Document Database Service User Guide

dds_faq

Issue 01

Date 2023-01-30

HUAWEI TECHNOLOGIES CO., LTD.

Trademarks and Permissions


and other Huawei trademarks are trademarks of Huawei Technologies Co., Ltd.
All other trademarks and trade names mentioned in this document are the property of their respective holders.

Notice

The purchased products, services and features are stipulated by the contract made between Huawei and the customer. All or part of the products, services and features described in this document may not be within the purchase scope or the usage scope. Unless otherwise specified in the contract, all statements, information, and recommendations in this document are provided "AS IS" without warranties, guarantees or representations of any kind, either express or implied.

The information in this document is subject to change without notice. Every effort has been made in the preparation of this document to ensure accuracy of the contents, but all statements, information, and recommendations in this document do not constitute a warranty of any kind, express or implied.

Security Declaration

Vulnerability

Huawei's regulations on product vulnerability management are subject to the *Vul. Response Process.* For details about this process, visit the following web page:
https://www.huawei.com/en/psirt/vul-response-process
For vulnerability information, enterprise customers can visit the following web page:
https://securitybulletin.huawei.com/enterprise/en/security-advisory

Contents

1 Product Consulting

1.1 What Is the Relationship Between DDS and MongoDB Community Edition?

DDS is fully compatible with MongoDB Community Edition 3.4, 4.0, and 4.2, and partially compatible with MongoDB Community Edition 4.4. For details, see **Version Compatibility**.

DDS supports most of MongoDB commands. Any client compatible with MongoDB can connect to DDS for data storage and execution of related operations.

You can learn more about the advantages of DDS by **comparing DDS with on-premises databases**.

For details about DDS, see **What Is DDS?**

1.2 Q&A About Switching Storage Engine to RocksDB for DDS 4.2 and Later Versions

What Are the Differences Between RocksDB and WiredTiger?

- Data write method
  - Data is in the B+ tree structure in WiredTiger. A key in the flushed data corresponds to a version. Inserting, updating, or deleting a key directly operates the corresponding data node.
  - Data is in the LSM-tree structure in RocksDB. Data is written in the log appending mode with a key, a value, as well as a version. A key corresponds to multiple versions on a disk. Inserting, updating, or deleting data at the service layer is converted to writing data in RocksDB. When data of multiple versions is stacked to a certain threshold, the data is merged automatically. This is how compaction in RocksDB works.
- Data organization
  - In WiredTiger, data of a collection or index corresponds to a disk file. The size of the collection is the same as that of the disk file.

- In RocksDB, files of 64 MB each are distributed on a disk.

What Are the Advantages of RocksDB?

Table 1-1 lists the advantages of RocksDB over WiredTiger in the following aspects: open-source ecosystem, storage space, customizability, high memory pressure, frequent update and deletion, and multi-table scenario.

Table 1-1 Comparison between RocksDB and WiredTiger

Storage Engine	RocksDB	WiredTiger
Open-source ecosystem	Compared with WiredTiger, RocksDB has a better open-source ecosystem (GitHub Star). Some well-known databases, such as TiDB, CRDB, and YUGADB, use RocksDB as the storage engine.	The open-source community ecosystem is not as good as RocksDB.
Storage space	During RocksDB data writing, data of multiple versions is temporarily stored in some service scenarios. The data of multiple versions will be asynchronously merged into the final version. The storage space may temporarily increase.	WiredTiger flushes only the latest version of data to disks, consuming less storage space.
Customizability	In RocksDB, there are many performance tuning parameters. You can modify the parameters based on the read and write requirements of your services.	In WiredTiger, there are a few performance tuning parameters.
High memory pressure	The read and write performance of RocksDB is stable and does not change with the memory pressure.	WiredTiger needs to frequently flush dirty pages and swap memory and disk data. As a result, the read and write performance fluctuates greatly.
Frequent update and deletion	After a certain amount of new data is accumulated, the compact thread is automatically triggered to merge and aggregate data of multiple versions to release redundant disk space. Disk fragmentation does not occur.	After deleting data, WiredTiger merges and aggregates data of multiple versions, causing disk space fragments. However, WiredTiger does not return the disk space to the operating system. WiredTiger marks the disk space for subsequent writes of the current collection. Disk fragmentation is severe.
Multi-table scenario	RocksDB supports more than 10,000 tables and indexes.	If there are more than 1,000 tables or indexes in WiredTiger, the read and write performance is affected.

Why Does Huawei Cloud DDS 4.2 or Later Use RocksDB as the Storage Engine?

According to **Table 1-1**, RocksDB has many advantages over WiredTiger. Using the community-friendly RocksDB, Huawei Cloud DDS is compatible with APIs of multiple MongoDB versions, such as MongoDB 4.2, 4.4, and 5.0.

Will Services Be Affected If the Storage Engine of Huawei Cloud DDS 4.2 or Later Is Switched to RocksDB?

- Function: A storage engine is a component of a database and manages the storage mode of data in the memory and disks. Although RocksDB and WiredTiger are different storage engines, DDS encapsulates its storage engine and is compatible with community APIs. Developers are unaware of the usage differences and services are not affected.
- Performance: RocksDB has experienced a long-term community evolution. Based on custom development by the DDS team, RocksDB has no obvious difference in performance with WiredTiger. In some query scenarios, RocksDB has better performance, and its storage or compute resource usage is slightly different from WiredTiger.

What Are the Performance Differences and Optimization Suggestions for the RocksDB Storage Engine in Typical Scenarios?

Table 1-2 describes the performance differences between RocksDB and WiredTiger and optimization suggestions in some typical service scenarios.

Table 1-2 RocksDB performance differences and optimization suggestions

Scenario	RocksDB Performance Differences	Optimization Suggestions	Example
Multi-document scanning	RocksDB consumes more CPU and I/O resources than WiredTiger in scenarios where many documents need to be scanned.	Add appropriate indexes for optimization to improve the scanning efficiency and reduce the consumption of resources such as CPU and I/O.	For example, if no index is added to field A in the **coll** table, run **db.coll.find({A: 1})** to query based on the field.

Why Do Disk and CPU Usage Fluctuations Occur When Lots of Addition, Deletion, and Modification Operations Are Performed on Huawei Cloud DDS Using RocksDB?

- In RocksDB, data is written in appending mode. Updating or deleting data will not modify the original data. Instead, the data is appended to the end. Therefore, the same key may have values of different versions on disks. This causes a piece of data may have multiple versions, increasing disk usage.

