Message Queue CKafka

Product Introduction

Product Documentation

Contents

Product Introduction
Product Overview

Last updated : 2020-08-19 15:24:59

CKafka Overview

Based on the open-source Apache Kafka message queuing engine, Tencent Cloud Kafka (CKafka) provides high-throughput and highly scalable message queuing services. It is perfectly compatible with Apache Kafka APIs v0.9, v0.10, v1.1, and v2.4 and has greater advantages in terms of performance, scalability, business security, and OPS, allowing you to enjoy powerful features at low costs while eliminating tedious OPS work.

Features

- **Message Decoupling**
  CKafka effectively decouples the relationship between message producer and consumer, allowing you to independently scale or modify the production/consumption processing procedure as long as they follow the same API constraints.

- **Peak Shifting**
  CKafka can withstand access traffic surges instead of completely crashing due to sudden overwhelming requests, which effectively boosts system robustness.

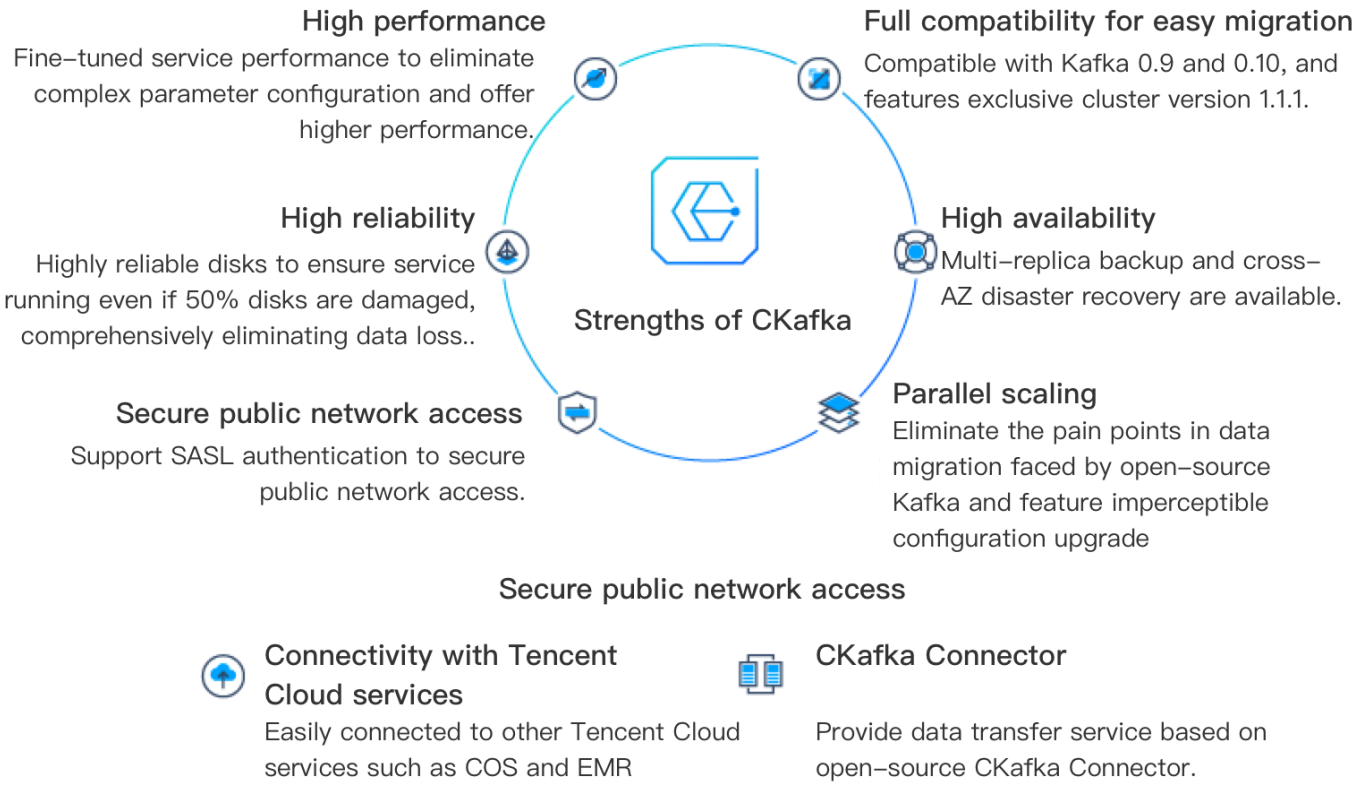
- **Sequential Read/Write**
  CKafka can guarantee the order of messages in a partition. Just like most message queue services, it can also ensure that data is processed in order, greatly improving disk efficiency.

