Requirements of a Distributed Messaging Queue's Design

Learn about the requirements of designing a distributed messaging queue using a strawman solution.

We'll cover the following

- Requirements
  - Functional requirements
  - Non-functional requirements
- Single-server messaging queue
- Building blocks we will use

Requirements

In a **distributed messaging queue**, data resides on several machines. Our aim is to design a distributed messaging queue that has the following functional and non-functional requirements.

Functional requirements

Listed below are the actions that a client should be able to perform:

- **Queue creation:** The client should be able to create a queue and set some parameters—for example, queue name, queue size, and **maximum message size.**↴
- **Send message:** Producer entities should be able to send messages to a queue that's intended for them.

- **Receive message:** Consumer entities should be able to receive messages from their respective queues.
- **Delete message:** The consumer processes should be able to delete a message from the queue after a successful processing of the message.
- **Queue deletion:** Clients should be able to delete a specific queue.

Non-functional requirements

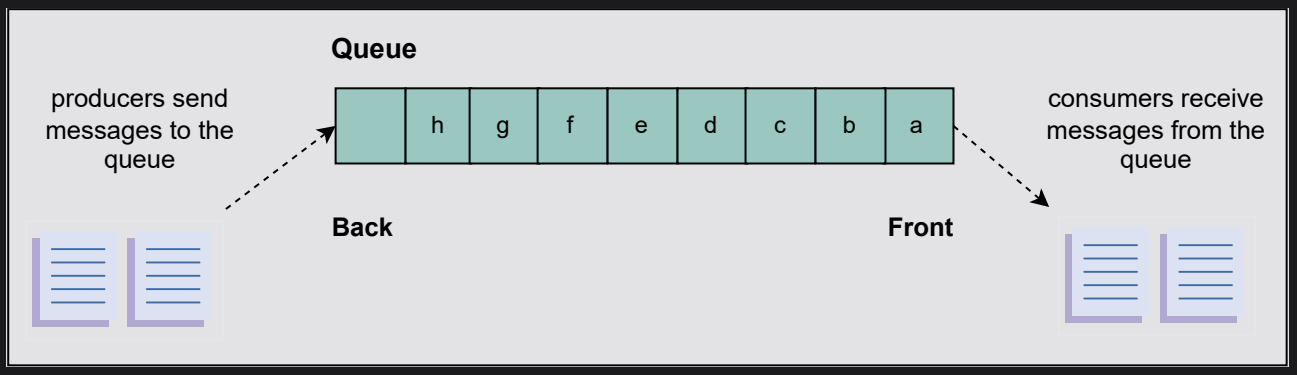
Our design of a distributed messaging queue should adhere to the following non-functional requirements:

- **Durability**: The data received by the system should be durable and shouldn't be lost. Producers and consumers can fail independently, and a queue with data durability is critical to make the whole system work, because other entities are relying on the queue.
- **Scalability**: The system needs to be scalable and capable of handling the increased load, queues, producers, consumers, and the number of messages. Similarly, when the load reduces, the system should be able to shrink the resources accordingly.
- **Availability**: The system should be highly available for receiving and sending messages. It should continue operating uninterrupted, even after the failure of one or more of its components.
- **Performance:** The system should provide high throughput and low latency.

Single-server messaging queue

Before we embark on our journey to map out the design of a distributed messaging queue, we should recall how queues are used within a single server where the producer and consumer processes are also on the same node. A producer or consumer can access a single-server queue by acquiring the locking mechanism to avoid data inconsistency. The queue is considered a critical section where only one entity, either the producer or consumer, can access the data at a time.

However, several aspects restrain us from using the single-server messaging queue in today's distributed systems paradigm. For example, it becomes unavailable to cooperating processes, producers and consumers, in the event of hardware or network failures. Moreover, performance takes a major hit as contention on the lock increases. Furthermore, it is neither scalable nor durable.



Multiple producers and consumers interact via a single messaging queue

Point to Ponder

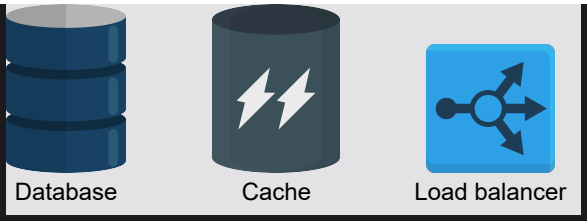
Question

Can we extend the design of a single-server messaging queue to a distributed messaging queue?

Show Answer

Building blocks we will use

The design of a distributed messaging queue utilizes the following building blocks:



The building blocks used to design a distributed messaging queue

- **Database(s)** will be required to store the metadata of queues and users.
- **Caches** are important to keep frequently accessed data, whether it be data pertaining to users or queues metadata.
- **Load balancers** are used to direct incoming requests to servers where the metadata is stored.

In our discussion on messaging queues, we focused on their functional and non-functional requirements. Before moving on to the process of designing a distributed messaging queue, it's essential for us to discuss some key considerations and challenges that may affect the design.

Back

Completed

Next