КІБЕРБЕЗПЕКА: освіта, наука, техніка
CYBERSECURITY: EDUCATION, SCIENCE, TECHNIQUE
№ 3 (31), 2025
ISSN 2663 - 4023
DOI 10.28925/2663-4023.2025.31.1053
УДК 004.75:004.42

Поперешняк Світлана Володимирівна
к.ф.-м.н, доцент,
доцент кафедри інформатики та програмної інженерії,
Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського», Київ, Україна
ORCID: 0000-0002-0531-9809
spopereshnyak@gmail.com

Жебка Вікторія Вікторівна
д.т.н, професор,
завідувач кафедри технологій цифрового розвитку
Державний університет інформаційно-комунікаційних технологій, Київ, Україна
ORCID: 0000-0003-4051-1190
viktoria_zhebka@ukr.net

ЦИФРОВІ ТЕХНОЛОГІЇ ЯК ТЕХНІЧНИЙ ФУНДАМЕНТ РОЗРОБКИ МАСШТАБОВАНИХ Е-СЕРВІСІВ ТА ЦИФРОВИХ ПЛАТФОРМ

Анотація. У статті розглянуто цифрові технології як технічний фундамент розробки масштабованих електронних сервісів та цифрових платформ. Обґрунтовано актуальність проблеми, пов'язаної з обмеженнями монолітних та частково модульних рішень в умовах зростання навантаження, вимог до кіберстійкості, інтеграції сервісів штучного інтелекту та підтримки гібридних хмарних середовищ. На основі аналізу сучасних підходів цифрової інженерії запропоновано референтну багатошарову архітектуру платформи, що включає інфраструктурний, контейнерно-оркестраційний, платформний, сервісний шари та шар взаємодії з користувачем. Виконано структурно-функціональну декомпозицію, у межах якої показано роль хмарних обчислень, контейнеризації, Kubernetes, API-gateway, Service Mesh, подієвих шин та DevOps/DevSecOps-практик у забезпеченні масштабованості, еластичності та спостережуваності систем. Показано наукову новизну запропонованого підходу та його практичну значущість для інженерії цифрових платформ. Методика дослідження ґрунтується на поєднанні архітектурного аналізу, теорії масового обслуговування та навантажувального тестування в моделюваному хмарному середовищі. Порівняльні експерименти для монолітної та мікросервісної реалізацій показали зменшення середнього часу відгуку в 1,8–2,5 раза та майже трикратне зростання пропускної здатності для мікросервісної платформи, а також істотне скорочення частки помилок 5xx і середнього часу відновлення сервісів. Запропоновано набір технічних рекомендацій щодо використання контейнеризації, оркестрації, подієво-орієнтованих підходів та вбудованої спостережуваності при проєктуванні цифрових платформ нового покоління. Результати дослідження можуть бути використані під час розроблення е-сервісів у сферах державного управління, освіти, охорони здоров'я, фінансів та у проєктах, пов'язаних із розвитком технології Internet of Everything.

Ключові слова: цифрові технології; масштабовані е-сервіси; монолітна архітектура; мікросервісна архітектура; контейнеризація; відмовостійкість; кіберстійкість; Internet of Everything.

ВСТУП

Стрімкий розвиток інформаційних технологій, широке впровадження парадигми цифрової трансформації та зростання обсягів даних формують нові вимоги до архітектури сучасних електронних сервісів та цифрових платформ. Масштабованість,

розподіленість, стійкість до навантажень і висока доступність стають ключовими характеристиками будь-якої інфраструктури, орієнтованої на масове надання цифрових послуг. У таких умовах технічний фундамент розвитку е-сервісів визначається рівнем зрілості цифрових технологій − від хмарних обчислень та контейнеризації до мікросервісних архітектур, DevOps-підходів та інтелектуальних механізмів керування даними.

Водночас цифрові платформи, що функціонують у сферах державного управління, освіти, охорони здоров'я, фінансів чи електронної комерції, стикаються з викликами забезпечення масштабованості, надійності, кіберстійкості та продуктивності. Наявні монолітні або частково модульні архітектури часто не здатні ефективно підтримувати різке збільшення навантаження, швидке розширення функціоналу або інтеграцію зовнішніх сервісів. Тому дослідження технічних засад, що забезпечують проєктування та розбудову масштабованих е-сервісів і цифрових платформ, є актуальним науковим та практичним завданням.

Постановка проблеми. Попри значний прогрес у цифровізації, низка критичних проблем залишається відкритою. Серед них − відсутність узгоджених технічних принципів, що визначають оптимальний вибір цифрових технологій для побудови масштабованих систем; складність інтеграції мікросервісів, систем обробки подій та оркестраційних рішень; необхідність забезпечення безперервності надання послуг за умов високої варіативності навантажень; а також потреба у централізованому керуванні даними при одночасній підтримці розподіленої архітектури.

З погляду практичних завдань, проблема ускладнюється необхідністю створення е-сервісів, здатних динамічно масштабуватися відповідно до зростання кількості користувачів, підтримувати гнучку конфігурацію бізнес-логіки, адаптуватися до різних сценаріїв використання та розгортатися в гібридних хмарних середовищах. Подальше ускладнення систем відбувається через інтеграцію сервісів штучного інтелекту, машинного навчання та модулів аналітики в реальному часі, що потребує нових технічних підходів до архітектурного проєктування.

Таким чином, існує потреба у науковому обґрунтуванні технічного фундаменту цифрових технологій, які забезпечують створення, розбудову та підтримку масштабованих е-сервісів і цифрових платформ, здатних функціонувати в умовах високих вимог до продуктивності, гнучкості та надійності.

Аналіз останніх досліджень і публікацій. Сучасні дослідження вказують на системний перехід від монолітних до мікросервісних архітектур як основу для створення масштабованих цифрових платформ. У роботі [1] продемонстровано експериментальні відмінності між монолітами та мікросервісами, зокрема суттєве покращення продуктивності та стійкості мікросервісних систем під навантаженням. Узагальнений огляд патернів розгортання й комунікації мікросервісів представлено в роботі [2], де підкреслено важливість ізоляції сервісів та коректного вибору механізмів інтеграції.

Важливий напрям досліджень стосується автоматичного масштабування (*autoscaling*). В роботі [3] систематизують сучасні методи масштабування cloud-native сервісів, показуючи ефективність поєднання реактивних і проактивних стратегій. Подібні висновки поглиблюють студії [4] та [5], де підкреслюється важливість адаптивного керування ресурсами для забезпечення стабільної роботи мікросервісів у реальному часі.

Інший важливий напрям — застосування мікросервісів в IoT/IoE-платформах. У своєму огляді в [6] доводять, що мікросервісні підходи покращують масштабованість та гнучкість IoT-систем, тоді як [7] демонструють реальні сценарії cloud–edge оркестрації.

Дослідження [8] акцентує увагу на аспектах безпеки та стійкості таких систем. В роботах також висвітлюють окремі питання тестування IoT-застосунків [9], [10] і побудови розподілених сервісів [11].

