УДК 004.7:004.42:004.8

С. М. Захарченко
Ю. О. Сеник

ПРОБЛЕМИ МАСШТАБУВАННЯ МОНОЛІТНИХ СИСТЕМ ЦІНОУТВОРЕННЯ В E-COMMERCE ТА ЇХ ВИРІШЕННЯ ЧЕРЕЗ МІКРОСЕРВІСНУ АРХІТЕКТУРУ

Вінницький національний технічний університет

Анотація
В роботі було проведено дослідження основних проблем масштабування монолітних систем динамічного ціноутворення для e-commerce платформ. Проаналізовано архітектурні обмеження традиційних підходів та запропоновано рішення на основі мікросервісної архітектури. Розглянуто аспекти декомпозиції системи для підвищення продуктивності.
Ключові слова: *мікросервіси, ціноутворення, e-commerce, масштабування, машинне навчання, архітектура.*

Abstract
The paper investigates the main problems of scaling monolithic dynamic pricing systems for e-commerce platforms. It analyzes the architectural limitations of traditional approaches and proposes a solution based on microservice architecture. It considers aspects of system decomposition to improve performance.
Keywords*: microservices, pricing, e-commerce, scaling, machine learning, architecture.*

Вступ

Ринок електронної комерції характеризується надзвичайно високою конкуренцією та динамічними змінами в поведінці споживачів що у таких умовах навіть незначна затримка в реакції на зміну ринкових умов може призвести до значних фінансових втрат [1]. Особливо критичним це стає для систем ціноутворення, де рішення мають прийматися в режимі реального часу.

Проблема полягає в тому, що більшість e-commerce платформ все ще використовує монолітні архітектури, які були ефективними на початкових етапах розвитку галузі, але тепер стають серйозним обмеженням. Такі системи демонструють неспроможність швидко адаптуватися до зростаючих вимог щодо обробки великих обсягів даних та інтеграції складних алгоритмів машинного навчання.

Сучасні e-commerce платформи стикаються з необхідністю одночасної обробки мільйонів товарних позицій, інтеграції з численними зовнішніми сервісами та персоналізації цінових пропозицій [1]. Монолітні архітектури створюють значні труднощі для реалізації такої функціональності через тісне зв'язування компонентів та централізовану обробку даних.

Проблеми утворюються при впровадженні алгоритмів машинного навчання, які потребують постійного експериментування, A/B тестування та швидкого розгортання нових моделей [2]. Монолітна архітектура не забезпечує необхідної гнучкості для таких операцій, оскільки будь-які зміни вимагають перезапуску всієї системи.

І додатковою проблемою є експоненційне зростання складності підтримки монолітних систем. Збільшення кодової бази призводить до ситуації, коли навіть незначні модифікації можуть мати непередбачувані наслідки для інших компонентів системи. У контексті динамічного ціноутворення такі ризики є неприйнятними.

Основна частина

Провівши аналіз існуючих рішень для динамічного ціноутворення в e-commerce показало, що значна частина платформ все ще використовує застарілі архітектурні підходи. Навіть великі гравці ринку часто покладаються на системи, розроблені 10-15 років тому, що створює серйозні обмеження для впровадження сучасних технологій та методів обробки даних. Основні проблеми централізованих систем ціноутворення:

Технічні обмеження:

- Неможливість незалежного масштабування окремих компонентів системи
- Складність інтеграції нових алгоритмів машинного навчання без зупинки всієї системи
- Обмежена можливість горизонтального масштабування при зростанні навантаження
- Високий ризик каскадних відмов при збоях в одному з компонентів

Операційні виклики:

- Довгі цикли розробки та тестування через взаємозалежність компонентів
- Складність паралельної роботи різних команд розробників
- Проблеми з відновленням після збоїв та забезпеченням високої доступності
- Неефективне використання ресурсів через неможливість точкового масштабування

Щоб вирішити зазначені проблеми було запропоновано архітектурне рішення на основі мікросервісного підходу. Ключовою ідеєю є декомпозиція монолітної системи на незалежні сервіси, кожен з яких відповідає за конкретну бізнес-функцію. Порівняння архітектурних підходів наведено на рисунку 1.

