Tìm hiểu và so sánh một số nền tảng hàng đợi thông điệp

Học viên: Tô Quang Minh

Ngày 5 tháng 5 năm 2016

1 Message queue

1.1 Định nghĩa

Message queue là một thành phần công nghệ phần mềm được sử dụng để giao tiếp giữa các tiến trình xử lý hoặc giữa các luồng trong cùng một tiến trình xử lý.

1.2 Giao thức trong message queue

- Message queue cung cấp các giao thức kết nối không đồng bộ (asynchronous communication protocols), có nghĩa là bên gửi (producer) và bên nhận (consumer) không cần phải tương tác với các hàng đợi cùng một lúc. Thông điệp được đặt vào hàng đợi cho đến lúc bên nhận lấy chúng.
- Chuẩn giao thức phổ biến được sử dụng trong mã nguồn mở:
  - Advance Message Queuing Protocol (AMQP)
  - Standard Text Oriented Messaging Protocol (STOMP)
  - MQ Telemetry Transport (MQTT)

1.3 Các hệ thống mã nguồn mở lựa chọn công nghệ message queue

- Apache ActiveMQ
- RabbitMQ
- ZeroMQ
- Kafka
- Sun Open Message Queue
- Apache QPid
- IronMQ
- ...

2 Các giao thức

2.1 AMQP

2.1.1 Định nghĩa

Advanced Message Queuing Protocol (AMQP) là một chuẩn mở định nghĩa một giao thức chung cho các hệ thống trao đổi thông điệp.

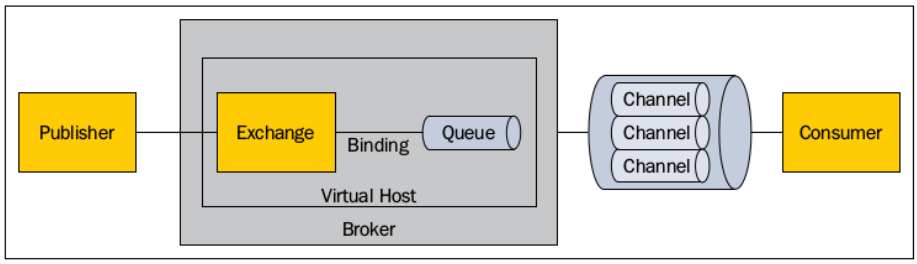
2.1.2 Các khái niệm trong AMQP

- Broker: là một ứng dụng trung gian có thể nhận thông điệp được gửi đến từ bên gửi và phân phối cho bên nhận hoặc các ứng dụng trung gian khác(brokers).
- Virtual host: Đây là một bộ phận ảo trong một ứng dụng trung gian (broker) cho phép sự phân biệt của bên gửi, bên nhận và tất cả các thành phần AMQP mà chúng phụ thuộc.
- Connection: Đây là một kết nối mạng vật lý (TCP) giữa một producer/consumer và broker.
- Channel: Đây là một kết nối logic giữa một producer/consumer và một broker. Nhiều kênh có thể được thành lập trong một kết nối duy nhất. Kênh cho phép cô lập của sự tương tác giữa một consumer cụ thể và broker để không cản trở nhau. Điều này xảy ra mà không có mở các kết nối TCP cá nhân tốn kém. Một kênh có thể đóng khi một có lỗi giao thức xảy ra.
- Exchange: Đây là đích cho tất cả các thông điệp được gửi đến và là thực thể chịu trách nhiệm áp dụng các quy tắc định tuyến đảm bảo cho các thông điệp gửi đến đúng điểm đến. Quy tắc định tuyến gồm: direct (point-to-point), topic (publish-subscribe) và fanout (multicast).
- Queue: Đây là điểm đến cuối cùng của thông điệp, thông điệp trong queue sẵn sàng được phân phối cho các consumers. Một tin nhắn có thể được sao chép và đưa đến nhiều queue nếu như quy tắc định tuyến của exchange quy định.
- Binding: là một kết nối ảo giữa một exchange và một queue cho phép thông điệp từ exchange sang queue để phân phối cho các consumers. Một routing key có thể liên kết với một ràng buộc trong mối quan hệ các quy tắc trao đổi định tuyến.

2.2 STOMP

2.2.1 Định nghĩa

Simple Text-Orientated Messaging Protocol là một giao thức tương thích đơn giản được thiết kế cho thông điệp không đồng bộ giữa clients và các máy chủ trung gian. Nó định nghĩa văn bản dựa trên wire-format cho các thông điệp giữa client - server.



Hình 1: Tổng quan về các khái niệm trong AMQP

3 Một số khái niệm

3.1 Consumers và Producers

4 Apache RabbitMQ

4.1 Định nghĩa

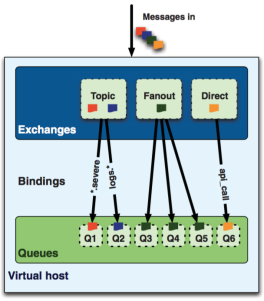
- Rabbit MQ là một message broker mã nguồn mở thực thi chuẩn AMQP.
- Message broker là một mô-đun chương trình trung gian có nhiệm vụ chuyển giao thức thông điệp chính thức từ bên gửi sang thông điệp chính thức của bên nhận.

