TDMQ for CKafka

Troubleshooting

Product Documentation

Contents

Troubleshooting
Topic Failures
Topic Creation Failure

Last updated：2024-01-09 14:57:56

Issue Description

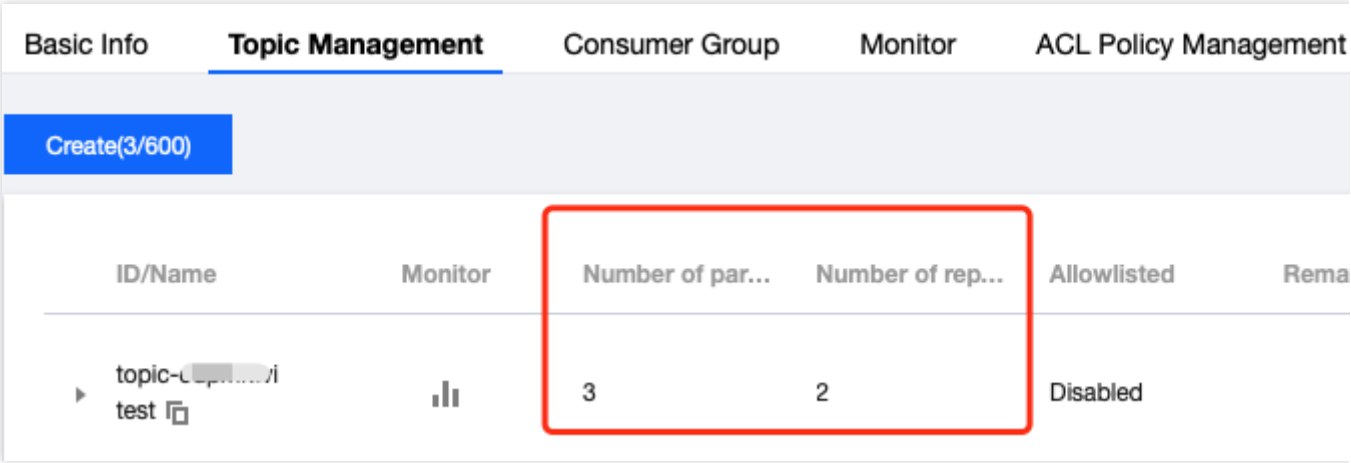
A topic fails to be created.

Possible Causes

The limit of an instance on the total number of topic partitions has been reached.
The topic is created while one with the same name is being deleted.
The topic already exists in the cluster.

Solutions

The limit of an instance on the total number of topic partitions has been reached
Try expanding the Kafka instance or delete unnecessary topics.



The topic is created while one with the same name is being deleted

Topic deletion is an async operation. After the deletion instruction is delivered, the system will delete the topic metadata asynchronously. During this period, if you try to create a topic with the same name as the deleted one, the system will prompt that the topic already exists. In this case, wait about 1 minute and try again later.

The topic already exists in the cluster

If a topic with the same name as the topic you want to create already exists in the cluster, the system will prompt that the topic already exists. In this case, you are recommended to rename the topic.

No Topic Monitoring Data

Last updated：2024-01-09 14:57:56

Issue Description

The topic has no monitoring data.

Possible Causes

The client does not produce/consume messages.
The monitoring system fails.

Solutions

The client does not produce/consume messages
Check whether the client can produce/consume messages by running the native production/consumption command and then viewing the monitoring data. For detailed directions, see Running Kafka Client (Optional).
The monitoring system fails
If the monitoring system fails, submit a ticket for assistance.

Linkage Failure of Other Tencent Cloud Services Caused by Topic ACL Policy Configuration

Last updated：2024-01-09 14:57:56

Issue Description

The linkage capabilities of other Tencent Cloud services fail after an ACL policy is configured for a topic.

Possible Causes

By default, no ACLs are set for a topic, and the topic can be accessed without limit by instances in the same VPC. If you want to control the permissions in the VPC, you can configure an ACL as instructed in Configuring ACL Policy.
When you add an ACL policy for a topic, the policy will prevent all other ineligible requests from accessing the topic, including those initiated by other Tencent Cloud services connected to CKafka (e.g., log shipping in CLS, message dump in SCF, and component consumption in EMR).
From a business point of view, the business wants to ensure that clients that don't meet the requirements cannot access Kafka data once an ACL is set; therefore, the rejection is reasonable.

Solutions

Before adding an ACL policy for a topic, you must determine whether the topic is being used in other scenarios through the service information or the monitoring information in the console; otherwise, problems with other linked features may occur.
In such cases, if you have to use an ACL policy, we recommend you produce messages to a new topic for permission grant instead of reusing the original topic.

Existence of Partition Message Heap

Last updated：2024-01-09 14:57:56

Issue Description

There are always heaped messages in some partitions as displayed on the monitoring page.

Possible Causes

The producer keeps producing messages to the partition while the consumer never consumes these messages.
There are insufficient consumers in the consumer group or the consumption speed is too slow.
The consumer does not consume messages in certain partitions due to bugs.

Solutions

The producer keeps producing messages to the partition while the consumer never consumes these messages

You can restart the client if messages in some partitions are not consumed. If the problem persists, you can either check the client logs first or submit a ticket for assistance.

There are insufficient consumers in the consumer group or the consumption speed is too slow

Try increasing the number of consumers to improve consumption speed.

The consumer does not consume messages in certain partitions due to bugs

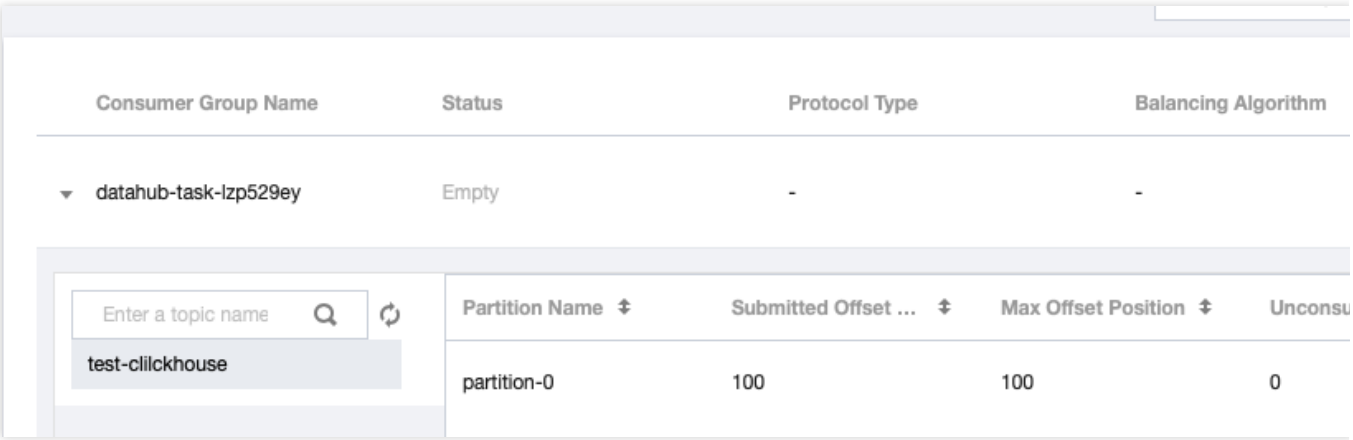
Check the local logs of the consumer to see whether exceptions exist.

