https://doi.org/10.31891/2219-9365-2025-83-41
УДК 004.46

ІВАНЧУК Ярослав
Вінницький національний технічний університет
https://orcid.org/0000-0002-4775-6505
e-mail: ivanchuck@vntu.edu.ua
ЯКОВЧУК Павло
Вінницький національний технічний університет
https://orcid.org/0009-0007-2376-5072
e-mail: pyakovchuk@gmail.com

РОЗВИТОК ВИСОКОНАВАНТАЖЕНИХ ІНФОРМАЦІЙНИХ СИСТЕМ: ЕВОЛЮЦІЯ ТА ВИКЛИКИ

Ця стаття аналізує ключові етапи розвитку, технологічні підходи та сучасні виклики, що стоять перед архітекторами та розробниками. В статті виконано аналіз розвитку високонавантажених інформаційних систем, їх еволюція та виклики щодо основних характеристик. Представлені різні підходи щодо розвитку високонавантажених систем, паралельних обчислень. Результати дослідження свідчить про те, що найбільш перспективною є мікросервісна архітектура. Хоча для невеликих проєктів може бути використана і монолітна архітектура. Розподілена сервісна архітектура також може бути запроваджена для корпоративних мереж, спеціалізованих рішень. Перспективи розвитку високонавантажених інформаційних систем пов'язані з подальшим використанням хмарних технологій, безсерверних обчислень та інтеграцією штучного інтелекту для оптимізації роботи систем і передбачення навантаження. Але, розширення інтелектуальних функцій може привести до зниження швидкості та відмовостійкості. Розробка ефективних високонавантажених систем вимагає формування балансу моніторингу та розширення функцій з кількістю мікросервісів, швидкістю роботи з запитами системи. Кожна високонавантажена система має свої особливості відповідно до предметної області. В планах подальших досліджень – вивчення особливостей роботи високонавантаженої системи для брокерської біржі.

Ключові слова: інформаційні технології, архітектура, паралельні обчислення, високонавантажені системи, системи моніторингу.

IVANCHUK Yaroslav, YAKOVCHUK Pavlo
Vinnytsia National Technical University

DEVELOPMENT OF HIGH-LOAD INFORMATION SYSTEMS: EVOLUTION AND CHALLENGES

This article analyzes the key stages of development, technological approaches and modern challenges facing architects and developers. The article analyzes the development of high-load information systems, their evolution and challenges regarding the main characteristics. Various approaches to the development of high-load systems, parallel computing are presented. The results of the study indicate that the most promising is the microservice architecture. Although for small projects, a monolithic architecture can also be used. Distributed service architecture can also be implemented for corporate networks, specialized solutions. The prospects for the development of high-load information systems are associated with the further use of cloud technologies, serverless computing and the integration of artificial intelligence to optimize the operation of systems and predict the load. However, the expansion of intelligent functions can lead to a decrease in speed and fault tolerance. The development of effective high-load systems requires the formation of a balance of monitoring and expansion of functions with the number of microservices, the speed of working with system requests. Each high-load system has its own characteristics in accordance with the subject area.

The plans for further research include studying the features of the operation of a high-load system for a brokerage exchange. High-Load Information Systems (High-Load Information Systems) are the basis of modern digital infrastructure for complex organizational systems, ensuring the functioning of powerful services in the areas of e-commerce, social networks, online banking, the activities of various dynamic trading platforms, cloud computing. such as e-commerce, social networks, online banking and cloud computing. The evolution of these systems reflects the constant search for optimal solutions for processing growing volumes of data, managing a large number of simultaneous requests and ensuring uninterrupted availability. The relevance of the development of high-load systems requires developers of new systems with the possibility of multi-tier architecture, special modules for working with dynamic data.

The use of a specific architecture is closely related to the selected parallelism model. Parallelism models are ways of organizing calculations that allow you to perform several tasks simultaneously to increase productivity. They are closely related to computer system architectures, which define how hardware and software interact to implement parallel computing.

Promising models of parallelism and practice, and scientists consider those that allow for the efficient use of hardware resources, such as multi-core processors and specialized accelerators, to solve complex tasks. They differ from traditional approaches, focusing on increasing performance, scalability and flexibility.

Keywords: information technology, architecture, parallel computing, high-load systems, monitoring systems.

