УДК 004.8

С.С.Чесноков
В.О. Денисюк

ДОСЛІДЖЕННЯ ЧАСУ ВИКОНАННЯ ПАРАЛЕЛЬНОЇ ПРОГРАМИ ІЗ ЗАСТОСУВАННЯМ РІЗНИХ ЗАСОБІВ СИНХРОНІЗАЦІЇ

Вінницький національний технічний університет

Анотація

Розглянуто дослідження ефективності різних засобів синхронізації для вирішення задачі взаємного виключення в паралельних обчислювальних системах мовою C#. Проаналізовано теоретичні аспекти «стану гонитви» та методи її усунення за допомогою блокуючих механізмів, активного очікування та атомарних операцій. Створено програмний комплекс за патерном «Стратегія», який дозволяє порівнювати продуктивність примітивів Monitor, Mutex, SpinLock та Interlocked. Результати експериментального тестування виявили залежність часу виконання програми від обраного засобу синхронізації та рівня конкуренції між потоками.

Ключові слова: *взаємне виключення, синхронізація, стан гонитви, паралельні обчислення, .NET, прискорення..*

Abstract

The paper examines the effectiveness of various synchronization methods for solving the mutual exclusion problem in parallel computing systems using C#. The theoretical aspects of the "race condition" problem are analyzed and methods for its elimination are investigated. A software implementation based on the "Strategy" pattern is created to compare the performance of Monitor, Mutex, SpinLock, and Interlocked. Experimental results revealed the dependence of execution time on the selected synchronization tool and the number of threads.

Keywords: *mutual exclusion, synchronization, race condition, parallel computing, .NET, speedup.*

Вступ

Багатопотокове програмування та розподілені обчислення є фундаментом сучасних програмних систем, починаючи від операційних систем і закінчуючи високопродуктивними серверами та хмарними платформами. У таких системах критично важливим є забезпечення коректної роботи зі спільними ресурсами [1, 2].

Коли кілька потоків одночасно намагаються змінити одні й ті ж дані, виникає стан гонитви (race condition), що призводить до непередбачуваних помилок. Для уникнення цього використовують механізми взаємного виключення (mutual exclusion), які гарантують цілісність даних, проте неминуче впливають на швидкодію системи.

Метою курсової роботи є проведення порівняльного аналізу часу виконання паралельної програми при використанні різних механізмів синхронізації та визначення умов їх доцільного використання.

Постановка задачі дослідження

Задачі роботи полягають у дослідженні наступних питань:

- теоретичні основи багатопотоковості, проблему стану гонитви та існуючі методи організації взаємного виключення;
- особливості роботи різних примітивів синхронізації (м’ютексів, семафорів, моніторів, атомарних змінних тощо);
- програмне забезпечення, яке реалізує задачу взаємного виключення з використанням різних засобів синхронізації;

- розгляд експериментального часу виконання програми для кожного методу при різному навантаженні та кількості потоків;
- порівняння отримані результати та сформулювати рекомендації щодо вибору оптимального засобу синхронізації для різних типів задач.

Виклад основного матеріалу

Ефективність сучасних обчислювальних систем значною мірою визначається обраною моделлю синхронізації при доступі до спільних ресурсів. Існує кілька фундаментальних підходів до організації цього процесу, які дозволяють уникнути «стану гонитви» (race condition), що виникає при одночасному зверненні кількох потоків до одних і тих самих даних. Для задач, пов'язаних із високопродуктивними обчисленнями, вибір між цими примітивами є критичним і залежить від тривалості критичної секції та архітектури процесора.

Для дослідження специфіки цих моделей у роботі використано платформу .NET та мову програмування C#, яка надає широкий спектр вбудованих засобів синхронізації [3]. Було досліджено та програмно реалізовано підходи: примітиви користувацького режиму, гібридні моделі та об'єкти режиму ядра [1, 2].

Примітиви користувацького режиму та гібридні моделі. Режим реалізації базується на механізмах, що мінімізують звернення до ядра операційної системи, що дозволяє суттєво знизити накладні витрати на перемикання контексту:

- атомарні операції (Interlocked) - використано для виконання неподільних інструкцій на апаратному рівні процесора, що забезпечує найвищу швидкодію, але обмежене лише простими математичними операціями;
- гібридне блокування (Monitor/Lock) - застосовано як універсальний засіб, який поєднує активне очікування (Spin-Wait) із подальшим переходом у стан сну, що дозволяє адаптуватися до рівня конкуренції між потоками;
- активне очікування (SpinLock) - потік не засинає, а виконується у циклі, перевіряючи доступність ресурсу, ефективно для коротких критичних секцій, проте, значно завантажує CPU при тривалих затримках.