4.2 Các chuẩn giao thức mà RabbitMQ hỗ trợ

- Hỗ trợ chuẩn giao thức AMQP 0.9.1
- Ngoài ra RabbitMQ cũng hỗ trợ các giao thức của message queue khác qua các plugins:
  - STOMP
  - MQTT
  - AMQP 1.0

4.3 Các định nghĩa

4.3.1 Queues



Hình 2: Amqp stacks: exchanges, Bindings and Queues

Là nơi các message được chứa và chờ được các consumers nhận. Consumers có thể nhận các thông điệp từ queue theo một trong hai cách sau:

- Đăng ký nhận thông qua lệnh basic.consume trong AMQP. Lệnh này sẽ tạo kênh (channel) đặt trong chế độ nhận đến khi nào huỷ đăng ký khỏi hàng đợi (queue). Khi subcribe vào một queue, consumber sẽ nhận tự động nhận các thông điệp nếu có từ queue sau khi xử lý xong thông điệp cuối cùng trong queue. Nên sử dụng basic.consume trong trường hợp consumser xử lý nhiều thông điệp một cách tự động ngay sau khi thông điệp này được chuyển tới queue.
- Đôi khi chỉ cần có một tin nhắc duy nhất từ hàng đợi mà không yêu cầu phải liên tục, sử dụng lệnh basic.get.

Nếu có nhiều consumers đăng ký vào cùng một hàng đợi, RabbitMQ sẽ chỉ chuyển thông điệp đến cho một hàng đợi theo cơ chế round-robin.
Nếu sử dụng cơ chế auto-ack, thì khi consumer nhận được thông điệp từ queue sẽ phải gửi lại cho RabbitMQ một acknowledge message để RabbitMQ biết là thông điệp đó đã được nhận và remove khỏi queue. Trong trường hợp consumer nhận thông điệp và mất kết nối tới RabbitMQ, RabbitMQ sẽ coi thông điệp đó vẫn chưa được gửi đi và sẽ vẫn lưu lại ở queue để phân phát cho consumer mới đăng ký.
Một số thuộc tính hay dùng cho queue:

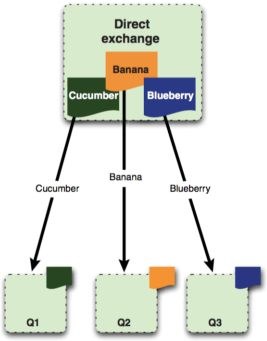
- exclusive: nếu set là true thì queue là private, chỉ được consume bởi một ứng dụng duy nhất. Nên dùng khi muốn hạn chế một queue chỉ được duy nhất một consumer.
- auto-delete: queue sẽ tự động delete khi queue không có consumer nào đăng ký nhận message. Nếu cần một queue tạm để cho duy nhất một consumer có thể kết hợp exclusive và auto-delete.

4.3.2 Exhanges và bindings

Bất cứ khi nào bạn muốn phân phát message tới queue, thì phải gửi message vào một exchange. Một queue sẽ được gắn (binding) vào exchange bởi một routing key. Exchange sẽ dựa vào routing-key để quyết định queue nào sẽ được phân phát message. Các loại exchanges:

4.3.2.1 Direct exchange: sẽ phân phát các thông điệp tới các hàng đợi dựa vào routing key. Routing key là một thuộc tính của thông điệp được gán vào trong header của thông điệp khi producer gửi thông điệp. Thông điệp được gửi đến một hàng đợi mà có routing key trùng với routing key của thông điệp (Hình 3). Trường hợp sử dụng:

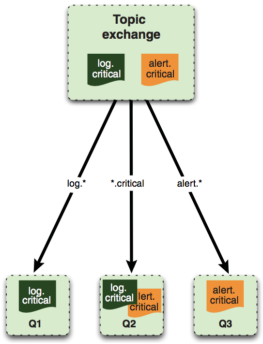
- Direct exchange được sử dụng trong định tuyến đơn hướng thông điệp.
- Direct exchange được sử dụng để phân phối nhiệm vụ giữa nhiều workers theo kiểu round robin (luân chuyển). Việc cân bằng tải này được hiểu giữa các consumers chứ không phải giữa các hàng đợi.



