AZ-204: Develop message-based solutions

Introduction

Azure підтримує два типи queue: service bus та storage queue.
Service Bus підтримує різні механізми: queue, publish/subscribe та інші.

Storage queue в свою чергу може зберігати велику кількість повідомлень, які доступні по HTTP.
Одне повідомлення може бути максимум 64KB. Це можуть бути мільйони повідомлень.

Storage queue часто використовують як беклог для роботи, яку потрібно зробити асинхронно.

Ці два механізми повністю відрізняються функціонально.

Choose Service Bus queues

- Якщо потрібно отримувати повідомлення без polling. Service Bus підтримує кілька протоколів поверх TCP. Це можуть бути long living connections.
- Якщо потрібно реалізувати FIFO підхід
- Якщо потрібно автоматично детектити дублікати в повідомленнях
- Якщо потрібно процесити повідомлення паралельно як окремі streams. Використовується session ID параметр. Споживач може перевіряти статус потоку використовуючи транзакції
- Є потреба в транзакціях та атомарності, коли читається батч повідомлень з черги
- Коли повідомлення можуть бути більше 64KB, але меньше 256KB

Choose Storage queues

- Потрібно зберігати майже 80 гігабіт повідомлень
- Потрібно трекати прогрес обробки повідомлення з черги. Якщо обробник падає, то інший обробник може підхватити це повідомлення та продовжити
- Потрібно серверні логи всіх транзакцій виконаних в чергах

Azure Service Bus

Підтримує різні формати повідомлень: JSON, XML, Apache Avro, Plain Text.

Common scenarios

- Messaging: переміщати дані по системі
- Decouple applications: підвищити reliability та scalability
- Topics and subscriptions: побудувати 1:n зв’язок між сервісами
- Message sessions: побудувати сценарій з впорядкованими або відстроченими повідомленнями

Service Bus tiers

Є стандартний та преміумний пакет. Хоча функціонально вони схожі, є різниця в призначенні та сценаріях.

Premium:

- High throughput
- Predictable performance
- Fixed pricing
- Ability to scale workload up and down
- Message size up to 100MB

Standard:

- Variable throughput
- Variable latency
- Pay as you go variable pricing
- No scaling
- Message size up to 256KB

Advanced features

- Message sessions - повідомлення можна зв’язувати в сесії. В рамках сессії вони будуть впорядковані
- Autoforwarding - можна автоматично переадресувати повідомлення в іншу чергу або топік в рамках одного namespace
- Dead-letter queue - черга для повідомлень, які не змогли опрацювати споживачі. Їх можна аналізувати окремо різними способами
- Scheduled delivery - можна контролювати, коли повідомлення буде відправлено. Це може бути зручно для background jobs
- Message deferral - споживач може відложити повідомлення на потім. Воно залишиться в черзі, але буде помічено відповідно
- Batching - можна відправляти повідомлення в чергу пачками
- Transactions - є так званий execution scope, який дозволяє поєднувати повідомлення в рамках однієї черги
- Filtering and actions - споживачі повідомлень можуть їх фільтрувати
- Autodelete on idle - можна автоматично видаляти чергу, якщо вона не використовується (мінімальний час простою 5 хвилин)
- Duplicate detection - два варіанти: або відправник перетирає старе повідомлення, або черга сама знаходить і видаляє дублікати
- Security protocols - SAS tokens, Role Based Access Control та Managed Identities for Azure resources
- Geo-disaster recovery - повідомлення зберігаються в декількох датацентрах
- Security - AMQP 1.0 та HTTP/Rest протоколи

Базовий протокол для Service Bus - AMQP 1.0.

Queues

Стандартна черга пропонує FIFO підхід відправлення повідомлень одному чи кільком споживачам. Вони конкурують між собою. Лише один споживач може прочитати повідомлення. Повідомлення зберігаються довговічно, то споживачам не обов’язково читати їх паралельно.