- Values of different versions will be merged when the amount of newly written data reaches a certain threshold. This process calls compaction in RocksDB. Data is merged asynchronously. After a round of data merging completes, data of different versions is merged. After the data merging, a new single-version data file is generated and the old multiple-version data file is deleted. While a new file is being created and an old file is being deleted, both old and new files are stored and disk usage increases temporarily. After data merging completes, the disk usage decreases immediately.
- Data merging occupies a small number of CPUs, and there may be a slight CPU usage fluctuation.

If the disk usage and CPU usage remain high, contact Huawei technical support.

Does DDS 4.2 or Later Support Customized Storage Engine Switchover?

No support plan is available. Please stay tuned for the service announcement.

1.3 What Are the Differences Between DDS and GeminiDB Mongo?

Document Database Service (DDS) is a high performance high availability MongoDB-compatible database service that is secure and scalable. It provides a variety of functions including instance creation in just a few clicks, capacity scaling, disaster recovery, backup, restoration, and instance monitoring. For more information about DDS, see **DDS Overview**.

GeminiDB Mongo, also called enhanced MongoDB, is a distributed NoSQL database service from Huawei with decoupled compute and storage. GaussDB(for Mongo) is fully compatible with MongoDB APIs and provides high-performance, high-reliability, and enterprise-class services. For more information about GeminiDB Mongo, see **Overview**.

1.4 What Precautions Should Be Taken When Using DDS?

1. Failover

   DDS uses multiple mongos, replica sets, and shards to ensure data reliability. When a mongos is faulty, the other mongos takes over services immediately to ensure service continuity. A replica set contains multiple secondary nodes. When the primary node is faulty, DDS selects a secondary node as the new primary within 30 seconds.
2. ECSs used for instances are invisible to you. Your applications can access only the IP addresses and ports corresponding to the database.
3. Backup files stored on OBS are invisible to you. They are only visible in the DDS backend management system.
4. With DDS, you do not need to perform basic database O&M operations, such as enabling HA and installing security patches, but you should still note:

a. Whether the vCPUs, IOPS, and storage space for DDS DB instances are sufficient. If any of them is insufficient, optimize or upgrade the related configuration.

b. Whether DDS DB instances have performance issues, whether there are a large number of **slow queries**, whether SQL statements need to be optimized, and whether there are redundant or missing indexes.

1.5 What Is the Availability of DDS DB Instances?

Formula for calculating the DDS DB instance availability:

DDS DB instance availability = (1 – Failure duration/Total service duration) × 100%

1.6 Will My DDS DB Instances Be Affected by Other Users' DDS DB Instances?

No. Your DDS DB instances and resources are isolated from others.

1.7 Does DDS Support Multi-AZ Deployment?

You can deploy a cluster or replica set instance across up to three AZs.

- Cluster instance:

  For details, see **Creating a Cluster Instance Across Multiple AZs**.
- Replica set instance:

  For details, see **Creating a Replica Set Instance Across Multiple AZs**.

1.8 Can I Change the VPC for a Created Instance?

After a DDS instance is created, the VPC of the instance cannot be changed on the DDS console. Execution caution when you select a VPC during instance creation.

1.9 Can I Change the Region for a Created Instance?

After a DDS instance is created, the region of the instance cannot be changed on the DDS console.

1.10 What Is Hidden Node?

Hidden nodes in DDS are used for backing up data of cluster and replica set instances.

Hidden nodes are the same as other backup nodes except that they have different names and cannot function as primary nodes.

1.11 What Are Logical Sessions?

Overview

In DDS, server sessions or logical sessions are used to maintain and manage the client operation status on the server. An application uses client sessions to interact with server sessions.

Session Lifecycle

1. Creating a Session
   - Explicitly creating a session

     A session is created by the server when the client sends a request. Sessions are usually created by the driver through APIs. For example, run the following commands to create a session using mongo shell:

     ```
     // Start a session.
     session = db.getMongo().startSession( { readPreference: { mode: "primary" } } );
     ```

     The created session maintains a unique session ID on the server to identify and manage the session.

     ```
     session { "id" : UUID("1c6480a3-8e8d-4726-9f1b-6bf2f6b2b606") }
     ```

   - Implicitly creating a session

     Implicitly creating a session means that when certain operations are performed, the driver automatically creates a session without explicitly calling startSession. Implicit sessions are mainly used to simplify code and make operations more convenient. When a client performs a database operation without explicitly providing session parameters, the driver automatically creates a session. For example, when a document is inserted or data is queried, the driver implicitly creates a session in the background and automatically ends the session after the operation is complete.

CAUTION

By default, the maximum number of sessions that can be cached on the server is 1,000,000. When the number of sessions exceeds the value, sessions cannot be created explicitly or implicitly.

The following scenarios may occur when the number of sessions reaches the upper limit:

- New session rejected: If the number of sessions reaches the upper limit, DDS cannot create new sessions. This means that new operation requests will be rejected or blocked until existing sessions are released.
- Resource contention: A large number of sessions may cause resource contention, for example, increasing memory and CPU usage. This may affect the performance and response time of a database.
- Transaction failure: If you attempt to start a new transaction when the number of sessions reaches the upper limit, the transaction may fail. The client receives an error message indicating that the session resources are exhausted.

– Using sessions

When a session is used, all database operations including read, write, and transaction operations in the session context are bound to the session. Example:

```
coll1 = session.getDatabase("mydb1").foo;
coll2 = session.getDatabase("mydb2").bar;
coll1.insertOne( { abc: 1 } );
coll2.insertOne( { xyz: 999 } ).
```

– Terminating a session

A session can be terminated explicitly or automatically after timeout. You can call **endSession** to explicitly terminate a session.

```
session.endSession();
```

The server automatically clears the resources occupied by the terminated session.