- **Async Communication**
  In the scenario where the business does not need to process messages immediately, CKafka provides an async message processing mechanism, that is, when the traffic is high, messages will be put into the queue only and processed after the traffic drops, which significantly relieves the system pressure.

Strengths

Last updated : 2020-07-20 09:44:44

This document describes the strengths of CKafka compared to Apache Kafka.



Strengths

Full Compatibility with Open-source Programs for Easy Migration

CKafka is compatible with open-source Kafka 0.9, 0.10, and 1.1, and features exclusive cluster version 1.1.1.
The CKafka service system is based on the existing code of open-source Apache Kafka; therefore, your code can be migrated to high-performance CKafka without any modification required. For more information on migration, see Migrating Data to CKafka.

High Performance

Tencent Cloud CKafka team has further fine-tuned the service performance to eliminate complex parameter configuration and offer higher performance.

High Availability

Backed by Tencent's many years of experience in technical monitoring, CKafka features comprehensive monitoring on clusters and has a professional OPS team in place to respond to alarms on a 24/7 basis and ensure high availability of CKafka.
Multi-AZ disaster recovery solutions are available with imperceptible failover.

High Reliability

The disks used are highly reliable, and the service will not be affected even if 50% disks are damaged. There are 2 replicas by default, and 3 replicas can be supported. The more replicas, the higher the reliability.

Parallel Scaling

CKafka eliminates the pain points in data migration faced by open-source Kafka and features imperceptible configuration upgrade.

Secure Public Network Access

SASL authentication is supported to secure public network access.

Data Security

CKafka provides various security features such as authentication, authorization, root account, and sub-account, enabling enterprise-level security protection.
VPC: Access from more secure VPC is supported.
Root account and sub-account: CAM features such as root account, sub-account, and collaborator are fully supported, enabling authorization between root account and sub-account as well as across organizational accounts.

Connectivity with Associated Services

Connectivity with Tencent Cloud Services

CKafka can be easily connected to other Tencent Cloud services such as COS and EMR.

Kafka Connector

Data transfer based on open-source Kafka Connector is supported, so that data can be transferred between two Kafka clusters.

Use Cases

Last updated : 2020-02-26 17:34:19

CKafka is widely used in big data scenarios, such as webpage tracking, behavior analysis, log aggregation, monitoring, streaming data processing, and online and offline data analysis.
You can make data integration easier by:

- Importing the messages in CKafka to data warehouses in Tencent Cloud such as COS and Oceanus.
- Connecting to other Tencent Cloud products via SCF triggers.

Webpage Tracking

CKafka processes website activities (such as PV, search, and other user behaviors) in real time and then publishes them to topics by type. These information flows can be used for real-time monitoring or offline statistical analysis.
Since a large amount of activity information is generated in each user's page views, website activity tracking requires a very high throughput. CKafka can perfectly meet the requirements of high throughput and offline processing.