Стаття надійшла до редакції / Received 26.07.2025
Прийнята до друку / Accepted 19.08.2025

ПОСТАНОВКА ПРОБЛЕМИ У ЗАГАЛЬНОМУ ВИГЛЯДІ ТА ЇЇ ЗВ'ЯЗОК ІЗ ВАЖЛИВИМИ НАУКОВИМИ ЧИ ПРАКТИЧНИМИ ЗАВДАННЯМИ

Високонавантажені інформаційні системи (High-Load Information Systems) є основою сучасної цифрової інфраструктури для складних організаційних систем, забезпечуючи функціонування потужних сервісів в напрямах електронної комерції, соціальних мереж, онлайн-банкінгу, діяльності різноманітних динамічних торговельних майданчиків, хмарних обчислень. таких як електронна комерція, соціальні мережі, онлайн-банкінг та хмарні обчислення. Еволюція цих систем відображає постійний пошук оптимальних рішень для обробки зростаючих обсягів даних, управління великою кількістю одночасних запитів та забезпечення безперебійної доступності. Актуальність розвитку високонавантажених систем вимагає від розробників нових систем з можливістю багаторівневою архітектури, спеціальних модулів для роботи з динамічними даними.

АНАЛІЗ ДОСЛІДЖЕНЬ ТА ПУБЛІКАЦІЙ

Серед відомих публікацій на тему еволюції розвитку високонавантажених інформаційних систем розглянемо роботи таких авторів як Мартін Клеппманн [1], Майкл Найгаард [2], а також практичні рекомендації щодо розвитку високонавантажених систем. Клеппманн М. фокусує свою увагу на масштабованості, надійності, консистентності даних, співвідношенні різних технологій в єдиній системі. Розробка та розгортання програмного забезпечення розглядається в книгах Найгарда М., де пропонуються патерни для побудови надійних та відмовостійких систем. Важливо також враховувати особливості реалізації багатопоточності та багатопроцесорності, що представлено в роботах Н. Брукса [3]. Розвиток різноманітних архітектур для високонавантажених систем свідчить про те, що такі системи потребують спеціальних підходів для оптимізації ресурсів, використання синхронних та асинхронних методів обробки даних [4-8]. Для управління комунікаціями між сервісами, доцільно використовувати фреймворки (Spring Boot) [6]. Питання оптимізації розміщення компонентів для мінімізації часу відгуку є ключовою проблемою у великих розподілених системах та розглядається не тільки з технічної точки зору, а і враховуючи предметну область [9; 10].

ФОРМУЛЮВАННЯ ЦІЛЕЙ СТАТТІ

Метою роботи є аналіз розвитку високонавантажених інформаційних систем з визначенням основних трендів для сучасних розробок.

ВИКЛАД ОСНОВНОГО МАТЕРІАЛУ

Історично розвиток високонавантажених систем можна розділити на кілька ключових етапів.

Монолітна архітектура.

Багаторівнева архітектура.

Сервіс-орієнтована архітектура.

На початкових етапах більшість веб-застосунків будувалися за принципом монолітної архітектури. Це єдиний, нероздільний блок коду, який виконує всі функції системи. . Основними перевагами були простота розробки та розгортання. Однак, з підвищенням навантаження, такий підхід виявив свої недоліки:

- низька масштабованість: масштабування всієї системи вимагало розгортання повного копіювання моноліту, що було неефективним;
- складність обслуговування внесення змін у великий монолітний код було ризикованим і трудомістким процесом;
- gроблеми з надійністю - відмова одного компонента могла призвести до збою всієї системи.

Відповіддю на виклики монолітних систем стало застосування багаторівневої архітектури, де систему розділяли на логічні шари, наприклад:

- рівень представлення (front-end), що відповідає за взаємодію з користувачем;
- рівень бізнес-логіки (back-end), що обробляє основні операції;
- рівень даних, що забезпечує зберігання та доступ до даних.

Цей підхід дозволив масштабувати кожен рівень окремо, що значно підвищило гнучкість.

Подальша еволюція призвела до концепції сервіс-орієнтованої архітектури (SOA), а згодом – до мікросервісної архітектури. У цьому підході великий застосунок розділяється на набір дрібних, незалежних сервісів, кожен з яких виконує одну конкретну бізнес-функцію. В таблиці 1 представлена порівняльна характеристика різних архітектур.

Масштабованість свідчить про те, наскільки легко і ефективно можна збільшити потужність системи для обробки зростаючого навантаження.

