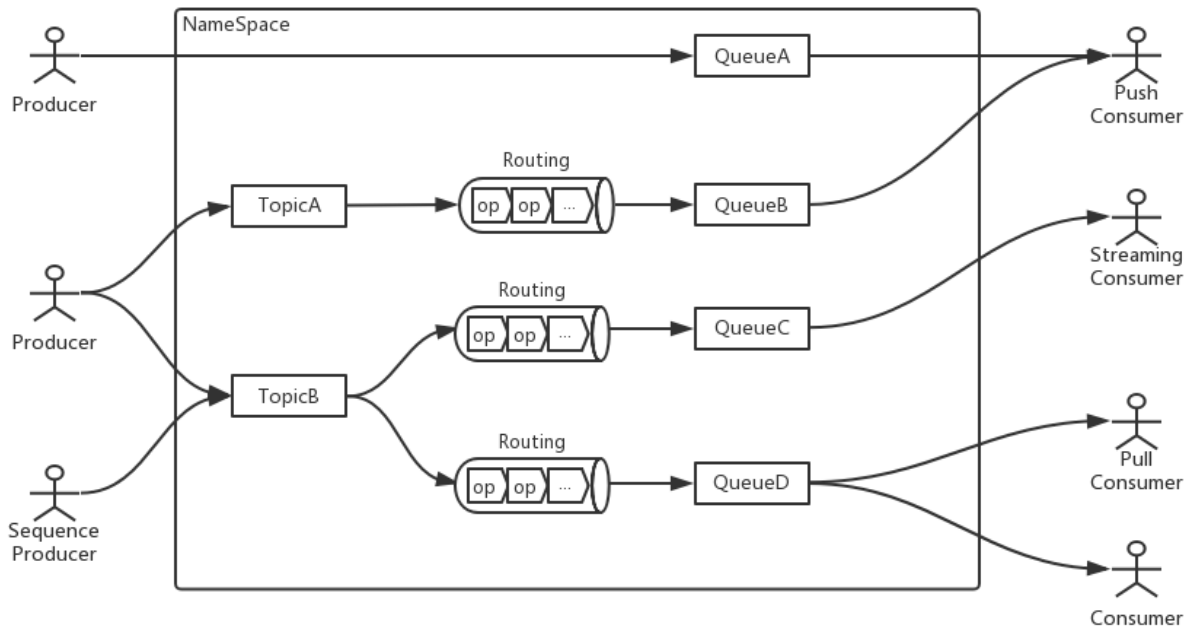
OpenMessaging Domain Architecture

Overview



Above is the domain architecture of Open Messaging, see JavaDoc for details.

OpenMessaging has released first alpha version recently, Apache RocketMQ has provided a partial implementation for OpenMessaging-0.1.0-alpha.

Namespace

Namespace likes a cgroup namespace, to create a isolated space with security guarantee. Each namespace has its own set of producer, consumer, topic, queue and so on. OpenMessaging uses **MessagingAccessPoint** to access/read/write the **resources** of a specified Namespace.

Producer

OpenMessaging defines two kinds of Producer: **Producer** and **SequenceProducer**.

- **Producer**, provides various send methods to send a message to a specified destination, **Topic** or **Queue**. Three ways are supported: synchronous, asynchronous and oneway.
- **SequenceProducer**, focuses on speed, the implementation can adopt the batch way, send many messages and then commit at once.

Consumer

OpenMessaging defines two kinds of Consumer: **PullConsumer**, **PushConsumer** and **StreamingConsumer**. Each consumer only supports consume messages from the **Queue**.

- **PullConsumer**, polls messages from the specified queue, supports submit the consume result by acknowledgement at any time. One PullConsumer only can pull messages from one fixed queue.
- **PushConsumer**, receives messages from multiple queues, these messages are pushed from the MOM server. PushConsumer can attach to multiple queues with separate **MessageListener** and submit consume result through **ReceivedMessageContext** at any time.
- **StreamingConsumer**, a brand-new consumer type, a stream-oriented consumer, to integrate messaging system with Streaming/BigData related platforms easily. StreamingConsumer supports consume messages from partitions of a specified queue like a iterator.

Topic Queue and Routing

These three concepts are closely connected, although **Topic** and **Queue** have different responsibilities, they are confusing.

Topic

Topic, the carrier of origin messages, is responsibility for holding messages. The distribution and sequential of messages in Topic is undefined.

Routing

The messages in Topic are original, waiting for processing, which always can't arouse the interests of consumers. In a word, the messages in Topic is producer-oriented, not consumer--oriented.

So the Routing is in charge of processing the original messages in Topic, and routing to Queue. Each Routing has a **operator pipeline**, consists of a series of operators. The messages will flow through the operator pipeline from Topic and Queue.

A **operator** is used to handle the flowing messages in Routing. There are many kinds of operator, expression operator, deduplicator operator, joiner operator, filter operator, rpc operator, and so on.

Queue

The messages have been routed to **Queue**, now can be consumed by consumers.
It is noteworthy that a Queue should be divided into **partitions**, a message will be routed to a specified partition by MessageHeader#SHARDING_KEY.

Queue also accepts messages from Producer directly, sometimes, we want to the shortest path from Producer to Consumer, for performance.

Topic vs. Queue

- Both Topic and Queue are the carrier of messages.
- Topic is producer-oriented, while Queue is consumer-oriented.
- Messages in Topic are from Producer, while messages in Queue is from Topic or Producer.
- Queue is divided into partitions, while the arch of Topic is undefined.
- Queue is a little sub set of a Topic in most cases.
- Create or destroy a Queue is easy and producer irrelevant.