УДК 004.4

ПІДХІД ДО БАЛАНСУВАННЯ НАВАНТАЖЕННЯ ВЕБ-СЕРВІСІВ У ГЕТЕРОГЕННОМУ КЛАСТЕРІ НА ОСНОВІ DOCKER SWARM

Сегеда С.А., Алєксєєв М.О., Педан С.І.
Навчально-науковий інститут телекомунікаційних систем
КПІ ім. Ігоря Сікорського, Україна
E-mail: ti71segeda@gmail.com, alexeyev@its.kpi.ua, stas.pedan@gmail.com

AN APPROACH TO WEB SERVICES LOAD BALANCING IN A HETEROGENEOUS CLUSTER BASED ON DOCKER SWARM

The work is devoted to the actual problem of effective load distribution of web services in a cluster environment. Currently, there are web services orchestration solutions in a cluster that solve this problem, but they are primarily intended for clusters with stationary architecture or for use in a cloud environment, while using as computing nodes the resources of user devices connected to the enterprise network, has advantages in terms of cost of ownership or data privacy. One of the most promising technologies that could ensure efficient functioning of the cluster in conditions of heterogeneity of working nodes and non-deterministic dynamics of architecture change is Docker Swarm, but it does not provide means of load balancing except for the cloud environment. The paper proposes an approach based on the use of Nginx load balancer to circumvent this limitation.

Робота присвячена актуальній проблемі ефективного розподілу навантаження веб-сервісів у кластерному середовищі. На даний момент існують рішення з оркестрації веб-сервісів у кластері, які вирішують цю проблему, але вони призначені насамперед для кластерів зі стаціонарною архітектурою або для використання у хмарному середовищі, у той час як використання у якості обчислювальних вузлів ресурсів користувальницьких пристроїв, підключених до корпоративної мережі, має переваги по вартості володіння або конфіденційності даних. Однією с найперспективних технологій, яка могла б забезпечити ефективне функціонування кластеру в умовах гетерогенності робочих вузлів та недетермінованої динамічності зміни архітектури це Docker Swarm, але вона не надає засобів балансування навантаження окрім як для хмарного середовища. У роботі для обходу цього обмеження пропонується підхід на основі використання балансувальника навантаження Nginx.

Кількість сервісів в Інтернеті зростає, тому проблема збільшення продуктивності серверів, які їх обслуговують, є актуальною. Цю проблему зазвичай вирішують на основі хмарних рішень, які дозволяють побудувати кластер серверів майже будь якої потужності, який буде автоматично масштабуватись, виходячи з поточних потреб. Але в умовах, коли у наявності компаній є обчислювальні пристрої, підключені до корпоративної мережі, доцільним може бути залучення простоюючих ресурсів таких пристроїв: це не потребує додаткових інвестицій і в організаціях є в наявності багато обладнання

- персональні комп’ютери, пристрої “розумного дому” та інтернету речей які не потребують автономності роботи та простоюють. У попередній роботі [1] було показано, що однією з перспективних технологій, яка могла б забезпечити ефективне функціонування кластеру в умовах гетерогенності зазначених робочих вузлів та очевидної недетермінованої динамічності зміни архітектури кластера, побудованого із залученням таких ресурсів є Docker Swarm, але вона має засоби балансування навантаження тільки для хмарного середовища [2]

Отже, у роботі поставлена задача ефективного балансування навантаження веб-сервісів у кластері під керуванням Docker Swarm, побудованому на основі незадіяних ресурсів пристроїв, підключених до корпоративної мережі.

Типовою архітектурою сучасного серверу може розглядатись наступна [2]: в нас існує кластер (сервер), який реалізує бізнес логіку і обслуговує застосунок. Для того, щоб запити могли потрапити всередину сервера необхідний балансувальник навантаження, наприклад Nginx або Apache HTTP Server [3]. Все, що робить балансувальник у більшості випадків, це перенаправляє запит у вказане місце, тобто в нашому випадку, на сервер з бізнес логікою. Проте, сервер може бути не один і балансувальник може розподілити запити між ними за заданим алгоритмом.

В даній роботі запропоновано використовувати алгоритм “Round Robin”. При цьому кожен сервер має свою “вагу” і запити розподіляються відповідно до неї [3].

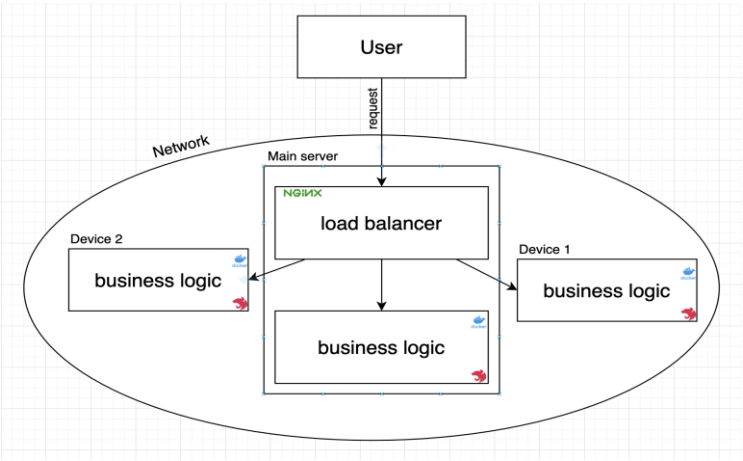

Рис.1. Структурна схема проекту.

