ДОСЛІДЖЕННЯ МЕХАНІЗМІВ СТІЙКОСТІ ЗНАННЯ-ОРІЄНТОВАНИХ МІКРОСЕРВІСНИХ СИСТЕМ У СЕРЕДОВИЩІ SERVICE MESH

Ільясов М.О., Гамзаєв Р.О.

Інформаційні управляючі системи та технології
ННІ "Комп'ютерних наук та штучного інтелекту",
Україна.

E-mail: iliasov2020ks11@student.karazin.ua
rustam.gamzayev@karazin.ua

Abstract

Resilience mechanisms in microservice architectures are investigated through the lens of service mesh technologies. The study outlines the essence of microservice architecture and the challenges of ensuring its resilience as sys-tem complexity grows. It then introduces the service mesh concept as an infrastructure solution to these challenges and examines two popular implementations, Istio and Linkerd, detailing their features, approaches to resiliency (fault tolerance), and technical aspects. A comparative analysis of Istio and Linkerd is presented based on experi-ments in a Kubernetes cluster with multiple microservices. The results demonstrate the performance and fault-tolerance characteristics of each service mesh. Conclusions highlight the trade-offs between Istio and Linkerd and provide guidance on leveraging service mesh technology to enhance microservice resilience.

Сучасні розподілені знання-орієнтовані [1] системи дедалі частіше реалізуються на основі мікросервісної архітектури, яка дозволяє досягти масштабованості, гнучкості розробки та ізоляції компонентів. Водночас розподілений характер таких систем підвищує їхню вразливість до часткових відмов: кожен мережевий запит між сервісами є потенційною точкою збоїв. У складних розгортаннях, що включають десятки або сотні мікросервісів, забезпечення стійкості та спостережуваності стає критично важливим.

В умовах підвищеної складності виникає потреба у впровадженні типових рішень для балансування навантаження, повторів запитів (retry), таймаутів, circuit breaking, трасування, шифрування трафіку та централізованого моніторингу. Реалізація цих механізмів у кожному сервісі окремо є ресурсозатратною та нестійкою до людського фактору. Саме тому виникла потреба в інфраструктурному рішенні, яке делегує ці функції з прикладного рівня на спеціальний мережевий шар [2-3].

Концепція Service Mesh

Service Mesh - це шаблон інфраструктури, який забезпечує прозоре керування мережевими запитами між сервісами в кластері. У загальному випадку, mesh складається з легковагових проксі (sidecar-контейнерів), що автоматично перехоплюють весь вхідний і вихідний трафік сервісу, та контрольної площини, яка централізовано керує правилами маршрутизації, безпеки й надійності.

Service Mesh надає наступні функціональні переваги: централізоване застосування шаблонів fault tolerance (повтори запитів, таймаути, автоматичне розмикання ланцюга викликів), автоматичне шифрування трафіку між сервісами (mTLS), збирання телеметрії - метрик, логів і трасування, балансування навантаження з урахуванням затримок, а також маршрутизація трафіку згідно з політиками (наприклад, canary-реліз або дублювання запитів на вторинні версії сервісів) [4].

До найпоширеніших реалізацій Service Mesh належать Istio та Linkerd. Обидва інструменти інтегруються з Kubernetes та надають перелічені можливості, однак відрізняються архітектурою, ступенем налаштовуваності, споживанням ресурсів та складністю розгортання [5-7].

Практичну оцінку Service Mesh проведено на основі експериментального середовища, до складу якого входять кілька мікросервісів: api-gateway, order-service, product-service, user-service і база даних

PostgreSQL. Всі сервіси працюють у кластері Kubernetes, в окремому про-сторі імен thesis, за винятком БД, що знаходиться у default (рис. 1).

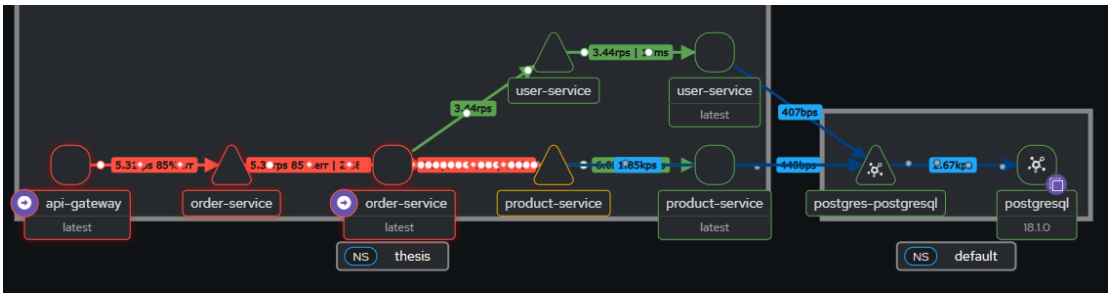

Рис. 1. Архітектура мікросервісної системи та маршрутизація запитів у середовищі Service Mesh

На рисунку зображено міжсервісні взаємодії, що перехоплюються Service Mesh, включаючи кількість запитів на секунду (RPS), середню затримку, статуси та наявність помилок. Архітектура дозволяє аналізувати ефективність механізмів fault tolerance та реакцію системи на збої в окремих компонентах. В експериментах використовувались два режими: лише fault injection (імітація збоїв без засобів стійкості) та активні політики стійкості (DestinationRule та resilient VirtualService).

У процесі дослідження було змодельовано кілька сценаріїв для порівняння поведінки сервісів під управлінням Istio та Linkerd у станах збоїв і навантаження. Запити надсилались на api-gateway з використанням інструменту hey, який імітує клієнтське навантаження. Порівнювались два режими: (1) fault injection без політик витривалості, (2) використання DestinationRule/ServiceProfile з налаштованим circuit breaking і retry.

