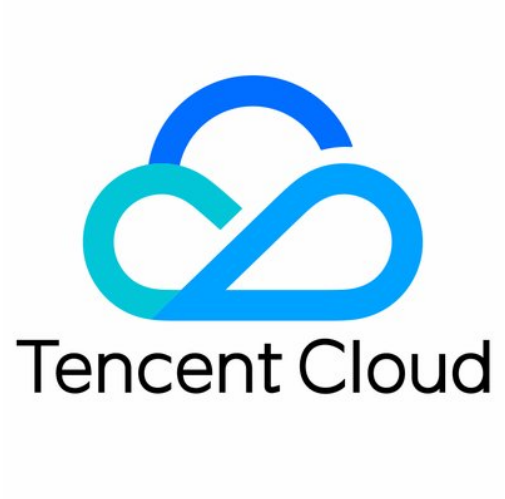
Cloud Message Queue

Product Introduction

Product Documentation



Contents

Product Introduction
Overview

Last updated：2020-12-18 13:02:01

CMQ Overview

Cloud Message Queue (CMQ) is a distributed message queue system based on Tencent's proprietary message engine. It ensures strong message consistency by leveraging Tencent's proprietary distributed Raft algorithm and offers high message reliability by storing three synchronous copies of messages. Specifically, with its advantages of high reliability, availability, performance, and scalability, it provides a rich set of services such as message queue, publish/subscribe model, message rewind, delayed message sending, sequential message sending, and message track. Over its development and iteration in more than seven years, CMQ asynchronously serves Tencent's major businesses such as WeChat, WeBank, QQ Show, and Mobile QQ.

CMQ has been put into formal commercial use and provides a highly available cloud message service in multiple Tencent Cloud regions around the globe. Its data center hardware is constructed in compliance with the high standards of Tencent Cloud IDCs. In a single region, CMQ is deployed across multiple data centers, so that it can still provide message services for applications even if a data center becomes entirely unavailable. In addition, it is also deployed in the Shenzhen and Shanghai Finance Zones to provide finance-grade high-reliability message queue services.

Currently, CMQ supports connection over HTTP, HTTPS, and TCP and can be integrated through SDKs for various programming languages such as PHP, Java, and Python.

Connection Method	HTTP/HTTPS Connection	TCP Connection
Application scenarios	It provides sync HTTP/HTTPS-based connection and can be simply and easily integrated through RESTful APIs and SDKs for multiple programming languages.	It supports sync/async TCP-based connection and SDKs for multiple programming languages, which improves the producer and consumer efficiency and increases the performance of the message queue service.
Strengths	It features unlimited message retention, finance-grade horizontal scalability and high reliability, and messages can be stored in disks in real time.	It features unlimited message retention, finance-grade horizontal scalability and high reliability, and messages can be stored in disks in real time. In addition, messages can be received and sent in an async non-blocking TCP method, which improves the efficiency.

Note：

- TCP-based connection to CMQ is currently in beta test. You can submit a ticket for application.
- CMQ supports deployment in private cloud. You can submit a ticket for application.

Use Case Overview

CMQ is recommended in scenarios where async communication is required; for example:

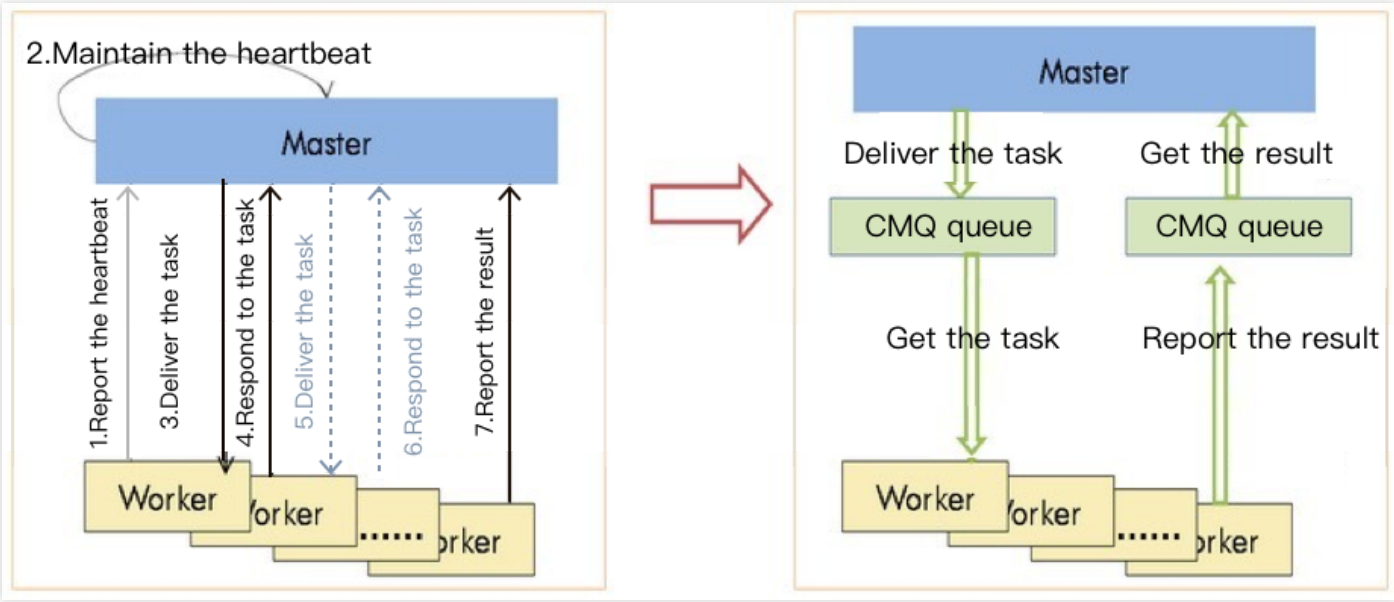
- Your application requires guaranteed message transfer reliability. When a message is sent, even if the recipient is unavailable due to problems such as power failure, server downtime, and CPU overload, the recipient can still receive it once becoming available. Traditional message queues store messages in the memory and therefore cannot achieve this effect. In the distributed message queues of CMQ, messages will be persistently stored until the recipient successfully gets them.

- Your business needs to run properly as the access traffic and number of messages retained in the queue soar. In traditional message queues, messages are stored in the local memory, and since the processing capabilities and memory capacity of a single server are limited, scalability is unavailable. In contrast, the distributed architecture of CMQ guarantees easy scalability, and more importantly, scaling is completely imperceptible to users.

- Two services need to communicate with each other when their networks cannot interconnect or the application route information (such as IP and port) is variable. For example, if two Tencent Cloud services want communication without knowing each other's address, they can agree on the queue name so that one service can send messages to the queue and the other can receive messages from it.

- The communication between system components or applications is frequent, they need to maintain the network connections to each other, and there are multiple types of communicated content. In this case, the system design will be very complicated if a traditional architecture is used. For example, when a central processing service needs to assign tasks to multiple task processing services (similar to the master/worker mode), the master needs to maintain connections with all workers and detect whether the workers start processing the tasks so as to determine whether task reassignment is required. At the same time, workers need to report the task results to the master. Maintaining such a system will result in complicated design and high implementation difficulty and costs. As shown below, the system can be made much simpler and more efficient when CMQ is used to reduce the coupling

between the master and workers.



- The coupling between system components or applications is tight, and you want to reduce the coupling degree especially when your control over the dependent components is weak. For example, the CGI of your business receives contents submitted by users, stores some data in its own system, and forwards the processed data to other business applications (such as data analysis and storage systems). In traditional solutions, services connect to each other through sockets, and if the IP or port of the recipient changes or the recipient is replaced, the data sender needs to modify the relevant information accordingly. In contrast, if CMQ is used, the recipient and sender are imperceptible to each other's information, which greatly reduces the coupling degree.

Product Features

Last updated：2020-05-12 21:11:43

Async Communication Protocol

The message sender can immediately get the returned result after sending a message to CMQ without the need to wait for the recipient's response. The message will be saved in the queue until fetched out by the recipient. The sending and processing of the message are completely async.

Improved Reliability

In traditional modes, a message request may fail due to long waits; however, if the recipient is unavailable when a message is sent in CMQ, CMQ will retain the message until it is successfully delivered.

Process Decoupling

CMQ helps reduce the degree of coupling between two processes. As long as the message format stays unchanged, no changes will be made to the sender even if the recipient's API, location, or configuration changes. Moreover, the message sender does not have to know who the recipient is, making the system design clearer; in contrast, if a remote procedure call (RPC) or socket connection is used between processes, when one party's API, IP, or port changes, the other party must modify the request configuration.