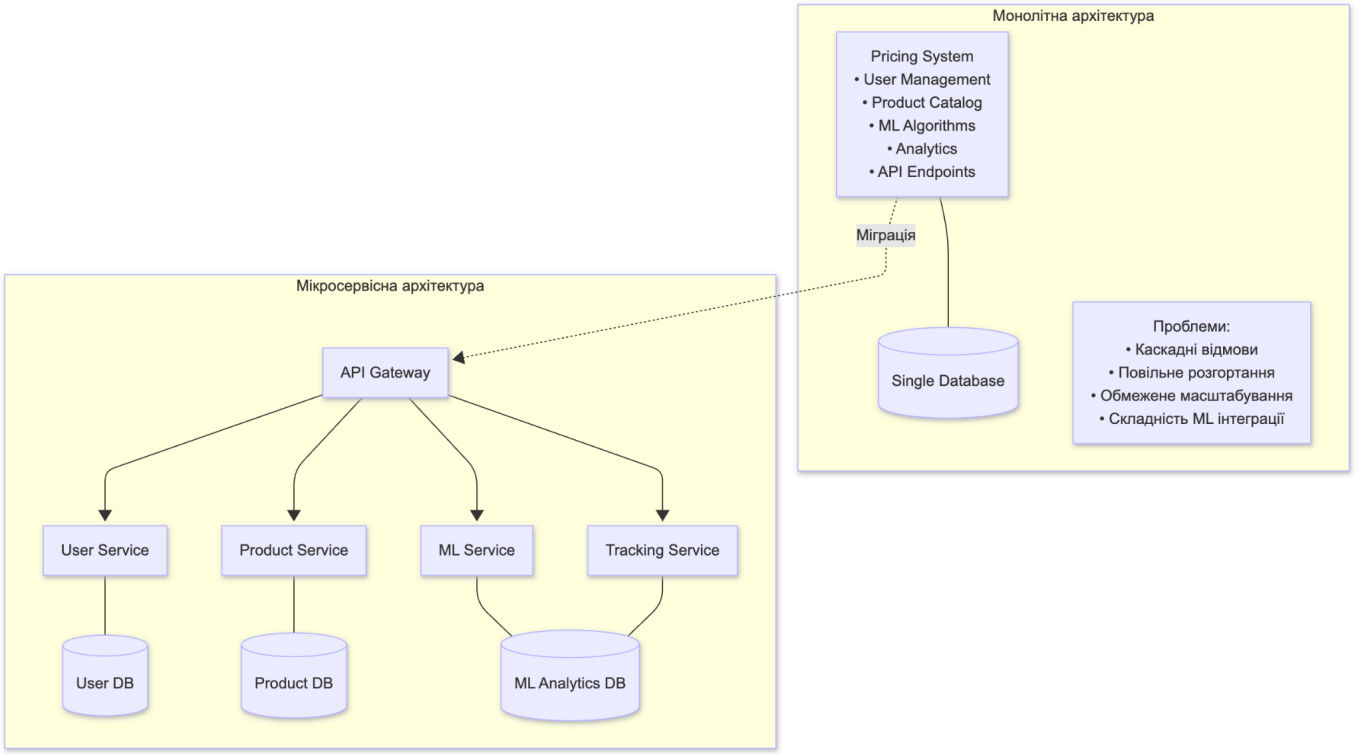

Рисунок 1 — Порівняння монолітної та мікросервісної архітектур

Архітектурні принципи декомпозиції системи

Одним з найбільш ефективних підходів до декомпозиції є аналіз потоків даних та бізнес-логіки системи. У доповіді “From Monolith to Microservices: A Dataflow-Driven Approach” [4] описано що демпозиця відіграє вирішальну роль у розробці dataflow-driven алгоритму, який дозволяє виділити раціональні та об'єктивні кандидати на мікросервіси через аналіз операцій та їх вихідних даних. Цей підхід особливо ефективний для систем ціноутворення, де потоки даних чітко визначені. Ключовими архітектурними принципами при цьому є:

Принцип єдиної відповідальності (Single Responsibility Principle) - кожен мікросервіс відповідає за одну бізнес-функцію. У системах ціноутворення це означає виділення окремих сервісів для управління користувачами, каталогу товарів, ML-алгоритмів та аналітики.

Принцип автономності сервісів - мікросервіси повинні мати власні бази даних та бути здатними функціонувати незалежно. Це забезпечує відмовостійкість та можливість незалежного розгортання.

Принцип слабкого зв'язування (Loose Coupling) - взаємодія між сервісами здійснюється через добре визначені API, що мінімізує залежності та спрощує еволюцію системи.

Knoche та Hasselbring [5] підкреслюють важливість архітектурного підходу "Strangler Fig Pattern" при модернізації застарілих систем. Цей патерн дозволяє поступово замінювати функціональність монолітної системи новими мікросервісами без зупинки роботи. Детальний план міграції представлено на рисунку 2.

