pp. 8. *He K, Zhang X, Ren S, Sun J.* Deep residual learning for image recognition. // In: IEEE Conference on Computer Vision and Pattern Recognition, 2016. 770-778 pp.

Надійшла до редколегії 14.05.2021

Ситніков Дмитро Едуардович, кандидат технічних наук, доцент, професор кафедри системотехніки ХНУРЕ. Наукові інтереси: Data Mining and Knowledge Discovery. Адреса: Україна, 61166, м. Харків, пр. Науки 14, тел. (057) 702 10 06.

Андрусенко Юлія Олександрівна, аспірантка кафедри електронних обчислювальних машин ХНУРЕ. Наукові інтереси: методи прогнозування часових рядів. Адреса: Україна, 61166, м. Харків, пр. Науки 14, тел. +38 (063) 407 06 09.

УДК 004.4 DOI: 10.30837/0135-1710.2021.177.047

Н.С. КРАВЕЦЬ

ВИЗНАЧЕННЯ ОБМЕЖЕНЬ РЕАЛІЗАЦІЇ ОБРОБКИ ПОДІЙ В ХМАРНОМУ ДОДАТКУ ЗА ДОПОМОГОЮ БЕЗСЕРВЕРНИХ ОБЧИСЛЕНЬ

У статті розглядаються безсерверні обчислення як нова і перспективна парадигма для розгортання хмарних додатків, зокрема таких що мають подієво-орієнтовану архітектуру. Платформи FaaS провідних постачальників хмарних послуг є найсучаснішою реалізацією безсерверної моделі. На основі порівняння характеристик та переліку джерел оброблюваних подій платформ FaaS від Amazon і Azure визначено переваги та обмеження використання цієї моделі для реалізації високопродуктивних систем, що обробляють події в режимі реального часу.

1. Вступ

У теперішній час розробникам все частіше доводиться вирішувати проблеми, пов'язані із обробкою в реальному часі потоку різноманітних подій, що надходять з різних джерел і вимагають різної обробки. Подібні проблеми покликана вирішувати подієво-орієнтована архітектура. Для вирішення проблем використовуються: асинхронна обробка подій, черги різних типів, механізм підписки на події тощо. Хмарні провайдери пропонують значну кількість рішень на базі своїх платформ, які також припускають використання безсерверних обчислень.

Безсерверні обчислення - це новий підхід, який концептуально ще більше дистанціює програмне забезпечення від інфраструктури, на якій воно виконується. Цей рівень абстракції істотно полегшує життя розробникам, дозволяючи зосередитися тільки на реалізації конкретних функцій. Хоча для реалізації безсерверних обчислень клієнта використовуються фізичні сервери, розробникам не потрібно думати про їх конфігурацію, характеристики та обслуговування. Компанія, що користується послугами безсерверного постачальника, по факту сплачує за використовувані ресурси і не повинна резервувати і оплачувати фіксовану пропускну здатність або кількість серверів, оскільки послуга автоматично масштабується.

У багатьох хмарних провайдерів, що пропонують безсерверні обчислення як послугу, є платформи Function-as-a-Service (FaaS), які дозволяють створювати прості функції, які незалежно запускаються при настанні якоїсь події і виконують одну задачу. За допомогою FaaS розробники можуть створювати модульну архітектуру, роблячи код більш масштабованим, не витрачаючи ресурси на підтримку бекенда. Основними перевагами безсерверних обчислень є низькі експлуатаційні витрати і ефективне управління та використання ресурсів. На теперішній час безсерверні обчислення пропонуються декількома постачальниками загальнодоступних хмарних сервісів. Безсерверні обчислення активно розвиваються, а їх реалізації від різних хмарних провайдерів мають істотні відмінності, аналіз яких є метою даної статті. AWS Microsoft і Azure на даний момент є лідерами ринку хмарних послуг, порівняємо можливості їх платформ FaaS з точки зору обробки подій.

2. Порівняння Azure Fuctions та AWS Lambda

Як правило, безсерверна технологія має наступні характеристики:

- відсутність керування серверами: не потрібно надавати або обслуговувати будь-які сервери, немає доступу до програмного забезпечення або середовища виконання для установки, обслуговування або адміністрування серверів;
- автоматичне масштабування: не потрібно турбуватися про масштабування додатка в разі різкого стрибка навантаження;
- більш «тонкий» білінг: до появи безсерверних хмарних сервісів в результаті автоматичного масштабування автоматично надавалися віртуальні машини, тобто резервувалися ресурси, які клієнт оплачував, навіть якщо вони простоювали; під час використання безсерверних обчислень оплата стягується на основі споживання ресурсів і часу, необхідного для виконання;
- виконання в результаті настання якоїсь події: зазвичай функція виконується після настання події, наприклад, завантаження нового файлу в сховище або спрацьовування таймера;
- висока доступність: висока доступність і відмовостійкість надаються безсерверним додаткам за замовчуванням.