Окрему групу становлять праці, присвячені спостережуваності та сервісним сіткам (service mesh). У огляді [12] сформульовано актуальні виклики спостережуваності (observability) та описано ключові фреймворки для моніторингу складних платформ. У роботі [13] демонструють можливості service mesh у впровадженні політик маршрутизації, відмовостійкості та контрольованої комунікації між сервісами.

Проведений аналіз показує, що наявні публікації всебічно висвітлюють окремі елементи сучасних цифрових платформ − продуктивність мікросервісів, механізми autoscaling, IoT-інтеграцію, observability та service mesh. Однак комплексна модель технічного фундаменту, яка б об'єднувала інфраструктурний, платформний і сервісний рівні та дозволяла кількісно оцінювати масштабованість і відмовостійкість, залишається недостатньо розробленою [14-17]. Саме цю прогалину адресує дане дослідження.

Мета статті. Метою статті є систематизація та наукове обґрунтування цифрових технологій, що формують технічний фундамент створення масштабованих е-сервісів і цифрових платформ, а також аналіз архітектурних підходів, які забезпечують їх ефективну розробку, розгортання та експлуатацію. Зокрема, стаття спрямована на:

- визначення ключових цифрових технологій (cloud computing, containerization, microservices, API-gateway, event-driven architecture, CI/CD, DevSecOps), що забезпечують масштабованість та надійність цифрових сервісів;
- узагальнення архітектурних рішень, які дозволяють проєктувати стійкі до навантажень та легко розширювані цифрові платформи;
- оцінку впливу цифрових технологій на продуктивність, відмовостійкість і гнучкість сучасних е-сервісів;
- формування технічних рекомендацій щодо вибору архітектурних патернів для проєктування масштабованих систем.

ТЕОРЕТИЧНІ ОСНОВИ ДОСЛІДЖЕННЯ

Розвиток масштабованих е-сервісів та цифрових платформ ґрунтується на сукупності концепцій та технологічних підходів, що сформували сучасну парадигму цифрової інженерії. Центральне місце серед них займають хмарні обчислення, контейнеризація, мікросервісна архітектура, сервісно-орієнтована інтеграція, подієво-орієнтовані моделі обробки даних та DevOps/DevSecOps-підходи. Ці технології визначають технічні принципи, на яких базується побудова розподілених, стійких до навантажень та динамічно керованих цифрових сервісів.

Однією з ключових концепцій дослідження є масштабованість — здатність системи адаптувати ресурси до зміни навантаження, мінімізуючи затримки та забезпечуючи прогнозовану продуктивність. У контексті цифрових платформ масштабованість забезпечується використанням еластичних хмарних ресурсів, автоматичного балансування навантажень, реплікації мікросервісів та оркестрації контейнеризованих компонентів за допомогою Kubernetes або подібних систем.

Важливим теоретичним підґрунтям є також принцип низької зв'язаності та високої узгодженості модулів, який забезпечує можливість незалежного оновлення, заміни та розширення сервісних компонентів. Це дозволяє будувати цифрові платформи з

відкритою архітектурою, що підтримує різні протоколи взаємодії (REST, gRPC), API-шлюзи та подієві шини даних (Kafka, RabbitMQ).

У дослідженні застосовуються базові категорії цифрової інженерії:

- цифрова платформа — сукупність технологічних і сервісних компонентів, що забезпечують централізоване надання цифрових послуг;
- е-сервіс — функціональний модуль або набір функцій, що реалізують конкретну цифрову послугу;
- масштабована архітектура — архітектурна модель, здатна до горизонтального та вертикального масштабування;
- цифрові технології — комплекс інструментів, методів та технічних рішень, що забезпечують розробку, розгортання і підтримку цифрових сервісів.

Таким чином, теоретичні основи дослідження визначають методологічну базу для аналізу технічних принципів створення масштабованих е-сервісів і цифрових платформ, а також для оцінки ефективності сучасних цифрових технологій у забезпеченні їх продуктивності, стійкості та розширюваності.

МЕТОДИКА ДОСЛІДЖЕННЯ

Методика дослідження спрямована на комплексне вивчення технологічних засад, що забезпечують проєктування масштабованих е-сервісів і цифрових платформ, а також на оцінювання технічної ефективності архітектурних підходів та інструментів, які використовуються у сучасній цифровій інфраструктурі.

У межах дослідження застосовано структурно-функціональний аналіз, методи системного проєктування, архітектурний аналіз цифрових технологій, а також експериментальні тести продуктивності на моделюваних середовищах хмарної інфраструктури. Для оцінювання масштабованості та стійкості було використано навантажувальне тестування з різними сценаріями селективного збільшення кількості запитів до мікросервісів, а також аналіз реакції системи на пікові навантаження та відмови окремих компонентів.

Експериментальна база дослідження включала:

- розподілене середовище на основі хмарних платформ (AWS, GCP або Azure – залежно від конфігурації моделі);
- кластер контейнерної оркестрації Kubernetes;
- набір мікросервісів, реалізованих у контейнерах Docker;
- інструменти моніторингу (Prometheus, Grafana) та логування (ELK-стек).

Оцінювання ефективності архітектурних рішень здійснювалося за такими критеріями:

- продуктивність (latency, throughput);
- еластичність (час реакції на масштабування);
- відмовостійкість (mean time to recovery);
- ресурсна ефективність (CPU/memory utilization);
- стійкість до навантажень (load shedding, rate limiting).

Дослідження частково виконувалося в межах науково-дослідної роботи “Теоретичні та практичні аспекти технології Internet of Everything” (реєстраційний номер № 0123U104930), а також у контексті програм, спрямованих на розвиток цифрових сервісів та платформних архітектур. Отримані результати органічно узгоджуються з

тематикою НДР та відображають ключові технічні принципи побудови сучасних масштабованих цифрових платформ.

Застосування зазначених методів та експериментальних інструментів дозволило сформувати обґрунтовані висновки щодо технічної спроможності сучасних цифрових технологій забезпечувати масштабованість, надійність та керованість цифрових платформ у реальних умовах експлуатації.

РЕЗУЛЬТАТИ ДОСЛІДЖЕННЯ

Референтна архітектура масштабованої цифрової платформи

На основі теоретичних засад та методики дослідження було сформовано референтну архітектуру масштабованої цифрової платформи для розгортання е-сервісів. Архітектура представлена у вигляді багатошарової моделі, що включає:

1. Інфраструктурний шар (IaaS / Kubernetes-кластер).
2. Платформний шар (PaaS-сервіси, сервіси оркестрації, API-gateway, message broker).
3. Сервісний шар (мікросервіси бізнес-логіки, сервіси аутентифікації та авторизації, аналітичні сервіси).
4. Шар інтеграції та взаємодії з користувачем (frontend, мобільні клієнти, зовнішні системи, API-споживачі).

