УДК 004.032.26, 004.8

**В.В. Грепечук**
**В.О. Денисюк**

# ДОСЛІДЖЕННЯ СИНХРОННОГО І АСИНХРОННОГО ОПРАЦЮВАННЯ ДАНИХ В ПАРАЛЕЛЬНИХ ТА РОЗПОДІЛЕНИХ СИСТЕМАХ

Вінницький національний технічний університет

***Анотація***

*Розглянуто розробку та дослідження алгоритмів синхронного та асинхронного опрацювання даних у паралельних системах з використанням мови програмування Python. Розглянуто теоретичні основи архітектур зі спільною пам'яттю та подієво-орієнтованих моделей, обґрунтовано вибір бібліотек threading та asyncio для програмної реалізації. У роботі створено програмний комплекс, який реалізує обробку великих масивів даних двома методами, та проведено порівняльний аналіз їх продуктивності. Використання результатів дозволить оптимізувати архітектурні рішення при проектуванні високонавантажених систем обробки інформації.*

***Ключові слова:*** *синхронність, асинхронність, паралельні обчислення, asyncio, threading, продуктивність.*

***Abstract***

*The development and research of synchronous and asynchronous data processing algorithms in parallel systems using the Python programming language is considered. Theoretical foundations of shared-memory architectures and event-driven models are considered, the choice of threading and asyncio libraries for software implementation is justified. In the work, a software complex was created that implements the processing of large data arrays by two methods, and a comparative analysis of their performance was carried out. Using the results will allow optimizing architectural solutions when designing high-load information processing systems.*

***Keywords:*** *synchronism, asynchronism, parallel computing, asyncio, threading, performance.*

## Вступ

Актуальність дослідження синхронного та асинхронного опрацювання даних полягає у стрімкому зростанні обсягів інформації та необхідності підвищення ефективності її обробки. Сучасні програмні системи вимагають від розробників вибору оптимальної моделі виконання коду: чи то детермінована синхронна модель, що гарантує послідовність, чи асинхронна, яка забезпечує масштабованість та неблокуюче введення-виведення [1-3].

Розуміння відмінностей у продуктивності цих підходів, особливо в контексті мови Python, є критично важливим для розробки серверних додатків, систем реального часу та наукових обчислень [2-4].

Метою роботи є розробка та порівняльний аналіз програмних реалізацій синхронного та асинхронного алгоритмів для визначення їх ефективності при обробці великих масивів даних.

## Постановка задачі дослідження

Задачі дослідження полягають у вирішенні наступних питань:

- аналіз архітектурних відмінностей між потоковою моделлю та моделлю подієвого циклу;
- розробка алгоритмів декомпозиції даних для рівномірного розподілу навантаження;
- програмна реалізація синхронного (threading) та асинхронного (asyncio) методів обробки даних;
- проведення експериментального тестування та оцінка накладних витрат ресурсів.

**Виклад основного матеріалу**

Ефективність сучасних обчислювальних систем значною мірою визначається обраною моделлю опрацювання даних. Існує два фундаментальні підходи до організації цього процесу: синхронний, що забезпечує сувору послідовність операцій, та асинхронний, орієнтований на події та неблокуюче виконання. Для задач, пов'язаних із обробкою великих масивів даних, вибір між цими підходами є нетривіальним і залежить від природи навантаження (CPU-bound або I/O-bound).

Для дослідження специфіки цих моделей у роботі використано мову програмування Python, яка підтримує обидві парадигми через стандартні бібліотеки. Було проаналізовано та програмно реалізовано два алгоритмічні підходи [4-6].

**1. Синхронна модель (багатопотокова).** Реалізація базується на бібліотеці threading, яка використовує нативні потоки операційної системи. У цій моделі всі потоки функціонують у спільному адресному просторі (Shared Memory). Ключовою особливістю реалізації є застосування механізмів синхронізації для уникнення станів гонитви (Race Conditions):

- **м'ютекси (Lock) - в**икористано для захисту критичних секцій коду, де відбувається запис часткових результатів у спільну структуру даних; це гарантує цілісність даних, але вводить затримки на очікування звільнення ресурсу;
- **бар'єри (Barrier) - з**астосовано для координації роботи потоків; головний процес очікує, доки всі N воркерів досягнуть точки завершення обчислень, що дозволяє коректно виміряти загальний час виконання; недоліком цього підходу в Python є наявність Global Interpreter Lock (GIL), який обмежує виконання байт-коду одним ядром процесора в кожен момент часу, а також накладні витрати операційної системи на перемикання контексту між потоками (context switching).