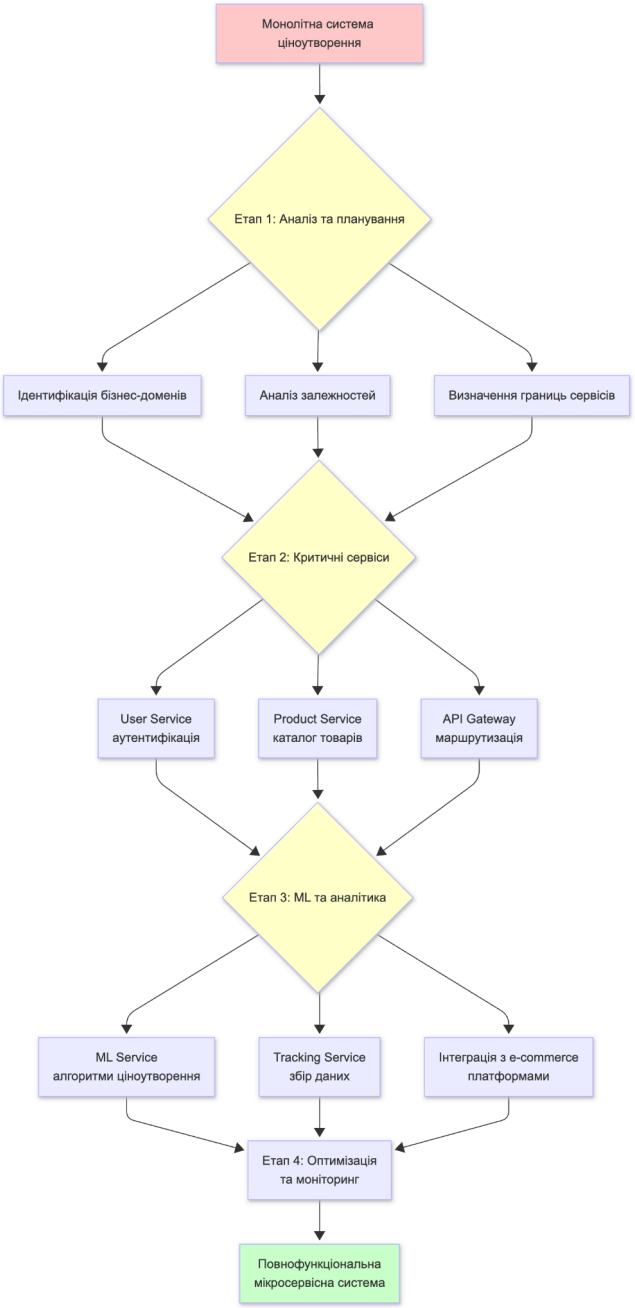

Рисунок 2 — Поетапна міграція до мікросервісної архітектури

Архітектурні патерни для систем ціноутворення

Database per Service Pattern [2] - кожен мікросервіс має власну базу даних, оптимізовану під його потреби. Для системи ціноутворення це означає використання PostgreSQL для транзакційних даних, TimescaleDB для часових рядів цін та Redis для кешування, що відповідає принципам архітектурних патернів для Python додатків [6].

API Gateway Pattern - централізована точка входу, що забезпечує маршрутизацію запитів, аутентифікацію, rate limiting та моніторинг. Критично важливо для систем ціноутворення через необхідність контролю доступу до цінової інформації.

Event Sourcing Pattern - зберігання історії всіх змін цін як послідовності подій. Це забезпечує повну аудованість та можливість відтворення стану системи на будь-який момент часу.

Порівняльний аналіз архітектурних підходів

Характеристика	Монолітна архітектура	Мікросервісна архітектура	Покращення
Масштабування	Вертикальне, всієї системи	Горизонтальне, окремих сервісів	Значне зниження витрат [1]
Час розгортання	Довготривалий процес	Швидке розгортання сервісів	Суттєве прискорення [2]
Відновлення після збоїв	Повна зупинка системи	Ізольовані збої	Підвищення надійності [1]
Розробка нових функцій	Послідовна, з блокуваннями	Паралельна, незалежна	Покращення продуктивності команд [6]

Архітектурні альтернативи для масштабування систем ціноутворення

Аналізуючи існуючі архітектурні підходи до вирішення проблем масштабування показують еволюцію від централізованих до розподілених рішень:

Вертикальне масштабування (Scale-Up Architecture) - збільшення потужності серверного обладнання з використанням shared-memory архітектури. Архітектурні обмеження включають memory bottleneck, single point of failure та експоненційне зростання вартості при досягненні апаратних лімітів.