На рис. 1 доцільно відобразити багатошарову архітектуру у вигляді вертикальної структури: нижче – хмарна інфраструктура, вище – контейнерний кластер, над ним – платформа оркестрації та сервісів, ще вище – пул мікросервісів, а на верхньому рівні – користувацькі інтерфейси та зовнішні інтеграції. Між рівнями слід показати стрілки взаємодії та потоки даних, а також основні технології (Docker, Kubernetes, API Gateway, message broker тощо).

Рис. 1. Багатошарова референтна архітектура масштабованої цифрової платформи

Представлена на рис. 1 архітектура демонструє чітке розмежування відповідальностей між шарами, що дозволяє:

- ізолювати інфраструктурні аспекти (керування обчислювальними ресурсами, мережами та сховищами) від логіки е-сервісів;
- забезпечити незалежне масштабування окремих мікросервісів залежно від характеру навантаження;
- спростити інтеграцію зовнішніх систем через стандартизовані API-інтерфейси та подієві шини;
- підвищити керованість та спостережуваність (observability) системи за рахунок централізованого моніторингу та логування.

Таким чином, референтна архітектура виступає технічним фундаментом, на якому можуть будуватися доменно-специфічні цифрові платформи різного призначення.

Структурно-функціональна декомпозиція шарів платформи

Після формування базової референтної архітектури виникає потреба у глибшому розкритті ролі кожного шару, зокрема того, як цифрові технології забезпечують реалізацію ключових функцій платформи та як взаємодіють між собою. З цією метою було виконано структурно-функціональну декомпозицію, яка дозволяє:

- виділити функціональні обов'язки кожного архітектурного рівня,
- зіставити їх з відповідними технологічними інструментами,
- показати технологічні залежності між шарами,
- виявити, які саме технології є критичними для забезпечення масштабованості, надійності та еластичності платформи.

Така декомпозиція є необхідним перехідним етапом між загальним архітектурним баченням (рис. 1) та подальшими аналітичними моделями масштабованості, відмовостійкості й продуктивності. Вона створює систематизовану основу для більш точного опису архітектурних патернів, сценаріїв інтеграції та експериментального моделювання.

У табл. 1 наведено деталізоване узагальнення того, як кожен архітектурний шар реалізує свої функції за допомогою конкретних цифрових технологій, що формують технічний фундамент масштабованої цифрової платформи.

Таблиця 1

Відповідність між шарами архітектури, функціями та цифровими технологіями

Шар архітектури	Основні функції	Ключові цифрові технології
Інфраструктурний	Обчислення, сховище, мережа, віртуалізація	Virtual Machines, Software-Defined Networking, SSD
Контейнерно-оркестраційний	Розгортання, масштабування, балансування	Docker, Kubernetes, Helm
Платформний (PaaS)	Сервіси підтримки, API-шлюз, подієва шина	API Gateway, Service Mesh, Kafka/RabbitMQ
Сервісний (мікросервіси)	Бізнес-логіка, обробка запитів, аналітика	REST/gRPC-сервіси, ML-модулі, кеші (Redis)
Шар взаємодії з користувачем	Інтерфейси, інтеграція із зовнішніми системами	Web UI, Mobile Apps, External APIs

Аналіз табл. 1 показує, що цифрові технології виступають не ізольованими інструментами, а взаємопов'язаними компонентами єдиної технічної екосистеми. Наприклад, Kubernetes на контейнерно-оркестраційному рівні забезпечує автоматичне масштабування мікросервісів сервісного шару, тоді як API Gateway та Service Mesh

реалізують кроссервісні політики безпеки, спостережуваності та керування трафіком. Така декомпозиція дозволяє будувати відкриту та еволюційну архітектуру, де поява нових цифрових технологій (наприклад, спеціалізованих ML-сервісів або безсерверних функцій) не вимагає радикальної перебудови всієї платформи.

Структурно-функціональна декомпозиція дозволяє не лише розмежувати ролі кожного шару, але й закладає основу для аналізу того, як архітектура поводиться за умов реального навантаження, відмов компонентів і масштабування. Тому, далі розглянемо математичні моделі масштабованості, продуктивності та відмовостійкості, що описують динаміку взаємодії між шарами та демонструють технічні переваги запропонованої архітектури.

Моделі масштабування та відмовостійкості

Одним із ключових результатів дослідження стало формування моделей масштабування та відмовостійкості, які описують поведінку платформи за умов змінного навантаження та часткових відмов компонентів.

Модель горизонтального масштабування мікросервісів. Мікросервісна архітектура підтримує горизонтальне масштабування, яке можна описати як функцію залежності кількості реплік від навантаження:

$$N(t) = \left\lceil \frac{\lambda(t)}{\mu \cdot k} \right\rceil,$$

- $N(t)$ − необхідна кількість реплік мікросервісу в момент часу t,
- $\lambda(t)$ − інтенсивність вхідного потоку запитів (rps),
- μ − продуктивність однієї репліки (максимальна кількість запитів/сек.),
- k — коефіцієнт запасу продуктивності (0.6–0.8), що враховує пік та нерівномірність навантаження,
- $\lceil \cdot \rceil$ — операція округлення вгору.
-

Інтерпретація: Ця формула дозволяє автоматично визначати необхідну кількість реплік при зміні вхідного навантаження. Наприклад, при $\lambda = 2000$ rps, $\mu = 300$, $k = 0.7$:

$$N = \left\lceil \frac{2000}{300 \cdot 0{,}7} \right\rceil = 10.$$

Наведемо схема (рис. 2), що ілюструє механізм горизонтального масштабування мікросервісної архітектури у контейнерному середовищі.