Найчастіше функція як сервіс взаємодіє із масивом хмарних служб для реалізації наступних сценаріїв: створення веб-API, обробка переданих файлів, створення безсерверного робочого процесу, реагування на зміни бази даних, виконання запланованих завдань, створення надійних систем черги повідомлень, аналіз потоків даних Інтернету речей, обробка даних в реальному часі. Однак безсерверні платформи мають власну специфіку, яка залежить від виробника.

Наприклад, і AWS Lambda і Azure Fuctions [1,2] дозволяють використовувати крім базового набору будь-яку додаткову мову програмування за допомогою API Runtime, Custom handler або gRPC language worker. При цьому AWS Lambda включає в себе підтримку довільних бібліотек, проте існують певні обмеження на модель програмування, наприклад, максимальний час виконання, відсутність збереження стану.

У Azure Fuctions від вибору варіанту розміщення залежить реалізація автоматичного масштабування, максимально можлива кількість примірників; перелік ресурсів, доступних для кожного примірника програми-функції; підтримка розширених функцій, таких як підключення до віртуальної мережі Azure. При розгортанні функції AWS Lambda в залежності від профілю робочого навантаження потрібно визначити тільки максимальний розмір виділеної пам'яті, потужність процесора і вартість виконання функції пропорційні виділеній пам'яті. Щоб запустити безсерверну функцію в AWS, її потрібно розгорнути в сервісі AWS Lambda. Microsoft дотримується іншого підходу: поняття моделі програмування Azure Fuctions відокремлено від безсерверної операційної моделі.

AWS Lambda має просту модель програмування: функція отримує об'єкт JSON в якості вхідних даних і може повертати інший JSON в якості вихідних [2]. Тип події визначає схему тих об'єктів, які задокументовані і визначені в мові SDK. Функції Azure мають більш складну модель, засновану на тригерах та прив'язках. Тригер задає подію, яка прослуховується функцією. Функція може мати будь-яку кількість вхідних і вихідних прив'язок для вилучення та/або повернення даних під час обробки [1].

Безсерверні функції - це невеликі блоки коду, що виконують тільки одне завдання. Тому створення великих додатків і систем з цих невеликих шматочків, є нетривіальною проблемою, але деякі рішення існують. Безсерверні обчислення засновані на концепції контейнеризації. Кожного разу, коли функція запускається, система створює контейнер, в якому її можна запустити. У той час, як запуск функції може зайняти всього мікросекунди, запуск контейнера займає 1-2 секунди. Для багатьох додатків це досить швидко, в порівнянні зі створенням віртуальної машини, проте в деяких випадках подібний час затримки може виявитися критичним. І AWS, і Azure мають спеціальні сервіси для оркестровки робочих процесів. Часто функції використовуються в якості кроків в цих робочих процесах, що дозволяє їм залишатися незалежними, але при цьому вирішувати важливі завдання.

Більш докладне порівняння характеристик платформ Faas від Amazon і Azure наведено в табл. 1.

Однією з важливих характеристик послуг FaaS є набір підтримуваних типів подій. У табл. 2 наведено джерела подій для AWS Lambda та Azure Fuctions.

Проблемою при обробці подій може стати некоректно сформована подія або збій одного з екземплярів функції. Для забезпечення надійності обробки повідомлень використовують-

Таблиця 1 -

Порівняння характеристик AWS Lambda та Azure Fuctions

Характеристика	AWS Lambda	Azure Fuctions
Підтримувані мови програмування	C#, Java, Python, Node.js, Ruby, Go, API Runtime	C#, JavaScript (Express.js, Node.js), F#, Java, Python, Type Script, **Custom handler/ gRPC language worker**
Автоматичне масштабування	Функції Lambda не зберігають стан, кількість оброблюваних запитів не обмежена	Визначається планом розміщення Consumption plan, Premium plan, and Dedicated (App Service) plan
Оркестратор	AWS Step Functions	**ASE, Kubernetes , Kubernetes c** Azure Arc
Тригери функцій	Підписка на події, тригери, Lambda API	Тригери і прив'язки
Інтеграція черг повідомлень	+	+
Інструмент моніторингу	Amazon Cloud Watch	Azure Application Insights
Підтримка CLI	+	+
Підтримка PowerShell	+	+
Підтримка шаблона ARM	+	+
Управління	Консоль AWS Lambda, AWS SDK	REST API, Visual Studio
Контекст виконання	У хмарі	Локально /у хмарі

ся черги. У Azure Functions при обробці повідомлення з черги функція може заблокувати некоректну подію, а також гарантує, що обробить подію хоча б один раз. У потоках подій (наприклад, в концентраторах подій Azure) блокування не використовуються. Ці служби налаштовані так, щоб забезпечити високу пропускну здатність, підтримувати кілька груп споживачів і можливість відтворення.

