DOI 10.34132/mspc2025.01.14.21

Євгеній Стоєв
Віктор Раленко

АРХІТЕКТУРНІ ПРИНЦИПИ ПОБУДОВИ ВІДМОВОСТІЙКИХ І ВИСОКОПРОДУКТИВНИХ ОБЧИСЛЮВАЛЬНИХ СИСТЕМ РЕАЛЬНОГО ЧАСУ: ВПЛИВ ПАТЕРНІВ ПРОЄКТУВАННЯ НА ФУНКЦІОНАЛЬНІ ХАРАКТЕРИСТИКИ

У дослідженні розглядаються архітектурні принципи побудови відмовостійких та високопродуктивних обчислювальних систем реального часу. Основна увага приділяється використанню патернів проєктування, таких як Circuit Breaker, CQRS та Saga Pattern, що сприяють підвищенню стабільності, продуктивності та узгодженості даних у розподілених системах. Оптимізація продуктивності досягається шляхом асинхронної обробки даних, подієво-орієнтованих підходів та використання платформ, таких як Apache Kafka і Spark Streaming. Додатково розглядається важливість автоматичного масштабування та ефективного кешування для забезпечення стабільної роботи високонавантажених систем.

Ключові слова: *Відмовостійкі системи, високопродуктивні обчислювальні системи, патерни проєктування, Circuit Breaker, CQRS, Saga Pattern, стабільність, продуктивність, узгодженість даних, асинхронна обробка даних, подієво-орієнтовані підходи, автоматичне масштабування, балансування навантаження, горизонтальне масштабування, доступність.*

У сучасному світі технологічний розвиток зумовлює необхідність створення обчислювальних систем, здатних працювати в режимі реального часу, гарантуючи при цьому високу продуктивність та відмовостійкість. Такі системи критично важливі для фінансових операцій, телекомунікацій, автоматизованого виробництва, транспорту та інших галузей, де швидкість обробки даних безпосередньо впливає на ефективність бізнес-процесів.

Архітектурні принципи проєктування відіграють ключову роль у формуванні надійних та масштабованих інфраструктур для обробки потоків даних реального часу. Одним із найважливіших аспектів є використання патернів проєктування, які допомагають структурувати систему таким чином, щоб вона могла ефективно обробляти запити, управляти транзакціями та гарантувати узгодженість даних.

Патерни проєктування в системах реального часу виконують низку функцій:

– Забезпечують відмовостійкість шляхом використання механізмів автоматичного відновлення та балансування навантаження (*Circuit Breaker*, *Failover Strategies*);

– підвищують продуктивність за допомогою структурованого розділення операцій читання та запису (*CQRS*), а також подієво-орієнтованого підходу (*Event-Driven Architecture*);

– оптимізують управління транзакціями в розподілених системах за допомогою *Saga Pattern*, що дозволяє гарантувати коректне виконання складних бізнес-операцій;

– спрощують обробку потокових даних через використання *Data Streaming Patterns* та спеціалізованих рішень на базі *Kafka*, *Flink* або *Spark Streaming*.

Таким чином, архітектурні патерни не лише підвищують стабільність роботи обчислювальних систем реального часу, але й дозволяють ефективно масштабувати їх, забезпечуючи максимальну продуктивність навіть при значних навантаженнях.

Обчислювальні системи реального часу мають низку критичних вимог, що визначають їхню ефективність та стабільність у високонавантажених середовищах. Однією з найважливіших характеристик є низька латентність, оскільки такі системи повинні оперативно реагувати на події та обробляти великі обсяги даних без затримок. У багатьох випадках необхідно дотримуватися жорстких часових меж виконання операцій, що особливо актуально для фінансових платформ, промислових автоматизованих процесів або медичних систем моніторингу.

Висока доступність є ще одним ключовим фактором, оскільки критично важливі сервіси повинні працювати безперервно, навіть за умов часткових збоїв або відмови вузлів. Для цього застосовуються механізми резервного копіювання, автоматичного переключення на резервні сервери та шаблони проєктування, такі як *Circuit Breaker*, які запобігають каскадним відмовам.

Масштабованість також відіграє важливу роль, оскільки системи реального часу повинні адаптуватися до динамічних змін навантаження. Горизонтальне масштабування дозволяє ефективно розподіляти ресурси та підтримувати обробку потоків даних на великих обсягах, що особливо важливо для телекомунікаційних платформ або фінансових ринків.

Узгодженість даних у розподілених обчислювальних середовищах потребує вибору правильного балансу між *Strong Consistency* та *Eventual Consistency*. Для складних бізнес-операцій використовується *Saga Pattern*, який забезпечує коректне управління довготривалими транзакціями у розподілених системах.

Крім того, захист і безпека інформації є важливими вимогами, оскільки системи реального часу обробляють конфіденційні та стратегічно важливі дані. Використання шифрування, механізмів автентифікації та інструментів для моніторингу активності допомагає запобігати загрозам та гарантує цілісність інформації.

Забезпечення всіх цих аспектів дозволяє системам реального часу ефективно працювати в умовах підвищеного навантаження, мінімізуючи ризики збоїв і забезпечуючи стабільну роботу в критично важливих галузях.

Продуктивність обчислювальних систем реального часу визначається їхньою здатністю обробляти великі обсяги даних з мінімальною затримкою. В умовах високонавантажених середовищ критично важливо досягти оптимального балансу між швидкістю виконання запитів, використанням ресурсів та узгодженістю даних.