Для швидких розробок найкраще підходять монолітна та мікросервісна архітектура. Швидкість розробки: Порівнює швидкість розробки на початкових етапах та при внесенні змін. Але, якщо розробка потребує внесення змін, високий рівень гнучкості, то мікросервісна архітектура має однозначну перевагу.

Відмовостійкість систем збільшувалась з розвитком архітектури і має найвищий рівень у мікросервісному варіанті. Але і складність розгортання є високою. Саме для цього були розроблені та впроваджуються спеціальні сервіси для розгортання та підтримки.

Для побудови сучасних високонавантажених систем застосовується широкий спектр технологій.

Таблиця 1

Порівняльна характеристика архітектур високонавантажених інформаційних систем

Характеристика	**МонолітSOA**	**Сервіс-орієнтована архітектура**	**Мікросервіси**
Особливості	Єдиний, нероздільний блок коду, що містить всю функціональність.	Набір великих, логічно згрупованих сервісів, які взаємодіють через централізовану шину (ESB).	Набір дрібних, незалежних сервісів, кожен з яких відповідає за одну бізнес-функцію.
Масштабованість	**Низька.** Масштабування вимагає дублювання всього додатка. Неможливо масштабувати окремі частини.	**Помірна.** Можливе масштабування окремих сервісів, але їхня залежність від шини (ESB) може створювати "вузькі місця".	**Висока.** Кожен сервіс можна масштабувати незалежно, що є дуже ефективним.
Швидкість розробки	**Висока.** На початковому етапі розробка простіша та швидша, завдяки єдиній кодовій базі.	**Помірна.** Розробка відбувається навколо визначених сервісів. Потребує багато часу на інтеграцію через ESB.	**Висока.** Команди працюють незалежно, що дозволяє швидко розробляти та тестувати окремі сервіси.
Гнучкість та незалежність	**Низька.** Всі компоненти тісно пов'язані. Зміна в одному місці може вплинути на весь додаток.	**Помірна.** Сервіси є відносно незалежними, але їхня взаємодія часто залежить від єдиного центрального компонента.	**Висока.** Сервіси повністю незалежні один від одного, що дозволяє використовувати різні технології для кожного сервісу.
Відмовостійкість	**Низька.** Відмова одного компонента може призвести до збою всієї системи.	**Помірна.** Збій одного сервісу не завжди призводить до краху всієї системи, але збій ESB може мати катастрофічні наслідки.	**Висока.** Відмова одного сервісу не впливає на інші. Система може продовжувати працювати з обмеженою функціональністю.
Складність розгортання	**Низька.** Простота розгортання одного додатка.	**Помірна.** Розгортання складніше через наявність шини та потреби в узгодженості сервісів	**Висока.** Розгортання кожного сервісу окремо вимагає складних інструментів автоматизації (наприклад, Kubernetes, Docker).
Приклади використання	Невеликі стартапи, внутрішні додатки.	Великі корпоративні системи, банківські системи, банківські додатки, урядові платформи	системи, банківські додатки, урядові Соціальні мережі (Facebook), потокові сервіси (Netflix), платформи електронної комерції (Amazon).платформи

Технології масштабування – горизонтальне та вертикальне. Горизонтальне масштабування (scale-out) використовується для збільшення пропускної здатності шляхом додавання нових вузлів (серверів) до системи.

Вертикальне масштабування (scale-up) використовується для збільшення ресурсів (CPU, RAM) на існуючому сервері.

Технології створення та використання баз даних – реляційні та нереляційні бази даних.

Реляційні бази даних (SQL), такі як PostgreSQL або MySQL, є ефективними для структурованих даних з високою консистентністю, але можуть мати обмеження при горизонтальному масштабуванні.

Нереляційні бази даних (NoSQL), такі як MongoDB, Cassandra, Redis, орієнтовані на високу продуктивність та масштабованість. Вони включають різні моделі: документо-орієнтовані, ключ-значення, графові тощо.

Технології кешування дозволяють використовувати такі системи як Redis або Memcached. Це дозволяє зберігати часто запитувані дані в оперативній пам'яті, значно знижуючи навантаження на основну базу даних.

Технології асинхронних обчислень та черги повідомлень використовують для обробки фонових завдань та розвантаження основних сервісів (наприклад, RabbitMQ, Apache Kafka). Це дозволяє системі швидко реагувати на запити, а трудомісткі операції виконувати асинхронно.