Об'єкти режиму ядра. Реалізовані на базі системних викликів ОС, що забезпечує ширші можливості синхронізації, проте збільшує часові затримки. Використовуються м'ютекси (Mutex), які виступають об'єктами ядра, що дозволяють синхронізувати потоки навіть між різними процесами. Головним недоліком є високі витрати на перемикання контексту (Context Switch), що робить цей метод найповільнішим серед досліджених.

Для експериментальної перевірки ефективності було розроблено програмний комплекс «SyncBenchmark», який імітував навантаження у вигляді 1 000 000 операцій інкременту на кожен потік. Тестування проводилося на 6-ядерній системі Intel Core i7 (табл.1).

Таблиця 1 – Результати тестування продуктивності

Метод реалізації	Час виконання, мс	Накладні витрати	Механізм синхронізації
Interlocked (Атомарний)	140	Мінімальні	Апаратні інструкції
Monitor (Гібридний)	750	Середні	Spin-wait -> Context Switch
SpinLock (Активний)	1 150	Високі (CPU Load)	Busy Wait
Mutex (Ядерний)	9 500	Дуже високі	Context Switch

Проаналізувавши отримані результати (табл. 1), можна зробити висновки:

- Interlocked продемонстрував найвищу швидкодію, оскільки не блокує потік у розумінні операційної системи;

- Monitor (lock) зарекомендував себе як оптимальний універсальний засіб («золота середина»), працюючи у 20-30 разів швидше за ядерні примітиви;
- SpinLock виявив явище деградації продуктивності при високій конкуренції через виникнення ефекту «голодування потоків» (Thread Starvation);
- Mutex залишається безальтернативним для міжпроцесної синхронізації, попри найнижчу продуктивність через витрати на системні виклики ядра.

Висновки

Проведено комплексне дослідження часу виконання паралельної програми, де задача взаємного виключення вирішується за допомогою різних засобів синхронізації. Розроблено програмне забезпечення для емуляції стану гонитви (race condition) та порівняльного аналізу продуктивності примітивів синхронізації платформи .NET [3].

У ході роботи було детально розглянуто теоретичні основи багатопотокового програмування та проблему доступу до спільних ресурсів. Визначено, що некоректна синхронізація призводить до непередбачуваних помилок та порушення цілісності даних. Проведено класифікацію засобів взаємного виключення на блокуючі (Mutex, Monitor), засоби активного очікування (SpinLock) та атомарні операції (Interlocked). Встановлено, що вибір конкретного інструменту суттєво впливає на масштабованість системи відповідно до закону Амдала [4].

Створено тестовий програмний комплекс мовою C# з використанням архітектурного патерну «Стратегія» [5], що забезпечило уніфікований підхід до тестування різних алгоритмів (Interlocked, Monitor, SpinLock, Mutex). Реалізовано гнучку систему бенчмаркінгу, яка дозволяє змінювати кількість потоків (від 2 до 16) та обсяг навантаження, забезпечуючи високу точність вимірювань за допомогою System.Diagnostics.Stopwatch.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Мартинюк А. Паралельні обчислення: навчальний посібник. Вінниця: ВНТУ, 2021. 78 с. URL: https://mpa.vntu.edu.ua/fdb/838/Lec_CITCSHI/Tema_2.pdf .
2. Минайленко Р.М. Паралельні та розподілені обчислення: Навчальний 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 .
3. Коноваленко І.В. Платформа .NET та мова програмування C# 8.0: навчальний посібник / Коноваленко І.В., Марущак П.О. Тернопіль: ФОП Паляниця В. А., 2020. 320 с. URL: https://elartu.tntu.edu.ua/bitstream/lib/32825/1/Konovalenko%20I.%20.NET-C%23.pdf .
4. Закон Амдала. URL: https://uk.wikipedia.org/wiki/Закон_Амдала.
5. Refactoring.Guru. URL: https://refactoring.guru/uk

Чесноков Сергій Сергійович – студент кафедри комп`ютерних наук, факультет інтелектуальних інформаційних технологій та автоматизації, Вінницький національний технічний університет, м.Вінниця, e-mail: chesnokovserhiivinn@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.

Chesnokov Serhii Serhiiovych – student of Computer Science Department, Faculty of Intelligent Information Technologies and Automation, Vinnytsia National Technical University, Vinnytsia, e-mail: chesnokovserhiivinn@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 .