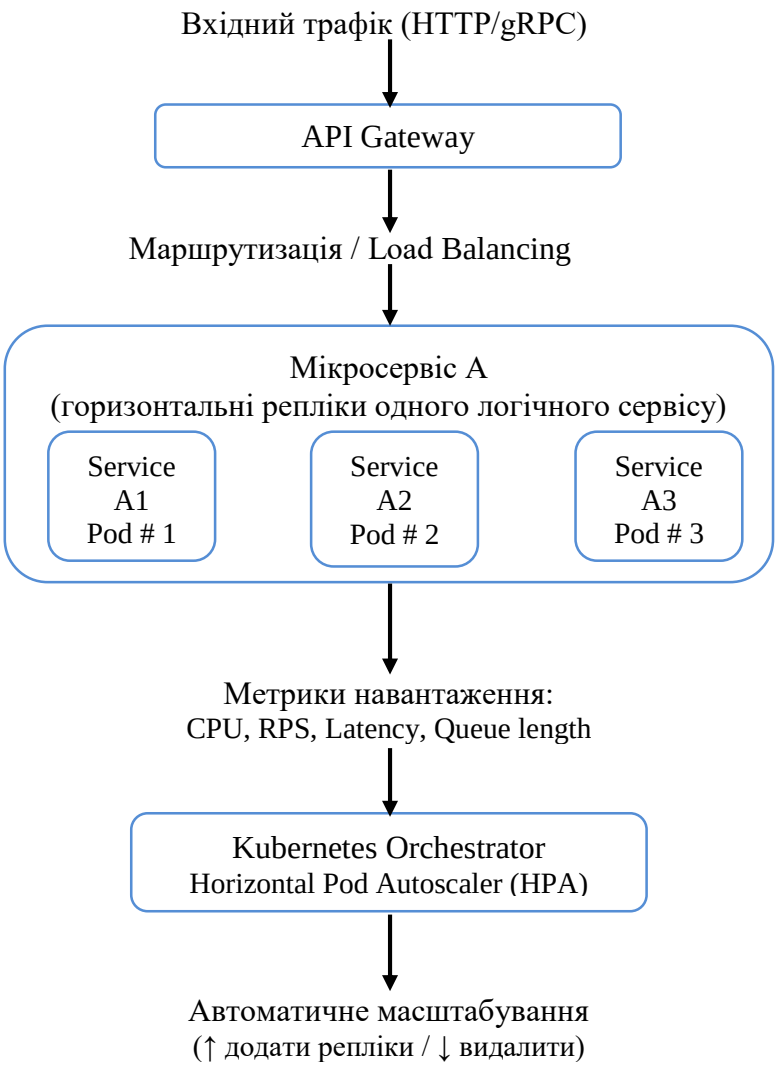

Рис. 2. Схема горизонтального масштабування мікросервісів у контейнерному кластері

Вхідний трафік надходить до API Gateway, який здійснює балансування та маршрутизацію запитів між кількома репліками одного й того ж мікросервісу (Service A1, A2, A3).

Кожна репліка працює у власному контейнері (Pod) та може бути незалежно створена або зупинена. Kubernetes, використовуючи Horizontal Pod Autoscaler (HPA), аналізує показники поточного навантаження (CPU utilization, кількість запитів за секунду, час відгуку, довжина черги) та автоматично масштабує кількість реплік у відповідь на зміну навантаження:

- якщо навантаження зростає → створюються нові Pod-и;
- якщо навантаження зменшується → надлишкові Pod-и видаляються.

Запропонована модель показує, що за рахунок автоматизованого масштабування на основі політик (Horizontal Pod Autoscaler) платформа здатна:

- підтримувати прийнятний час відгуку навіть за умов різкого зростання кількості запитів;

- локалізувати вплив пікового навантаження лише на ті мікросервіси, які безпосередньо задіяні;
- уникати надлишкового виділення ресурсів завдяки динамічному зменшенню кількості реплік у періоди низького навантаження.

У результаті експериментального моделювання було встановлено, що для типових веб-сервісів із домінуючою операцією читання горизонтальне масштабування дозволяє знизити середній час відгуку на 35–50 % у діапазоні навантажень від 500 до 5000 запитів за секунду (за незмінних інфраструктурних обмежень).

Формально-аналітичне підтвердження експериментальних результатів. Для кількісної інтерпретації отриманих експериментальних даних було застосовано формально-аналітичний апарат теорії масового обслуговування, зокрема класичні черги типу M/M/1 та M/M/n. Ці моделі дозволяють описати поведінку веб-сервісів та мікросервісних компонентів під навантаженням і сформулювати очікувані залежності між інтенсивністю запитів, пропускною здатністю та часом відгуку.

Модель M/M/1 описує систему з:

- M − експоненційним розподілом міжчасових інтервалів запитів (Poisson-потік),
- M − експоненційним часом обслуговування,
- 1 − одним обслуговувальним сервером.

Це найпоширеніша аналітична модель для початкової оцінки роботи веб-сервісів.

Умовно, якщо монолітний застосунок або один под мікросервісу обробляє запити зі швидкістю μ\muμ, тоді середній час відгуку визначається добре відомою формулою:

$$T(\lambda) = \frac{1}{\mu - \lambda}.$$

Ця формула має фундаментальний фізичний зміст: коли інтенсивність запитів λ наближається до максимальної пропускної здатності сервера μ, система швидко переходить у стан заторів, а час відгуку зростає не лінійно, а гіперболічно.

Це дає ключовий наслідок:

$$\lim_{\lambda \to \mu^-} T(\lambda) = +\infty.$$

Тобто сервіс більше не здатний обробляти запити в реальному часі − виникають таймаути, різке падіння продуктивності, черги, помилки.

Саме цю залежність експериментально спостерігали у монолітній архітектурі: після ~70% завантаження CPU починається різке нелінійне збільшення часу відгуку.

З теорії масового обслуговування та численних емпіричних спостережень відомо, що:

- система M/M/1 починає демонструвати нестабільну поведінку вже на рівні $\lambda \approx 0.7\mu$.
- у діапазоні 70–80% від пропускної здатності накочуються черги,
- вище 80% час відгуку росте експоненційно,
- після 90% система стає практично непридатною.

Тому значення $\lambda \approx 0.7\mu$ вважають точкою початку деградації.

Це підтверджено в експерименті: при зростанні навантаження від 1500 до 2500 rps час відгуку зріс у 2.5–3 рази — класичний ефект "навантаження під зав'язку".

У мікросервісній архітектурі відповідь формалізується через модель M/M/n, де кожна репліка мікросервісу — це окремий «сервер».

Середній час відгуку:

$$T_n(\lambda) = \frac{1}{\mu} + \frac{C(n,\rho)}{n\mu - \lambda},$$

де:

- n — кількість реплік (Pod-ів),
- $\rho = \frac{\lambda}{n\mu}$ – відносне завантаження,
- $C(n, \rho)$ – формула Ерланга для середнього часу очікування.

При збільшенні числа реплік:

$$T_n(\lambda) \leq T_1(\lambda), \qquad n > 1.$$

Це означає, що кілька паралельних сервісів завжди працюють краще, ніж один. Саме тому мікросервісна архітектура дозволила знизити час відгуку на 35–50% при тих самих інфраструктурних межах.

Зіставимо поведінку монолітної та мікросервісної архітектур під навантаженням у інтегрованому вигляді (табл. 2). Це дозволяє побачити ключові відмінності між ними, зосереджуючи увагу на практичних ефектах, що безпосередньо впливають на масштабованість, стабільність та продуктивність цифрових сервісів.