Для вирішення поставленої задачі у роботі пропонується архітектура, показана на рис.1. - це мережа, в якій знаходиться головний сервер, в якому в свою чергу знаходиться балансувальник та бізнес логіка, упакована в Docker контейнер. Всі комп’ютери які планується застосувати, також мають встановлений Docker і знаходяться в одній мережі з головним сервером, хоча останнє і не є обов’язковим.

Docker зручно використовувати для контейнеризації застосунку, бо він дозволяє абстрагуватися від операційної системи, на якій буде запущено додаток, а також дозволяє запакувати все в образ, який потім може бути скачаний з репозиторія для їх зберігання, такого як Docker Hub або AWS ECR. При запуску головний сервер публікує посилання на певній адресі, перейшовши за яким починається завантаження скрипта, який після виконання скачує образ зі схожою бізнес логікою, встановить його та запустить. Бізнес логіка в свою

чергу звертається в балансувальник з певним запитом. Для того щоб обробити цей запит у даній роботі використано CGI. CGI має певні переваги над іншими методами для зміни конфігурації під час роботи - це програмне рішення, яке не використовує багато системних ресурсів, не потребує сторонніх технологій, функціоналу яких буде забагато для цих цілей, має багато варіантів реалізації на різних мовах програмування та не потребує багато зусиль в розробці []. Отримуючи запит, ми маємо ip адресу комп'ютера з якого цей запит було отримано і додаємо його до списку серверів, на які запити будуть транслюватися в подальшому.

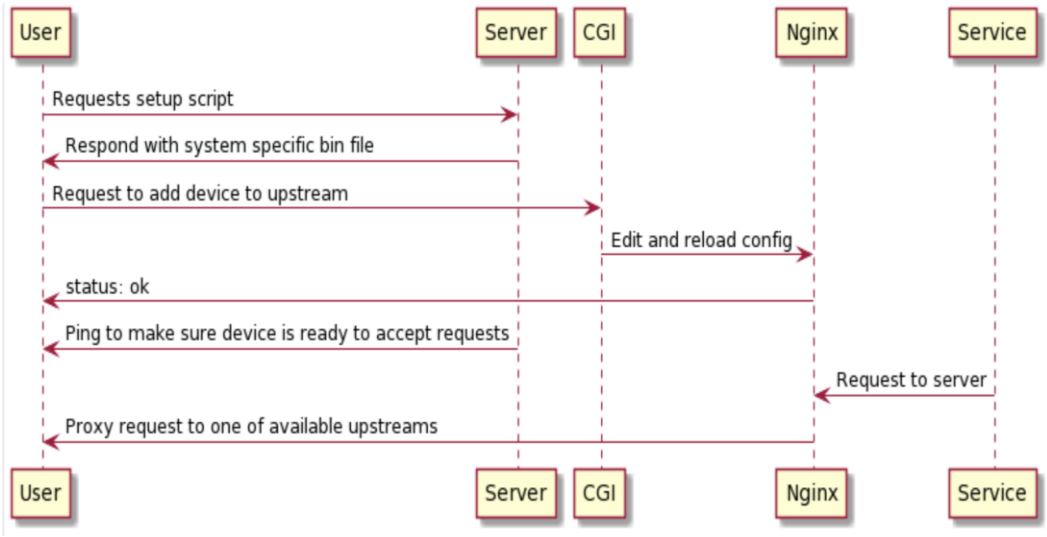

Рис. 2. Діаграма станів проекту.

Після цього необхідно лише перезавантажити конфігурацію і подальші запити будуть розподілятись між серверами.

Висновок. Запропоноване рішення легке у розробці, швидкодіюче, гнучке у налаштуванні та здатне вирішити поставлену задачу – збалансувати навантаження між контейнеризованими сервісами у кластері під керуванням Docker Swarm, побудованому на основі незадіяних ресурсів пристроїв, підключених до корпоративної мережі.

Література

1. Swarm mode key concepts. URL: https://docs.docker.com/engine/swarm/key-concepts/#load-balancing (дата звернення: 20.03.2024).
2. Шелест Є.В., Алєксєєв М.О., Педан С.І. Оглядовий аналіз технологій контейнеризації для використання у гетерогенному середовищі із динамічною архітектурою / ІТ-2024: Збірник матеріалів конференції. К.: КПІ ім. Ігоря Сікорського, 2024.
3. Tony Mauro of F5, "Choosing an NGINX Plus Load-Balancing Technique", October 29, 2015, URL: https://www.nginx.com/blog/choosing-nginx-plus-load-balancing-techniques (дата звернення: 20.03.2024).