Consumer Group Failures
No Consumer Group Details Displayed

Last updated：2024-01-09 14:57:56

Issue Description

The consumer group list in the CKafka Console contains consumer group name, but no consumption details are displayed on the details page; for example, the consumer group CR has no details displayed in the figure below:

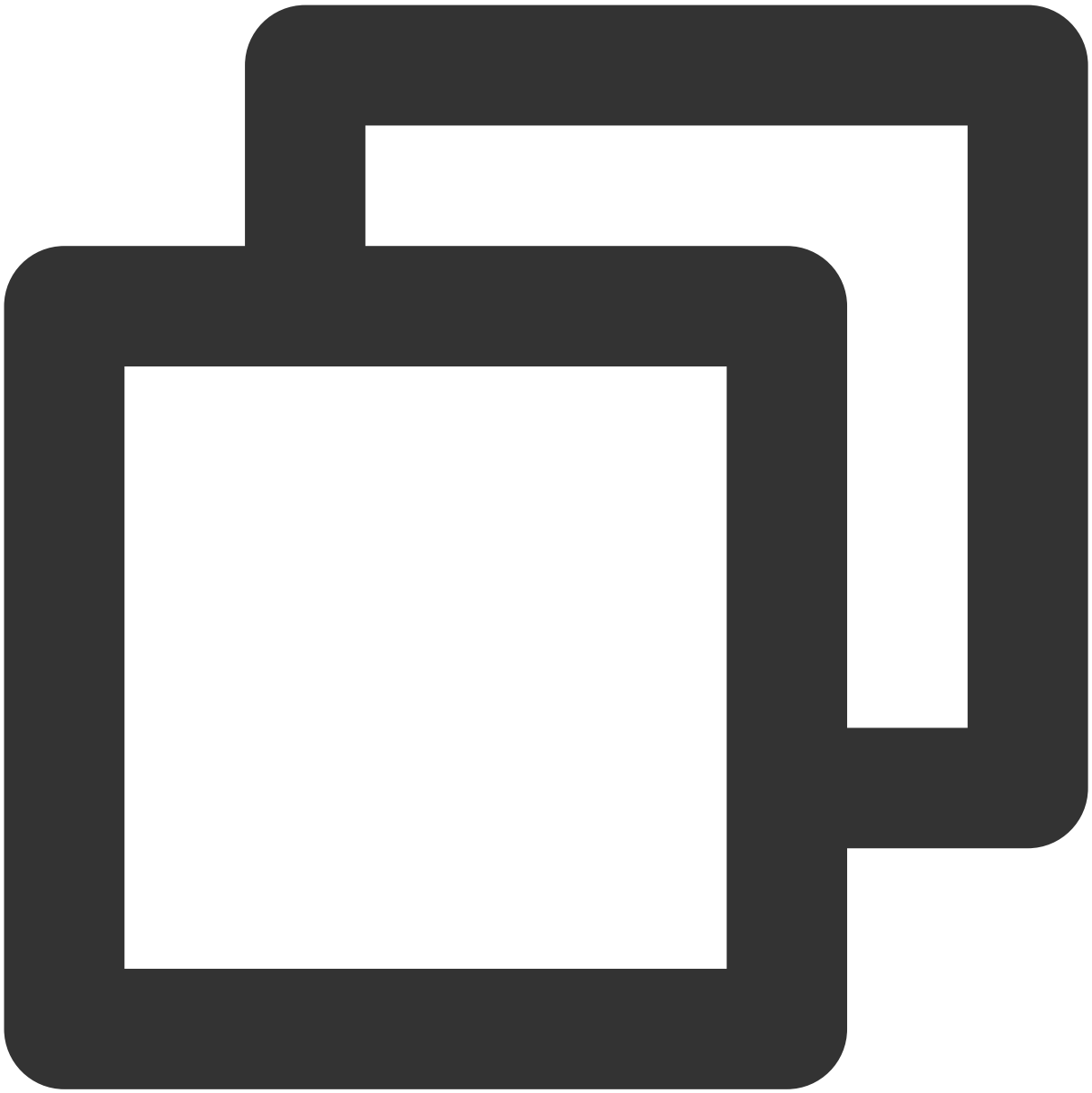


Possible Causes

There are two data consumption modes in Kafka: consumer group mode and custom partition consumption mode. When consumption is performed in consumer group mode, the client will coordinate consumption through the consumption coordinator, and after data consumption is completed, it will send an offset storage request to the server, which will store information such as the consumed topic, partition progress, and client.
When consumption is performed in custom partition consumption mode, the client will not automatically submit an offset storage request to the server. In this case, the server will not be able to see information related to consumption.
After an ACL is set for a topic, you may not be able to view the consumer group details on some instances. In this case, check whether there are any ACLs, and if so, submit a ticket for assistance.

Solutions

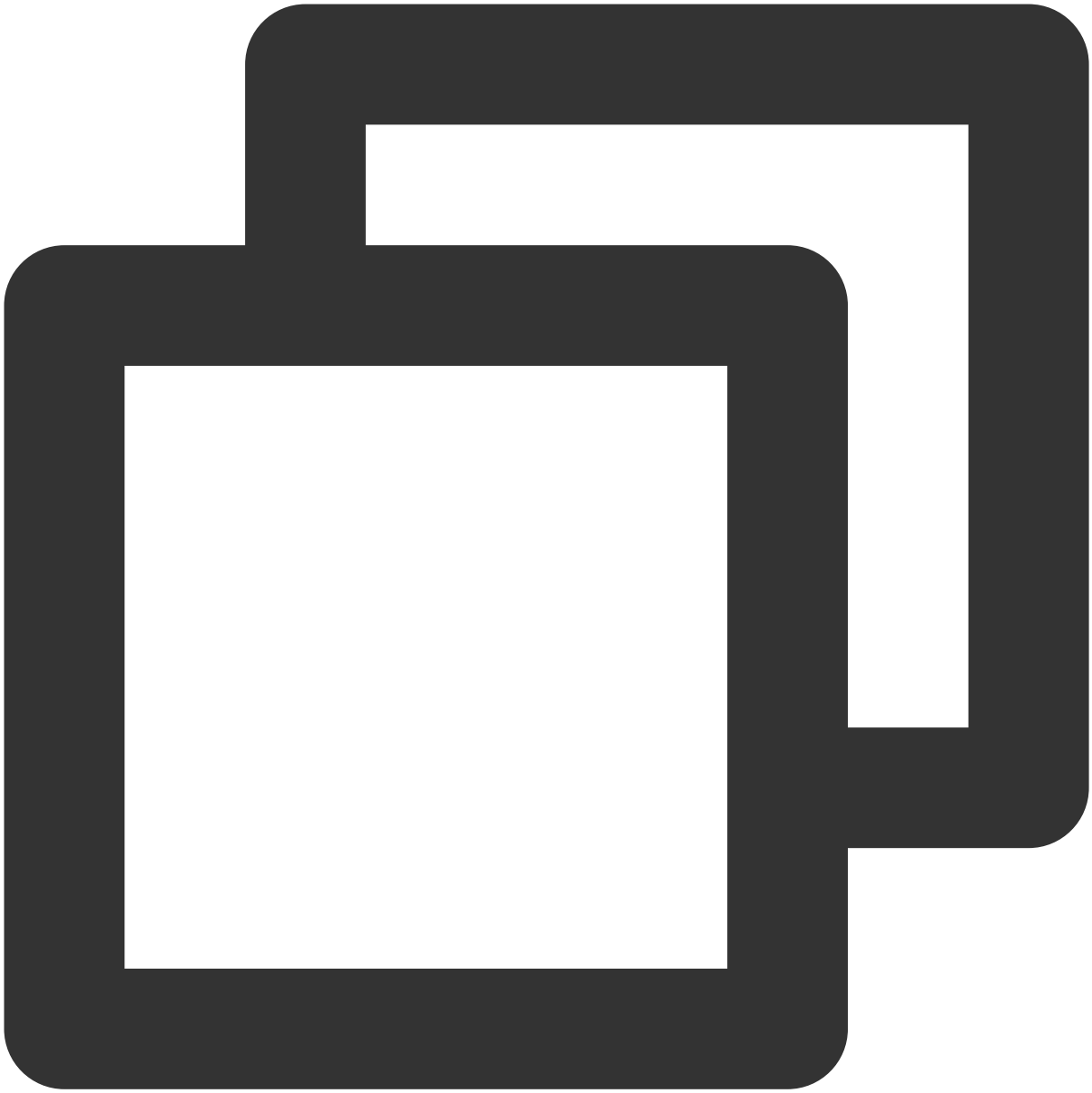
1. View the consumer groups of the instance.



```
]$ bin/kafka-consumer-groups.sh --bootstrap-server 9.146.153.249:9092 --list
CR
```

You should be able to see the names of all current consumer groups.

2. View the details of a specific consumer group of the instance.

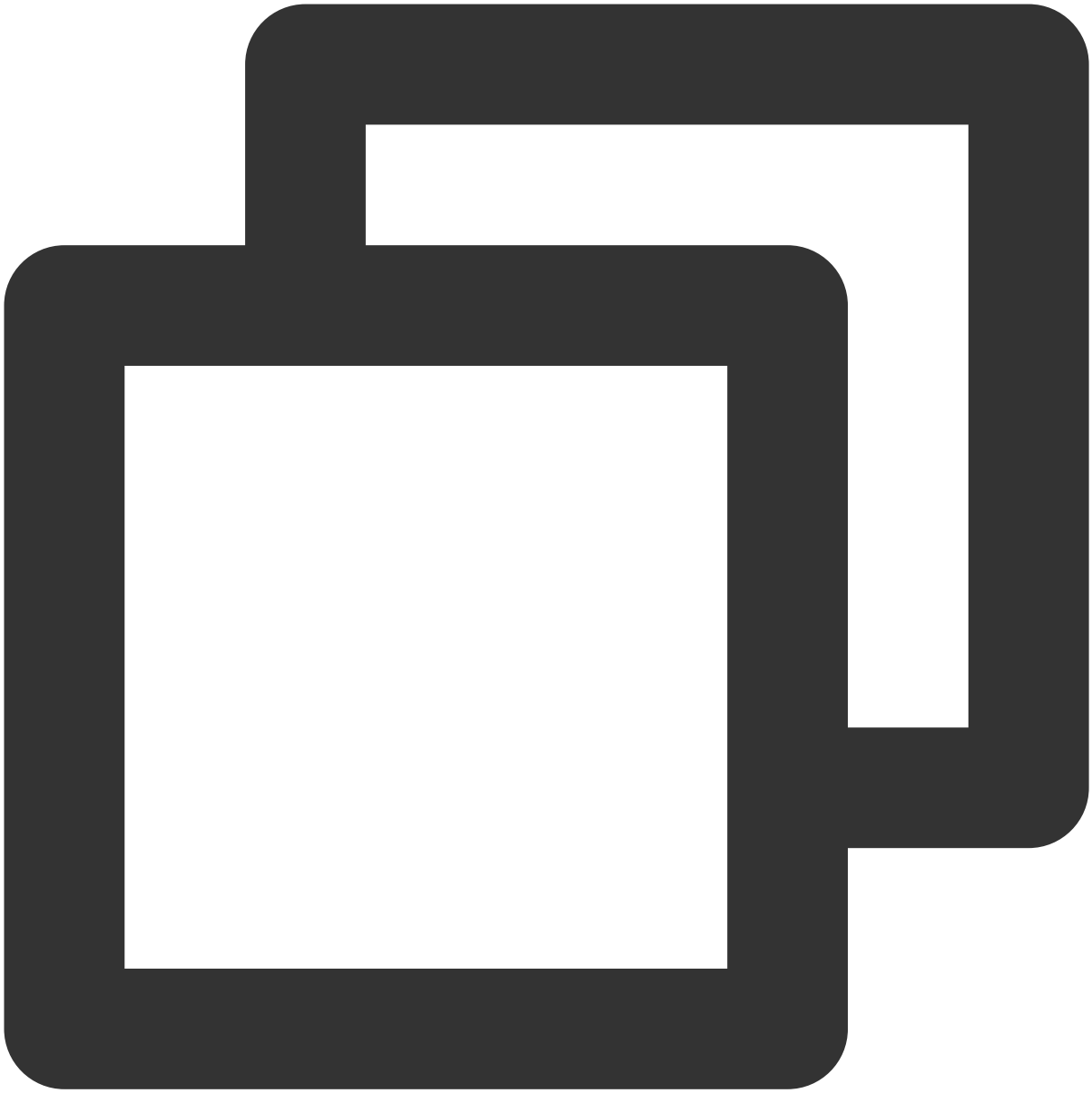


```
]$ bin/kafka-consumer-groups.sh --bootstrap-server 9.146.153.249:9092 --describe --
Note: This will not show information about old Zookeeper-based consumers.
```

You will find that there are no details for this consumer group, which means that the consumer client did not use the `consumerGroup` mechanism to consume data, that is, the client did not submit consumption details to the server. As the server did not store the consumption data, no details will be displayed.

3. Check whether the problem is caused by the server.

Use the native consumer group command to specify the consumer group `test1` for consumption as shown below:



```
]$ bin/kafka-console-consumer.sh --bootstrap-server 127.0.0.1:9092 --from-beginning
```

You should be able to see the normally displayed consumer group in the console, and you can run the `--describe` command to view the details as shown below:

```
$ bin/kafka-consumer-groups.sh --bootstrap-server 127.0.0.1:9092 --describe --group test

GROUP     TOPIC    PARTITION  CURRENT-OFFSET  LOG-END-OFFSET  LAG   CONSUMER-ID
test      test     0          -               67              -     sarama/127.0.0.12020-06-11 23:33:03:110-a113e7d1-e3
```

Consumer Group Constantly in PreparingRebalance Status

Last updated：2024-01-09 14:57:56

Issue Description

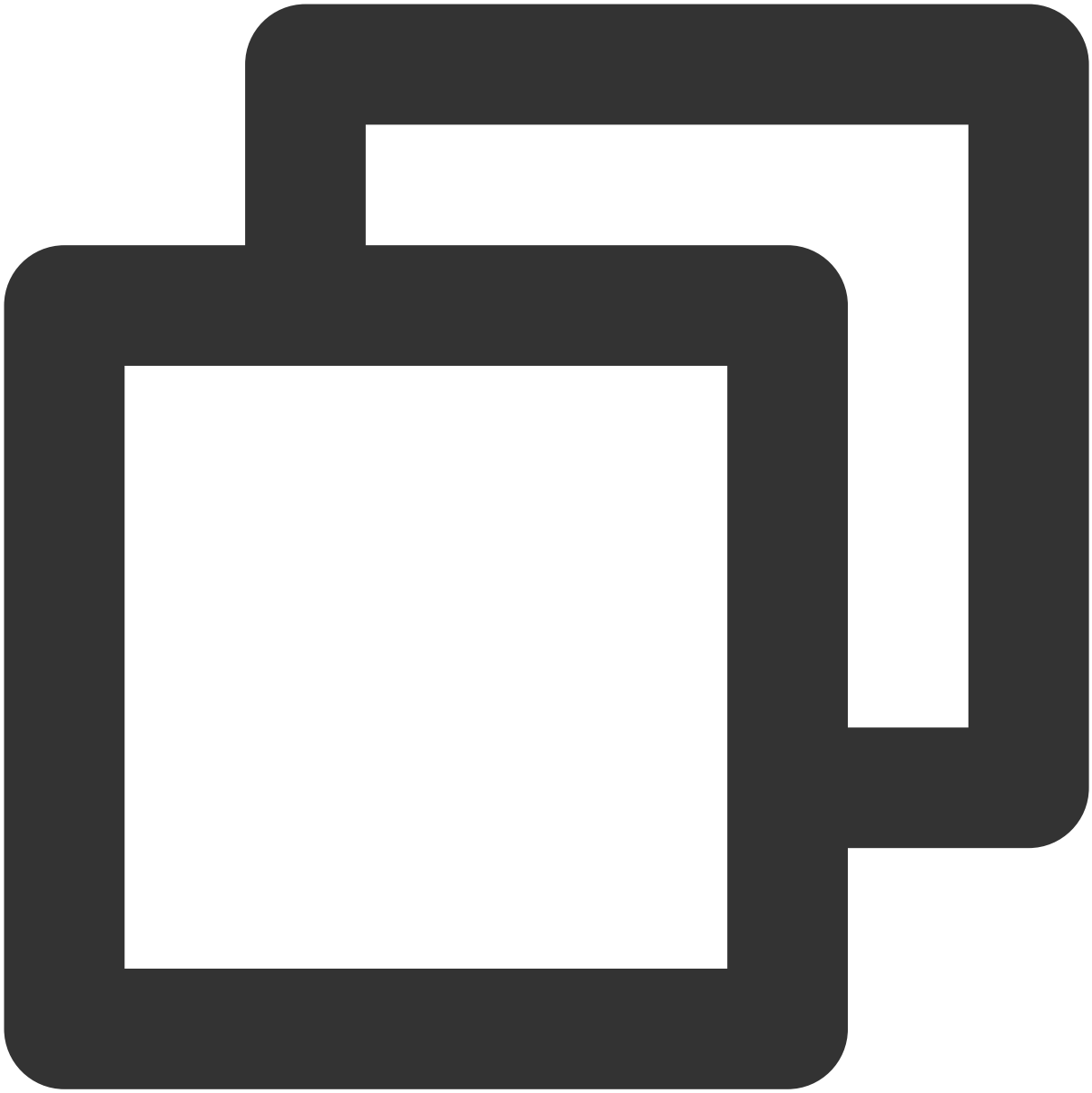
A consumer group is constantly in `PreparingRebalance` status.

