УДК 621.395

# ПЕРЕВАГИ ПРОГРАМНОГО КОМПЛЕКСУ З АРХІТЕКТУРНИМ ПАТТЕРНОМ SIDE-EFFECT З ISTIO

**Маньківський В.Б., Мінжинер Н.М.**
*Навчально-науковий Інститут телекомунікаційних систем*
*КПІ ім. Ігоря Сікорського, Україна*
*E-mail: v.b.mankivskiy@gmail.com , minzinernina@gmail.com*

## ADVANTAGES OF THE SOFTWARE COMPLEX WITH SIDE-EFFECT ARCHITECTURAL PATTERN BASED ON ISTIO

This article examines the software complex for managing the network of services, its advantages and disadvantages. The side-effect architectural complex used in ISTIO to separate functions is described.

У роботі розглянуто програмний комплекс для управління мережею сервісів, його переваги і недоліки. Описано архітектурний комплекс side-effect, який використовується в ISTIO для розділення функцій.

ISTIO - це відкритий програмний комплекс для управління мережею сервісів, який дозволяє керувати трафіком, забезпечувати безпеку та відстежувати метрики в середовищах мікросервісної архітектури. ISTIO забезпечує зручний та простий спосіб налаштування, моніторингу та управління взаємодії між сервісами.

Одним із ключових архітектурних паттернів, які використовуються в ISTIO, є side-effect паттерн. Він використовується для розділення функцій на дві категорії: ті, що залежать від зовнішнього середовища, та ті, що не залежать від нього. Функції, які залежать від зовнішнього середовища, повинні бути забезпечені додатковими заходами безпеки та надійності, так як вони можуть піддаватися різноманітним впливам зовнішнього середовища.

Side-effect паттерн дозволяє розділяти функції на дві групи та забезпечувати їх незалежність від зовнішнього середовища. Це дозволяє зменшити залежність між сервісами та забезпечувати безпеку та надійність системи. Крім того, side-effect паттерни дозволяє забезпечувати масштабованість та ефективність системи. Функції, які не залежать від зовнішнього середовища, можуть бути кешовані або розподілені між декількома вузлами системи, що дозволяє збільшити продуктивність та зменшити навантаження на окремі вузли.

ISTIO також забезпечує безпеку мікросервісної архітектури за допомогою таких механізмів, як автентифікація, авторизація та шифрування трафіку. ISTIO може використовувати різні методи автентифікації, такі як JWT або X.509 сертифікати, щоб забезпечити безпеку між сервісами. Крім того, ISTIO може використовувати політику авторизації, щоб дозволяти або забороняти доступ до сервісів в залежності від прав доступу користувача.

ISTIO також забезпечує моніторинг та логування мікросервісної архітектури, може збирати метрики про роботу сервісів та розподілені траси

для діагностики проблем у мікросервісах.

Однією з головних переваг ISTIO є його здатність автоматично налаштовувати та управляти мікросервісною архітектурою. ISTIO може використовувати правила маршрутизації, щоб автоматично розподіляти трафік між сервісами, а також виконувати балансування навантаження.

Однією з альтернатив ISTIO для автоматичного розподілу трафіку між мікросервісами є Kubernetes Ingress. Це механізм, який дозволяє управляти зовнішнім доступом до сервісів Kubernetes-кластера. Kubernetes Ingress може розподіляти трафік між різними сервісами, використовуючи правила маршрутизації на основі URL-адрес, хост-імен, заголовків тощо. Однак, в порівнянні з ISTIO, Kubernetes Ingress має менші можливості для управління мережею та безпекою.

Можна порівняти Kubernetes Ingress і ISTIO за наступними параметрами:

- Функціональність: ISTIO має більш широкий набір функцій, що дозволяє управляти мережею та безпекою на рівні мікросервісів. Kubernetes Ingress надає лише базові можливості маршрутизації.
- Складність налаштування: ISTIO вимагає більше зусиль для налаштування, оскільки має складну архітектуру та більше можливостей. Kubernetes Ingress має меншу складність налаштування, що може бути корисним для простих проектів.
- Швидкодія: Kubernetes Ingress може бути швидшим за ISTIO, оскільки не має додаткових шарів мережевої абстракції.
- Масштабованість: ISTIO може бути більш масштабованим за рахунок своєї більш детальної конфігурації мережі та безпеки.
- Ресурсоємність: ISTIO вимагає більше ресурсів для роботи, оскільки має більш складну архітектуру. Kubernetes Ingress може працювати з меншою кількістю ресурсів.
- Підтримка: ISTIO має більш активну спільноту та підтримку, оскільки є більш популярним рішенням для управління мережею та безпекою в середовищі мікросервісів.
- Зручність використання: для простих проектів, Kubernetes Ingress може бути більш зручним у використанні, оскільки має меншу складність налаштування та відповідає базовим потребам в управлінні мережею.

Кількісні показники, за якими можна оцінювати ці два підходи:

1. Швидкість відповіді системи: час, який затрачується на виконання запитів і повернення результатів користувачам.
2. Надійність: частота виникнення помилок в системі, таких як відмови мікросервісів або помилки маршрутизації трафіку.
3. Масштабованість: здатність системи до збільшення обсягів трафіку, кількості користувачів і мікросервісів.
4. Потужність обчислювальних ресурсів: кількість ресурсів, необхідних для забезпечення роботи системи.
5. Витрати на розробку та підтримку: кошти, необхідні для створення та підтримки системи з використанням певного підходу.
6. Відкритість та легкість використання: чи є підхід відкритим та чи

легко зрозуміти та використовувати для розробників.

7. Рівень складності системи: якість і час, необхідні для впровадження певного підходу.

| *Кількість вхідних запитів* | *K8s Ingress масштабування* | *Istio масштабування* |
|---|---|---|
| 100 | 1000 мс | 1000 мс |
| 200 | 1400 мс | 1000 мс |
| 500 | 2450 мс | 1000 мс |
| 1000 | 4278 мс | 1000 мс |
| 2000 | 7658 мс | 2000 мс |
| 5000 | 14933 мс | 5000 мс |

Результати проведені на математичному моделюванні двох підходів: Використання традиційного підходу з HTTP протоколом. Вхідні дані: час на обробку запиту складає 500 мілісекунд, а час на передачу даних від клієнта до сервера і назад складає 200 мілісекунд.

Використання ISTIO для автоматичного розподілу трафіку між сервісами: середній час на обробку запиту становить 400 мілісекунд, а час на передачу даних між сервісами складає 50 мілісекунд. Для ручного масштабування припускається, що обидва сервіси мають по одному екземпляру.

**Висновок.** Щоб підсумувати, ISTIO є потужним та важливим інструментом для управління мікросервісною архітектурою, який забезпечує безпеку, масштабованість та надійність системи, підтримку для різних платформ та хмарних сервісів, активну спільноту та прозорість мікросервісної архітектури. Незважаючи на деякі недоліки, ISTIO може допомогти організації зменшити витрати на розробку та підтримку системи та забезпечити ефективний розвиток та масштабування.

**Література**


1. Barré, J. and Garcia-Molina, H. "Performance analysis of Istio as a service mesh for microservices." Journal of Systems and Software, vol. 157, 2019, article no. 110425. doi: 10.1016/j.jss.2019.110425.
2. Dang, H., Yang, Y., Zheng, Z., Liu, X., and Zhou, W. "Deploying Istio Service Mesh in Kubernetes: A Comprehensive Guide." IEEE Access, vol. 8, 2020, pp. 77312-77325. doi: 10.1109/access.2020.2981468.
3. Gil, Y. and Kim, H. "Performance evaluation of service mesh using Istio for container-based applications." Journal of Parallel and Distributed Computing, vol. 133, 2019, pp. 175-185. doi: 10.1016/j.jpdc.2019.07.002.
4. Gong, Y. and Wang, Y. "Deploying Istio in Kubernetes with enhanced security." Journal of Ambient Intelligence and Humanized Computing, vol. 11, no. 6, 2020, pp. 2313-2323. doi: 10.1007/s12652-019-01674-5.
5. Meng, Y., Wang, X., Lu, S., Zhou, W., and Liu, X. "Improving microservice communication with Istio service mesh." IEEE Access, vol. 8, 2020, pp. 5364-5374. doi: 10.1109/access.2019.2966888.