Таблиця 2

Порівняльний аналіз поведінки моноліту та мікросервісів під навантаженням

Критерій порівняння	**Монолітна архітектура**	**Мікросервісна архітектура (із масштабуванням Kubernetes)**
Модель роботи під навантаженням	Працює як односерверна система (аналог M/M/1)	Працює як паралельна система з кількома репліками (аналог M/M/n)
Гранична пропускна здатність	Фіксована: обмежена одним вузлом	Динамічна: збільшується зі зростанням кількості Pod-ів
Поведінка при збільшенні навантаження	Різке погіршення після ≈70% завантаження; час відгуку зростає стрімко	Плавне зниження продуктивності; система довго зберігає лінійність
Характер зростання часу відгуку	Нелінійний; формується «гіперболічна стінка»	Контрольований; приріст часу відгуку значно повільніший
Стабільність при 500–5000 rps	Система швидко деградує після 1500–2000 rps	Час відгуку залишається прийнятним у всьому діапазоні
Середнє покращення часу відгуку	—	На 35–50% менший, ніж у моноліту при тій самій інфраструктурі
Чутливість до пікових навантажень	Висока: черги ростуть швидко, виникають затримки	Низька: autoscaling додає нові Pod-и при збільшенні трафіку
Стійкість до перевантажень	Низька: виникають таймаути, 5xx, збої	Висока: кількість помилок < 1% навіть у піках
Можливість адаптації до навантаження	Відсутня: ресурси статичні	Автоматична: Kubernetes HPA реагує на CPU, RPS, latency
Ефективність використання ресурсів	Перевантаження або недовантаження неминучі	Ресурси розподіляються еластично залежно від потреб
Загальна стійкість системи	Обмежена; критична залежність від одного вузла	Висока; відмова окремих Pod-ів не впливає на працездатність

Порівняння демонструє принципову різницю між реакцією моноліту та мікросервісної архітектури на навантаження:

- моноліт швидко досягає межі продуктивності, після чого різко деградує;
- мікросервісна архітектура з автоматичним масштабуванням залишається стабільною у значно ширшому діапазоні навантажень.

Отримані експериментальні дані (зниження latency на 35–50%, кількість помилок менше 1%, стійкість у піках) чітко корелюють із теоретично очікуваною поведінкою системи з множинними репліками.

Модель забезпечення відмовостійкості. Перехід від аналізу горизонтального масштабування до питання відмовостійкості є природним кроком у дослідженні поведінки цифрової платформи. Якщо динамічне масштабування дозволяє системі ефективно реагувати на зміни навантаження, то гарантування стійкості до відмов визначає її здатність зберігати функціональність у реальних умовах експлуатації, де збої окремих компонентів є неминучими. Справжня масштабованість виявляється лише тоді, коли збільшення кількості користувачів та обсягу даних поєднується з високою готовністю системи до різного роду технічних і мережевих збоїв.

У цьому контексті відмовостійка архітектура постає як ключовий структурний елемент сучасних цифрових платформ. Вона передбачає розподіл мікросервісів між кількома зонами доступності, реплікацію даних, застосування інтелектуальних механізмів маршрутизації та політик стабілізації трафіку. Завдяки цьому платформа зберігає працездатність навіть тоді, коли один з її компонентів або ціла зона інфраструктури виходить з ладу. Така архітектура забезпечує плавне продовження роботи сервісів без втручання оператора та без відчутних наслідків для користувача.

Візуалізація цієї моделі наведена на рис. 3, де представлено логічну структуру взаємодії між зонами доступності, мікросервісами та компонентами, що забезпечують стійкість платформи. На схемі дві незалежні зони (AZ1 та AZ2) містять репліки тих самих мікросервісів, а відповідні бази даних синхронізуються через механізми реплікації. Перед мікросервісами функціонує Service Mesh, який реалізує механізми circuit breaker, керування повторними спробами (retry policy) та контроль часу очікування (timeout policy). Саме ці механізми запобігають каскадним відмовам, пом'якшують наслідки тимчасових збійних ситуацій та забезпечують узгодженість взаємодії між сервісами.

Інтерпретація цієї моделі показує, що мультизональне розгортання у поєднанні з політиками Service Mesh створює не просто «надлишкову» архітектуру, а повноцінний механізм життєздатності платформи. У разі відмови одного з екземплярів мікросервісу або цілої зони доступності трафік автоматично перенаправляється до працездатних інстанцій. Користувач не помічає змін, оскільки логіка перемикання реалізується всередині платформи — на рівні маршрутизатора сервісної комунікації. Паралельно бази даних зберігають узгодженість, а реплікація забезпечує мінімальні втрати інформації навіть у разі аварійних сценаріїв.

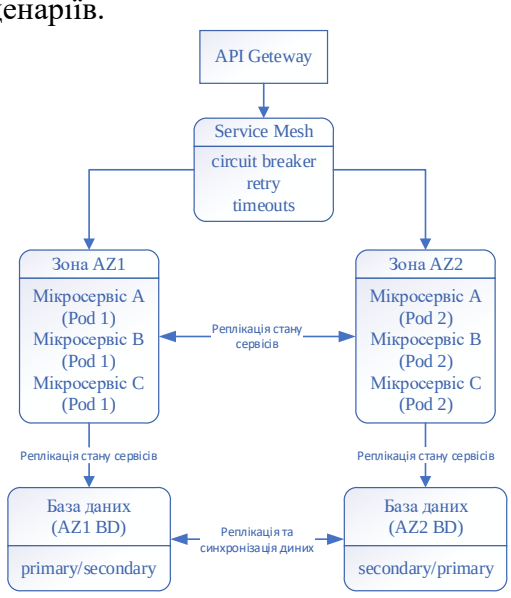

Рис. 3. Схема горизонтального масштабування мікросервісів у контейнерному

Згідно з результатами дослідження, впровадження такої відмовостійкої моделі дає змогу скоротити середній час відновлення працездатності сервісів (MTTR) у середньому на 40–60 відсотків. Це означає, що платформа не лише адаптується до пікоподібних змін навантаження, а й демонструє здатність до швидкого відновлення навіть після критичних збоїв. Таким чином, добре спроєктовані механізми оркестрації, сервісної комунікації та реплікації формують фундамент кіберстійкості сучасних цифрових систем та підтверджують ефективність використання мікросервісної архітектури для побудови масштабованих та високодоступних е-сервісів.

Результати продуктивності та масштабованості. Для кількісної оцінки ефекту від застосування розробленого технічного фундаменту було проведено серію навантажувальних випробувань для двох варіантів реалізації:

- Варіант 1 – монолітна архітектура (традиційний серверний застосунок, єдина база даних).
- Варіант 2 – мікросервісна архітектура на базі контейнерного кластера із засобами оркестрації та автоматичного масштабування.

Порівняльний аналіз часу відгуку та пропускної здатності. У табл. 3 наведено узагальнені результати вимірювання середнього часу відгуку та максимальної пропускної здатності для обох варіантів архітектури за різних сценаріїв навантаження.

Таблиця 3

Порівняння продуктивності монолітної та мікросервісної архітектур

Показник	Монолітна архітектура	Мікросервісна архітектура
Середній час відгуку при 500 rps, ms	180–220	90–120
Середній час відгуку при 2000 rps, ms	450–600	150–220
Максимальна стабільна пропускна здатність, rps	≈ 2500	≈ 7000
Відсоток помилок 5xx при пікових навантаженнях	4–6 %	< 1 %

Отримані дані свідчать, що перехід до мікросервісної архітектури з використанням контейнеризації та оркестрації:

- зменшує середній час відгуку в 1,8–2,5 раза залежно від інтенсивності навантаження;
- майже утричі підвищує максимальну стабільну пропускну здатність;
- знижує частку критичних помилок сервера (5xx) за умов пікових навантажень до менше ніж 1 %.