2. Managing Sessions

– Transaction management

Sessions are the core of transaction management. At the beginning of a transaction, the server records the transaction operations in the session context to ensure that these operations are either all committed or all rolled back. Example:

```
db.getSiblingDB("mydb1").foo.insertOne(
{abc: 0},
{ writeConcern: { w: "majority", wtimeout: 2000 } }
)
db.getSiblingDB("mydb2").bar.insertOne(
{xyz: 0},
{ writeConcern: { w: "majority", wtimeout: 2000 } }
)
// Start a session.
session = db.getMongo().startSession( { readPreference: { mode: "primary" } } );
coll1 = session.getDatabase("mydb1").foo;
coll2 = session.getDatabase("mydb2").bar;
// Start a transaction
```



```
session.startTransaction( { readConcern: { level: "local" }, writeConcern: { w:
"majority" } } );
// Operations inside the transaction
try {
coll1.insertOne( { abc: 1 } );
coll2.insertOne( { xyz: 999 } );
} catch (error) {
// Abort transaction on error
session.abortTransaction();
throw error;
}
// Commit the transaction using write concern set at transaction start
session.commitTransaction();
session.endSession();
```

- Session timeout

  A session has a default timeout interval, which is 30 minutes by default. The server automatically terminates inactive sessions after timeout to release resources.

How Do Logical Sessions Work

1. Session ID

   Each session has a unique session ID, which remains unchanged throughout the life cycle of the session. A session ID is used to trace and manage operations within a session.

2. Session resource management

   The server dynamically allocates resources based on session activities. Unused session metadata is automatically cleared. When **writeConcern** for clearing session metadata is set to **majority**, residual session metadata cannot be cleared if the majority of standby nodes have a latency. This can result in accumulated session metadata on the server and increase the number of sessions, even after the session has been terminated. To prevent excessive resource usage, the server periodically checks the status of sessions that have been running for a long time.

Best Practices

1. You are advised to specify **w** to **majority** when connecting to a DDS instance to prevent implicit residual sessions caused by long primary/secondary latency. In addition, if data is not written using **w=majority**, the data that is not synchronized to the secondary node may be lost when a primary/secondary switchover occurs.
2. Explicitly invoke **endSession** after operations are complete to terminate a session to release resources.
3. Minimize operations performed within a transaction to reduce locking and resource usage.

2 Database Versions

2.1 Does DDS Support Version Upgrade?

Major Version Upgrade

DDS does not support major engine version upgrade on the console. You can use DRS to migrate data as required. For details, see **Upgrading a Major Version**.

Minor Version Upgrade

DDS supports manual minor version upgrades, which can improve performance, add new functions, and fix bugs.

For details, see **Upgrading a Minor Version**.

3 Resource Freezing, Release, Deletion, and Unsubscription

Why Are My DDS Resources Released?

If your subscriptions have expired but not been renewed, or you are in arrears due to insufficient balance, your resources enter a grace period. If you still do not complete the payment or renewal after the grace period expires, your resources will enter a retention period. During the retention period, the resources are not available. If the renewal is still not completed or the outstanding amount is still not paid off when the retention period ends, the stored data will be deleted and the cloud service resources will be released. For details, see **Service Suspension and Resource Release**.

Why Are My DDS Resources Frozen?

Your resources may be frozen for a variety of reasons. The most common reason is that you are in arrears.

Can I Still Back Up Data If My DB Instance Is Frozen?

No. If your DDS instance is frozen due to arrears, you need to unfreeze the instance first.

How Do I Unfreeze My Resources?

Frozen due to arrears: You can renew your resources or top up your account. DDS instances frozen due to arrears can be renewed, released, or deleted. Yearly/Monthly DDS instances that have expired cannot be unsubscribed from, while those that have not expired can be unsubscribed from.

What Happens When My Resources Are Frozen, Unfrozen, or Released?

- After your resources are frozen:
  - They cannot be accessed, causing downtime. For example, if your DDS instance is frozen, it cannot be connected.
  - If they are yearly/monthly resources, no changes can be made to them.

  - They can be unsubscribed from or deleted manually.
- After your resources are unfrozen, you can connect to them again.
- If your resources are released, your instances will be deleted. Before the deletion, DDS determines whether to **move the instance to the recycle bin** based on the recycling policy you specified.

How Do I Renew My Resources?

After a yearly/monthly DDS instance expires, you can renew it on the **Renewals** page. For details, see **Renewal Management**.

Can My Resources Be Recovered After Being Released? /Can I Retrieve an Incorrect Unsubscription?

If your instance is moved to the recycle bin after being deleted and is within the retention period, you can **rebuild** it from the recycle bin. Otherwise, data cannot be restored.

Before unsubscribing from a resource, confirm the resource information carefully. If you have unsubscribed from an instance by mistake, purchase a new one.

How Can I Delete a DDS Instance?

- For pay-per-use instances, see **Deleting a Pay-per-Use Instance**.
- For yearly/monthly instances, see **Unsubscribing from a Yearly/Monthly Instance**.

4 Resource and Disk Management

4.1 Which Items Occupy the Storage Space of DDS Instances?

The following types of data will occupy the storage space:

- The storage space you applied for will contain the system overhead required for inode, reserved block, and database operation.
- User data (backups not included)
- Data required to ensure instances run properly, such as system database data, rollback logs, and search indexes.
- Log output files that are generated by DDS ensure the stable operating of DDS instances. For example, oplogs occupy 10% of the storage space by default. The retention period of rotation oplogs depends on the oplog generation rate. To change the defalut value, change the value of **oplogSizePercent** on the console. For details, see **Modifying DDS DB Instance Parameters**.

4.2 Which Types of Logs and Files Occupy DDS DB Instance Storage Space?

Logs and files listed in the following table occupy storage space.

Database Type	File Type
DDS	Log files: mongod.log, mongos.log, configsvr.log and mongodb_error.log
	Data files: database content and index file
	Other files: DDS temporary files and YAML configuration files.

4.3 Why Is the Storage Space Usage Displayed on the GUI Smaller Than the Actual Usage?

Data stored on DDS disks is compressed before being stored, so the storage usage displayed on the GUI is less than the amount of data stored.

4.4 Why Does Available Disk Space Not Increase After Data Is Deleted?

Operations such as write, update, and delete (including index insert and delete) are actually converted to write operations in the background. When data of an instance in use is deleted, the disk space is not reclaimed. Such unreclaimed disk space is called disk fragments. When new data is inserted, these fragments are reused without applying for new disk space. Different underlying storage engines (RocksDB and WiredTiger) vary according to specific scenarios.

After deleting data, RocksDB directly converts the **delete** operation to append write. After a certain amount of redundant data is accumulated, the background compact thread is automatically triggered to merge and aggregate data of multiple versions to release redundant disk space. You are advised to wait for the system to automatically reclaim the disk space. If the disk space usage is high and close to the **read-only** threshold, contact Huawei technical support.

After deleting data, WiredTiger merges and aggregates data of multiple versions, causing disk space fragments. However, WiredTiger does not return the disk space to the operating system. WiredTiger marks the disk space for subsequent writes of the current collection, the reserved disk space is preferentially used for subsequent writes of the collection. To release the disk space, run the **compact** command. (Note: This command blocks normal services and is disabled by default.)

4.5 Why Is the Resident Memory of a 4 vCPUs/8 GB Memory Replica Set Instance Only 4 GB?

If 100% of the resident memory is occupied, the system is stuck and services cannot run properly.

5 Capacity Expansion and Specification Changes

5.1 Can a DDS Instance Type Be Changed?

The instance type cannot be changed, but you can use Data Replication Service (DRS) to migrate data between different types of instances.

For example, if you want to change a replica set instance to a cluster instance, you can purchase a cluster instance and use DRS to migrate data from the existing replica set instance to the cluster instance.

For details about data migration, see **Migrating Data Using DRS**.

5.2 Can Nodes Be Added to DDS Instances?

You can add nodes to cluster and replica set instances. Single-node instances do not support adding nodes.

- Cluster instance

  A cluster instance consists of a config node, and multiple dds mongos and shard nodes. Additional shard and dds mongos nodes can be added, but the number of config nodes is fixed.

  For details, see **Adding Cluster Instance Nodes**.

- Replica set instance

  A replica set consists of the primary node, secondary node, and hidden node. By default, the system provides a three-node replica set. You can add nodes to a five-node or seven-node replica set as required. The newly added nodes are all secondary nodes.

  For details, see **Adding Replica Set Instance Nodes**.

5.3 Is DDS Available When Nodes Are Being Added?

Yes. Services are not affected by adding shard nodes. The existing shard nodes are not affected.

6 Database Performance

6.1 When Will a Primary/Standby Switchover Be Triggered for a Cluster or Replica Set?

Causes

The trigger for a switchover depends on the architectures: cluster or replica set.

- A cluster consists of three types of nodes: dds mongos, shard, and config. The shard and config nodes use a three-node replica-set architecture. If a primary node becomes faulty, it triggers a switchover.
- In a replica-set architecture there are also three nodes: primary, secondary, and hidden. The primary and secondary nodes provide IP addresses for external access. The primary node of a replica set instance is not fixed. If the configuration of the replica set instance changes, or if the primary node fails, or a switchover is triggered. A secondary node is promoted to primary, and the originally primary node is demoted to secondary.
- If the memory usage is too high, the instance is under heavy load and primary/standby switchover will occur.

For more information, see **Cluster Architecture** and **Replica Set Architecture**.

Impacts

- If a primary node fails, the system selects a standby node as the new primary within 30 seconds.
- If your applications are connected to a primary node that is demoted to standby due to a primary/standby switchover, the read and write operations on services will be affected.

Service Deployment Suggestion

Ensure that your application supports automatic reconnection, so the application can be automatically reconnected to the instance and avoid write errors in the event of a brief disconnects.

- It is recommended that you use URLs to connect the cluster and replica set instances. When a node is faulty, the primary/standby switchover does not affect service read and write operations. For details, see **Connecting to a Cluster Instance** and **Connecting to a Replica Set Instance for Read and Write Separation and High Availability**.
- If an instance is under heavy load, you can change the instance class. Take a cluster instance as an example. For details, see **Changing a Cluster Instance Class**.

6.2 High Storage Usage

If the storage usage of a DDS instance is too high or fully used, the instance becomes unavailable.

This section describes how to analyze and fix high storage usage.

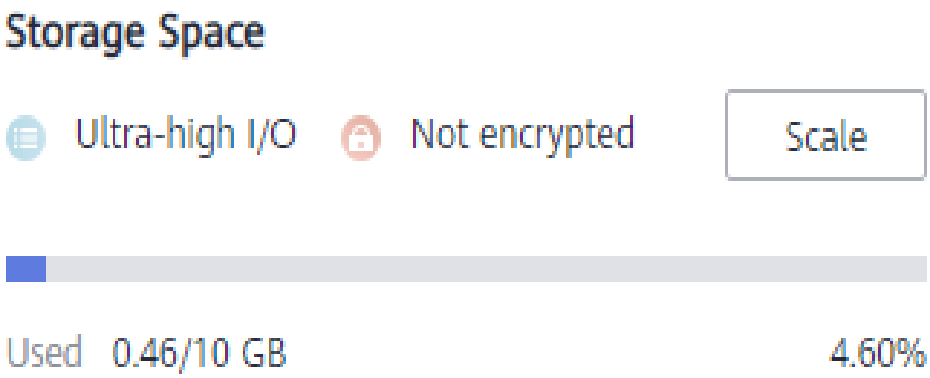
Checking the Storage Usage

DDS provides the following two methods to check the storage usage of an instance:

1. Check the storage usage on the DDS console.

   You can log in to the DDS console and click the instance. On the **Basic Information** page, you can view the storage space of the instance in the **Storage Space** area.

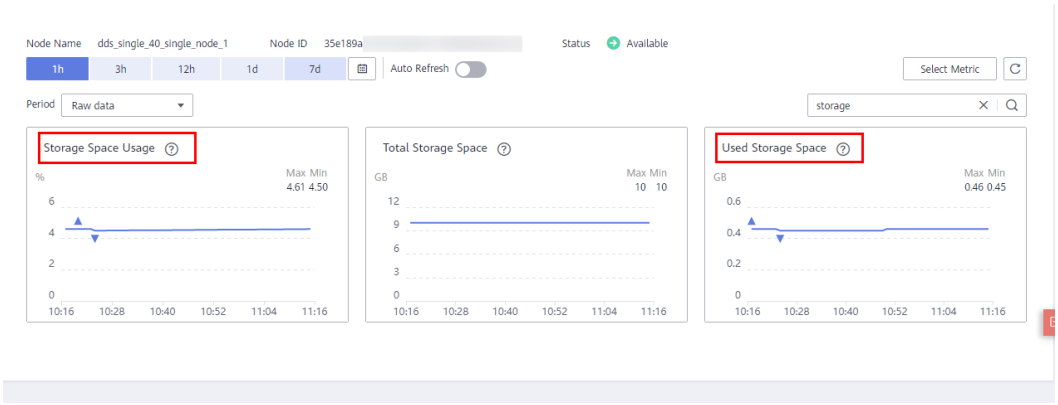
   Figure 6-1 Checking the storage usage

2. View the monitoring metrics (storage usage and used storage).

   To view monitoring metrics, see **Viewing Monitoring Metrics**.