Possible Causes

1. A new consumer joins the consumer group.
2. A running consumer stops running and leaves the consumer group. Common causes for this include consumer restart, consumer application crash, and heartbeat timeout of consumer process reporting. For more information, see Configuration Guide for Common Parameters in CKafka.
3. The number of partitions changes (partitions are added or deleted).

Solutions

Rebalancing is inevitable in cases 1 and 3. Normally, rebalancing can be completed in 30s. If there is a longer rebalancing process, submit a ticket for assistance.
If the rebalancing is caused by heartbeat timeout or excessive interval between two polls, you can adjust the following parameters:



```
# Consumer timeout period when the Kafka consumer grouping mechanism is used. If th
session.timeout.ms=10000

# Interval at which the consumer sends a heartbeat when the Kafka consumer grouping
heartbeat.interval.ms=3000

# Maximum interval allowed for calling the poll again when the Kafka consumer group
max.poll.interval.ms=300000
```

If a consumer subscribes to many topics, you can try reducing the topics to which the consumer group subscribes.

Client Failures
Common Client Errors and Solutions

Last updated：2024-01-09 14:57:56

Client Configuration or Service Exceptions

The following describes client configuration or service exceptions. If any of the following exceptions occurs, the client will not automatically retry.

Exception	Description	Analysis
UnknownServerException	An unknown error occurred when the server processed the request.	The error will be returned during traffic throttling in the legacy version. If it occurs i the new version, it may be caused by a server bug.
RecordTooLargeException	Message is too large.	The current configuration is `message.max.bytes=1000012` .
InvalidRequiredAcksException	The `acks` parameter configured for the producer is invalid.	-
InconsistentGroupProtocolException	Group protocol is inconsistent with the client protocol.	Check whether the same `group.id` is configured for the consumer and the connector. They cannot join the same grou if different protocols are used.
InvalidGroupIdException	Consumer group ID is invalid.	Use no more than 128 characters such as z, A-Z, 0-9, and ._-.
InvalidTopicException	Topic is invalid.	When topic auto-creation is enabled, an exception will be returned if the client uses an invalid topic. Check whether the topic name contains invalid characters or excee the length limit.
InvalidSessionTimeoutException	`Session.timeout.ms` configured for the consumer is invalid.	Current minimum and maximum values allowed by the server are 6000 (value of `group.min.session.timeout.ms`) and 300000 (value of `group.max.session.timeout.ms`)

InvalidCommitOffsetSizeException	Submitted offset information is too large and exceeds the maximum message size, so `__consumer_offsets` cannot be written.	Current configuration is `message.max.bytes=1000012` .
OffsetMetadataTooLarge	Metadata contained in the request to submit the offset is too large.	`offset.metadata.max.bytes=409` is configured for the server.
UnsupportedVersionException	Broker does not support requests in this version.	We recommend using a v0.10.2.x client.

Brief Exceptions That Occur During Normal Program Operation

The following exceptions may occur briefly during normal program operation, and the client will automatically retry. If an exception persists, the service will run improperly.

Exception	Description	Analysis
TimeoutException	Request timeout.	If the initial connection reports a request timeout, check whether the address is correct and run telnet to confirm whether the network works properly. If this exception occurs only occasionally during program execution, it may be caused by network jitters.
CorruptRecordException	Message is invalid.	Possible causes include CRC error or invalid data size. This exception may also occur if the compression method used is **gzip** or **the version is below 0.9**.
UnknownTopicOrPartitionException	Topic or partition does not exist.	Go to the console to check whether the corresponding topic has been created. Note: the client produces and consumes through `TopicName` rather than

		`TopicId` . This exception will also occur if the client does not have permission to access the topic.
LeaderNotAvailableException	Partition has no leader.	When the topic has just been created yet the server has not selected the appropriate leader, an error will be returned to the client, and the client will automatically retry to get the leader information. **Only the legacy version has this exception, which has been removed from 0.10.2.1.**
NotLeaderForPartitionException	Partition leader is unavailable.	As the client caches the metadata of the topic, when the partition leader changes, production or consumption requests may still be sent to the original leader. An error will be returned to the client and the client will automatically update the metadata.
NetworkException	Client connection is closed by the server.	Network exception or the number of connections exceeds the limit.
NotEnoughReplicasException	Number of ISRs is insufficient.	The number of ISRs in the partition when data is written is smaller than the value of `min.insync.replicas` configured for the topic, which may be caused by ISR jitters.
NotEnoughReplicasAfterAppendException	ISR jitters occur after data is written to the local broker, making the configuration of `min.insync.replicas` unsatisfied.	-
BrokerNotAvailableError	Partition leader is not found.	As the client caches the metadata of the topic, when the partition leader changes, the

		production or consumption requests may still be sent to the original leader. An error will be returned to the client and the client will automatically update the metadata. After the leader changes, new production requests sent to the original leader will be automatically forwarded to the new leader upon this error reporting. Theoretically, the integrity of consumption data written will not be affected.
NotLeaderForPartitionError	Partition leader is not found.	As the client caches the metadata of the topic, when the partition leader changes, production or consumption requests may still be sent to the original leader. An error will be returned to the client and the client will automatically update the metadata. After the leader changes, new production requests sent to the original leader will be automatically forwarded to the new leader upon this error reporting. Theoretically, the integrity of consumption data written will not be affected.

Exceptions That Occur When Log Level Is Configured as DEBUG

The following exceptions will occur if the log level is configured as DEBUG and will be automatically processed by the client.

Exception	Description	Analysis
OffsetOutOfRangeException	Offset passed in was out of range when consumer pulled messages.	If an offset reset policy (earliest or latest) is configured for the client, the client will reset the offset according to the policy; otherwise, the user

		application needs to handle this exception.
GroupLoadInProgressException	Coordinator of the consumer group is being loaded.	This may occur briefly when the server is being upgraded. The client will automatically retry.
GroupCoordinatorNotAvailableException	Coordinator is not available.	This may occur briefly when the server is being upgraded. The client will automatically retry.
NotCoordinatorForGroupException	The current node is not the coordinator of the consumer group, and the coordinator has been migrated to another node.	This may occur briefly when the server is being upgraded. The client will automatically retry.
IllegalGenerationException	`Generation` of the consumer group is invalid.	Possible causes include heartbeat timing out or a new consumer joining the group. The consumer will automatically retry joining the consumer group.
RebalanceInProgressException	Consumer group is rebalancing.	Possible causes include heartbeat timing out or a new consumer joining the group. The consumer will automatically retry joining the consumer group.

Blockage of Messages Produced by Client

Last updated：2024-01-09 14:57:56

Issue Description

Messages produced by the client are blocked mainly because they cannot be sent, or message sending is slower than message production.

If messages cannot be sent, "time out" will be prompted. In this case, you can use the command line for production and consumption first and view the basic performance of the cluster. For more information, see Running Kafka Client (Optional).