Сучасні високонавантажені системи стикаються з новими викликами підтримки консистентності даних між різними сервісами стає складним завданням.

Такі системи потребують постійного моніторингу показників і підтримки відмовостійкості. Саме для цього використовують комплексні системи моніторингу (Prometheus, Grafana) та механізми автоматичного відновлення (Kubernetes).

Розподілена природа мікросервісів створює нові вектори загроз, вимагаючи посилених заходів безпеки на всіх рівнях.

Використання визначеної архітектури тісно пов’язано з вибраною моделлю паралелілізму. Моделі паралелізму – це способи організації обчислень, що дозволяють виконувати кілька завдань одночасно для підвищення продуктивності. Вони тісно пов'язані з архітектурами комп'ютерних систем, які визначають, як апаратне та програмне забезпечення взаємодіють для реалізації паралельних обчислень.

Виділяють три основні моделі – даних, завдань, конвеєрний.

Паралелізм даних (Data Parallelism) – ця модель фокусується на розподілі даних між кількома процесорами, де кожен процесор виконує ідентичну операцію над своєю частиною даних. . Це ідеально підходить для завдань, які вимагають масової обробки великих обсягів незалежних даних, наприклад, в наукових обчисленнях або обробці зображень.

Паралелізм завдань (Task Parallelism) – у цій моделі різні завдання розподіляються між різними процесорами. Кожен процесор виконує унікальне завдання. Цей підхід ефективний, коли завдання не залежать одне від одного. Наприклад, в багатопроцесорних системах один процесор може обробляти запити, а інший – взаємодіяти з базою даних.

Конвеєрний паралелізм (Pipeline Parallelism) – ця модель розділяє одне велике завдання на послідовні етапи, де результат одного етапу стає вхідними даними для наступного. Кожен етап виконується окремим процесором або потоком. Це дозволяє досягти високої пропускної здатності, оскільки всі етапи можуть працювати одночасно.

За моделлю Фліна потоки інструкцій і даних поділяються на однопроцесорну однопотокову; однокомандну багатопотокову, багатокомандну, однопотоковуі багатокомандну багатопотокову архітектури. Крім того існують асихронна, реактивна та безсерверна моделі [11].

Перспективними моделями паралелізму і практики, і науковці вважають ті, що дозволяють ефективно використовувати апаратні ресурси, такі як багатоядерні процесори та спеціалізовані акселератори, для вирішення складних завдань. Вони відрізняються від традиційних підходів, фокусуючись на підвищенні продуктивності, масштабованості та гнучкості.

ВИСНОВКИ З ДАНОГО ДОСЛІДЖЕННЯ І ПЕРСПЕКТИВИ ПОДАЛЬШИХ РОЗВІДОК У ДАНОМУ НАПРЯМІ

Перспективи розвитку високонавантажених інформаційних систем пов'язані з подальшим використанням хмарних технологій, безсерверних обчислень та інтеграцією штучного інтелекту для оптимізації роботи систем і передбачення навантаження. Але, розширення інтелектуальних функцій може привести до зниження швидкості та відмовостійкості. Розробка ефективних високонавантажених систем вимагає формування балансу моніторингу та розширення функцій з кількістю мікросервісів, швидкістю роботи з запитами системи. Кожна високонавантажена система має свої особливості відповідно до предметної області.

В планах подальших досліджень – вивчення особливостей роботи високонавантаженої системи для брокерської біржі.

Література

1. Klippmann Martin Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems. O’Reilly.2017.714 p.
2. Nygard Michael T. Release It!: Design and Deploy Production-Ready Software 2nd Edition, O’Reilly.2018.378 p.
3. Implementing Microservicilites with Istio [Електронний ресурс] // InfoQ. – 2021. – Режим доступу до ресурсу: https://www.infoq.com/articles/microservicilitiesistio/?topicPageSponsorship=3a992d7f-f0cc-4b4a-ae93-f8ca4031520b.
4. Raible M. Java Microservices with Spring Boot and Spring Cloud [Електронний ресурс] / Matt Raible // Auth0. – 2023. – Режим доступу до ресурсу: https://auth0.com/blog/java-spring-boot-microservices/.
5. Брукс Н. Багатопотоковість проти багатопроцесорності – різниця між ними [Електронний ресурс] / Натаніель Брукс // Guru99. – 2024. – Режим доступу до ресурсу: https://www.guru99.com/uk/difference-betweenmultiprocessing-and-multithreading.html
6. Goetz B. Virtual Threads: New Foundations for High-Scale Java Applications [Електронний ресурс] / B. Goetz, D. Bryant // InfoQ. – 2022. – Режим доступу до ресурсу: https://www.infoq.com/articles/java-virtualthreads/?topicPageSponsorship=cc235d0e-fc59-4e62-b242-d9b5d7b55939
7. Naveen M. Mastering Advanced Java Concepts: Multithreading, Generics, Reflection and Beyond [Електронний ресурс] / Metta Naveen // Medium. – 2023. – Режим доступу до ресурсу: https://naveen-