Message Routing

A direct connection is not required between the sender and the recipient, as CMQ guarantees that the message can be routed from the former to the latter. Message routing is even available for two services that are not easily interconnectable.

Multi-Device Interconnection

Messages can be sent or received among multiple parts in a system, and CMQ controls the availability of messages through message status.

Diversity

Each queue can be configured independently and not all of them must be identical. Queues in different business scenarios can be customized. For example, if a queue has a longer message processing time, it can be optimized for queue properties.

SCF Trigger

The CMQ topic model can pass messages to SCF and invoke functions by using the message content and relevant information as parameters.

SCF product documentation >>

Strengths

Last updated：2021-08-17 11:25:32

Advantages over RabbitMQ

- **QPS:** CMQ features higher QPS. When high reliability is ensured and the same physical device is used, the throughput of CMQ is four times higher than that of RabbitMQ, with a single CMQ cluster providing over 100,000 QPS.

- **Message rewind:** RabbitMQ does not support message rewind, while CMQ allows you to rewind messages by time; for example, messages can be consumed again from a specified time point on the day before. In a typical scenario where a consumer needs to analyze orders, if the messages consumed today have all become invalid due to problems such as program logic errors or dependent system faults, then the messages need to be consumed again from 00:00 yesterday on, and message rewind will be much helpful in this case.

- **Consistency algorithm:** CMQ and RabbitMQ both support hot backup with multiple servers to improve availability. CMQ implements this feature based on the Raft algorithm which is simpler and easier to be maintained. RabbitMQ uses its proprietary Guaranteed Multicast (GM) algorithm which is difficult to learn.

- **OPS difficulty:** OPS in RabbitMQ is more difficult, as it is developed in Erlang, a less popular programming language that has higher learning costs.

Advantages over RocketMQ

- **Data loss**: in extreme cases, data may be lost in RocketMQ. Because RocketMQ allows ACK to be returned to the client before data flushing, messages will be lost when the server is down due to exceptions.

- **Multiple masters and slaves**: multiple masters and slaves need to be set up for RocketMQ to ensure high business availability. RocketMQ can ensure availability and reliability only when there are healthy nodes in ISR; otherwise, the availability and reliability cannot be guaranteed, and the overheads will be high.

Therefore, compared with traditional open-source message queue applications, Tencent Cloud CMQ has the following advantages:

Comparison Items	Tencent Cloud CMQ	Open-Source Messaging Middleware
High performance	High performance and reliability can be guaranteed at the same time, and the QPS of a single CMQ instance reaches 5,000	High performance and reliability cannot be guaranteed at the same time
High scalability	• The number of queues and queue storage capacity are highly scalable • The underlying system can be automatically scaled based on the business volume, which is imperceptible to upper-layer businesses • Hundreds of millions of messages can be received, sent, pushed, and retained efficiently with an unlimited capacity • The message service is provided in multiple regions: Beijing, Shanghai, and Guangzhou	• The numbers of queues and retained messages are limited • Each IDC needs to have devices purchased and deployed, which is complicated
High reliability	• Backed by Tencent's proprietary Cloud Reliable Message Queue (CRMQ) distributed framework, CMQ has been widely used in Tencent businesses such as QQ and WeChat red packet systems and lottery • CMQ ensures that data is replicated to different physical servers in three copies before a successful write of each message is returned to the user, and the backend data replication mechanism guarantees that data can be quickly migrated when any physical server fails, so that the three copies of user data are always available with a reliability of 99.999999% • The improved Raft consistency algorithm is integrated to delivery a strong data consistency • The business availability is guaranteed at 99.95%	• Data is stored in single servers or a simple master/slave architecture, where data cannot be rewound once lost due to single points of failure • The open-source replica algorithm will cause rebalancing of global data when a server is added to or removed from the cluster, drastically bringing down the availability • If Kafka flushes and replicates data asynchronously, strong data consistency cannot be ensured
Business security	• Multi-dimensional security protection and anti-DDoS services are provided • Each message service has an independent namespace to strictly isolate data of different customers • HTTPS access is supported • Cross-region secure message service is supported	• The security protection features are limited • To avoid public network threats, cross-region and cross-IDC services over the public network usually cannot be provided