There are three reasons why message sending is slower than production:

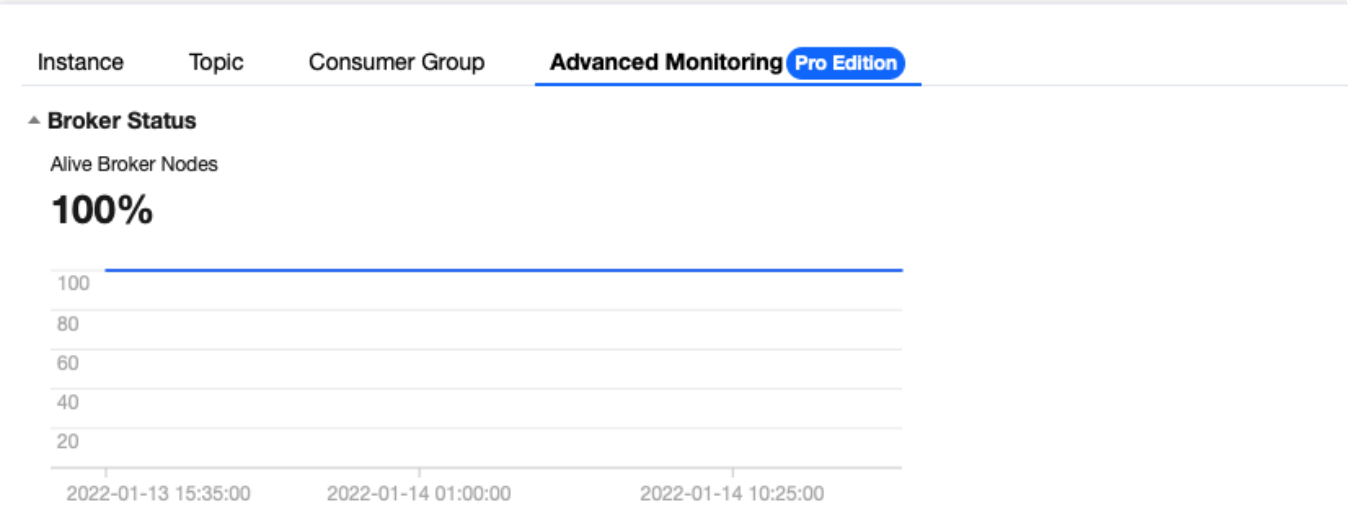
Producer instances are not enough. As the production performance of a single producer is limited, if there are insufficient producers while the traffic is high, message sending may be blocked.

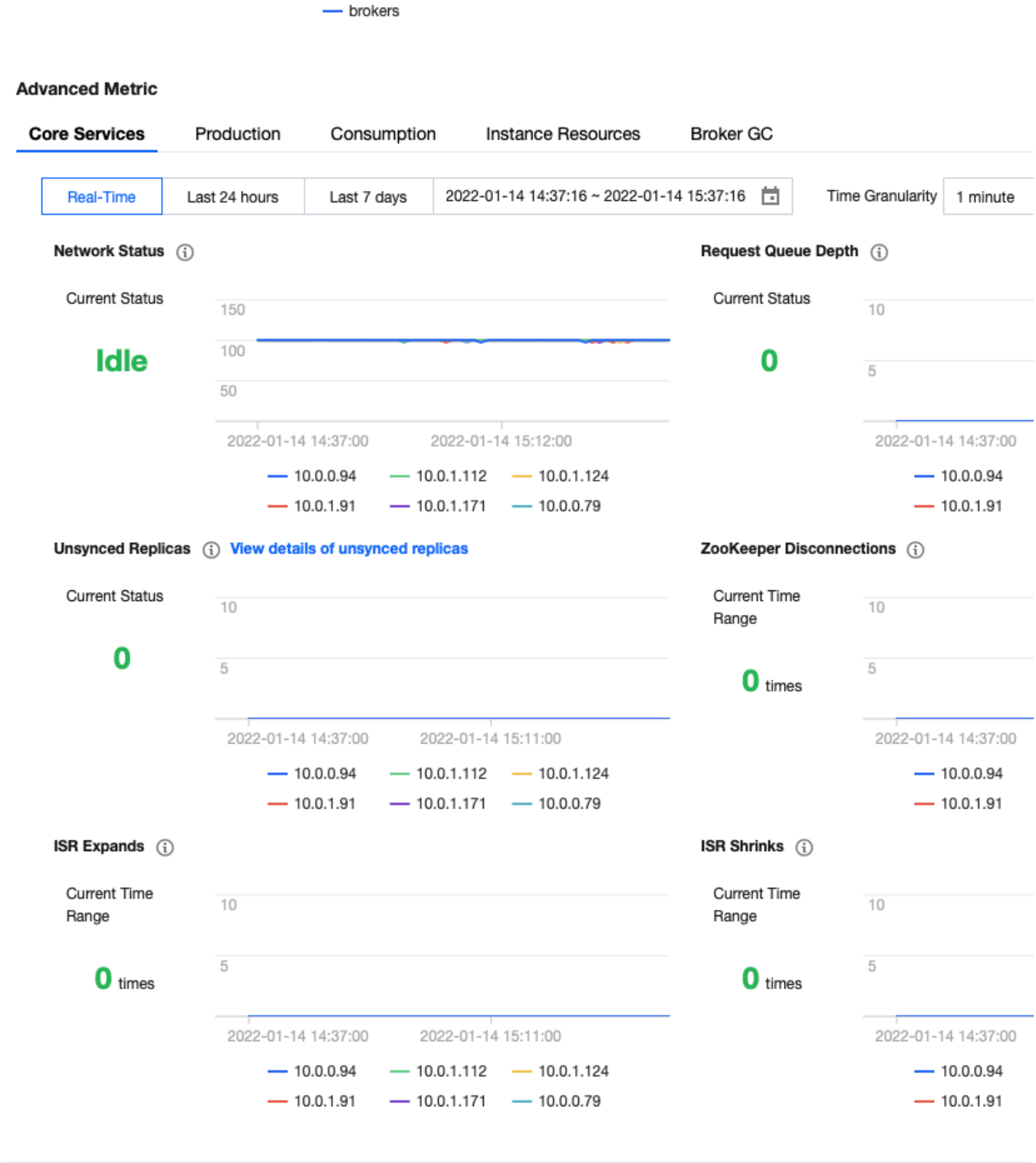
The traffic is high, but the topic partitions are not enough, resulting in low write concurrency.

Another reason is low service quality. For example, the network quality is low, the broker load is high, or the client load is high (in the case of GC on the client). This will prolong message sending from the client to the server and reduce the production efficiency, causing a heap in the local buffer of the client and eventually blocking messages.

Possible Causes

1. For Pro Edition instances, you can view advanced monitoring metrics such as the request queue depth and server production and consumption time in the console to check the overall load of the server and whether the server has performance problems. For Standard Edition instances, you can submit a ticket to view such metrics.





2. Check the client load such as local server CPU and memory utilization (if the client is in Java, you should pay attention to GC).

3. If blocking occurs occasionally, check whether the local network is stable, especially in a container network environment.

4. Check whether producers are not enough based on the traffic of a single server. If the throughput traffic of a server is high while producers send messages in a single thread, you should pay attention to the number of producers.