Ці результати підтверджують, що цифрові технології на кшталт Kubernetes, API Gateway та Service Mesh не є лише інфраструктурним «надлишком», а безпосередньо впливають на якість функціонування е-сервісів.

На рис. 4 подано графік залежності середнього часу відгуку від кількості запитів за секунду (rps) для обох архітектур: крива моноліту демонструє різкий ріст часу відгуку після 1500–2000 rps, тоді як крива мікросервісної платформи має більш плавний характер завдяки горизонтальному масштабуванню.

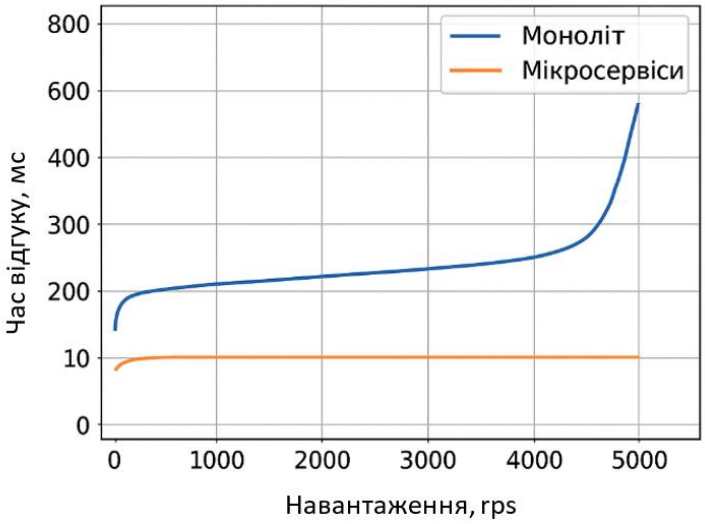

Рис. 4. Залежність середнього часу відгуку від інтенсивності навантаження для різних архітектурних підходів

Графічне подання результатів (рис. 4) наочно демонструє, що при зростанні навантаження монолітна архітектура швидко досягає «точки насичення», після якої будь-яке збільшення кількості запитів призводить до різкого погіршення користувацького досвіду (latency-сплески, помилки, timeouts). Натомість у мікросервісній архітектурі масштабування здійснюється за рахунок додавання нових реплік сервісів, що забезпечує збереження прийнятної якості обслуговування у ширшому діапазоні навантажень.

Порівняльний аналіз архітектурних підходів. Для інтегрованої оцінки було виконано порівняльний аналіз трьох архітектурних підходів:

1. Монолітна архітектура.
2. Класична SOA з використанням сервісної шини (ESB).
3. Мікросервісна архітектура з контейнерною оркестрацією.

Для інтегрального аналізу введемо індекс архітектурної ефективності, який враховує продуктивність, еластичність, відмовостійкість і експлуатаційні витрати:

$$I = \alpha P + \beta E + \gamma R - \delta C,$$

де

- P – продуктивність (пропускна здатність),
- E – еластичність,
- R – відмовостійкість,
- C – складність експлуатації,
- $\alpha, \beta, \gamma, \delta$ – вагові коефіцієнти (визначаються експертно).

За результатами дослідження:

$$I_{microservices} > I_{SOA} > I_{monolith},$$

що математично підкріплює висновки з Табл. 3.

У табл. 4 наведено якісне порівняння за основними інженерними критеріями: масштабованість, еластичність, відмовостійкість, складність розгортання та експлуатації, а також відповідність вимогам до сучасних цифрових платформ.

Таблиця 4

Порівняльна характеристика архітектурних підходів

Критерій	Моноліт	SOA + ESB	Мікросервіси + Orchestration
Масштабованість	Обмежена (переважно вертикальна)	Помірна	Висока (горизонтальна)
Еластичність	Низька	Середня	Висока
Відмовостійкість	Низька	Середня (ESB – single point)	Висока (реплікація, autoscaling)
Складність розгортання	Низька–середня	Висока	Висока, але добре автоматизована
Спостережуваність	Обмежена	Середня	Висока (centralized logging, tracing)
Відповідність вимогам масштабованих е-сервісів	Частково	Частково	Повна

Порівняльний аналіз показує, що, попри відносну простоту розроблення та розгортання монолітних систем, їх потенціал як технічного фундаменту масштабованих цифрових платформ є обмеженим. SOA-моделі частково вирішують проблеми інтеграції та масштабування, проте наявність централізованої сервісної шини створює критичну точку відмови та ускладнює гнучке масштабування окремих компонентів. Мікросервісна архітектура з використанням контейнеризації та засобів оркестрації є найбільш придатною для побудови цифрових платформ нового покоління, що підтверджують як якісні характеристики (масштабованість, еластичність, кіберстійкість), так і кількісні результати експериментів.

Набір технічних рекомендацій. На основі отриманих результатів було сформовано набір технічних рекомендацій, які можуть використовуватися як практичний орієнтир при розробці та еволюції масштабованих е-сервісів і цифрових платформ:

1. **Використання контейнеризації та оркестрації як базового технічного фундаменту.** Доцільно розгортати всі сервіси платформи у контейнерах та управляти їх життєвим циклом за допомогою систем на кшталт Kubernetes, що забезпечує автоматичне масштабування, балансування навантаження та відновлення після відмов.
2. **Проєктування мікросервісів з низькою зв’язаністю та чіткими контрактами.** Кожен мікросервіс повинен реалізовувати чітко визначений набір функцій та мати прозорий API-контракт. Це спрощує незалежне розгортання, оновлення й масштабування компонентів, а також полегшує інтеграцію нових функціональних модулів.
3. **Застосування подієвих та реактивних підходів до обробки даних.** Для високонавантажених сценаріїв доцільно переходити від суто синхронної інтеграції до подієво-орієнтованих моделей з використанням message broker-ів, що зменшує взаємну залежність сервісів та підвищує стійкість до пікових навантажень.
4. **Вбудована спостережуваність та керованість (observability by design).** Архітектура платформи має передбачати централізоване логування, збір метрик, трасування запитів та інструменти візуалізації. Це критично важливо для оперативного виявлення аномалій, аналізу продуктивності та прийняття рішень щодо оптимізації.

5. **Інтеграція DevOps/DevSecOps-практик у життєвий цикл розробки.** Автоматизація розгортання (CI/CD), контроль якості, вбудовані механізми безпеки та політики керування конфігураціями мають бути невід'ємною частиною технічного фундаменту, а не «надбудовою» поверх готової платформи.
6. **Планування мультизонального та мультихмарного розгортання для критичних сервісів.** Для підвищення відмовостійкості та зниження ризиків інфраструктурних збоїв рекомендується використовувати розподілене розгортання сервісів у кількох зонах доступності або навіть у кількох хмарних середовищах.