**2. Асинхронна модель (подієво-орієнтована).** Реалізована на базі бібліотеки asyncio та концепції кооперативної багатозадачності. Архітектура базується на патерні «Виробник–Споживач» (Producer-Consumer):

- **подієвий цикл (Event Loop) - в**иступає центральним планувальником, який перемикає виконання між задачами (корутинами) лише у моменти їх простою або явного очікування (await);
- **неблокувальні черги (Queue) -** використано для розподілу блоків даних між воркерами; це дозволяє організувати конвеєрну обробку без необхідності використання блокувань, оскільки код у межах корутини виконується послідовно; перевагою методу є відсутність витрат на системні перемикання та менше споживання пам'яті, оскільки корутини є значно «легшими» об'єктами, ніж потоки ОС.

Для експериментальної перевірки ефективності було розроблено програмний комплекс, який генерує масив випадкових чисел обсягом 100 000 елементів. Навантаження на процесор імітувалося виконанням арифметичної операції $f(x) = x^2 mod\ 17$ для кожного елемента. Тестування проводилося на 4-ядерній системі (табл. 1).

Таблиця 1 – Результати порівняльного аналізу продуктивності

| Метод реалізації | Час виконання, с | Накладні витрати | Механізм синхронізації |
|---|---|---|---|
| Синхронний (Threading) | | Високі (OS Context Switching) | |
| Асинхронний (Asyncio) | | Низькі (Event Loop Switching) | |

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

- асинхронний метод продемонстрував менший час виконання (0.03980 с проти 0.04215 с), що свідчить про ефективність кооперативної багатозадачності навіть для CPU-bound задач за рахунок відсутності витрат на перемикання контексту потоків;
- синхронний метод, хоча й забезпечує паралелізм, стикається з обмеженнями GIL та накладними витратами на синхронізацію доступу до ресурсів через м'ютекси.

Отже, основними чинниками, які впливають на ефективність опрацювання даних, є тип архітектури (блокуюча чи неблокуюча), накладні витрати на керування контекстом виконання та обсяг вхідних даних. Асинхронна модель демонструє кращу масштабованість та меншу ресурсоємність, що робить її перспективною для високонавантажених систем.

## Висновки

Досліджено синхронне та асинхронне опрацювання даних, як фундаментальні підходи до організації обчислень у сучасних програмних системах, визначено основні сфери їх застосування. Розглянуто теоретичні основи паралельних та розподілених обчислень, зокрема архітектури зі спільною пам'яттю та моделі, керовані подіями. На основі літературних джерел досліджено особливості реалізації багатопотоковості та асинхронності у мові Python, проаналізовано вплив Global Interpreter Lock (GIL) та специфіку роботи подієвого циклу (Event Loop) відносно розподілу обчислювального навантаження.

Програмно реалізовано задану задачу обробки великих масивів даних двома методами: синхронним (із використанням threading) та асинхронним (із використанням asyncio), та описано всі компоненти коду. Також наведено UML-діаграми класів, послідовності та діяльності. Проведено тестування розробленої програми, проаналізовано її результати, досліджено показники часу виконання та ефективності використання ресурсів.

Визначено, що асинхронна модель демонструє менший час виконання та вищу продуктивність порівняно із синхронною моделлю завдяки відсутності накладних витрат на перемикання контексту операційною системою, що підтверджує доцільність її використання у масштабованих системах.

## СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ


1. Семеренко В. П. Технології паралельних обчислень: навч. посіб. Вінниця: ВНТУ, 2018. 104 с. URL: https://www.researchgate.net/publication/334710599
2. Новотарський М. А. Алгоритми та методи обчислень. Київ: КПІ ім. Ігоря Сікорського, 2019. 407 с. URL: https://ela.kpi.ua/server/api/core/bitstreams/7421218e-d7dd-4e75-aa3e-bd7979db4e6d/content.
3. 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 .
4. Блонський Д. О., Денисюк В. О. Реалізація паралельного алгоритму обчислень засобами Python multiprocessing module. Молодь в науці: дослідження, проблеми, перспективи (МН-2025). Вінниця : ВНТУ, 2025. URL: https://conferences.vntu.edu.ua/index.php/mn/mn2025/paper/viewFile/23223/19223
5. Fowler M. Python Concurrency with asyncio. Manning Publications, 2022. 350 p.
6. Документація Python 3.12. Бібліотека threading – Потоковий паралелізм. URL: https://docs.python.org/3/library/threading.html



***Грепечук Вікторія Вадимівна*** – студентка кафедри комп`ютерних наук, факультет інтелектуальних інформаційних технологій та автоматизації, Вінницький національний технічний університет, м.Вінниця, e-mail: vikhrep5885@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.

***Hrepechuk Victoriia Vadimovna***– student of Computer Science Department, Faculty of Intelligent Information Technologies and Automation, Vinnytsia National Technical University, Vinnytsia, e-mail: vikhrep5885@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