Figure 6-2 Checking the storage usage



Solution

1. For cluster instances, data may be unevenly distributed because the database collection is not properly sharded. As a result, the storage usage is high.

   To shard the database collection properly, see **How Do I Improve Database Performance by Configuring Sharding?**

2. As service data increases, the original database storage is insufficient. You can expand the storage space to fix this problem.
   - To scale up storage for cluster instances, see **Scaling Up a Cluster Instance**.
   - To scale up storage for replica set instances, see **Scaling Up a Replica Set Instance**.
   - To scale up storage for single node instances, see **Scaling Up a Single Node Instance**.

   If the storage space has reached the upper limit of your instance class, change the instance class first.
   - To change the cluster instance class, see **Changing a Cluster Instance Class**.
   - To change the replica set instance class, see **Changing a Replica Set Instance Class**.
   - To change the single node instance class, see **Changing a Single Node Instance Class**.

3. If a large number of expired files occupy the storage space, delete the expired files in time. For example, if the entire database is no longer used, run **dropDatabase** to delete it.

4. The background data processing mechanism is faulty.

   Operations such as write, update, and delete (including index insert and delete) are actually converted to write operations in the background. When data of an instance in use is deleted, the disk space is not reclaimed. Such unreclaimed disk space is called disk fragments. When new data is inserted, these fragments are reused without applying for new disk space. Different underlying storage engines (RocksDB and WiredTiger) vary according to specific scenarios.

   After deleting data, RocksDB directly converts the **delete** operation to append write. After a certain amount of redundant data is accumulated, the

background compact thread is automatically triggered to merge and aggregate data of multiple versions to release redundant disk space. You are advised to wait for the system to automatically reclaim the disk space. If the disk space usage is high and close to the **read-only** threshold, contact Huawei technical support.

After deleting data, WiredTiger merges and aggregates data of multiple versions, causing disk space fragments. However, WiredTiger does not return the disk space to the operating system. WiredTiger marks the disk space for subsequent writes of the current collection, the reserved disk space is preferentially used for subsequent writes of the collection. To release the disk space, run the **compact** command. (Note: This command blocks normal services and is disabled by default.)

6.3 What Is the Time Delay for Primary/Secondary Synchronization in a Replica Set?

The delay for primary/secondary synchronization cannot be calculated using a formula. The delay is affected by the following factors:

1. Network communication status
2. Transaction pressure on the primary node, that is, transactions per second (TPS) of the primary node
3. Transaction size executed by the primary node, that is, the duration of a transaction execution
4. Load of the secondary node

If the primary node bears heavy pressure within a period and executes a large number of transactions per second, the synchronization to the secondary node is delayed.

You can view metric "Delay Between Primary and Secondary Nodes" of the secondary node on the Cloud Eye console to know the synchronization delay.

6.4 How Is Data Transferred Between the Primary and Secondary Nodes of a Replica Set?

Data between the primary and secondary nodes is transferred in asynchronous mode.

6.5 How Do I Clear an Alarm Saying the Shard Memory Usage Exceeds 90%?

You can change the value of **enableMajorityReadConcern** to **false** and then restart the node.

6.6 What Can I Do If a Query Error Is Reported After Data Is Written Into the DDS Cluster?

Symptom

Data is continuously written into the DDS cluster. After the data is written into the DDS cluster, an error is reported during query.

Example:

```
W SHARDING [Balancer] Failed to enforce tag ranges :: caused by :: ExceededTimeLimit: Unable to obtain shard utilization information for shard01 due to Operation timed out, request was RemoteCommand -- target: 192.168.*.*:8635, db: admin, cmd:{ getShardStatistics: 1, maxTimeMS: 30000 }
```

Possible Cause

Data is continuously written into the storage system, causing chunk splitting timeout in the background.

Solution

This is a normal warning error and does not affect operations. You can try again later.

7 Database Permissions

7.1 What Permissions Does User root Have?

After an instance is created, DDS creates a user **root** in the **admin** database. This user has the built-in root permission of MongoDB. For details about MongoDB built-in roles, see **Built-In Roles**.

You can log in to the instance and run the following commands to view the permissions of current users:

show users

show roles

7.2 Default Permission Mechanism

DDS has a series of security enhancements to deal with increasingly severe security challenges. The Community Edition allows you to connect to a database without authentication. In DDS, you must pass the authentication before connecting to the database. Otherwise, the database cannot be used.

- After a DB instance is created, the system creates the default administrator **rwuser**. The administrator must be specified by the customer and meet the password complexity requirements.
- The administrator is used to create and manage user-defined roles and users. The administrator does not have a default password. The password must be specified by the customer and meet the password complexity requirements.

Scenario

During the execution of mongodump and mongorestore, if you back up and restore the entire DB instance, the permission verification fails. This is because user **rwuser** has limited permissions on the **admin** and **config** databases of the DB instance. You need to grant permissions on certain databases and tables to the user.

7.3 Role Management

DDS uses role-based management to control users' data access permissions. Roles are classified into built-in roles and user-defined roles.

7.3.1 Built-In Roles

Built-in roles are automatically generated by the system. The built-in roles read and readWrite can be used by clients.

MongoDB uses roles to manage databases, so you need to assign a role to a user when creating the user. In addition to built-in roles, you can also create user-defined roles (**User-Defined Roles**).

Table 7-1 Common built-in roles

Role	Permission	Actions
read	The read role provides permissions to read data on all non-system collections and some system collections (system.indexes, system.js, and system.namespaces).	changeStream, collStats, dbHash, dbStats, find, killCursors, listIndexes, listCollections
readWrite	The readWrite role provides all the permissions of the read role plus ability to modify data on all non-system collections and the system.js collection.	collStats, convertToCapped, createCollection, dbHash, dbStats, dropCollection, createIndex, dropIndex, find, insert, killCursors, listIndexes, listCollections, remove, renameCollectionSameDB, update
readAnyDatabase	The readAnyDatabase role provides the read-only permissions on all databases except local and config. The role also provides the listDatabases action on the cluster as a whole.	In MongoDB 3.4 and earlier, this role provides the read permission for the config and local databases. In the current version, to provide read permissions on the config and local databases, create a user in the admin database with read role in the config and local databases.
readWriteAnyDatabase	The readWriteAnyDatabase role has the read and write permissions for all databases except config and local. The role also provides the listDatabases action on the cluster as a whole.	In MongoDB 3.4 and earlier, this role has the read and write permissions for the config and local databases. In the current version, if you want to read or write data from or to the config and local databases, create a user in the admin database with the readWrite role in the config and local databases.

Role	Permission	Actions
dbAdmin	The dbAdmin role provides the ability to perform administrative tasks such as schema-related tasks, indexing, and gathering statistics. This role does not grant permissions for user and role management.	• For system collections (system.indexes, system.namespaces, and system.profile), the actions include collStats, dbHash, dbStats, find, killCursors, listIndexes, listCollections, dropCollection, and createCollection (applicable only to system.profile). • For non-system collections, the actions include bypassDocumentValidation, collMod, collStats, compact, convertToCapped, createCollection, createIndex, dbStats, dropCollection, dropDatabase, dropIndex, enableProfiler, reIndex, renameCollectionSameDB, repairDatabase, storageDetails, and validate.
dbAdminAnyDatabase	The dbAdminAnyDatabase role provides the same database management permissions as dbAdmin on all databases except local and config. The role also provides the listDatabases action on the cluster as a whole.	In MongoDB 3.4 and earlier, this role has the management permissions for the config and local databases. In the current version, if you want to manage the two databases, create a user in the admin database with the dbAdmin role in the config and local databases.
clusterAdmin	The clusterAdmin role has the greatest cluster-management access.	This role combines the permissions granted by the clusterManager, clusterMonitor, and hostManager roles, and provides the dropDatabase action.

Role	Permission	Actions
userAdmin	The userAdmin role contains the permissions to create and modify roles and users in the current database. This role allows users to grant any permission to any other user (including themselves). This role also indirectly provides the superuser (root) access to either the database or, if scoped to the admin database, the cluster.	changeCustomData, changePassword, createRole, createUser, dropRole, dropUser, grantRole, revokeRole, setAuthenticationRestriction, viewRole, viewUser
userAdminAnyDatabase	The userAdminAnyDatabase role has the permissions similar to the userAdmin role. It manages all databases except the config and local databases.	• This role contains the following cluster commands: the authSchemaUpgrade, invalidateUserCache, and listDatabases • For system collections (admin.system.users and admin.system.roles), the actions include collStats, dbHash, dbStats, find, killCursors, planCacheRead, createIndex, and dropIndex.

7.3.2 User-Defined Roles

A user-defined role is a customized role created by a user by running a command. It contains only one or more CRUD operations or one or more built-in roles. You can customize roles based on different resources and actions. User-defined roles are applied in the same way as built-in roles.

Creating, Modifying, and Deleting Roles

- Before creating a role, connect to the DB instance as a user with the required permission (for example, rwuser). For details, see **Connecting a Database**.
- You can use createRole to create a user-defined role to control permissions for different databases and collections or inherit permissions from other roles.
- After a role is created, you can run grantPrivilegesToRole, grantRolesToRole, revokeRolesFromRole, or revokePrivilegesFromRole to obtain or revoke permissions of the role. For details, see **Creating and Managing Roles**.

7.3.3 Creating and Managing Roles

Creating a Role

db.createRole(role, writeConcern)

- **role** is mandatory and its type is document. The details are as follows:

```
{
  role: "<name>",
  privileges: [
    { resource: { <resource> }, actions: [ "<action>", ... ] },
    ...
  ],
  roles: [
    { role: "<role>", db: "<database>" } | "<role>",
     ...
  ],
  authenticationRestrictions: [
   {
     clientSource: ["<IP>" | "<CIDR range>", ...],
     serverAddress: ["<IP>" | "<CIDR range>", ...]
   },
   ...
  ]
}
```

Parameter description

Field	Type	Description
role	string	Role name
privileges	Array	This parameter is mandatory. The array elements indicate the permissions of a role. If this parameter is set to an empty collection, the role does not have any permission.
resource	Documents	The database name or collection name.

Field	Type	Description
actions	Array	List of available operations. Common actions are as follows: • find • count • getMore • listDatabases • listCollections • listIndexes • insert • update • remove For more actions, see the **official document**.
roles	Array	Array element. This parameter is mandatory. The array element is the name of a role inherited by the role. The role can be a preset role read or readWrite or a user-defined role.
authenticationRestrictions	Array	Optional. This parameter specifies the IP address or IP address segment that can be accessed by the role.

- writeConcern specifies the write concern level of a command.

Updating a Role

db.grantPrivilegesToRole(rolename,privileges,writeConcern)

db.revokePrivilegesFromRole(rolename,privileges,writeConcern)

The preceding commands are used to obtain or revoke specified permissions for a role.

- **rolename** specifies the name of the role to be updated. This parameter is mandatory.
- **privileges** indicates the permissions to be adjusted for the role.

```
db.grantPrivilegesToRole(
  "< rolename >",
```



```
  [
    { resource: { <resource> }, actions: [ "<action>", ... ] },
    ...
  ],
  { < writeConcern > }
)
```

Table 7-2 privileges description

Field	Type	Description
resource	Document	The database name or collection name.
actions	Array	For details, see description about createRole.

In addition to the preceding commands, updateRole can also be used to update role information.

db.updateRole(role, update, writeConcern)

Table 7-3 Parameter description

Field	Type	Description
role	string	Role name
update	Array	Mandatory. Its meaning is the same as that of privileges in the command for creating a role. It is used to replace all permission information of a role.
writeConcern	Document	writeConcern specifies the write concern level of a command.

Deleting a Role

db.dropRole(rolename, writeConcern)

- **rolename** specifies the name of the role to be deleted. This parameter is mandatory.
- **writeConcern** specifies the write concern level of a command.

7.4 User Management

In DDS, user permissions are managed based on roles. Differentiated permission control is implemented by assigning different roles to users.

To provide management services for DDS DB instances, admin, monitor, and backup accounts are automatically created when you create a DDS DB instance. Attempting to delete, rename, change the passwords, or change privileges for these accounts will result in errors.

You can change the password of the database administrator **rwuser** and any accounts you create.

7.4.1 Creating a User

Precautions

- All the following operations require permissions. By default, user **rwuser** has the required permissions. If a user-defined user is used for management, check whether the user has the required permissions.
- Connect to a DB instance as a user who has the required permission (for example, **rwuser**).
- You can use createUser to create required users and configure roles to control user rights. Note that the **passwordDigestor** parameter must be set to **server**. Otherwise, the command fails to be executed. This restriction is added to prevent security risks.

Creating a User

db.createUser(user, writeConcern)

- In the command, **user** is mandatory and the type is document. It contains the identity authentication and access information of the user to be created.
- **writeConcern** is an optional parameter of the document type. It contains the write concern level of the creation operation.

The **user** document defines users. The format is as follows:

```
{
  user: "<name>",
  pwd: "<cleartext password>",
  customData: { <any information> },
  roles: [
    { role: "<role>", db: "<database>" } | "<role>",
    ...
  ],
  authenticationRestrictions: [
     {
       clientSource: ["<IP>" | "<CIDR range>", ...],
       serverAddress: ["<IP>" | "<CIDR range>", ...]
     },
     ...
  ]
  mechanisms: [ "<SCRAM-SHA-1|SCRAM-SHA-256>", ... ],
  passwordDigestor: "<server|client>"
}
```


Table 7-4 Description of parameter user

Field	Type	Description
user	string	The new username.
pwd	string	User password. If you run **db.createUser()** on the **$external** database to create a user who stores credentials outside of MongoDB, the **pwd** field is not required.
customData	Document	Optional. Any information, which can be used to store any data that the administrator wants to associate with this particular user. For example, this could be the user's full name or employee ID.
roles	Array	The role assigned to the user. You can specify an empty array [] to create a user without a role.
authenticationRestrictions	Array	Optional. The authentication restrictions forcibly imposed by the server on the created user. It is used to specify the IP address or IP address segment that can be accessed by the role.
mechanisms	Array	Optional. The specific SCRAM mechanism or mechanisms for the user credentials. Valid values are SCRAM-SHA-1 and SCRAM-SHA-256.
passwordDigestor	string	Optional. Whether to verify the password on the server or client. The default value is server.

Example

- **Assigning Different Roles to Different Databases During User Creation**

  The following describes how to use **db.createUser()** to create user **accountAdmin01** in database **products**.

  ```
  use products
  db.createUser( { user: "accountAdmin01",
                   pwd: "Changeme_123",
                   customData: { employeeId: 12345 },
                   roles: [ { role: "clusterAdmin", db: "admin" },
                            { role: "readAnyDatabase", db: "admin" },
                            "readWrite"] },
                 { w: "majority" , wtimeout: 5000 } )
  ```

  The preceding operations assign the following roles to user **accountAdmin01**:

  – Roles clusterAdmin and readAnyDatabase in the **admin** database

  – Role readWrite in the **products** database

- **Assigning Different Roles to a Database During User Creation**

  The following describes how to create a user named **accountUser** whose roles are readWrite and dbAdmin in the **products** database.

  ```
  use products
  db.createUser(
    {
      user: "accountUser",
      pwd: "Changeme_123",
      roles: [ "readWrite", "dbAdmin" ]
    }
  )
  ```

- **No Assigning Any Role During User Creation**

  The following describes how to create a user named **reportsUser** with no role assigned in the **admin** database.

  ```
  use admin
  db.createUser(
    {
      user: "reportsUser",
      pwd: "Chagneme_123",
      roles: [ ]
    }
  )
  ```

- **Creating an Administrator and Assigning a Role to the Administrator**

  The following describes how to create a user named **appAdmin** in the **admin** database and grant the user the read and write permissions on the **config** database so that the user can change some settings of a sharded cluster, such as the shard balancer settings.

  ```
  use admin
  db.createUser(
    {
      user: "appAdmin",
      pwd: "Changeme_123",
      roles:
        [
          { role: "readWrite", db: "config" },
          "clusterAdmin"
        ]
    }
  )
  ```


- **Creating a User with Authentication Restrictions**

  The following describes how to create a user named **restricted** in the **admin** database. User authentication is required only when the user connects 192.0.2.0 to 198.51.100.0.

  ```
  use admin
  db.createUser(
    {
      user: "restricted",
      pwd: "Changeme_123",
      roles: [ { role: "readWrite", db: "reporting" } ],
      authenticationRestrictions: [ {
         clientSource: ["192.0.2.0"],
         serverAddress: ["198.51.100.0"]
      } ]
    }
  )
  ```

- **Creating a User Using Only the SCRAM-SHA-256 Certificate**

  The following describes how to create a user with only the SCRAM-SHA-256 certificate.

  ```
  use reporting
  db.createUser(
    {
      user: "reportUser256",
      pwd: "Changeme_123",
      roles: [ { role: "readWrite", db: "reporting" } ],
      mechanisms: [ "SCRAM-SHA-256" ]
    }
  )
  ```

  If the **authenticationMechanisms** parameter is set, the **mechanisms** field can contain only the values specified in the **authenticationMechanisms** parameter.

7.4.2 Updating a User

db.updateUser(username, update, writeConcern)

- **username** indicates the username to be updated.
- **update** is a document containing the replacement data for the user.
- **writeConcern**: The write concern level of the update operation. This parameter is optional.

```
db.updateUser(
  "<username>",
  {
    customData : { <any information> },
    roles : [
      { role: "<role>", db: "<database>" } | "<role>",
      ...
    ],
    pwd: passwordPrompt(),      // Or  "<cleartext password>"
    authenticationRestrictions: [
      {
        clientSource: ["<IP>" | "<CIDR range>", ...],
        serverAddress: ["<IP>", | "<CIDR range>", ...]
      },
      ...
    ],
```



```
    mechanisms: [ "<SCRAM-SHA-1|SCRAM-SHA-256>", ... ],
    passwordDigestor: "<server|client>"
  },
  writeConcern: { <write concern> }
)
```

Table 7-5 update description

Field	Type	Description
customData	Documents	Optional. Any information.
roles	Array	Optional. The role assigned to the user. An update to the roles array overrides the previous array's values.
pwd	string	Optional. The user's password.
authenticationRestrictions	Array	Optional. The IP address or CIDR blocks that can be accessed by a role.
mechanisms	Array	Optional. The specific SCRAM mechanism or mechanisms for the user credentials. Valid values are SCRAM-SHA-1 and SCRAM-SHA-256.
passwordDigestor	string	Optional. Whether to verify the password on the server or client. The default value is server.