У AWS Lambda при безпосередньому виклику функції потрібно визначити стратегію обробки помилок: повторити спробу, відправити подію в чергу для налагодження або проігнорувати помилку. Якщо функція викликається побічно, можливі наступні сценарії: асинхронний виклик, зіставлення джерел подій. При асинхронному виклику AWS Lambda повторює помилковий запит двічі. Якщо у функції недостатньо можливостей для обробки всіх вхідних запитів, події можуть чекати в черзі протягом декількох годин або днів, щоб бути відправленими в функцію. Якщо використовується зіставлення джерел подій, які зчитуються з потоків, повторюється весь пакет елементів. Повторні помилки блокують обробку порушеного блоку доти, поки помилка не буде усунена або термін дії елементів не закінчиться.

З точки зору реалізації паралельної обробки подій, як AWS, так і служби Azure здатні здійснювати кілька виконань однієї і тієї ж функції одночасно.

Таким чином реалізація обробки подій у хмарі за допомогою безсерверних технологій має наступні переваги: відсутність необхідності керувати інфраструктурою; суттєве скорочення часу випуску додатка; «тонкий» білінг; висока продуктивність, доступність, відмовостійкість та масштабованість. До ризиків слід віднести обмеження на модель програмування; проблеми з оркестрацією, якщо додаток складається із великої кількості незалежних функцій; збільшення часу відгуку внаслідок так званого «холодного» запуску, якщо щільність потоку оброблюваних подій недостатньо велика; потреба у додаткових налаштуваннях задля надійності обробки подій, якщо щільність потоку оброблюваних подій досить велика.

3. Висновки

Для високопродуктивних систем, що обробляють події в режимі реального часу, головні характеристики - це час відгуку на запит і доступність, час відгуку (response time) - те, що бачить клієнт: крім фактичного часу обробки запиту (час обслуговування, service time), він включає затримки при передачі інформації по мережі і затримки повідомлень в черзі.

Таблиця 2 -

Джерела подій, що обробляються AWS Lambda та Azure Fuctions

Джерело оброблюваних подій	**Azure Fuctions**	**AWS Lambda**
Сховища даних	Сховище BLOB-об'єктів, Azure Cosmos DB	Amazon DynamoDB, Amazon Cognito, Amazon Simple Storage Service Batch, Amazon Simple Storage Service
Черги	Queue storage, Service Bus	Amazon Simple Queue Service
Середовище для створення мікросервісних додатків	Dapr	
Брокер подій	Event Grid, Event Hubs	Amazon CloudWatch Events
HTTP-запити та веб-перехоплювачі	HTTP тригери, API, веб-перехоплювачі, Azure SignalR	Amazon API Gateway
Інтернет речей	IoT Hub	AWS IoT, AWS IoT Events
Брокер повідомлень	RabbitMQ, Kafka	Amazon MQ (Apache ActiveMQ), Amazon Kinesis, Amazon Managed Streaming for Apache Kafka, self-managed Apache Kafka, Amazon Connect, Amazon Simple Notification Service, Amazon Simple Email Service
Таймер	Тригер таймера	Можливий запуск функцій за розкладом, використовуючи Amazon CloudWatch
Сервіси моніторингу та управління	Через Event Grid та Event Hubs	Elastic Load Balancing, Amazon CloudWatch, AWS CodeCommit, AWS CodePipeline
Сервіси із голосовим управлінням		Amazon Lex, Alexa
Сервіси доставки даних	Через Event Grid та Event Hubs	Amazon Kinesis Data Firehose, CloudFront
Сервіси роботи з інфраструктурою як з кодом		AWS Cloud Formation, AWS Config

Обмеженням безсерверних обчислень є так званий холодний запуск: запуск «холодних» функцій, тобто тих, які не запускалися протягом тривалого часу, займає додатковий час. Можна вирішувати цю проблему, періодично запускаючи функції, це гарантує, що вони залишаться «теплими», але разом з тим це призводить до додаткових витрат і зводить нанівець одну із основних переваг безсерверної системи - «тонкий» білінг. Час запуску функції можна оптимізувати, але цього не достатньо [3]. Таким чином, проблеми, пов'язані із холодним запуском, означають, що висока продуктивність, яку пропонують безсерверні обчислення, досяжна тільки за певних умов. І AWS Lambda, і Azure дозволяють уникнути холодного запуску в преміальних і виділених планах.