Hình 3: Direct exchange message flow

4.3.2.2 Topic exchange: sẽ định tuyến thông điệp đến hàng đợi có routing key trùng hoặc gần đúng mẫu routing key của thông điệp. Do đó, thông điệp được gửi đến một hoặc nhiều hàng đợi có routing key trùng hoặc gần đúng mẫu. Ví dụ về routing key: *agreements.us*, *agreements.eu.stockholm* hoặc theo mẫu *agreements.*.*.b.** - chỉ chấp nhận routing key với từ bắt đầu là **agreements** và từ thứ 4 là **b**; *agreements.eu.berlin.#* - chấp nhận routing key bắt đầu bằng **agreements.eu.berlin** (Hình 4). Bất cứ khi nào một vấn đề liên quan đến nhiều consumers/appications có chọn lọc loại thông điệp muốn nhận, topic exchange là lựa chọn phù hợp. Một số ca sử dụng thường gặp:

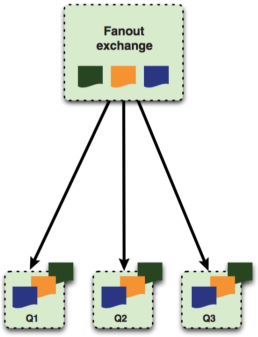
- Phân phối các dữ liệu có liên quan đến vị trí địa lý cụ thể, ví dụ, các điểm bán hàng.
- Cập nhật tin tức liên quan đến việc phân loại và gắn nhãn (ví dụ, chỉ cho một môn thể thao hoặc nhóm cụ thể).
- Cập nhật giá cổ phiếu (và cập nhật về các loại dữ liệu tài chính).



Hình 4: Topic exchange message flow

4.3.2.3 Fanout exchange: sao chép và phân phát tất cả thông điệp đến tất cả hàng đợi được gán vào exchange mà không cần quan tâm đến routing key như direct hoặc topic exchange. Fanout exchange được sử dụng trong trường hợp cùng một thông điệp cần gửi đến một hoặc nhiều hàng đợi với nhiều consumers xử lý thông điệp theo những cách hác nhau (Hình 5). Ca sử dụng thường gặp:

- Các trang web tin tức thể thao có thể sử dụng để phân phối bản cập nhật điểm cho các khách hàng điện thoại di động trong thời gian gần thực.
- Phân phát thông tin cập nhật cổ phiếu cho các ứng dụng như bảng giá, mobile đặt lệnh, vẽ biểu đồ mã chứng khoán thời gian thực.



Hình 5: Fanout exchange message flow

5 Apache ActiveMQ

5.1 Định nghĩa

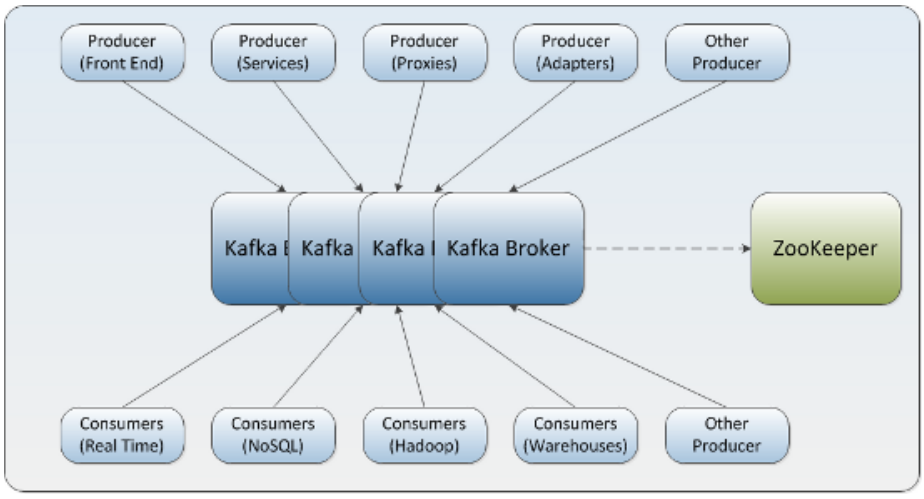
5.2 ZeroMQ

5.3 Apache Kafka

5.3.1 Giới thiệu

Kafka là hệ thống log message hỗ trợ phân tán, phân vùng và nhân bản; hệ thống thông điệp dạng publish-subcribe. Kafka được thiết kế với những tính chất:

- *Persistent messaging*: Hạn chế mất mát thông tin với dữ liệu lớn, độ phức tạp in/out là hằng số (O(1)) kể cả với dữ liệu lớn (TB). Với Kafka thông điệp được lưu trữ trên ổ cứng cũng như được nhân rộng trong cluster để ngăn ngừa việc mất dữ liệu.
- *High throughput*: Hỗ trợ vài triệu messages/s đối với những thiết bị phần cứng thông thường
- *Distributed*: Message được phân vùng trên những Kafka server khác nhau và việc consume cũng có thể phân tán trên nhóm consumer mà vẫn đảm bảo được tính thứ tự trên mỗi phân vùng.
- *Multiple client support*: Tương thích với nhiều client trên các nền tảng khác nhau (java, php, .net, python....)
- *Real time*: Consumer ngay lập tức có thể nhận được message tạo bởi producer



Hình 6: Apache Kafka messaging system

Các thuật ngữ cơ bản:

- *Topic*: Kafka lưu trữ message trong các nhóm (categories).
- *Producer*: tiến trình đẩy message vào Kafka topic
- *Consumer*: tiến trình nhận message từ kafka topic mà nó subscribe
- *Broker*: Kafka chạy trên một nhóm các server, mỗi server trong đó gọi là một broker

5.3.2 Các thuật ngữ cơ bản: