УДК 004.42

DOI: 10.31673/2412-9070.2020.063639

О. Б. ПРИДИБАЙЛО[1], ст. викладач, здобувач;
Р. В. ПРИДИБАЙЛО[2], провідний інженер,
[1] Державний університет телекомунікацій, Київ
[2] ТОВ Krusche & Company, Київ

МАРШРУТИЗАЦІЯ ТРАФІКУ ЗА ДОПОМОГОЮ КЕРОВАНОЇ ПЛАТФОРМИ ISTIO

Розглянуто, як за допомогою керованої платформи відбувається керування трафіком. Керована платформа Istio вигідна тим, що вирішує складності, які виникають у додатках, заснованих на наборі невеликих сервісів, кожний з яких працює у власному процесі і комунікує з іншими механізмами, як правило, HTTP, тобто з мікросервісами. Запропонована технологія має істотні переваги, за допомогою яких уможливлюється підімкнення мережі мікросервісів, а також керування ними, забезпечуючи їхню безпеку незалежно від середовища виконання, джерела і розробника. Керована платформа дає змогу керувати вхідним та вихідним обсягами інформації з можливістю відслідковувати час очікування відклику на запит, повторні спроби запитів та балансування навантаження на комп'ютерні системи; забезпечувати спостереження за процесом покрокового виконання програми за системою постійного вистежування, а також гарантувати безпеку користувача. Дуже важливим є те, що платформа Istio функціонує в мережі і реалізує подання інформації про систему користувача та процес збору, агрегації та аналізу цих даних для вдосконалення характеристик і поведінки компонентів системи. Для коректної роботи керованої платформи існує програмна частина архітектури додатку — сервісна сітка, що забезпечує безпечну, швидку та надійну взаємодію між дискретними програмними компонентами — сервісами. Запропоновано кілька схем, які наочно показують архітектуру керованої платформи і загальну архітектуру додатку, який можна реально створити, перевірити та розпочати виконання на хмарних платформах. Запуск керованої платформи відбувається на платформах Google, що є дуже зручним для використання, оскільки не потребує додаткових затрат і має дуже простий код для підімкнення, який також наведено у статті.

Ключові слова: керована платформа; мікросервіси; керування трафіком; сервісна сітка; панель даних; панель керування; маршрутизація трафіку; мережний шлюз; віртуальні сервіси; мікросервісні сітки; кластер; середовище налаштування; інтерфейс користувача; простір імен; моніторинг; метрики.

Вступ

Хмарні платформи надають безліч переваг організаціям, які їх використовують. Проте не можна заперечувати, що впровадження хмари може спричинити деякі труднощі для команд розробників. Розробники мають використовувати мікросервіси для створення перенесення, тоді як оператори керують надзвичайно великими гібридними і мультихмарними розгортаннями. Деякі технології дають можливість під'єднувати, захищати, контролювати і спостерігати за сервісами.

На високому рівні дані технології допомагають зменшити складність цих розгортань і знизити навантаження на команди розробників. Це сервісна сітка з повністю відкритим вихідним кодом, яка прозоро розміщується на наявних розподілених додатках. Це також платформа, зокрема API-інтерфейси, що уможливлюють інтегрування в будь-яку платформу ведення журналів, телеметрію або систему телекомунікацій. Різноманітний набір функцій технологій розгортання додатків дає змогу успішно і ефективно запускати архітектуру розподілених мікросервісів і забезпечує єдиний спосіб захисту, підімкнення та моніторингу мікросервісів.

Основна частина

Однією з технологій, що допомагають розгортанню різноманітних додатків на будь-яких платформах, є технологія Istio.

Istio — це відкрита технологія, що дає розробникам можливість під'єднувати мережі мікросервісів, а також керувати ними, забезпечуючи їхню безпеку — незалежно від платформи, джерела і розробника. Istio це так званий ***Service mesh***, технологія, яка додає рівень абстракції над мережею. Зазвичай перехоплюється вся або частина трафіку в кластері і виробляється певний набір операцій із ним. Наприклад, створюється розумний роутинг або реалізується підхід circuit breaker, можна організовувати «canary deployment», частково перемикаючи трафік на нову версію сервісу, а можна обмежувати зовнішні взаємодії і контролювати всі походи з кластера в зовнішню мережу. Є можливість задавати

policy правила для контролю походів між різними мікросервісами. Нарешті, можна отримати всю карту взаємодії по мережі і зробити уніфікований збір метрик повністю прозоро для додатків.