Горизонтальне масштабування монолітів (Scale-Out Monolith) - реплікація всієї системи з використанням load balancer та shared database архітектури. Архітектурні проблеми: database становиться вузьким місцем, складність синхронізації стану між інстансами, неефективне використання ресурсів через реплікацію невикористаних компонентів.

Сервіс-орієнтована архітектура (SOA) - архітектурний стиль з Enterprise Service Bus (ESB) як центральним компонентом. Використовує SOAP/XML протоколи та централізовану оркестрацію сервісів. Архітектурні недоліки: ESB як single point of failure, складність управління версіями контрактів, overhead від XML processing.

Мікросервісна архітектура - повністю розподілена архітектура з децентралізованим управлінням даними та хореографією замість оркестрації. Використовує lightweight протоколи (HTTP/REST, gRPC), асинхронну комунікацію через message brokers та архітектурний принцип "smart endpoints, dumb pipes". Архітектурні виклики: distributed transactions, eventual consistency, network latency та складність debugging розподілених систем.

Кожен з підходів має свої переваги та недоліки, проте для систем динамічного ціноутворення з високими вимогами до швидкості реакції та необхідністю інтеграції ML-алгоритмів найбільш перспективним є мікросервісний підхід.

Висновки

Поточне дослідження демонструє, що наразі традиційні системи ціноутворення не здатні ефективно справлятися з вимогами до масштабованості та продуктивності. Особливо це проявляється при зростанні навантаження на e-commerce платформи, коли система починає демонструвати критичні проблеми з продуктивністю.

Важливо розуміти, що мікросервісна архітектура не є універсальним рішенням всіх проблем. Вона вимагає ретельного планування, глибокого розуміння бізнес-процесів та поетапного впровадження. Однак переваги у такого підходу є очевидними.

Головною перевагою мікросервісної архітектури є можливість незалежного масштабування окремих компонентів системи, що означає, що ресурсоємні ML-алгоритми можуть масштабуватися незалежно від стабільно працюючих сервісів управління користувачами, що забезпечує оптимальне використання обчислювальних ресурсів.

У системі динамічного ціноутворення це особливо критично, оскільки навіть короткочасні збої

можуть призвести до значних фінансових втрат і до прикладу досвід впровадження мікросервісної архітектури підтверджує ефективність поетапного підходу до міграції. Кількісні показники покращення представлені на рисунку 3.

Але і мікросервісна архітектура створює нові технічні виклики, які потребують спеціальних підходів до вирішення, до прикладу управління розподіленими транзакціями вимагає використання складних патернів, таких як Saga або Two-Phase Commit, що значно ускладнює архітектуру системи.

Основною проблемою є забезпечення консистентності даних у розподіленому середовищі що на відміну від монолітних систем, де транзакційність забезпечується автоматично, мікросервіси вимагають прийняття концепції eventual consistency. Для систем ціноутворення, де точність даних є критично важливою, це створює додаткові архітектурні складності.

Перспективні архітектурні напрямки

Подальший розвиток архітектури систем ціноутворення спрямований на впровадження гібридних підходів, що поєднують переваги різних архітектурних стилів. Зокрема, перспективною є інтеграція serverless архітектури для обробки пікових навантажень, що дозволяє автоматично масштабувати систему під час періодів високої активності без постійних витрат на ресурси.

Важливою тенденцією є розвиток service mesh архітектур, які забезпечують централізоване управління міжсервісною комунікацією, моніторингом та безпекою. Використання event-driven архітектури з Apache Kafka для real-time обробки цінових подій відкриває нові можливості для створення високореактивних систем ціноутворення.