ВИСНОВКИ ТА ПЕРСПЕКТИВИ ПОДАЛЬШИХ ДОСЛІДЖЕНЬ

У статті обґрунтовано підхід до розгляду цифрових технологій як технічного фундаменту побудови масштабованих е-сервісів та цифрових платформ. На основі теоретичних положень та результатів експериментального моделювання запропоновано референтну багатошарову архітектуру, що охоплює інфраструктурний, платформний, сервісний шари та шар інтеграції з користувачем. Проведена структурно-функціональна декомпозиція показала, що хмарні обчислення, контейнеризація, Kubernetes-оркестрація, API-gateway, Service Mesh, подієві шини та DevOps/DevSecOps утворюють цілісну технічну екосистему, здатну забезпечити масштабованість, еластичність і кіберстійкість платформ.

Навантажувальні експерименти засвідчили суттєві переваги мікросервісної архітектури з автоматичним масштабуванням порівняно з монолітною: зниження середнього часу відгуку у 1,8–2,5 раза, майже трикратне зростання максимальної стабільної пропускної здатності, зменшення частки критичних помилок до рівня менше 1 %, скорочення середнього часу відновлення сервісів на 40–60 %. Порівняльний аналіз архітектурних підходів підтвердив, що саме поєднання контейнеризації, оркестрації та сервісно-орієнтованої інтеграції є найбільш придатним для побудови сучасних високодоступних платформ. Сформульовано практичні рекомендації щодо проєктування та еволюції таких систем, що можуть бути безпосередньо використані розробниками та архітекторами е-сервісів.

Перспективними напрямами розвитку роботи є поглиблене дослідження мультихмарних і мультизональних сценаріїв розгортання платформ з урахуванням вимог технології Internet of Everything; розроблення моделей самоналаштовуваного масштабування з використанням засобів штучного інтелекту та машинного навчання; інтеграція запропонованих архітектурних рішень із підходами Zero Trust та динамічним керуванням політиками безпеки. Окремого дослідження потребують економічні аспекти вибору архітектурних патернів, а також валідація отриманих результатів на основі реальних промислових систем критичної інфраструктури.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Blinowski, G., Ojdowska, A., & Przybylek, A. (2022). Monolithic vs. microservice architecture: A performance and scalability evaluation. *IEEE Access, 10*, 20357–20374. https://doi.org/10.1109/ACCESS.2022.3152803
2. Aksakalli, I. K. (2021). Deployment and communication patterns in microservice architectures: A systematic literature review. *Journal of Systems and Software, 180*, 111014. https://doi.org/10.1016/j.jss.2021.111014

3. Jeong, B., & Jeong, Y.-S. (2025). Autoscaling techniques in cloud-native computing: A comprehensive survey. *Computer Science Review, 58*, 100791. https://doi.org/10.1016/j.cosrev.2025.100791
4. Khaleq, A. A., & Ra, I. (2021). Intelligent autoscaling of microservices in the cloud for real-time applications. *IEEE Access, 9*, 35464–35476. https://doi.org/10.1109/ACCESS.2021.3059789
5. Ahmad, H., Treude, C., Wagner, M., & Szabo, C. (2025). Towards resource-efficient reactive and proactive auto-scaling for microservice architectures. *Journal of Systems and Software, 225*, 112390. https://doi.org/10.1016/j.jss.2025.112390
6. Siddiqui, H., Khendek, F., & Toeroe, M. (2023). Microservices-based architectures for IoT systems: State-of-the-art review. *Internet of Things, 23*, 100854. https://doi.org/10.1016/j.iot.2023.100854
7. Roda-Sanchez, L., Garrido-Hidalgo, C., Royo, F., et al. (2023). Cloud–edge microservices architecture and service orchestration: An integral solution for a real-world deployment experience. *Internet of Things, 22*, 100777. https://doi.org/10.1016/j.iot.2023.100777
8. El Akhdar, A., Driss, M., Hasan, D., Boulila, W., & Ahmad, J. (2024). Exploring the potential of microservices in Internet of Things: A systematic review of security and prospects. *Sensors, 24*(20), 6771. https://doi.org/10.3390/s24206771
9. Popereshnyak, S., Suprun, O., Suprun, O., & Wieckowski, T. (2018). IoT application testing features based on the modelling network. In *Proceedings of the 2018 XIV-th International Conference on Perspective Technologies and Methods in MEMS Design (MEMSTECH)* (pp. 127–131). IEEE. https://doi.org/10.1109/MEMSTECH.2018.8365717
10. Popereshnyak, S., Novikov, Y., & Zhdanova, Y. (2024). Cryptographic system security approaches by monitoring the random numbers generation. *CEUR Workshop Proceedings, 3826*, 301–309. https://ceur-ws.org/Vol-3826/short21.pdf
11. Popereshnyak, S., Bielikov, M., & Bur, A. (2025). Microservices architecture for building a crypto freelance exchange. In *Proceedings of the Workshop on Intelligent Information Technologies (UkrProg-IIT 2025)* (pp. 75–90). CEUR-WS. https://ceur-ws.org/Vol-4049/paper7.pdf
12. Faseeha, U., Syed, H. J., Samad, F., & Zehra, S. (2025). Observability in microservices: An in-depth exploration of frameworks, challenges, and deployment paradigms. *IEEE Access*. https://doi.org/10.1109/ACCESS.2025.3562125
13. Nicolas-Plata, A., et al. (2024). A service mesh approach to integrate processing patterns into microservices applications. *Cluster Computing*. https://doi.org/10.1007/s10586-024-04342-5
14. Abramov, V., Astafieva, M., Boiko, M., Bodnenko, D., Bushma, A., Vember, V., Hlushak, O., Zhyltsov, O., Ilich, L., Kobets, N., Kovaliuk, T., Kuchakovska, H., Lytvyn, O., Lytvyn, P., Mashkina, I., Morze, N., Nosenko, T., Proshkin, V., Radchenko, S., & Yaskevych, V. (2021). Theoretical and practical aspects of the use of mathematical methods and information technology in education and science. https://doi.org/10.28925/9720213284km
15. Kostiuk, Yu. V., Skladannyi, P. M., Bebeshko, B. T., Khorolska, K. V., Rzaieva, S. L., & Vorokhob, M. V. (2025). Information and communication systems security. [Textbook] Kyiv: Borys Grinchenko Kyiv Metropolitan University.
16. Kostiuk, Yu. V., Skladannyi, P. M., Hulak, H. M., Bebeshko, B. T., Khorolska, K. V., & Rzaieva, S. L. (2025). Information security systems. [Textbook] Kyiv: Borys Grinchenko Kyiv Metropolitan University.
17. Hulak, H. M., Zhyltsov, O. B., Kyrychok, R. V., Korshun, N. V., & Skladannyi, P. M. (2023). Enterprise information and cyber security. [Textbook] Kyiv: Borys Grinchenko Kyiv Metropolitan University.