Керована платформа Istio забезпечується в складі IBM Cloud™ Kubernetes Service. У рамках цієї послуги надаються установка Istio, автоматичні оновлення, керування життєвим циклом компонентів середовища контролю й інтеграція з інструментами ведення журналів і моніторингу платформи [1].

Керована платформа Istio вирішує складності, які виникають у додатках, заснованих на мікросервісах, наприклад такі:

♦ *керування трафіком:* час очікування, повторні спроби, балансування навантаження;

♦ *безпека:* автентифікація й авторизація кінцевого користувача;

♦ *спостереження:* трасування, моніторинг, запис файлів, що містять системну інформацію роботи сервера або комп'ютера, у загальний файл програми.

Керована платформа пропонує спеціалізоване вирішення, повністю відокремлене від сервісів і функціонує завдяки втручанню в мережну взаємодію. А отже, вона реалізує:

♦ *відмовостійкість:* спираючись на код статусу у відповіді, керована платформа здійснює перевірку, визначає збій у запиті та виконує його повторно;

♦ *перенаправлення* на нову версію сервісу фіксовано лише відсотком від кількості запитів;

♦ *моніторинг і метрики*, тобто, за який час сервіс відповів на запит;

♦ *трасування і спостереження:* додає спеціальні заголовки в кожний запит і виконує їх трасування в кластері;

♦ *безпеку:* авторизує користувачів [2].

Перш ніж розглядати керовану платформу, необхідно розібратися з сервісною сіткою Service Mesh, як вона працює та її призначення. Сервісна сітка — це шар архітектури, який відповідає за надійне доставляння запитів через складну мережу мікросервісів. Він забезпечує безпечну, швидку та надійну взаємодію між дискретними програмними компонентами — сервісами. Під час створення додатку для запуску в хмарі (тобто cloud native) саме й необхідна сервісна сітка.

На етапі розширення додатку з моноліту в мікросервісну архітектуру стає все складніше ним керувати та відстежувати. У такому разі настає потреба переходити на такі вирішення, які допоможуть розв'язати частину проблем, пов'язаних із мікросервісами:

- балансування навантаження всередині мікросервісної сітки;
- виявлення сервісів (Service discovery);
- відновлення після збоїв (Failure recovery);
- метрики;
- моніторинг (відстежування).

Також існують більш складніші завдання, які постають перед розробником, які також необхідно вирішувати в найкоротші терміни, а саме:

- тестування;
- канаркові викочування (Canary rollouts);
- контроль доступу (Access control);
- наскрізна автентифікація (end-to-end authentication).

Саме з такими завданнями дає змогу дуже швидко впоратися керована платформа Istio, яка створена компаніями Google, IBM і Lyft (рис. 1).

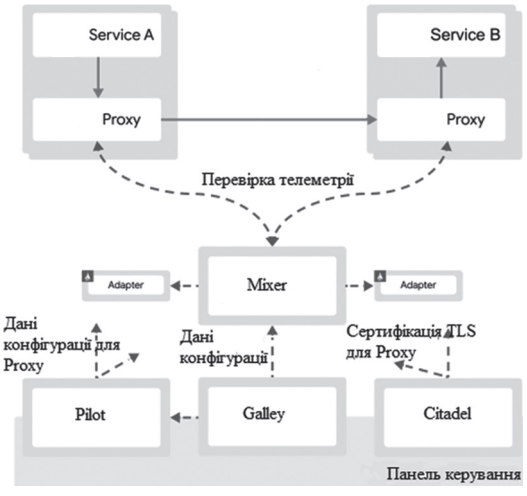

Рис. 1. Загальна схема архітектури Istio

Керовану платформу з сервісною сіткою логічно розділено на *панель даних* та *панель керування*.

• ***Панель даних*** — це набір віддалених комп'ютерів-посередників (проксі), розгорнутих як sidecars. Ці проксі забезпечують і контролюють весь мережний зв'язок між сервісами разом із Mixer центром політик і телеметрії.

• ***Панель керування*** налаштовує проксі для маршрутизації трафіку, а також Mixer для застосування політик і збору телеметрії [3].

Компоненти:

• *Envoy* — це високопродуктивний проксі, розроблений на C++ для передавання всього вхідного і вихідного трафіку для всіх сервісів.