metta.medium.com/masteringadvanced-java-concepts-multithreading-generics-reflection-and-beyondd7dc442e31d9.

8. Engstrand G. Microservice Threading Models and their Tradeoffs [Електронний ресурс] / Glenn Engstrand // InfoQ. – 2019. – Режим доступу до ресурсу: https://www.infoq.com/articles/engstrand-microservice-threading/.

9. Empowering Microservices: A Deep Dive into Intelligent Application Component Placement for Optimal Response Time [Електронний ресурс] // Springer – 2024. – Режим доступу до ресурсу: https://link.springer.com/article/10.1007/s10922-024-09855-3

10. Trung Luu Q. Difference Between Parallel and Distributed Computing [Електронний ресурс] / Q. Trung Luu, M. Aibin // Baeldung. – 2024. – Режим доступу до ресурсу: https://www.baeldung.com/cs/parallel-vs-distributedcomputing.

11. Flynns-taxonomy-classification-parallel-systems . – Режим доступу до ресурсу: https://fiveable.me/parallel-and-distributed-computing/unit-2/flynns-taxonomy-classification-parallel-systems/study-guide/Ohzf44x4HCtFZRjK

References

1. Klippmann Martin Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems. O'Reilly.2017.714 p.

2. Nygard Michael T. Release It!: Design and Deploy Production-Ready Software 2nd Edition, O'Reilly.2018.378 p.

3. Implementing Microservicilites with Istio [Електронний ресурс] // InfoQ. – 2021. – Режим доступу до ресурсу: https://www.infoq.com/articles/microservicilitiesistio/?topicPageSponsorship=3a992d7f-f0cc-4b4a-ae93-f8ca4031520b.

4. Raible M. Java Microservices with Spring Boot and Spring Cloud / Matt Raible // Auth0. – 2023. https://auth0.com/blog/java-spring-boot-microservices/.

5. Bruks N. Bahatopotokovist′ proty bahatoprotsesornosti – riznytsya mizh nymy [Elektronnyy resurs] / Nataniel′ Bruks

6. Goetz B. Virtual Threads: New Foundations for High-Scale Java Applications [Електронний ресурс] / B. Goetz, D. Bryant // InfoQ. https://www.infoq.com/articles/java-virtualthreads/?topicPageSponsorship=cc235d0e-fc59-4e62-b242-d9b5d7b55939.

7. Naveen M. Mastering Advanced Java Concepts: Multithreading, Generics, Reflection and Beyond [Електронний ресурс] / Metta Naveen // Medium. – 2023. https://naveen-metta.medium.com/masteringadvanced-java-concepts-multithreading-generics-reflection-and-beyondd7dc442e31d9.

8. Engstrand G. Microservice Threading Models and their Tradeoffs [Електронний ресурс] / Glenn Engstrand // InfoQ. – 2019. https://www.infoq.com/articles/engstrand-microservice-threading/.

9. Empowering Microservices: A Deep Dive into Intelligent Application Component Placement for Optimal Response Time [Електронний ресурс] // Springer – 2024. https://link.springer.com/article/10.1007/s10922-024-09855-3.

10. Trung Luu Q. Difference Between Parallel and Distributed Computing [Електронний ресурс] / Q. Trung Luu, M. Aibin // Baeldung. – 2024. https://www.baeldung.com/cs/parallel-vs-distributedcomputing.

11. Flynns-taxonomy-classification-parallel-systems .https://fiveable.me/parallel-and-distributed-computing/unit-2/flynns-taxonomy-classification-parallel-systems/study-guide/Ohzf44x4HCtFZRjK