Solutions

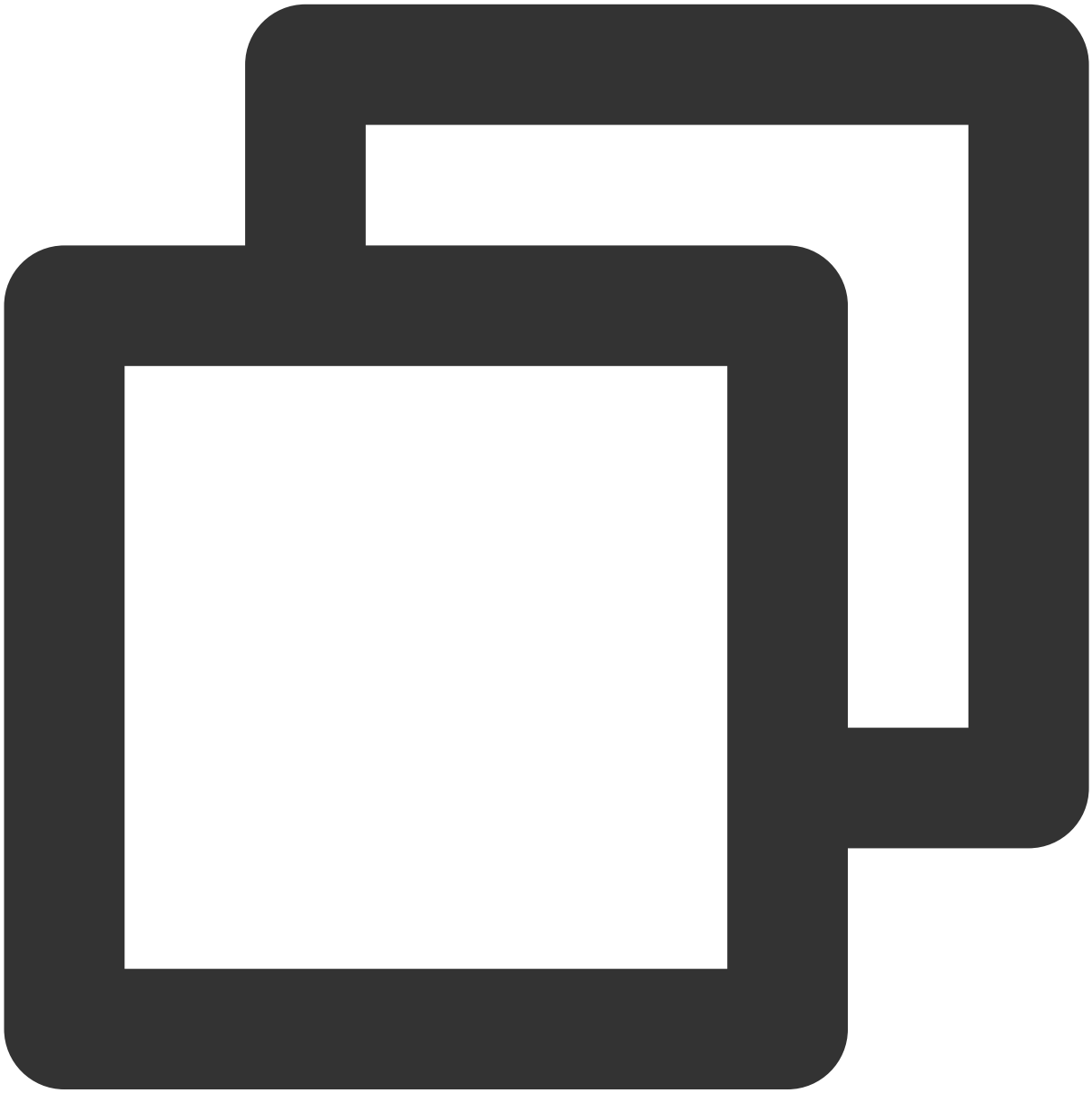
Below are some solutions:

1. If messages are produced faster than sent by the `Sender` thread to the broker, the memory configured by `buffer.memory` will be used up, blocking the sending operations by the producer. This parameter sets the maximum blocking time. If a larger `send buffer` is required, you can increase the value of `buffer.memory`, which is 32 MB by default.

```
# Maximum blocking time
```

```
max.block.ms=60000
# Configure the memory that the producer uses to cache messages to be sent to the b
buffer.memory=33554432
```

2. If the topic traffic is high while the producer instances for client message sending are not enough, you can add more producers for production as follows:



```
KafkaProducer<byte[],byte[]> producer = new KafkaProducer<>(props);
```

3. Add more partitions to the cluster.
4. Submit a ticket for assistance.

Client's Failure to Consume Messages

Last updated：2024-01-09 14:57:56

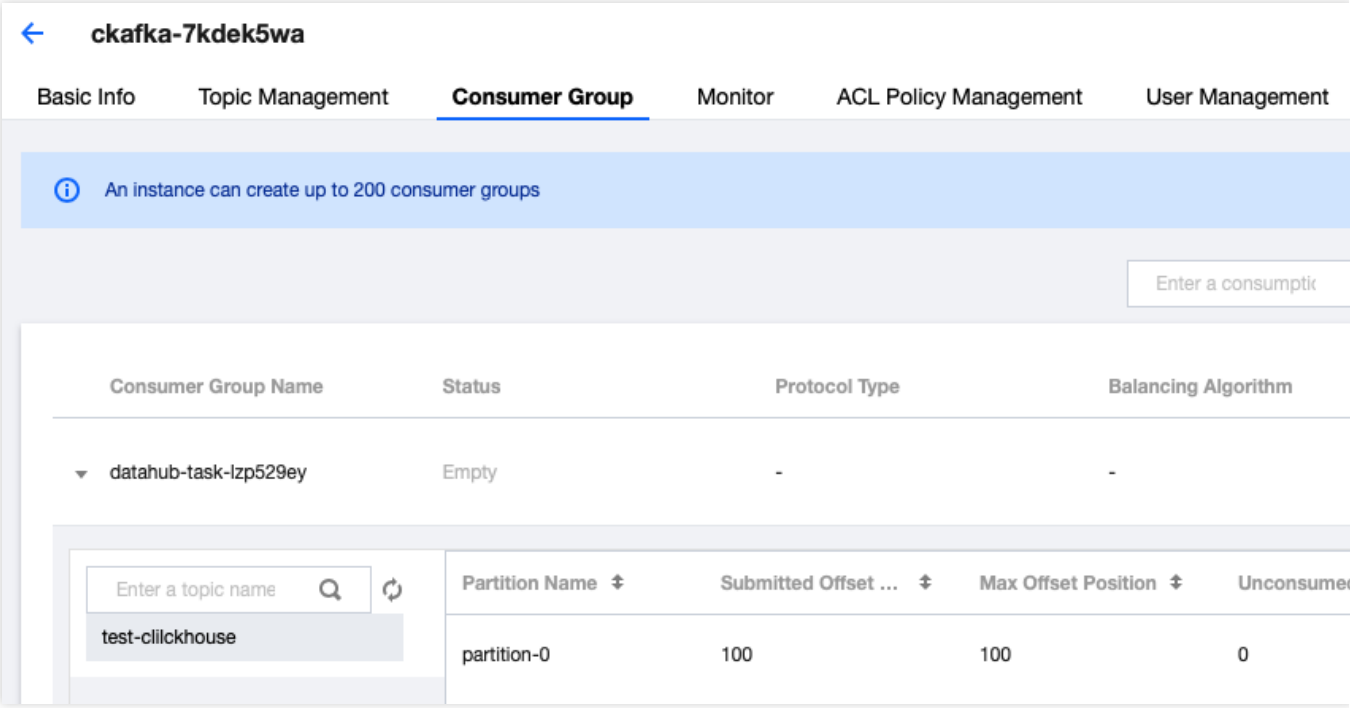
Issue Description

The consumer cannot pull messages.

Cause

Check whether the consumer group has a heap. If there is no heap, an empty message will be returned after the `fetch.max.wait` time elapses. This parameter is configured by the consumer client and 500 ms by default as shown below:

```
# Fetch request wait time.
fetch.max.wait.ms=500
```



Solutions

If messages heap up, but no messages are pulled, we recommend you check the client SDK version. If the SDK version is low, upgrade it as instructed in SDK Overview.

You can also submit a ticket for assistance.

Sarama Client

Last updated：2024-01-09 14:57:56

Issue Description

Sarama is a Kafka client written in Go with a high message throughput.
After you actively increase the number of CKafka partitions due to a performance bottleneck, Sarama may be unable to perceive the partition rebalance, causing a failure to produce/consume messages in the new partitions.

Common Causes

After CKafka rebalances partitions for various reasons, it will take about 10 minutes for Sarama to perceive the partition changes and then pull the metadata of the current topic to parse the latest partition data. As the pull is time-consuming, it may be regarded as a failure sometimes.
In addition, the CKafka team has found occasional failures of Sarama to pull the latest partition information, causing message retention and random exception reporting.
This problem has been repeatedly reported and fixed in the Sarama community. It occurs less frequently on the latest version but still exists.

