СТРАТЕГІЇ БАЛАНСУВАННЯ ТРАФІКУ В КЛАСТЕРІ KUBERNETES

Колтаков О.А., Савченко Р.О.

Харківський національний університет радіоелектроніки,
кафедра інфокомунікаційної інженерії ім. В.В. Поповського,
Україна

E-mail: oleksandr.koltakov@nure.ua;
roman.savchenko1@nure.ua

Abstract

Virtualization based on container technology is gaining more and more popularity due to its simplicity and convenience. Containerization technology provides users with more predictable, isolated, and lightweight runtime environments than traditional virtual machines. However, when the level of external traffic increases, the container cluster may experience a decrease in performance. In this work, an analysis of existing load balancing methods was carried out, their differences, shortcomings and application scenarios were given. The relevance of this work is due to the provision of effective load balancing in the container orchestration system, by analyzing existing traffic management models and algorithms for their further improvement.

Використання контроллеру входу. Контролер входу — це компонент, який абстрагує складність маршрутизації трафіку програми в кластері Kubernetes, забезпечуючи міст між службами Kubernetes і зовнішніми службами. Вхідний контролер налаштовується за допомогою об'єктів під назвою Ingress Resources, якими керується через Kubernetes API. Основними функціями вхідного контролера Kubernetes є отримання та балансування навантаження трафіку ззовні Kubernetes до модулів, що працюють у кластері Kubernetes, а також керування вихідним трафіком від служб усередині кластера до служб поза кластером. Переваги:

- Надання безпечного доступу до служб через HTTP або HTTPS.
- Створює маршрути доступу до кількох служб із повним контролем над маршрутизацією служб та умовами зовнішнього доступу.
- Впровадження обмеження швидкості та формування трафіку для контролю потоку вхідних запитів.

Недоліки впровадження даного модуля:

- Контролер Ingress обробляє лише трафік рівня 7.
- Ingress використовується в одному просторі імен. Це означає, що все в просторі імен Kubernetes може посилатися лише на служби в цьому ж просторі імен.

Використання зовнішніх балансувальників. Зовнішній балансувальник навантаження — це хмарна служба, яка розподіляє вхідний трафік між кількома екземплярами додатку, забезпечуючи високу доступність і надійність. Даний балансувальник навантаження керується хмарними провайдерами та використовує такі функції, як автоматичне масштабування, перевірка працездатності та інтегрована безпека. До даних типів балансувальників відносять AWS ELB, Google Cloud Load Balancer або Azure Load Balancer.

Зовнішні балансувальники навантаження підходять для сценаріїв, коли потрібно ефективно обробляти великі обсяги вхідного трафіку. Вони особливо корисні для:

- Мікросервісів з високим трафіком.
- Багатохмарного або гібридного середовища, де трафік необхідно розподіляти між різними компонентами інфраструктури.

До недоліків даного рішення можна віднести необхідність стабільного каналу з'єднання з хмарним провайдером та неповну відкритість структури балансувальника для клієнта.

Оптимізація з'єднання DNS. Ефективне налаштування DNS має ключове значення для зменшення затримок в каналі. Kubernetes надає внутрішні DNS-сервіси, такі як kube-dns або CoreDNS, які керують розпізнаванням імен у кластері. Оптимізація DNS – це налаштування кешування, скорочення часу запитів та балансування їх між серверами DNS, моніторинг DNS та збір показників його продуктивності. Наприклад, налаштування кешування результатів протягом 20 секунд дає змогу зменшити кількість запитів DNS і час відповіді від сервера.

Використання модуля HPA. Horizontal Pod Autoscaling (HPA) — це модуль Kubernetes, який дозволяє масштабувати робочі навантаження на основі попиту. HPA працює, збільшуючи або зменшуючи кількість обчислюваних вузлів в кластері на основі таких показників, як використання ЦП і пам'яті або кількість запитів. Приклад: якщо середнє використання ЦП перевищує 50%, HPA збільшить кількість реплік максимум до 10. Якщо використання ЦП опуститься нижче 50%, масштаб буде зменшено до мінімум 1 репліки.

Перед використанням модуля треба визначити мінімальну та максимальну кількість реплік, щоб запобігти надмірному або недостатньому забезпеченню ресурсами. В процесі використання необхідно відстежувати продуктивність HPA та за потреби коригувати цільові показники та порогові значення. Для цього використовуються інструменти моніторингу Prometheus і Grafana.

HPA використовують в будь-якому середовищі Kubernetes, де робоче навантаження програми значно змінюється з часом. А саме:

- Додатки з різними піками активності впродовж доби, як веб-сервіси.
- Забезпечення високої доступності ресурсів шляхом автоматичного налаштування відповідно до потреб.
- Завдання пакетної обробки, які вимагають додаткових ресурсів у час пікової обробки.