• *Mixer* забезпечує контроль доступу та політики використання в мережі сервісів і збирає дані телеметрії з проксі-сервера Envoy та інших сервісів.

• *Pilot* забезпечує відкриття послуги для Envoy, дає можливості для інтелектуальної маршрутизації трафіку (наприклад, A/B-тести, розгортання) і відмовостійкість (тайм-аути, повторні спроби, автоматичні вимикачі).

• *Citadel* забезпечує надійне сервісне обслуговування і автентифікацію кінцевого користувача.

• *Galley* є компонентом валідації конфігурації Istio. Він відповідає за ізоляцію інших компонентів керованої платформи від конфігурації користувача базової платформи.

Модель маршрутизації і конфігурації трафіку керованої платформи Istio використовує такі ресурси програмного інтерфейсу додатку (API):

♦ віртуальні сервіси — налаштовує правила для маршрутизації трафіку Envoy всередині створеної сервісної сітки (service mesh);

♦ правила призначення — налаштовування політики після застосувань правил маршрутизації у віртуальних сервісах;

♦ мережний шлюз — для налаштування способу розподілу навантаження Envoy (HTTP, TCP або gRPC);

♦ службові записи — для налаштування зовнішніх залежностей сітки.

Для запуску Istio на платформі Google Kubernetes Engine (GKE) спочатку необхідно створити кластер. Для цього потрібно скласти код :

```
gcloud container clusters create <cluster-name> \
 --cluster-version latest \
 --num-nodes 4 \
 --zone <zone> \
 --project <project-id>
```

Після цього користувач отримує ключі доступу (credentials) для роботи з платформою:

```
gcloud container clusters get-credentials <cluster-name> \
  --zone <zone> \
  --project <project-id>
```

А також права адміністратора:

```
kubectl create clusterrolebinding cluster-admin-binding \
  --clusterrole=cluster-admin \
  --user=$(gcloud config get-value core/account)
```

Підготувавши кластер, можна переходити до встановлення компонентів Istio: cd istio-1.3.0 та запуску додатку [4]. Архітектуру додатку зображено на рис. 2, до якої належать:

• інтерфейс користувача (UI — англ. *User interface*) — інтерфейс, що забезпечує передавання інформації між користувачем-людиною і програмно-апаратними компонентами комп'ютерної системи;

• мережний шлюз (Gateway) — апаратний маршрутизатор або програмне забезпечення для сполучення комп'ютерних мереж, що використовують різні протоколи (наприклад, локальної і глобальної);

• об'єкти бази даних — виведення об'єктів із баз даних. Сервіс об'єктів вибирає тільки ті елементи, які є необхідними за поставленим завданням;

• рейтинг об'єктів.

Спочатку для додатку необхідно створити механізм ядра, що забезпечує ізоляцію процесів один від одного, тобто простір імен (namespace):

```
kubectl create ns mesh
kubectl label namespace mesh istio-injection=
enabled
```

Після створення простору імен формується файл objects deployment.yaml із певним змістовним навантаженням [5; 6] (рис. 3).

Рис. 2. Архітектура додатку

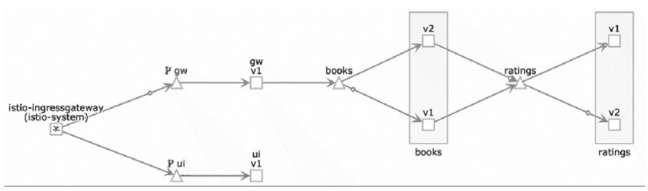

Рис. 3. Топологія додатку

Висновки

Отже, запропонований підхід дозволяє вирішити проблему оптимізації використання апаратного та програмного забезпечення завдяки високій продуктивності та можливості зберігання інформації на хмарних платформах замість жорстких

дисків на комп'ютерах та серверах. Це дає можливість отримати не тільки набір високоефективних інструментів, а також цілу систему додаткових програмних продуктів та сервісів, що допомагатиме у розробленні запуску додатків, використовуючи технологію віртуалізації з перших кроків створення програми.