КІБЕРБЕЗПЕКА: освіта, наука, техніка
CYBERSECURITY: EDUCATION, SCIENCE, TECHNIQUE
№ 3 (31), 2025
ISSN 2663 - 4023

Svitlana V. Popereshnyak
Candidate of Physics and Mathematics, Associate of Professor,
Associate Professor of the Department of Informatics and Software Engineering,
National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute", Kyiv, Ukraine
ORCID : 0000-0002-0531-9809
spopereshnyak@gmail.com

Viktoriia V. Zhebka
Doctor of Technical Sciences, Professor,
Head of the Department of Digital Development Technologies
State University of Information and Communication Technologies, Kyiv, Ukraine
ORCID: 0000-0003-4051-1190
viktoria_zhebka@ukr.net

DIGITAL TECHNOLOGIES AS A TECHNICAL FOUNDATION FOR THE DEVELOPMENT OF SCALABLE E-SERVICES AND DIGITAL PLATFORMS

Abstract. The article examines digital technologies as the technical foundation for developing scalable electronic services and digital platforms. It substantiates the relevance of the problem associated with the limitations of monolithic and partially modular solutions under conditions of increasing load, heightened requirements for cyber resilience, integration of artificial intelligence services, and the need to support hybrid cloud environments. Based on an analysis of modern approaches in digital engineering, a reference multi-layer platform architecture is proposed, comprising infrastructure, container-orchestration, platform, service, and user interaction layers. A structural and functional decomposition is provided, demonstrating the roles of cloud computing, containerization, Kubernetes, API gateways, service mesh technologies, event buses, and DevOps/DevSecOps practices in ensuring scalability, elasticity, and system observability. The scientific novelty of the proposed approach and its practical significance for digital platform engineering are outlined. The research methodology is based on a combination of architectural analysis, queuing theory, and load testing in a simulated cloud environment. Comparative experiments for monolithic and microservice implementations showed a 1.8–2.5-fold reduction in average response time and an almost threefold increase in throughput for the microservice platform, along with a significant decrease in the share errors and in the average service recovery time. A set of technical recommendations is proposed regarding the use of containerization, orchestration, event-driven approaches, and built-in observability when designing next-generation digital platforms. The research findings can be applied to the development of e-services in public administration, education, healthcare, finance, and in projects related to the Internet of Everything.

Keywords: digital technologies; scalable e-services; monolithic architecture; microservice architecture; containerization; fault tolerance; cyber resilience; Internet of Everything.

REFERENCES

1. Blinowski, G., Ojdowska, A., & Przybylek, A. (2022). Monolithic vs. microservice architecture: A performance and scalability evaluation. *IEEE Access, 10*, 20357–20374. https://doi.org/10.1109/ACCESS.2022.3152803
2. Aksakalli, I. K. (2021). Deployment and communication patterns in microservice architectures: A systematic literature review. *Journal of Systems and Software, 180*, 111014. https://doi.org/10.1016/j.jss.2021.111014
3. Jeong, B., & Jeong, Y.-S. (2025). Autoscaling techniques in cloud-native computing: A comprehensive survey. *Computer Science Review, 58*, 100791. https://doi.org/10.1016/j.cosrev.2025.100791
4. Khaleq, A. A., & Ra, I. (2021). Intelligent autoscaling of microservices in the cloud for real-time applications. *IEEE Access, 9*, 35464–35476. https://doi.org/10.1109/ACCESS.2021.3059789

5. Ahmad, H., Treude, C., Wagner, M., & Szabo, C. (2025). Towards resource-efficient reactive and proactive auto-scaling for microservice architectures. *Journal of Systems and Software, 225*, 112390. https://doi.org/10.1016/j.jss.2025.112390
6. Siddiqui, H., Khendek, F., & Toeroe, M. (2023). Microservices-based architectures for IoT systems: State-of-the-art review. *Internet of Things, 23*, 100854. https://doi.org/10.1016/j.iot.2023.100854
7. Roda-Sanchez, L., Garrido-Hidalgo, C., Royo, F., et al. (2023). Cloud–edge microservices architecture and service orchestration: An integral solution for a real-world deployment experience. *Internet of Things, 22*, 100777. https://doi.org/10.1016/j.iot.2023.100777
8. El Akhdar, A., Driss, M., Hasan, D., Boulila, W., & Ahmad, J. (2024). Exploring the potential of microservices in Internet of Things: A systematic review of security and prospects. *Sensors, 24*(20), 6771. https://doi.org/10.3390/s24206771
9. Popereshnyak, S., Suprun, O., Suprun, O., & Wieckowski, T. (2018). IoT application testing features based on the modelling network. In *Proceedings of the 2018 XIV-th International Conference on Perspective Technologies and Methods in MEMS Design (MEMSTECH)* (pp. 127–131). IEEE. https://doi.org/10.1109/MEMSTECH.2018.8365717
10. Popereshnyak, S., Novikov, Y., & Zhdanova, Y. (2024). Cryptographic system security approaches by monitoring the random numbers generation. *CEUR Workshop Proceedings, 3826*, 301–309. https://ceur-ws.org/Vol-3826/short21.pdf
11. Popereshnyak, S., Bielikov, M., & Bur, A. (2025). Microservices architecture for building a crypto freelance exchange. In *Proceedings of the Workshop on Intelligent Information Technologies (UkrProg-IIT 2025)* (pp. 75–90). CEUR-WS. https://ceur-ws.org/Vol-4049/paper7.pdf
12. Faseeha, U., Syed, H. J., Samad, F., & Zehra, S. (2025). Observability in microservices: An in-depth exploration of frameworks, challenges, and deployment paradigms. *IEEE Access*. https://doi.org/10.1109/ACCESS.2025.3562125
13. Nicolas-Plata, A., et al. (2024). A service mesh approach to integrate processing patterns into microservices applications. *Cluster Computing*. https://doi.org/10.1007/s10586-024-04342-5
14. Abramov, V., Astafieva, M., Boiko, M., Bodnenko, D., Bushma, A., Vember, V., Hlushak, O., Zhyltsov, O., Ilich, L., Kobets, N., Kovaliuk, T., Kuchakovska, H., Lytvyn, O., Lytvyn, P., Mashkina, I., Morze, N., Nosenko, T., Proshkin, V., Radchenko, S., & Yaskevych, V. (2021). Theoretical and practical aspects of the use of mathematical methods and information technology in education and science. https://doi.org/10.28925/9720213284km
15. Kostiuk, Yu. V., Skladannyi, P. M., Bebeshko, B. T., Khorolska, K. V., Rzaieva, S. L., & Vorokhob, M. V. (2025). Information and communication systems security. [Textbook] Kyiv: Borys Grinchenko Kyiv Metropolitan University.
16. Kostiuk, Yu. V., Skladannyi, P. M., Hulak, H. M., Bebeshko, B. T., Khorolska, K. V., & Rzaieva, S. L. (2025). Information security systems. [Textbook] Kyiv: Borys Grinchenko Kyiv Metropolitan University.
17. Hulak, H. M., Zhyltsov, O. B., Kyrychok, R. V., Korshun, N. V., & Skladannyi, P. M. (2023). Enterprise information and cyber security. [Textbook] Kyiv: Borys Grinchenko Kyiv Metropolitan University.