Log Aggregation

The low-latency processing capability of CKafka makes it easier to sustain distributed processing (consumption) of data from multiple sources. Compared to a centralized log aggregation system, CKafka can achieve stronger persistence guarantee and lower end-to-end latency while providing the same performance.
The features of CKafka make it an ideal "log collection center". Multiple servers/applications can send operation logs to a CKafka cluster "in batches" and "asynchronously" with no need to store them locally or in a database. CKafka can submit/compress messages in batches, so that the producer can hardly perceive the performance overhead. In this case, the consumer can use systematic storage and analysis systems such as Hadoop to perform statistical analysis on the pulled logs.

Big Data Scenarios

In some business scenarios involving big data, massive amounts of concurrent data need to be processed and aggregated, so high cluster processing performance and scalability are required. Thanks to its advantages in data distribution mechanism, allocation of disk storage space, processing of message formats, server selection, and data compression, CKafka is suitable for processing high numbers of real-time messages and can aggregate the data generated by distributed applications for easier system OPS.
Specifically, CKafka can process offline data or streaming data effectively and aggregate and analyze data easily.

Function Trigger

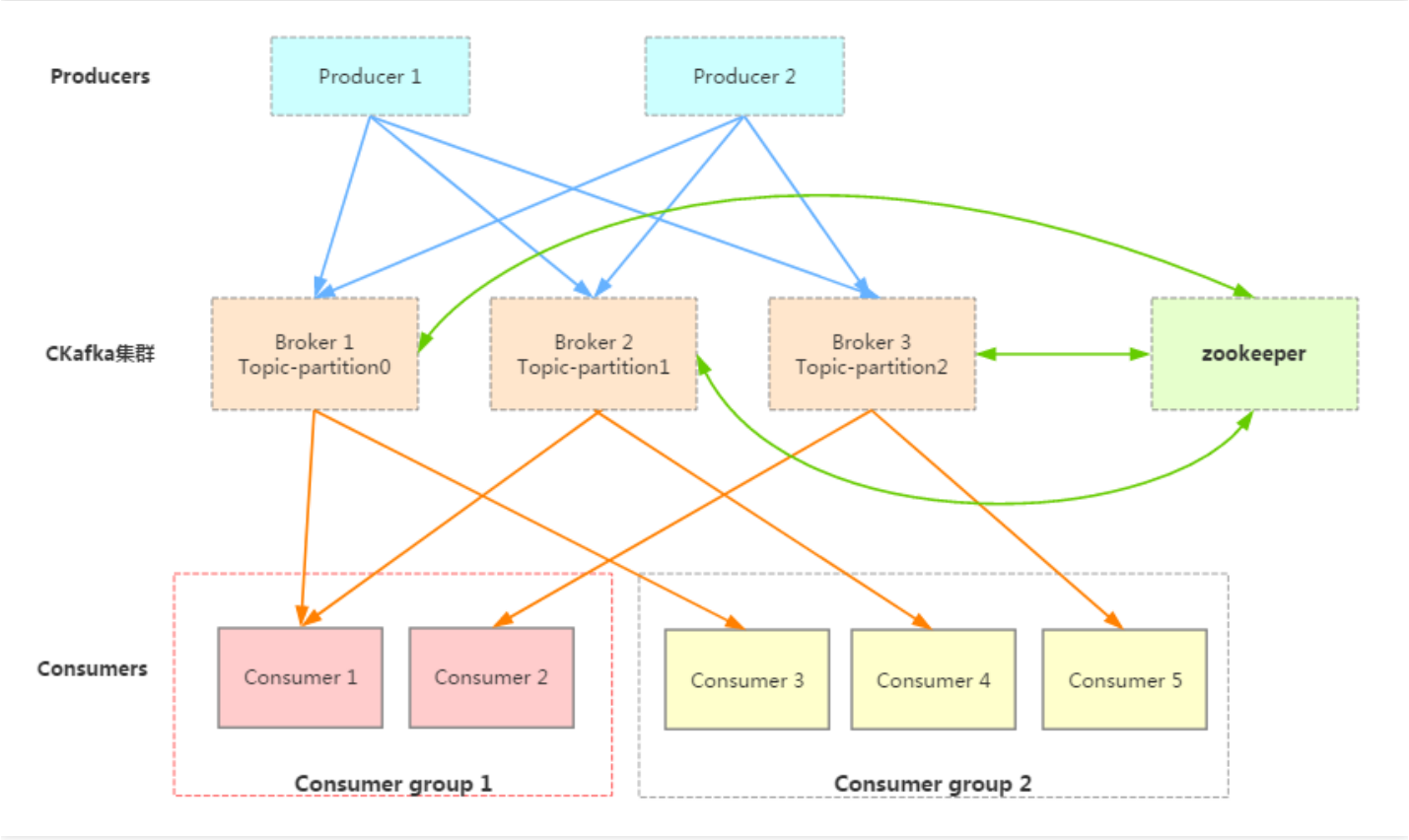
CKafka can be used as SCF function triggers, and when a message is received, a function can be triggered and the message will be passed to the function as event content. For example, when CKafka triggers a function, the function can transform the message structure, filter the message contents, or deliver the message to Elasticsearch Service (ES).

