цього можна підключати різні системи запису журналів. В контексті даної системи було використано наступні інструменти для забезпечення відмовостійкості: перевірка життєздатності робочих вузлів за допомогою періодичних сигналів, алгоритм консенсусу для вибору лідера, підняття кластерних RabbitMQ, Redis. Відповідно до результатів тестування дані механізми забезпечили стабільність системи за великих навантаження та під час падіння лідера.

Список літератури

1. Таненбаум Э. 1. Распределенные системы. Принципы и парадигмы / Э. Таненбаум, М. ван Стеен. – СПб: Питер, 2003. – 877 с. – (Серия «Классика computer science»).
2. Consumer Acknowledgements and Publisher Confirms [Online]. Available: https://www.rabbitmq.com/confirms.html. [Accessed: Nov. 14, 2021].
3. Shaiekhova, I. F. and Oliinyk, Y. O., “Approach to developing architecture of a heterogeneous multicomputer task planning system” *Applied Questions of Mathematical Modelling*, 4(2.1). [Online]. Available: https://doi.org/10.32782/kntu2618-0340/2021.4.2.1.29 [Accessed: Nov. 14, 2021].
4. IBM. *IBM Db2 documentation: High availability through failover* [Online]. Available: https://www.ibm.com/docs/en/db2/11.5?topic=strategies-failover [Accessed: Nov. 14, 2021].
5. D. Ongaro, “Consensus: bridging theory and practice” [Online]. Available: https://github.com/ongardie/dissertation/blob/master/stanford.pdf [Accessed: Nov. 14, 2021].
6. *Distributed locks with Redis* [Online]. Available: https://redis.io/topics/distlock [Accessed: Nov. 14, 2021].

УДК 004.03

ЄРШОВА Ю.В.

МЕТОД ТА ЗАСІБ ПОБУДОВИ ВЕБ-СЕРВІСУ НА ОСНОВІ SERVERLESS ОБЧИСЛЕНЬ

У даній статті розглянуто застосування архітектурного рішення на основі serverless обчислень. На прикладі реалізованого веб-додатку для контролю фінансів, де традиційна архітектура поєднується з serverless обчисленнями, наведено результати тестів продуктивності та аналіз вартості реалізації. Мета дослідження – відповісти на такі запитання: чи доцільно використовувати архітектурне рішення, де традиційна архітектура поєднується з serverless обчисленнями? Наскільки ефективним є впровадження такого архітектурного рішення з точки зору вартості та продуктивності.

КЛЮЧОВІ СЛОВА: SERVERLESS ОБЧИСЛЕННЯ, МІКРОСЕРВІСНА АРХІТЕКТУРА, ВЕБ-СЕРВІС, ВЕБ-ДОДАТОК, ПРОДУКТИВНІСТЬ, FAAS, PAAS.

The subject of the article is an application of an architectural solution based on serverless computing. The results of performance tests and analysis of the implementation cost are demonstrated are provided with regard to an implemented web application for financial control, where the traditional architecture is combined with serverless computing. The goal of the study is to answer the following questions: Is it feasible to use an architectural solution where the traditional architect is combined with serverless computing? How effective is the implementation of such an architectural solution in terms of cost and performance?

KEYWORDS: SERVERLESS COMPUTING, MICROSERVICE ARCHITECTURE, WEB-SERVICE, WEB-APPLICATION, PERFORMANCE, FAAS, PAAS.

1. Введення

У цій статті розглянуто застосування архітектурного рішення на основі serverless обчислень. На прикладі реалізованого веб-додатку для контролю фінансів, де традиційна архітектура поєднується з

serverless обчисленнями, наведено результати тестів продуктивності та аналіз вартості реалізації. Мета дослідження – відповісти на такі запитання: чи доцільно використовувати архітектурне рішення, де традиційна архітектура поєднується з serverless обчисленнями? Наскільки ефективним є впровадження такого архітектурного рішення з точки зору вартості та продуктивності.

2. Хмарні обчислення

Національний інститут стандартів і технологій США визначає хмарні обчислення як модель, яка забезпечує універсальний, простий доступ до мережі за запитом до пулу обчислювальних ресурсів. Цими обчислювальними ресурсами можуть бути, наприклад, сервери, сховище, мережі, програми або сервіси. Користувач повинен мати можливість надавати нові ресурси з мінімальним зусиллям управління або людської взаємодії з постачальником послуг [1]. Хмарні обчислення дозволяють швидше надавати обчислювальні ресурси, а також абстрагують від користувача безліч завдань управління. Це дозволяє користувачеві зосередитися на впровадженні програмного забезпечення та сервісів без управління базовою інфраструктурою. Наскільки обсяг базової інфраструктури абстрагований та наскільки користувачу надається контроль, залежить від моделі сервісу хмарних обчислень. Усього таких моделей 4: інфраструктура як сервіс (IaaS), контейнери як сервіс (CaaS), платформа як сервіс (PaaS), ПЗ як сервіс (SaaS).