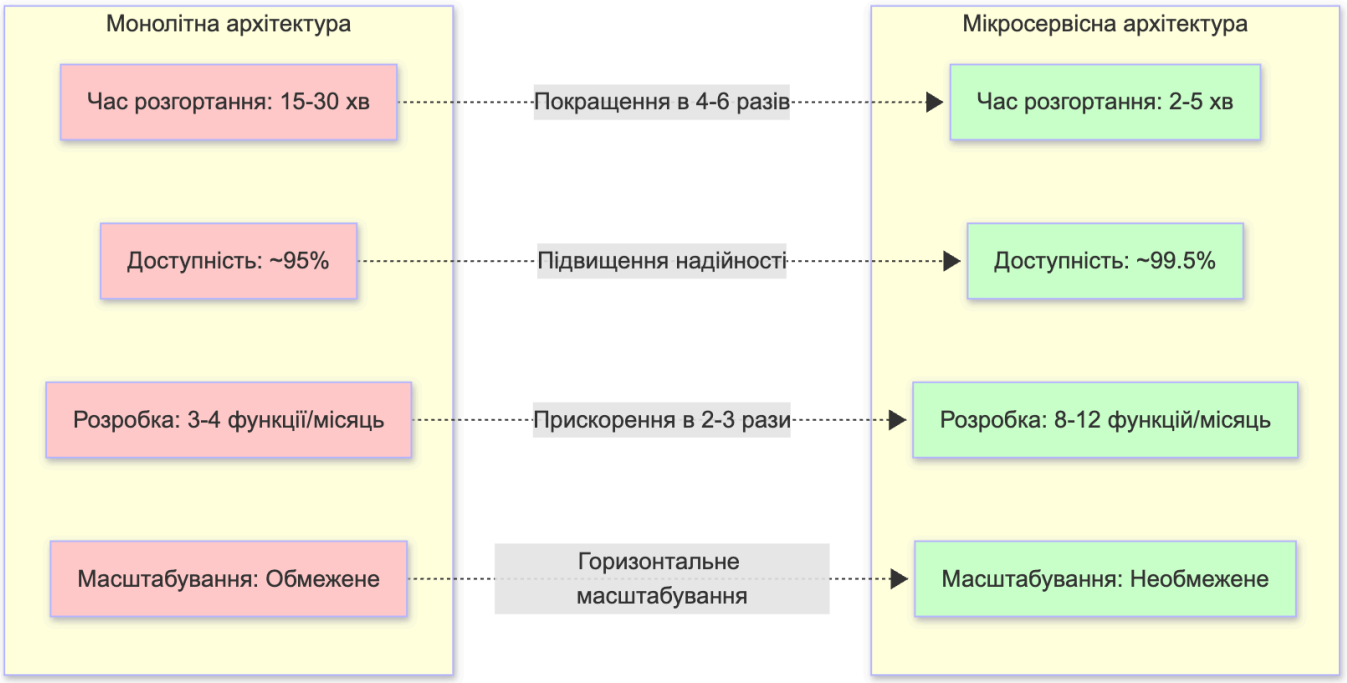

Рисунок 3 — Порівняльний аналіз ключових метрик продуктивності

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Newman S. Building Microservices: Designing Fine-Grained Systems. 2nd Edition. O'Reilly Media, 2021. 616 p.
2. Taibi D., Lenarduzzi V., Pahl C. Architectural patterns for microservices: a systematic mapping study. Information and Software Technology. 2020. Vol. 118. P. 106221.
3. Bucchiarone A., Dragoni N., Dustdar S., Larsen S.T., Mazzara M. From monolithic to microservices: An experience report from the banking domain. IEEE Access. 2022. Vol. 10. P. 25612-25628.
4. Chen R., Li S., Li Z. From Monolith to Microservices: A Dataflow-Driven Approach. IEEE Transactions on Services Computing. 2021. Vol. 14. No. 6. P. 1466-1479.
5. Knoche H., Hasselbring W. Using Microservices for Legacy Software Modernization. IEEE Software. 2020. Vol. 37. No. 3. P. 44-49.
6. Percival H., Gregory B. Architecture Patterns with Python: Enabling Test-Driven Development, Domain-Driven Design, and Event-Driven Microservices. O'Reilly Media, 2020. 304 p.

Захарченко Сергій Михайлович — к.т.н., професор, Вінницький національний технічний університет, Вінниця, e-mail: zakharchenko.sergii@vntu.edu.ua

Сеник Юрій Олександрович — студент групи 2КІ-24м, факультет інформаційних технологій та комп'ютерної інженерії, Вінницький національний технічний університет, Вінниця, e-mail: redstone.yuri@gmail.com

Zakharchenko Serhii Mykhailovych — Candidate of Technical Sciences, Professor, Vinnytsia National Technical University, Vinnytsia, e-mail: zakharchenko.sergii@vntu.edu.ua

Senyk Yuriy Oleksandrovych — student of group 2KI-24m, Faculty of Information Technologies and Computer Engineering, Vinnytsia National Technical University, Vinnytsia, e-mail: redstone.yuri@gmail.com