> For more information on the availability of SCF, see Service Level Agreement for SCF.

Technical Principle

Last updated : 2020-07-14 16:05:44

The architecture of CKafka is as follows:



- A producer can be information such as messages generated by webpage activities and service logs. It publishes messages to CKafka's broker cluster in push mode.
- ZooKeeper is used to manage the cluster configuration, elect the leader, implement fault tolerance, and do more in the cluster.
- Consumers are divided into several consumer groups. A consumer consumes messages from the broker in pull mode.

For the advantages of CKafka over self-built open-source Apache Kafka, please see Strengths.

High Throughput

In CKafka, a huge amount of network data is permanently stored in disks and high numbers of disk files are sent over the network. The performance of this process directly affects Kafka's overall throughput, especially in the following aspects:

- *Improved disk utilization*: data is read and written sequentially in the disk, which helps increase the disk utilization.
  - Message write: messages are written to the page cache and flushed by the async thread.
  - Message read: messages are transferred directly from the page cache into the socket and then sent out.
  - If the corresponding data is not found in the page cache, disk IO will be caused, and the messages will be loaded from the disk to the page cache and then sent out directly from the socket.
- **Broker's zero copy mechanism**: the sendfile system is called to send data directly from the page cache to the network.
- **Reduced network overheads**
  - Data compression reduces the network load.
  - Batch processing mechanism: the producer writes data to the broker in batch, while the consumer pulls data from the broker in batch.

Data Persistence

Data persistence is mainly implemented in CKafka through the following principles:

- **Partition storage distribution in topic**
  In the file storage of CKafka, a topic has multiple different partitions, each of which physically corresponds to a folder. Messages and index files are stored in these partitions. For example, if two topics are created where topic 1 has 5 partitions and topic 2 has 10 partitions, then a total of 5 + 10 = 15 folders will be generated in the cluster.

- **File storage method in partition**
  A partition is physically composed of multiple segments of equal size. These segments are read from/written to sequentially and are deleted quickly upon expiration, which improves the disk utilization.

Scale-out

- One topic can include multiple partitions distributed in one or more brokers.
- One consumer can subscribe to one or more of these partitions.
- A producer is responsible for equally assigning messages to partitions.
- Messages in partitions are sequential.

Consumer Group

- CKafka does not delete consumed messages.
- A consumer must belong to a group.
- Consumers in the same consumer group do not consume the same partition at the same time.
- Different groups consume the same message at the same time, which is more diversified (queuing model and publish-subscribe model).

Multiple Replicas

The multi-replica design can enhance the system availability and reliability.
Replicas are evenly distributed across the entire cluster. The replica algorithm is as follows:

1. All brokers (assuming n brokers in total) and the partitions to be assigned are sorted.
2. The ith partition is assigned to the (i mod n)th broker.
3. The jth replica of the ith partition is assigned to the ((i + j) mode n)th broker.