Solutions

If you are sensitive to rebalance and use the Go technology stack, we recommend you migrate to Confluent-Kafka-Go, a client maintained by Confluent™.

Comparison of Common Go Clients

Go Client	Strengths	Limitations
Sarama	It is very active in the community and widely used. Questions raised in the community are answered and solved quickly. It performs well. As it is written in native Go, it supports async and high-concurrency operations.	It is moderately stable. It may experience unknown errors after the number of partitions is increased and all partitions are rebalanced.

Confluent-Kafka-go (recommended)	It is very stable. As it actually provides a layer of encapsulation for librdkafka, and librdkafka has been running for many years in multiple languages, it can be reliable enough. It performs well. As it is implemented in C++ at the underlying layer, it requires fewer resources and computes faster.	As it imports a C++ library, the Go compiler needs additional compilation configurations, which add compilation dependencies and make compilation more complex. In addition, as the logic for C++ library implementation varies by compilation environment, importing the `librdkafka` library may affect the cross-compilation of Go projects.
kafka-go	It provides top-level APIs and exposes underlying APIs, which can be called more flexibly to configure the Kafka client. It is easy to use since it requires less code to produce and consume messages and has many default configuration items.	It underperforms the above two clients and may be unable to sustain a high concurrency.

Message Failures
Consumption Data Exception

Last updated：2024-01-09 14:57:56

Issue Description

Consumption data is exceptional.

Troubleshooting Approaches

You can view the traffic monitoring data on the monitoring page in the CKafka console to check whether there is a surge, and if so, upgrade the instance specification for solution.
Check whether the limit of consumer groups has been reached.
If rebalancing occurs frequently for network reasons, we recommend you adjust the client timeout period.
Check whether expired offsets are pulled. Messages will be deleted after expiration, and if they are pulled with expired offsets, the pull will fail.

Failure to Delete Expired Messages Promptly

Last updated：2024-01-09 14:57:56

Issue Description

Expired messages are not deleted promptly.

Possible Causes

The message deletion mechanism of Kafka may cause a problem where expired messages are not deleted promptly in certain business scenarios. You may feel confused if you are unfamiliar with this mechanism. For example, the message timestamps in partition 0 and partition 7 are obviously different, but the expired messages in partition 0 are not deleted promptly.

Kafka Message Deletion Mechanism

Kafka data is stored in three dimensions: topic, partition, and data segment. The message data deletion conditions are as follows:
Message data is deleted in the unit of data segment based on the retention period.
Currently, the maximum size of a data segment is set to 1 GB. After a data segment reaches 1 GB in size, a new segment will be generated, and so on.
Only after all messages in a data segment expire will the segment be deleted.
If a message in a data segment is still within the retention period, such as the last row of a segment file, then the file will not be deleted.
For some reasons, messages are written unevenly and concentrated in a certain partition (such as partition 7), while some other partitions (such as partition 0) have little data. In this case, the size of the data segment in partition 0 does not reach 1 GB, so no new segments are generated; however, there is data in the entire data segment within the retention period. Therefore, messages in partition 0 will not be deleted.

Slow Consumption Speed

Last updated：2024-01-09 14:57:56

Issue Description

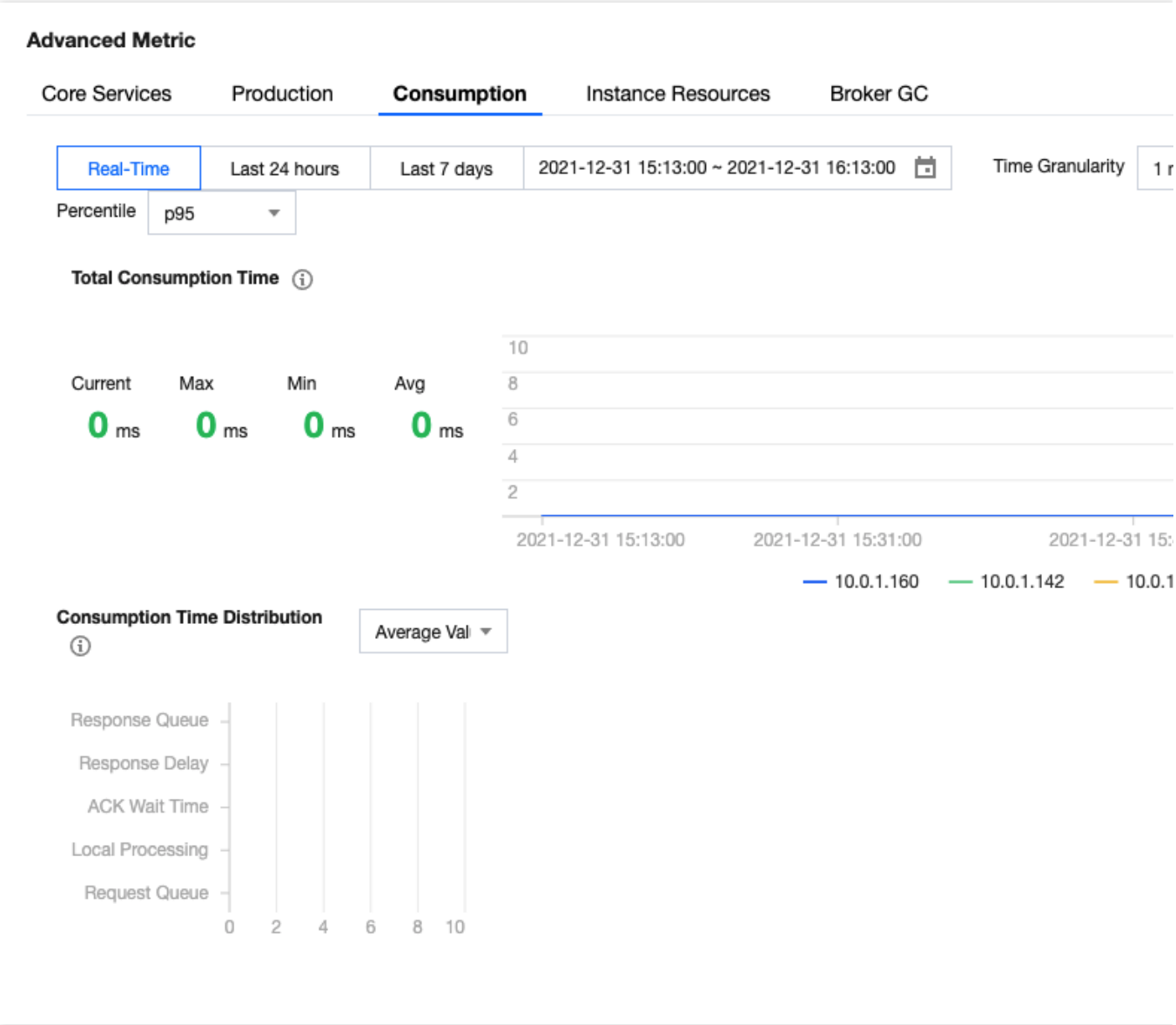
Message consumption is slow.