Віртуальні сервіси, що працюють за певними правилами, використовують ключові елементи компонентів функціональності маршрутизації трафіку Istio. Віртуальний сервіс дозволяє користувачеві налаштовувати сервіси в сітці сервісу Istio, засновуючись на базових можливостях підімкнення та обертання. Кожний віртуальний сервіс є наявним з набору правил маршрутизації, які оцінюються за дотриманням, отже, Istio уможливлюють встановлення зв'язку кожного запиту до віртуальної служби з конкретним реальним пунктом призначення в сітці. Така сітка може вимагати кількох віртуальних сервісів або жодного залежно від конкретного варіанта використання.

Віртуальні сервіси є ключовими моментами в тому, щоб зробити керування трафіком Istio гнучким і потужним, чітко відокремлюючи, куди клієнти відправляють свої запити від головних робочих навантажувачів, які фактично їх реалізують. Віртуальні сервіси також мають чималу здатність задавати різні правила маршрутизації трафіку для відправлення трафіку в ці робочі навантаження.

За допомогою віртуальної служби можна вказати про створення трафіку для одного або кількох імен хостів. Маршрутними отримувачами можуть бути версії одного і того самого сервісу або досконало розроблені сервіси.

Список використаної літератури

*1. **Никульчев Е. В., Паяин С. В., Плужник Е. В.** Динамическое управление трафиком программно-конфигурируемых сетей в облачной инфраструктуре // Вестник РГРТУ. 2013. № 3.*

*2. **Анализ** моделей управления трафиком в сетях АСУП на основе технологии MPLS / В. Т. Еременко, С. В. Еременко, Д. В. Анисимов [и др.] // Информационные системы и технологии. 2013. №1.*

*3. **Алиев Т. И., Муравьева-Витковская Л. А.** Приоритетные стратегии управления трафиком в мультисервисных компьютерных сетях // Известия высш. учеб. заведений. Приборостроение. 2011. Т. 54, № 6.*

*4. **Контроль,** измерение и интеллектуальное управление трафиком / А. А. Алейников, К. З. Билятдинов, А. В. Красов, М. В. Левин // Центр научно-информационных технологий «Астерион». СПб. 2016. 92 с.*

*5. **Exploring** and troubleshooting istio issues / T. Lange, A. Shribman, E. Raichstein, K. Barabash // Publication: SYSTOR '19: Proceedings of the 12th ACM International Conference on Systems and Storage. 2019. С. 196.*

*6. **Rahul** Sharma, Avinash Singh. Getting Started with Istio Service Mesh // Apress, Berkeley, CA. 2020. 321 с.*

О. Б. Придыбайло, Р. В. Придыбайло

МАРШРУТИЗАЦИЯ ТРАФИКА ПРИ ПОМОЩИ УПРАВЛЯЕМОЙ ПЛАТФОРМЫ ISTIO

Рассмотрено, как с помощью управляемой платформы осуществляется управление трафиком. Управляемая платформа Istio выгодна тем, что решает сложности, которые возникают в приложениях, основанных на наборе небольших сервисов, каждый из которых работает в своем процессе и коммуницирует с другими механизмами, как правило, HTTP, иначе говоря, с микросервисами. Предложенная технология обладает большими преимуществами, благодаря которым возможно подключение сети микросервисов, а также управление ими с обеспечением их безопасности независимо от среды выполнения, источника и разработчика. Управляемая платформа позволяет управлять входным и выходным объемами информации с возможностью отслеживания времени ожидания отклика на запрос, повторных попыток запросов и балансировки нагрузки на компьютерные системы, обеспечивать наблюдение за процессом пошагового выполнения программы по системе постоянного отслеживания, а также гарантировать безопасность пользователя. Очень важно, что платформа Istio функционирует в сети и реализует представление информации о системе пользователя и процесс сбора, агрегации и анализа этих данных для совершенствования характеристик и поведения компонентов системы. Для корректной работы управляемой платформы существует программная часть архитектуры приложения — сервисная сетка, обеспечивающая безопасное, быстрое и надежное взаимодействие между дискретными программными компонентами — сервисами. Предложены несколько схем, которые наглядно показывают архитектуру управляемой платформы и общую архитектуру приложения, которое можно реально создать, проверить и запустить на выполнение на облачных платформах. Запуск управляемой платформы происходит на платформах Google, что очень удобно для использования, поскольку не требует дополнительных затрат и имеет очень простой код для подключения, который также приводится в статье.

Ключевые слова: управляемая платформа; микросервисы; управления трафиком; сервисная сетка; панель данных; панель управления; маршрутизация трафика; сетевой шлюз; виртуальные сервисы; микросервисные сетки; кластер; среда настройки; интерфейс; пространство имен; мониторинг; метрики.