Метод із закріпленням сеансу. Даний метод гарантує що клієнт постійно спрямовується до одного модуля протягом сеансу. Це досягається за допомогою сеансових файлів cookie та IP-адреси клієнта для відстеження сеансу. Kubernetes підтримує закріплені сеанси через **sessionAffinity** атрибут у специфікації служби. Увімкнути дане налаштування можна у службі Kubernetes із назвою **sessionAffinity** та значенням **ClientIP**.

Закріплені сеанси особливо корисні для додатків із збереженням стану, які вимагають збереження даних сеансу користувача за кількома запитами. Наприклад:

- Додатки, які зберігають стан входу користувача та дані сеансу.
- Додатки реального часу, де безперервний сеанс важливий для роботи з користувачем.

Використання проксі-сервера з власним алгоритмом балансування. Щоб реалізувати власні алгоритми балансування навантаження, зазвичай використовують розширені проксі-сервери, такі як Envoy або NGINX. Ці проксі надають гнучкі конфігурації для визначення поведінки балансування.

Власні алгоритми балансування навантаження слід використовувати, якщо стратегії за замовчуванням не відповідають конкретним потребам системи. Вони особливо корисні для:

- Модулів, які потребують низької затримки та високої пропускної здатності.
- У сценаріях, де додатки потребують постійності сеансу або підключень із збереженням стану.
- Сервісів з нерівномірним розподілом навантаження, де певні вузли можуть обробляти більше трафіку.

На графіку на рис.1 представлено залежність затримки (latency) від інтенсивності навантаження для різних алгоритмів під час використання різних алгоритмів балансування трафіку для NGINX та Envoy, де: low є 100 запитів за секунду, medium - 500 запитів за секунду, high - 1000 і більше.

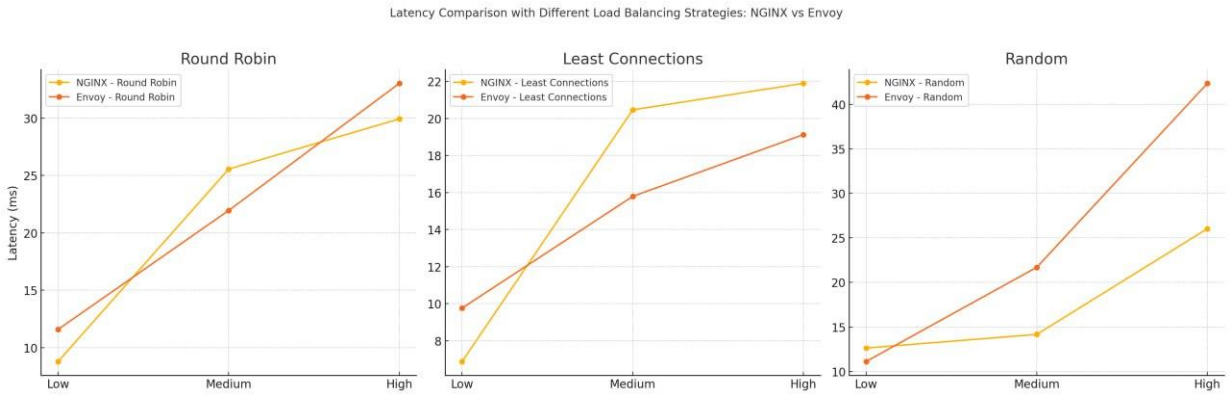

Рис. 1. Залежність затримки від інтенсивності навантаження для різних алгоритмів балансування трафіку

Висновки. Деякі з наведених методів є необхідними для роботи системи і налаштовуються обов'язково, як ,наприклад, контролер входу та DNS. До необов'язкових можна віднести проксі-сервера з власним алгоритмом балансування та хмарні балансувальники. Вибір стратегії керування трафіком залежить від складності та завантаженості системи в цілому. Кожен з них потребує додаткового налаштування моніторингу системи.

Література

1. А.В. Марчук. Контейнеризація в технологіях віртуалізації інформаційнокомунікаційної інфраструктури: навчальний посібник для студентів спеціальності 126 «Інформаційні системи та технології» : підручник. Харків : ХНУРЕ, 2024. 205 с.
URL: https://catalogue.nure.ua/document=266357 (дата звернення: 16.09.2024).

2. Abdulkareem Shafiq D. A., Jhanjhi N., Abdullah A. Load balancing techniques in cloud computing environment: A review. Journal of King Saud University – Computer and Information Sciences. 2022. Vol. 34, no. 7. P. 3910–3933. URL: https://doi.org/10.1016/j.jksuci.2021.02.007 (дата звернення: 02.09.2024).

3. Services, Load Balancing, and Networking kubernetes.io. URL: https://kubernetes.io/docs/concepts/services-networking/ (дата звернення: 16.09.2024).