Черги дозволяють контролювати навантаження (повідомлення можуть лежати в черзі та чекати свого часу) та зменшують зв’язність елементів системи. Класика для system design.

Receive modes

Є два типи обробки повідомлень: receive and delete OR Peek lock

Receive and delete

Перед відправленням повідомлення споживачу Service Bus помічає його прочитаним. Це більш дешевий варіант, який може приводити до втрати повідомлень, якщо споживач впав з помилкою.

Peek lock

Спочатку споживач читає повідомлення (Service Bus його блокує), а потім споживач робить ще один запит, щоб помітити його як вже оброблене.

Також споживач може відмовитись від повідомлення, тоді інший споживач може його прочитати.

Також можна налаштувати timeout для блокування.

Topics and subscriptions

В звичайній черзі лише один споживач може обробити конкретне повідомлення. В топіку такого обмеження немає. Кожне повідомлення доступне кожному споживачу. Це типова Кафка.

Відправка повідомлень в топік працює так само. А ось зчитування відрізняється. Є один реальний фізичний топік з повідомленнями, але кожен споживач створює свою віртуальну чергу, яка нагадує View в базах даних. Цю віртуальну чергу він читає вже як звичайну.

Rules and actions

Можна створити декілька підписок (віртуальних черг) та налаштувати фільтри для повідомлень. Фільтрувати можна системні та власні параметри. Синтаксис щось по типу SQL expression.

Потім споживач підключається до цієї підписки та читає вже відфільтровані повідомлення.

Metadata and payload

Кожне повідомлення має body та metadata. Останнє - це key-value пари (іншими словами хедери). Іноді достатньо лише хедерів навіть без body.

Body = binary payload. Хедери мають дві секції: системні та кастомні (user defined).

Message routing and correlation

Є базовий перелік системних параметрів: To, ReplyTo, ReplyToSessionId, MessageId, CorrelationId, SessionId.

З їх допомогою можна реалізовувати різні сценарії:

- Request/reply: це по суті паттерн request + response queue в одній черзі. Ти вказуєш в ReplyTo кому повернути відповідь. Також обробник повідомлення копіює MessageId реквесту в CorrelationId відповіді. Можна вказувати декілька отримувачів в ReplyTo
- Multicast request/reply: це сценарій request/reply для топіків. Кожен споживач сам вирішує, чи потрібно обробляти повідомлення, а потім так же само може відправити відповідь. В цьому випадку в ReplyTo також можна вказати ім'я топіку.
- Multiplexing: повідомлення можна об'єднувати в сесії в рамках одного топіку. Далі споживач може вичитувати та блокувати всі повідомлення в рамках однієї сесії.
- Multiplexed request/reply: по суті такий же підхід request/reply, але реквест відправляється з ReplyToSessionId, а відповідь потім потрапляє в конкретну сесію

В рамках одного namespace можна налаштувати auto-forwarding. Routing між namespaces потрібно налаштовувати через Azure LogicApps.

Не потрібно спиратись на To параметр. Споживачі повинні читати повідомлення використовуючи фільтри.

Payload serialization

В самій черзі body - це звичайний бінарний блок. Є параметр ContentType, щоб споживач розумів формат повідомлення. Використовується MIME стандарт для опису типів.

AMQP серіалізує повідомлення в свій власний формат. Інші протоколи мають свої нюанси та обмеження. З AMQP десеріалізація працює максимально просто GetBody<T>();

Microsoft рекомендує не робити свої костилі та використовувати AMQP serialization.

Practice

https://github.com/telamar/AzureLearning/tree/main/src/ServiceBus

Storage Queue

Intro

Можна зберігати мільйони повідомлень. Вони доступні по HTTPs.

Формат лінки:
https://<storage account>.queue.core.windows.net/<queue>

Доступ та менеджмент відбувається через Storage Account.

Ім’я черги повинно містити лише малі літери.

Потрібно вказувати час життя повідомлення (time to live). За замовчуванням це 7 днів.

Practice

https://github.com/telamar/AzureLearning/tree/main/src/StorageQueue