Таблиця 1. Зведене порівняння середніх показників для режимів mesh

Mesh	Resilience Mode	Avg Latency (s)	Requests/sec	201 Created (кількість / %)	502 Error (кількість / %)
Istio	Fault Injection only	4.09	2.23	99 / 66%	52 / 34%
Istio	Resilience v1	1.61	5.88	60 / 16%	309 / 84%
Istio	Resilience v2	1.03	9.20	287 / 51%	280 / 49%
Linkerd	Fault Injection only	1.14	8.11	320 / 63%	187 / 37%
Linkerd	Resilience enabled	1.60	5.86	355 / 95%	20 / 5%

Аналіз середніх затримок та кількості запитів свідчить про загальні переваги використання механізмів відмовостійкості. Проте середнє значення не повною мірою відображає поведінку системи під навантаженням, тому для глибшого розуміння критичних ситуацій додатково було проаналізовано латентність на рівнях p50, p90 та p99 - тобто для 50%, 90% та 99% найшвидших відповідей відповідно.

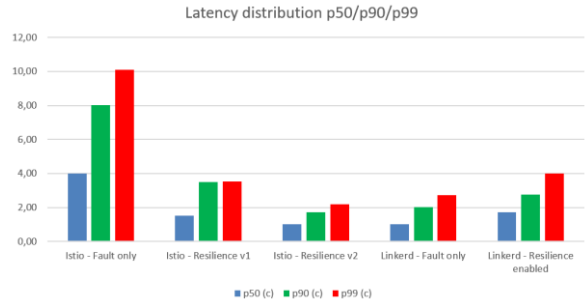

Рис. 2. Розподіл затримки (p50/p90/p99)

З наведеного графіка (рис. 2) видно, що у сценарії fault injection Istio має найбільше значення затримки p99 — понад 10 секунд. Це означає, що 1% запитів виконуються з великою затримкою, що неприйнятно в реальному сервісі. Після активації resilience-механізмів ситуація покращується,

однак Linkerd демонструє стабільніші результати навіть без додаткового налаштування. У режимі resilience enabled Linkerd зберігає затримку p99 на рівні 4 секунд, що є найкращим показником серед усіх конфігурацій.

На підставі проведеного аналізу та експериментального дослідження встановлено, що обидві реалізації Service Mesh - Istio та Linkerd - здатні забезпечувати механізми стійкості мікросервісної архітектури, включно з fault injection, повтори запитів, circuit breaking та телеметрію. Однак результати навантажувального тестування свідчать, що Linkerd демонструє більш стабільну поведінку як у стандартному режимі, так і при активації політик витривалості. Зокрема, Linkerd досягає вищого відсотка успішних відповідей (до 95%), нижчої середньої затримки та менших коливань у p99 затримки, що є критичним показником для користуваць-кого досвіду.

Istio, попри розширений функціонал і гнучкість конфігурацій, вимагає ретельного налаштування політик, аби зменшити кількість помилок 502, і навіть у таких умовах може демонструвати високі пікові затримки. Це робить його менш передбачуваним у сценаріях із підвищеним навантаженням або збоями. З урахуванням результатів, Linkerd є оптимальним вибором для критичних до стабільності мікросервісних систем із невеликими обсягами трафіку або обмеженими ресурсами, тоді як Istio може бути доцільним у великих інфраструктурах із необхідністю детального контролю та розширених функцій безпеки.

Література

1. Гамзаєв Р. О. Знання-орієнтована інформаційна технологія забезпечення мінливості процесів і компонентів у життєвому циклі кіберфізичних систем. Проблеми інформатики та моделювання (ПІМ–2023): Тези 23-ї міжнар. наук.-техн. конф., Харків – Одеса, 20-22 вересня 2023 р. / наук. ред. В. Д. Дмитрієнко; Нац. акад. наук України [та ін.]. – Харків : Контраст, 2023. – С. 39.

2. Mohammad Reza Saleh Sedghpour, Paul Townend (2022), "Service Mesh and eBPF-Powered Microservices: A Survey and Future Directions", Proceedings of the 2022 IEEE Interna-tional Conference on Service-Oriented System Engineering (SOSE), San Fransisco, USA, August 15-18, 2022 p. 176-184. DOI: https://doi.org/10.1109/SOSE55356.2022.00027 (дата звернення: 20.11.2025).

3. Fowler M., Lewis J. Microservices - a definition of this new architectural term [Електронний ресурс], 2014. Режим доступу: https://martinfowler.com/articles/microservices.html. (дата звернення: 20.11.2025).

4. Morgan W. Service mesh: A critical component of the cloud native stack // CNCF Blog. [Електронний ресурс] 2017. Режим доступу: https://www.cncf.io/blog/2017/04/26/service-mesh-critical-component-cloud-native-stack. (дата звернення: 20.11.2025).

5. Istio Documentation: Architecture [Електронний ресурс]. Режим доступу: https://istio.io/latest/docs/ops/deployment/architecture/. (дата звернення: 20.11.2025).

6. Linkerd Documentation: Architecture [Електронний ресурс]. Режим доступу: https://linkerd.io/2-edge/reference/architecture/index.html. (дата звернення: 20.11.2025).

7. Linkerd vs Istio // Buoyant: a service mesh comparison [Електронний ресурс]. Режим доступу: https://www.buoyant.io/linkerd-vs-istio. (дата звернення: 20.11.2025).