O. B. Prydybailo, R. V. Prydybailo

TRAFFIC ROUTING USING THE ISTIO MANAGED PLATFORM

The article examines how traffic is managed through a managed platform. The managed Istio platform is advantageous in that it solves the complexities of applications based on a set of small services, each of which works in its own process and communicates with other mechanisms, usually HTTP, in other words, works with microservices. The technology discussed in this article has great advantages that allow you to connect, manage, and secure your microservices networks regardless of the runtime, source, and developer. The managed platform allows you to manage the input and output of the information so that it can track the response time of the request, retry requests and load balancing on computer systems; provides step-by-step program monitoring, continuous tracking, and user safety. It is very important that the Istio platform operates on the network and implements the submission of information about user systems and the process of collecting, aggregating and analyzing this data to improve the characteristics and behavior of system components. For the proper operation of the managed platform, there is a software part of the application architecture, such as a service grid, which provides safe, fast and reliable interaction between discrete software components — services. There are also several schematics in this article that clearly illustrate the managed platform architecture and the overall application architecture that can be realistically created, tested, and run on cloud platforms. Running a managed platform takes place on Google platforms, which is very easy to use, because it requires no extra cost and has a very simple code to connect. This code is also given in this article.

Keywords: managed platform; microservices; traffic management; service grid; data panel; control panel; traffic routing; network gateway; virtual services; microservice grids; cluster; configuration environment; user interface; namespace; monitoring; metrics.

УДК 004.04

DOI: 10.31673/2412-9070.2020.064043

А. О. БАВЕНКО, студент;
А. Б. КОБА, ст. викладач,
Державний університет телекомунікацій, Київ

ВИКОРИСТАННЯ ТЕХНОЛОГІЇ WebSocket ДЛЯ ПЕРЕДАВАННЯ ДАНИХ У РЕЖИМІ РЕАЛЬНОГО ЧАСУ

Запропоновано застосування сучасної технології для передавання даних у режимі реального часу — WebSocket. Відображено статистику кількості веб-сайтів та використання інтернету населенням. Аргументується, чим використання веб-застосувань ефективне для проектів.

Визначено проблему швидкого передавання даних, а також актуальності інформації. Встановлено частини, на які поділяють веб-розроблення, та чим вони відрізняються. Наведено кілька прикладів.

Досліджено технологію для отримання даних на веб-сторінку AJAX. Як саме реалізується використання цієї технології, можливості налаштування та функціонал, який вона надає. Також розглянуто її переваги щодо отримання даних порівняно зі способом при перезавантаженні сторінки. Проаналізовано недоліки використання цієї технології у разі, коли потрібні найактуальніші дані.

Запропоновано технологію отримання даних на веб-сторінку Server-sent Event, її використання та функціонал, який вона надає. Визначено переваги, які вона дає в отриманні даних порівняно зі способом при використанні технології AJAX. Проаналізовано недоліки використання цієї технології.

Розглянуто технологію WebSocket як найкраще вирішення для проектів, де необхідно у найкоротший термін передавати та отримувати данні. Показано принцип роботи, переваги використання порівняно з технологією Server-sent Event.

Проілюстровано переваги та недоліки кожної технології, за допомогою яких ми можемо реалізувати передавання даних у веб-середовищі. Проаналізовано ефективність використання кожної технології порівняно між собою. Пояснюється чому технологія WebSocket заслуговує уваги як технологія для передавання даних у веб-середовищі і чому її можна назвати технологією для передавання даних у режимі реального часу.

Ключові слова: WEB; HTTP; WebSocket; веб-сервер; браузер; передавання даних; AJAX; Server-Sent Event; актуальність даних.

ВСТУП

У наш час мережа Інтернет набуває стрімкого розвитку. У повсякденні задля власних потреб ним користується більшість людей. На початок 2019 року на планеті налічувалось 4 100 667 287 інтернет-користувачів, що перевищує 53% усього населення Землі. Станом на січень 2019 року в інтернеті було зареєстровано 1,94 млрд веб-сайтів. Це пошукові системи, блоги, електронні бібліотеки, сайти-візитки, соціальні мережі, сайти компаній, поштові сервіси тощо. Із розвитком мобільних технологій та покриттям мобільного інтернету