Одним із ключових методів оптимізації є асинхронна обробка даних, яка дозволяє мінімізувати блокуючі виклики та збільшити загальну швидкодію системи. Застосування подієво-орієнтованої архітектури (*Event-Driven Architecture*, *EDA*) дозволяє реагувати на зміни у даних без необхідності чекати завершення всіх процесів, що значно пришвидшує роботу у критичних сценаріях.

Розділення операцій читання та запису за допомогою *CQRS* (*Command Query Responsibility Segregation*) також сприяє оптимізації продуктивності. Цей підхід дає змогу розділити бізнес-логіку на незалежні компоненти, що дозволяє масштабувати запити та підвищити ефективність обробки даних. *CQRS* широко використовується у фінансових системах та інших середовищах, де важливо миттєво отримувати результати запитів без затримок у транзакціях.

Для підвищення продуктивності також застосовується обробка потокових даних (*Data Streaming Patterns*). Використання платформ на кшталт *Apache Kafka*, *Flink* або *Spark Streaming* дозволяє працювати з постійними потоками інформації, що надходять від датчиків, користувачів або зовнішніх API, забезпечуючи швидку реакцію та адаптивність до змін.

Ще одним важливим фактором є автоматичне масштабування, яке дозволяє розподіляти навантаження між серверами залежно від поточної активності. Це досягається за допомогою оркестрації контейнерів у *Kubernetes*, який автоматично додає чи видаляє ресурси відповідно до запитів у режимі реального часу.

Крім того, оптимізація продуктивності включає ефективне кешування та застосування технологій попередньої обробки запитів. Використання *Redis* або *Memcached* допомагає значно знизити навантаження на основні бази даних та забезпечити швидкий доступ до критичних даних.

Побудова відмовостійких та високопродуктивних обчислювальних систем реального часу вимагає комплексного підходу, що включає сучасні архітектурні патерни, ефективне управління ресурсами та оптимізацію продуктивності. Таким чином, ефективна архітектура систем реального часу базується на принципах динамічного управління даними, відмовостійких механізмах та адаптивних підходах до продуктивності. Використання перевірених патернів проєктування дає змогу побудувати стабільну, масштабовану та ефективну інфраструктуру для критично важливих бізнес-процесів.

References

1. Top 10 integration patterns for enterprise use cases. MuleSoft URL: https://blogs.mulesoft.com/api-integration/patterns/top-10-integration-patterns/ (date accessed: 28.04.2025)
2. Understanding Enterprise Integration Patterns. HALSIMPLIFY. URL: https://www.halsimplify.com/knowledge-center/enterprise-integration-patterns-guide (date accessed: 28.04.2025)
3. A comprehensive guide to integration patterns in modern business systems. LONTI. URL: https://www.lonti.com/blog/a-comprehensive-guide-to-integration-patterns-in-modern-business-systems (date accessed: 30.04.2025)

DOI 10.34132/mspc2025.01.14.21

Yevhenii Stoiev
Viktor Ralenko,

ARCHITECTURAL PRINCIPLES FOR BUILDING FAULT-TOLERANT AND HIGH-PERFORMANCE REAL-TIME COMPUTING SYSTEMS: THE IMPACT OF DESIGN PATTERNS ON FUNCTIONAL CHARACTERISTICS

This study examines the architectural principles for building fault-tolerant and high-performance real-time computing systems. The focus is on the use of design patterns such as Circuit Breaker, CQRS, and Saga Pattern, which contribute to the improvement of stability, performance, and data consistency in distributed systems. Performance optimization is achieved through asynchronous data processing, event-driven approaches, and the use

of platforms such as Apache Kafka and Spark Streaming. Additionally, the importance of auto-scaling and effective caching is considered to ensure the stable operation of high-load systems.

Keywords: *Fault-tolerant systems, high-performance computing systems, design patterns, Circuit Breaker, CQRS, Saga Pattern, stability, performance, data consistency, asynchronous data processing, event-driven approaches, auto-scaling, load balancing, horizontal scaling, availability.*

Відомості про авторів / Information about the Authors

Євгеній Стоєв, викладач кафедри інженерії програмного забезпечення, Чорноморського національного університету імені Петра Могили, м. Миколаїв, Україна. E-mail: stoiev.yevhenii@chmnu.edu.ua, orcid: https://orcid.org/0009-0008-8999-2267.

Yevhenii Stoiev, Lecturer at the Department of Software Engineering, Petro Mohyla Black Sea National University, Mykolaiv, Ukraine. E-mail: stoiev.yevhenii@chmnu.edu.ua, orcid: https://orcid.org/0009-0008-8999-2267.

Віктор Раленко, викладач кафедри інженерії програмного забезпечення, Чорноморського національного університету імені Петра Могили, м. Миколаїв, Україна. E-mail: ralenko.v@chmnu.edu.ua, orcid: https://orcid.org/0009-0009-4161-8468.

Viktor Ralenko, Lecturer at the Department of Software Engineering, Petro Mohyla Black Sea National University, Mykolaiv, Ukraine. E-mail: ralenko.v@chmnu.edu.ua, orcid: https://orcid.org/0009-0009-4161-8468.

Стаття отримана редакцією 19.05.2025.