3. Визначення та роль serverless обчислень

Serverless обчислення відносяться до концепції, де користувачу не потрібно керувати будь-якою інфраструктурою сервера. Користувач не запускає сервери, а розгортає код програми на платформі постачальників послуг. Логіка програми виконується, масштабується, а рахунок виставляється на запит без будь-яких поточних витрат. Cloud Native Computing Foundation (CNCF) поділяє serverless архітектуру на дві моделі послуг хмарних обчислень: Бекенд як сервіс (Backend as a Service BaaS) та Функція як сервіс (Function as a Service FaaS). У цій статті розглядається модель FaaS.

Платформи Функція як сервіс (FaaS) запускають невеликі одиниці коду, що виконуються у відповідь на подію. Подією може бути, наприклад, запит HTTP, операція в базі даних або повідомлення в черзі повідомлень. Розробники розгортають код на платформах FaaS, і код виконується за запитом на основі подій. Постачальник послуг піклується про масштабування та інфраструктуру, тому сервіс видається розробникам без сервера [2].

4. Реалізація веб-сервісу на основі Serverless архітектури

Веб-додаток призначений для організації та контролю особистих фінансів: вносити регулярні доходи та витрати, стежити за накопиченнями, рахувати статистику та прогнози. Веб-додаток реалізований на основі мікросервісної архітектури та має 3 мікросервіси:

1. Сервіс Облікового запису (Account service) – реалізує логіку та валідацію збереження доходів, витрат, накопичень та налаштувань облікового запису.
2. Сервіс Статистики (Statistics service) – здійснює розрахунки основних статистичних параметрів облікового запису, зберігає дані у вигляді, зручному для подальшого аналізу.
3. Сервіс Нотифікації (Notification service) – зберігає налаштування повідомлень (частоти нагадувань, періодичність резервного копіювання). За розкладом здійснює розсилку повідомлень.

Сервіс Облікового запису та сервіс Статистики використовується користувачами доволі часто. Впродовж дня користувач може здійснювати безліч транзакцій тобто витрат і фіксує їх у додатку. З кожною такою фіксацією спрацьовує сервіс Статистики для обчислення основних статистичних параметрів облікового запису. Тобто частота використання сервісу Статистики та Облікового запису однакова. Робота сервісу Нотифікацій значно відрізняється від попередніх двох. Цей сервіс викликається один або максимум двічі на день для надсилання повідомлень користувачам. Тобто такий сервіс простоює майже всю добу, займаючи виділені обчислювальні ресурси, за які клієнту доводиться платити. У такій ситуації має сенс трансформувати мікросервіс Нотифікацій в serverless архітектуру.

5. Результати дослідження

Початкові данні:

- Веб-додаток, який включає сервіс, реалізований на платформах PaaS (традиційний підхід) і FaaS (serverless архітектура).
- Apache JMeter – програма з відкритим вихідним кодом, призначена для завантаження веб-додатків та перевірки їхньої продуктивності.
- Тестова машина: процесор: Intel Core i5; пам'ять 8 ГБ.
- Швидкість підключення до Інтернету – 100/100 Мбіт/с.
- Регіон – Київ, Україна

Параметризація:

- Продуктивність вимірювалась шляхом перевірки часу відповіді на успішні HTTP-запити. Тестові запуски тривали 10 хвилин і навантаження повільно збільшувалося до 100 запитів.
- Витрати вимірювалися на підставі виставлених рахунків провайдерами хмарних платформ.

Таблиця 1. Результати тестування навантаження веб-сервісу. Найкращі результати виділені жирним шрифтом.

Характеристика	**Google App Engine**	**Google Cloud Functions**
Кількість запитів	**412 251**	395 234
Середній час відгуку	**80** мс	81 мс
Мінімальний час відгуку	62 мс	**60** мс
Максимальний час відгуку	**1272** мс	2398 мс
Стандартне відхилення	**13** мс	26 мс
Пропускна здатність	**687** запитів/секунду	658 запитів/секунду

Таблиця 2. Порівняння щомісячних витрат платформи PaaS та FaaS. Найкращі результати виділені жирним шрифтом.

Пропускна здатність	Google App Engine	Google Cloud Functions
1 запитів/секунду	43.00 $	**1.72** $
5 запитів/секунду	43.00 $	**8.7** $
10 запитів/секунду	43.00 $	**17.19** $
20 запитів/секунду	43.00 $	**33.50** $
30 запитів/секунду	**43.00** $	49.08 $

Таблиця 3. Щомісячна оцінка вартості сервісу Нотифікацій.