Possible Causes

High server load
Traffic throttling
Client load issue
Consumer's consumption capability
Network issue

Solutions

High server load
If you want to check whether it is a server issue, you can view the consumption time in **Advanced Monitoring** in the console, which indicates the time it takes for the server to process requests. If there is a server load issue, the time values in the statistical periods are relatively high as shown below:

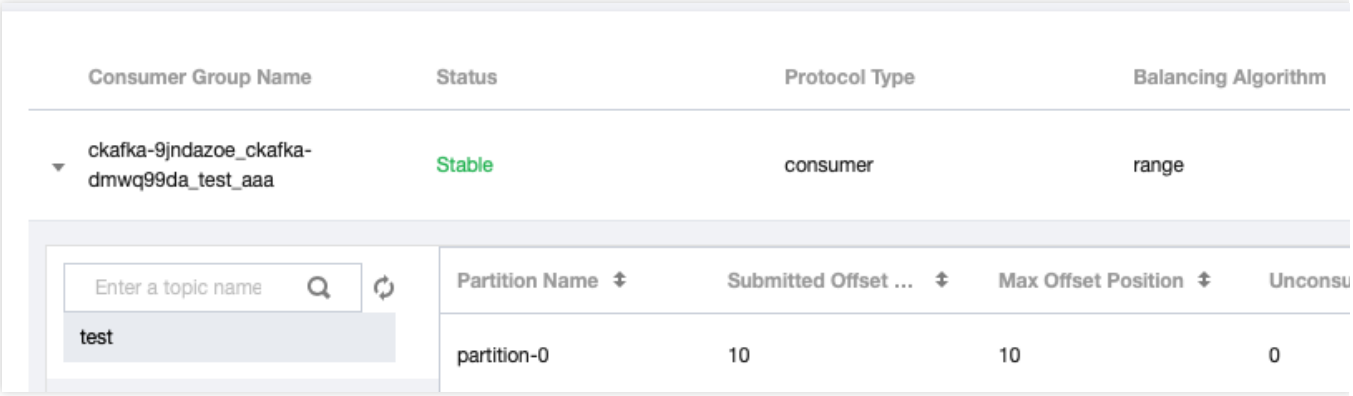


Traffic throttling

If you want to check whether it is caused by traffic throttling, you can configure an alarm for excessive bandwidth and then check whether the bandwidth peak of the instance has been reached in **Monitoring** > **Instance**; if so, you need to upgrade the bandwidth peak. For more information on how to upgrade the instance configuration, see Upgrading Instance.

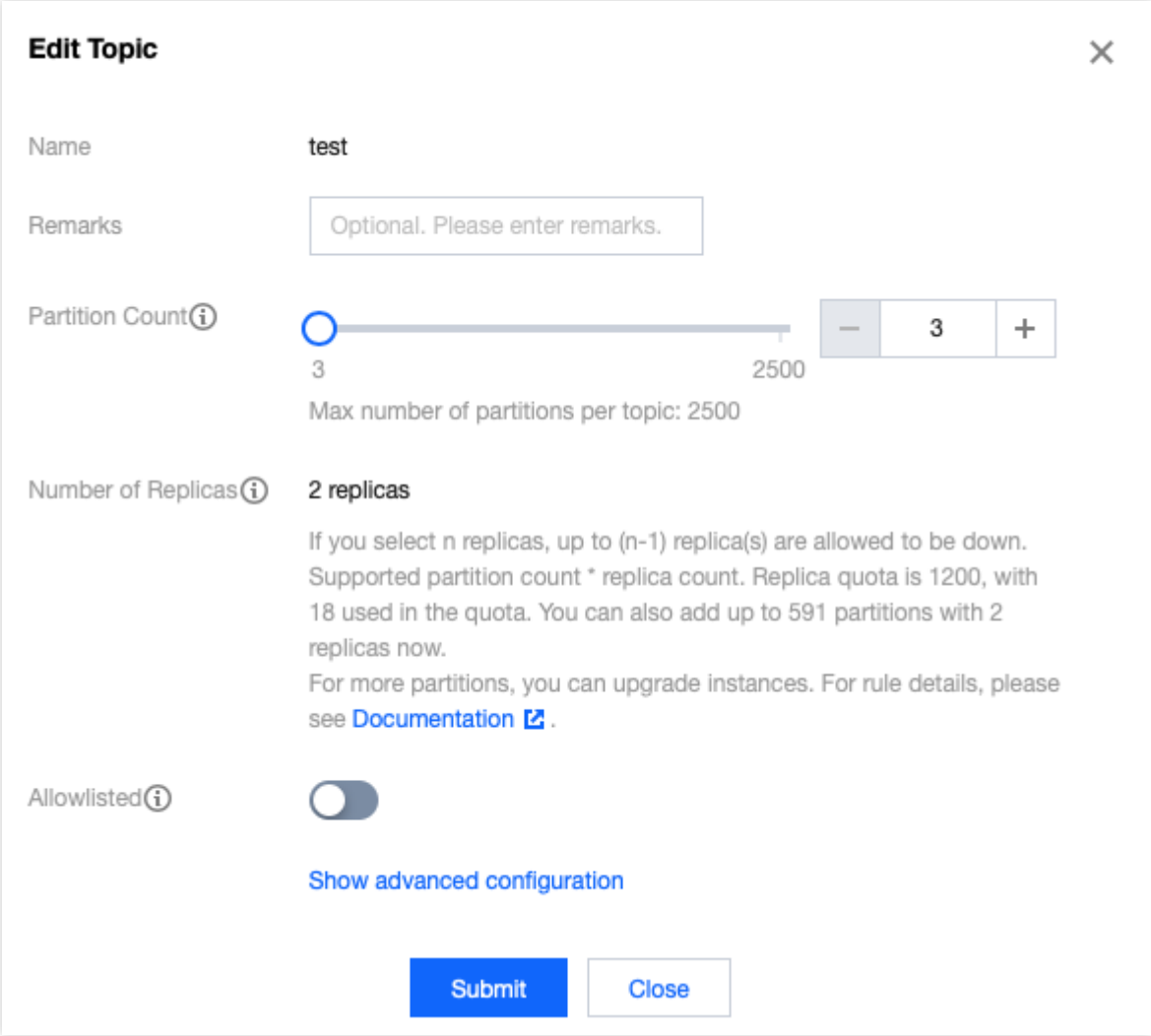
Client load issue

If the server doesn't have a performance issue, there is a high probability that the consumption capacity of the client is insufficient. First, take a look at the correspondence between partitions and consumers as shown below. If a consumer consumes too many partitions, we recommend you add more consumers, so that one consumer consumes one partition.



Consumer's consumption capability

If the assignment relationship between consumers and partitions is normal, you can add more partitions in the console to increase the concurrency of data consumption as shown below. Adding partitions in the console is instant and lossless, so your business won't be affected.



Network issue

Check the client load metrics, such as the client CPU, memory size, and network interface. If it is a Java process, you should pay attention to GC and heap memory usage.

Warning Displayed for Message Heap

Last updated：2024-01-09 14:57:56

Issue Description

A warning is displayed for message heap.

Possible Causes

The client has not consumed messages.
The client's consumption speed is slow.

Solutions

The client has not consumed messages
You can check the consumption speed of partitions to confirm whether there is consumption as shown below:
The client's consumption speed is slow
For more information, see Slow Message Consumption.

Recommended Settings

Open-Source Kafka supports setting a timestamp field and type in a message. Currently, two timestamp types are supported: `CreateTime` and `LogAppendTime`.
`CreateTime` indicates the local time on the client. As client time may vary from server time, check whether the entered time is correct. If it differs too much from the China Standard Time, the CKafka service will not be able to delete data promptly upon expiration based on the normal message retention period, which may cause exceptional message heap.
`LogAppendTime` indicates the time when a message is produced to the CKafka service, which is the CKafka server time and thus recommended.

Server Stress Testing

If you have questions about the server performance, you can run the following stress test commands to check whether there is a problem on the server:

Sample command for production testing:

```
bin/kafka-producer-perf-test.sh
--topic test
--num-records 123
--record-size 1000
--producer-props bootstrap.servers= ckafka vip : port
--throughput 20000
```

Sample command for consumption testing:

```
bin/kafka-consumer-perf-test.sh
--topic test
--new-consumer
--fetch-size 10000
--messages 1000
--broker-list bootstrap.servers=ckafka vip : port
```

For more information, see Conducting Production and Consumption Stress Testing on CKafka.

Error Persistence After a Period of Production

Last updated：2024-01-09 14:57:56

Issue Description

An error persists after a period of production.

Troubleshooting Approaches

View whether traffic throttling is performed. Check whether there is a surge in traffic on the monitoring page and upgrade the instance specification for solution.
View whether capacity control is performed. Check the instance disk capacity on the monitoring page and upgrade the instance specification or modify the data retention period for solution.