аналогічне сортування без використання потоків.

У ході тестування здійснено серію запусків для масивів різного розміру з вимірюванням часу виконання. Результати показали, що при малих об'ємах даних перевагу має послідовна реалізація, натомість для більших масивів паралельна модель демонструє покращення продуктивності. Отримані дані відображено у таблиці порівняння часу. Загалом, реалізована система підтверджує ефективність застосування механізму події для керування паралельними потоками та демонструє можливість масштабування обчислювальних задач за рахунок розподілу навантаження [7].

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

Таблиця 1 – Порівняння часу виконання паралельної програми з послідовною

№ тесту	Кількість елементів масиву	Час послідовного сортування, мс	Час паралельного сортування, мс
1	100 000	12.759	12.211
2	1 000 000	135.628	131.895
3	10 000 000	1 359.103	1 325.269

Результати тестування демонструють коректність роботи обох реалізацій та підтверджують ефективність використання паралельної моделі синхронізації потоків. Для всіх трьох тестів зі збільшенням розміру масиву (від 100 тис. до 10 млн елементів) час виконання зростав пропорційно, що вказує на стабільність алгоритму й відсутність деградації продуктивності.

Паралельний варіант показав дещо менший час виконання порівняно з послідовним. Це пояснюється тим, що після активації події чотири робочі потоки можуть одночасно обробляти частини масиву, що дає змогу частково компенсувати накладні витрати на створення й синхронізацію потоків. Незважаючи на обмеження багатопоточності в Python, паралельна модель демонструє покращення часу обробки завдяки поділу задачі на незалежні фрагменти.

Отже, реалізована система синхронізації потоків із використанням події забезпечує: зменшення часу виконання порівняно з повністю послідовним підходом; коректну та передбачувану взаємодію між потоками; відсутність взаємоблокувань і збоїв під час тестування; можливість масштабування при збільшенні обсягу даних.

Отримані результати підтверджують доцільність використання механізму події threading.Event для координації роботи кількох потоків у задачах паралельної обробки даних.

Висновки

У межах курсової роботи було реалізовано та досліджено паралельну модель синхронізації потоків, у якій один керуючий потік керує запуском чотирьох робочих потоків за допомогою механізму події. Побудовані UML-діаграми та блок-схема відобразили структуру системи й логіку взаємодії потоків. Програмна реалізація мовою Python із використанням threading.Event підтвердила правильність роботи моделі: робочі потоки коректно очікують сигнал, одночасно переходять до виконання задачі та коректно завершують роботу без взаємоблокувань.

Для порівняння продуктивності було створено програму, яка виконує сортування цілого масиву в одному потоці. Проведене експериментальне тестування показало очікувану динаміку: паралельна модель демонструє скорочення часу виконання для великих масивів завдяки розподілу даних між кількома потоками, тоді як для малих обсягів переваги зменшуються через накладні витрати на синхронізацію.

Отже, обидві реалізації виконують поставлену задачу коректно. Паралельна програма підтвердила ефективність механізму події для координації потоків та можливість масштабування обчислень, а послідовна реалізація надала базовий орієнтир для об’єктивного порівняння продуктивності. Результати роботи демонструють доцільність застосування синхронізованої багатопоточності в задачах, що передбачають обробку великих обсягів даних.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Семеренко, В. П. Технології паралельних обчислень : навч. посіб. Вінниця: ВНТУ, 2018. – 104 с. URL: https://www.researchgate.net/publication/334710599_V_P_Semerenko_TEHNOLOGII_PARALELNIH_OBCISLEN_Ministerstvo_osviti_i_nauki_Ukraini_Vinnickij_nacionalnij_tehnicnij_universitet
2. Python Threading. URL: https://docs.python.org/3/library/threading.html
3. Multithreading and Synchronization. URL: https://www.geeksforgeeks.org/multithreading-in-python/
4. Кормен Т., Лейзерсон Ч., Рівест Р., Штайн К. Алгоритми: побудова та аналіз. Київ: Книжковий клуб «Клуб сімейного дозвілля», 2015. 328 с.
5. Семенов А. В. Паралельне програмування: навчальний посібник. Київ : КНЕУ, 2019. 268 с.
6. Bentaleb, A.; Yifan, L.; Xin, J.; et al. (2016). Parallel and Distributed Algorithms URL: https://www.comp.nus.edu.sg/~rahul/allfiles/cs6234-16-pds.pdf
7. Гнатюк О. В. Паралельні та розподілені обчислення: навчальний посібник. Львів: ЛНУ, 2018. 212 с.

Савлук Дмитро Юрійович – студент кафедри комп`ютерних наук, факультет інтелектуальних інформаційних технологій та автоматизації, Вінницький національний технічний університет, м.Вінниця, e-mail: savlukdi@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.

Savluk Dmytro Yuriyovych – student of Computer Science Department, Faculty of Intelligent Information Technologies and Automation, Vinnytsia National Technical University, Vinnytsia, e-mail: savlukdi@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