Безсерверні обчислення стають новою і привабливою парадигмою для розгортання хмарних додатків, багато в чому завдяки недавньому переходу від архітектури корпоративних додатків до контейнерів і мікросервісів. Але найчастіше запускати весь додаток у безсерверному середовищі недоцільно або економічно невигідно, тому ця модель передбачає можливість запускати в ній певні частини програми, яким постійно потрібна висока продуктивність і значні обчислювальні ресурси. FaaS - це більше розширення, яке працює разом із контейнерами або віртуальними машинами. Таким чином, потрібно розглядати

безсерверні рішення скоріше як спосіб доповнити різні типи архітектур додатків і подієво-орієнтовану архітектуру зокрема.

Список літератури: 1. *Sreeram P. K.* Azure Serverless Computing Cookbook: Build and monitor Azure applications hosted on serverless architecture using Azure functions. - Packt Publishing Ltd, 2020. 2. *Sbarski P., Kroonenburg S.* Serverless architectures on Aws: with examples using Aws Lambda. - Shelter Island : Manning Publications Company, 2017. - С. 376. 3. *McGrath G., Brenner P. R.* Serverless computing: Design, implementation, and performance //2017 IEEE 37th International Conference on Distributed Computing Systems Workshops (ICDCSW). - IEEE, 2017. - С. 405-410.

Надійшла до редколегії 02.06.2021

Кравець Наталя Сергіївна, кандидат технічних наук, доцент, доцент кафедри програмної інженерії ХНУРЕ. Наукові інтереси: хмарні технології, паралельні обчислення. Адреса: Україна, 61166, м. Харків, пр. Науки, 14, тел. (057) 702 14 46.

УДК 004.62 DOI: 10.30837/0135-1710.2021.177.051

М.С. ТІТОВСЬКОЙ, О.В. ХРЯПКІН

ДОСЛІДЖЕННЯ МЕТОДІВ МІГРАЦІЇ ДАНИХ В СИСТЕМАХ ЕЛЕКТРОННОЇ КОМЕРЦІЇ НА ПРИКЛАДІ CMS MAGENTO

Проведено аналіз особливостей та ризиків міграції даних на прикладі системи електронної комерції Magento. Визначено основні стратегії міграції даних. Для вибору найкращих методу та стратегії міграції даних проведено опитування експертів. За результатами опитування обґрунтовано вибір найкращих методу та стратегії міграції даних для систем електронної комерції різних масштабів.

1. Вступ

Система електронної комерції - це інформаційна система (ІС), яка інтегрує відповідне апаратне і програмне забезпечення для досягнення певних функціональних можливостей [1]. Серед таких можливостей можна виділити основні: прийом і оформлення замовлень по каталогах і прайс-листах через Інтернет на товари різних категорій, зберігання замовлень в єдиній базі даних, реєстрація користувачів, підтримка віддаленого адміністрування; обробка замовлень за стандартною схемою (обробка, поставка, звітно-фінансові документи).

Тема електронної комерції є популярною в Європі, що підтверджують статистичні дані eurostat [2]. Як показано на рис. 1, з 2010 року частка онлайн-продажів щорічно зростає серед покупців всіх вікових груп, що свідчить про існування величезного

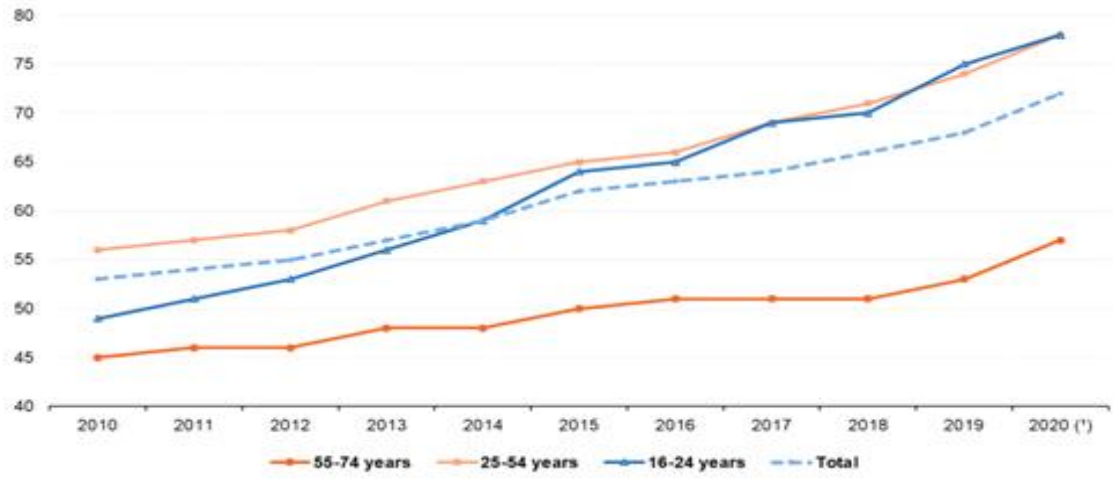

Рис.1. Змінювання частки онлайн-продажів серед користувачів Інтернету різного віку