	Google App Engine	Google Cloud Functions
Робоче навантаження	720 годин роботи серверу для розміщення веб-додатку	5 800 000 запитів
Вартість	43.00 $	4.22 $

Інтерпретація отриманих результатів:

У Таблиці 1 представлені результати тесту продуктивності сервісу на різних платформах. Платформи працювали приблизно однаково, і платформа PaaS дала лише трохи кращі результати. Найбільш помітна відмінність між платформами – максимальний час відгуку та стандартне відхилення. Пропускна здатність понад 600 запитів на секунду – це більш ніж достатньо для веб-сервісу, тому загальна продуктивність на обох платформах є

відмінною. На підставі тестів продуктивності можна зробити висновок, що платформа FaaS може використовуватися як серверна частина веб-додатку, але тільки в тих випадках, коли сервіс має низький обсяг трафіку. Саме такій вимогі відповідає сервіс Нотифікацій, який призначений для розсилки повідомлень кілька разів на добу.

У Таблиці 2 порівнюються щомісячні витрати платформ PaaS та FaaS. Виходячи з розрахунків вартості, можна зробити висновок, що платформа FaaS дуже економічно ефективне рішення для веб-сервісів, якщо їх пропускна здатність низька, але при збільшенні обсягів запитів платформа PaaS є дешевшим варіантом. Наприклад, із пропускною здатністю 5 запитів на секунду, близько 13 мільйонів запитів на місяць, Google Cloud Functions більш ніж у п’ять раз дешевший, ніж Google App Engine.

У Таблиці 3 наведено щомісячну оцінку вартості сервісу Нотифікацій. За оцінками, платформа FaaS Google Cloud Functions виходить у 10 разів економніше, ніж платформа PaaS Google App Engine. Хоча FaaS не є найкращим варіантом, якщо розглядати продуктивність, це може бути дуже прибутковим варіантом, якщо розглядати витрати.

Висновки

У дослідженні була розглянута можливість використання архітектурного рішення, де традиційна архітектора поєднується з serverless обчисленнями в якості серверної частини веб-додатку. Дослідницька робота проводилася шляхом вивчення рішень на основі FaaS з точки зору продуктивності та фінансових витрат. Продуктивність серверної частини веб-додатку на основі платформи FaaS порівнювалась з більш традиційною архітектурою – архітектурою мікросервісів, яка була реалізована за допомогою платформи PaaS. Витрати було перевірено шляхом розрахунку вартості інфраструктури, побудованої на основі PaaS та FaaS. Також було проведено аналіз вартості різних постачальників послуг FaaS.

На підставі результатів тестів було виявлено, що продуктивність серверної частини веб-додатку на основі FaaS можна порівняти з бекендом на основі PaaS. Платформи FaaS забезпечують низькі середні затримки, і вони добре масштабуються, щоб відповідати зростаючому навантаженню. Однак, хоч і рідко, але на платформах FaaS іноді бувають відгуки з високою затримкою, і це може бути проблемою для деяких веб-додатків з високим трафіком. Як висновок, FaaS може бути можливим підходом для серверної частини веб-додатків, якщо сервіс має відносно невеликий трафік.

З погляду витрат, платформа FaaS виявилася дуже економічно ефективним рішенням для веб-додатків із низьким обсягом трафіку. Це робить FaaS вигідним рішенням для невеликих комерційних веб-застосунків або деяких компонентів системи, наприклад, як це було реалізовано для веб-додатку Контролю фінансів, для якого сервіс Нотифікацій був реалізований на основі FaaS. Проведені розрахунки вартості показали, що FaaS - це вигідний підхід до створення сервісів з низьким рівнем трафіку. Для веб-додатків з великим трафіком слід проводити порівняння вартості, оскільки інші рішення, такі як платформи PaaS, можуть бути значно економнішими. На закінчення, при прийнятті архітектурного рішення необхідно ретельно розглядати та порівнювати різних хмарних провайдерів і платформ, які вони пропонують та порівнювати їх з точки зору витрат.

Список літератури

1 Мелл П., Гранс Т. The NIST Definition of Cloud Computing. – 2011.
2 CNCF Serverless Working Group. – 2018. Дата доступу: 23.10.2021. https://github.com/cncf/wg-serverless/raw/master/whitepapers/serverless-overview/cncf_serverless_whitepaper_v1.0.pdf.
3 Фаулер М. Patterns of enterprise application architecture // Эддісон-Уэслі Лонгман Паблішинг. – 2002.