Leader Election Mechanism

CKafka dynamically maintains a set of in-sync replicas (ISR) in ZooKeeper, and all replicas in ISR catch up to the leader. Only members of the ISR can be elected as leaders.

- If there are f + 1 replicas in ISR, a partition can
  tolerate f replica failures while guaranteeing that committed message will not be lost,
- There is a total of 2f + 1 replicas (including the leader and followers), and it must be guaranteed that f+1
  Replicas have replicated the messages before the commit operation. To ensure that a new leader can be correctly elected, the number of failed replicas cannot exceed f.

Product Comparison

Last updated : 2020-07-14 16:05:44

The performance comparison between CKafka and other messaging services is as follows:

Feature	CKafka	Apache Kafka	RabbitMQ	RocketMQ	CMQ
Advantages	Very high throughput; very elastic scalability; very low OPS costs	High throughout	High reliability	High reliability	Very high reliability; suitable for finance and other scenarios where strong consistence is required
Disadvantages	Occasionally message loss in extreme circumstances	Occasionally message loss; Less flexible scalability; Multiple dependent components, leading to heavy OPS work; Limited security protection, poor isolation and compatibility	Poor performance; Less flexible scalability	Manual (not automatic) high-availability switch	Average throughout for guaranteed strong consistency
Programming language	Scala	Scala	Erlang	Java	C++

Feature	CKafka	Apache Kafka	RabbitMQ	RocketMQ	CMQ
Scalability	Very flexible and easy to scale. Only the VIP address needs to be specified for message sending, and broker changes are imperceptible for both message sending and receiving	Not flexible enough. The broker address needs to be specified to send messages, and ZooKeeper coordination scheduling is required for message receiving	Not flexible enough. The broker address needs to be specified for message sending	Relatively flexible. The sender and receiver are connected to the name server	Flexible, smooth, and horizontally scalable. Logically, a single queue can provide services across multiple clusters
Throughput	Very high	High	Average	Average	Average
General performance	Million-level QPS	Million-level QPS	100-thousand-level QPS	100-thousand-level QPS	100-thousand-level QPS
"2-core 4 GB" stress test	220,000 read/write QPS	200,000 read/write QPS	100,000 read/write QPS	100,000 read/write QPS	120,000 read/write QPS
Sync algorithm	ISR (Replica)	ISR (Replica)	GM	Synchronous double write	Raft

Feature	CKafka	Apache Kafka	RabbitMQ	RocketMQ	CMQ
Availability	Very high availability. Automatic master/slave switch is supported. CKafka guarantees an availability of 99.95%	High availability. Automatic master/slave switch is supported. Messages may be lost after switch due to async flush and replication	Automatic master/slave switch is supported. The mirror queue is used to support m/s where the master provides services and the slave implements backup only	Automatic master/slave is not supported. The slave only reads but does not write when the master is not available	Very high availability. The broker can provide high-availability services as long as it contains two nodes
Consumption method	Pull	Pull	Pull and push	Pull and push	Pull and push
Message reliability	High reliability; Reliability can be further improved based on the three-copy mechanism, and the cluster features better disaster recovery where failures rarely occur	Low reliability; The broker has only async flush and async master/slave replication mechanisms, which may cause message loss	High reliability; When messages are sent, if they are specified as persistent, they will be written into the disk	High reliability; The broker supports sync doublewrite, and a success will be returned only after the message is written into both the master and the slave	Extremely high reliability; Message loss is eliminated with sync flush. The data persistence is 99.999999%
Data verification	CRC	CRC	None	CRC	Checksum
Message rewind	Yes	Yes	No	No	Yes
Security protection	Yes	No	No	No	Yes

Feature	CKafka	Apache Kafka	RabbitMQ	RocketMQ	CMQ
Monitoring and alarming	Yes	No	No	No	Yes
Service support	Yes	No	No	No	Yes

"2-core 4 GB stress test" indicates the result of a stress test on a server with 2 CPU cores and 4 GB memory.