Example

- **Updating User Information**

  The information about the **appClient01** user in the **products** database is as follows:

```
{
   "_id" : "products.appClient01",
   "token" : NumberLong("8424642624807814713"),
   "user" : "appClient01",
   "db" : "products",
   "customData" : {
      "empID" : "12345",
      "badge" : "9156"
   },
   "roles" : [
      {
         "role" : "readWrite",
         "db" : "products"
```



```
        },
        {
            "role" : "read",
            "db" : "inventory"
        }
    ],
    "mechanisms" : [
        "SCRAM-SHA-1",
        "SCRAM-SHA-256"
    ]
}
```

The following describes how to update the user-defined data and role data.

```
use products
db.updateUser( "appClient01",
{
  customData : { employeeId : "0x3039" },
  roles : [
    { role : "read", db : "assets"  }
  ]
} )
```

The updated information about the **appClient01** user in the **products** database is as follows:

```
{
    "_id" : "products.appClient01",
    "token" : NumberLong("8424642624807814713"),
    "user" : "appClient01",
    "db" : "products",
    "customData" : {
        "employeeId" : "0x3039"
    },
    "roles" : [
        {
            "role" : "read",
            "db" : "assets"
        }
    ],
    "mechanisms" : [
        "SCRAM-SHA-1",
        "SCRAM-SHA-256"
    ]
}
```

- **Updating User Information to Be a User with Only the SCRAM-SHA-256 Certificate**

  The information about the **reportUser256** user in the **reporting** database is as follows:

```
{
    "_id" : "reporting.reportUser256",
    "token" : NumberLong("2827251846225877395"),
    "user" : "reportUser256",
    "db" : "reporting",
    "roles" : [ ],
    "mechanisms" : [
        "SCRAM-SHA-1",
        "SCRAM-SHA-256"
    ]
}
```


The following describes how to change the current user with both SCRAM-SHA-256 and SCRAM-SHA-1 certificates to a user with only the SCRAM-SHA-256 certificate.

CAUTION

- If the password is not specified with mechanisms, mechanisms can only be updated to a subset of the user's current SCRAM mechanism.
- If the password is specified with mechanisms, you can specify any supported SCRAM mechanism.
- For SCRAM-SHA-256, **passwordDigestor** must be set to the default value **server**.

```
db.updateUser(
  "reportUser256",
  {
    mechanisms: [ "SCRAM-SHA-256" ]
  }
)
```

The updated information about the **reportUser256** user in the **reporting** database is as follows:

```
{
   "_id" : "reporting.reportUser256",
   "token" : NumberLong("2827251846225877395"),
   "user" : "reportUser256",
   "db" : "reporting",
   "roles" : [ ],
   "mechanisms" : [
      "SCRAM-SHA-256"
   ]
}
```

7.4.3 Deleting a User

db.dropUser(username, writeConcern)

- **username** is the name of the user to be deleted from the database.
- **writeConcern**: The level of write concern for the removal operation. This parameter is optional.

Example

The following describes how to delete user **reportUser1** from database **products**.

```
use products
db.dropUser("reportUser1", {w: "majority", wtimeout: 5000})
```

After the user is deleted, **null** is displayed when you run the **db.getUser** command.

```
replica:PRIMARY> db.getUser("reportUser1")
null
```


8 Creation and Deletion

8.1 How Do I Select Instance Specifications and Nodes?

- Higher specifications can provide better performance. For details about the instance specifications supported by DDS, see **Instance Specifications**.
- For any given set of instance specifications, more nodes provide better performance. For details about performance, see **Performance White Paper**.
- Select specifications that can meet your service requirements based on test data and reserve some extra resources for reliability and future service growth.

NOTE

- For a cluster instance with one vCPU, select 16 GB of memory for better performance.
- For replica sets instances, the most popular configurations are listed as recommended for your convenience. For details, see **Buying a Replica Set Instance > Quick Config**.

8.2 Why Is an Instance Not Displayed on the Console After It Is Created?

If a DDS instance you created is not displayed on the console, there are a couple of possibilities:

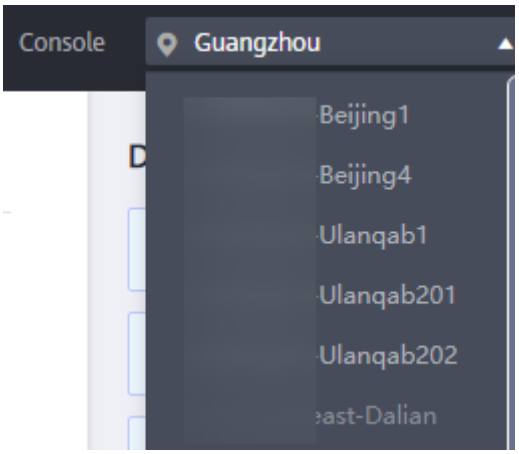
- Insufficient permissions

  Your account may not have the required **permissions**. You can **create a user and grant DDS permissions to the user**.
- Incorrect region

  You can view the created instance only in the region where the instance is located.

  You can click  in the upper left corner of the management console and select the region and project where the instance is located.

Figure 8-1 Selecting a region



8.3 Can I Use a Template to Create DDS DB Instances?

You do not need a template to create DDS DB instances. When you create a DB instance, DDS provides different DB instance specifications which are similar to templates.

8.4 Why Is Data Missing from My Database?

DDS does not delete or perform any operations on any user data. If this problem occurs, check whether a misoperation has been performed. Restore the data using backup files, if necessary.

Solution:

- Restore a DB instance from a backup. For details, see **Backup and Restoration**.
- Import a backup to a DB instance. For details, see **Data Migration**.

8.5 Will My Backups Be Deleted If I Delete My Cloud Account?

If your cloud account is deleted, both your automated and manual backups are deleted.

8.6 What Are the Differences Between Instance Deletion and Unsubscription?

The method of releasing resources varies depending on the DB instance billing mode.

- For a DB instance billed in pay-per-use mode, no order is generated after you purchase it. To release resources based on service requirements, manually delete the DB instance on the **Instance Management** page.

- For a yearly/monthly DB instance, an order is generated after you purchase it. You need to unsubscribe from the order to release the DB instance resources. For details, see **Unsubscribing from a Yearly/Monthly DB Instance**.

9 Database Connection

9.1 What Should I Do If I Fail to Connect to a DDS Instance?

Possible Causes

Locate the fault from the following aspects:

1. **Check whether the DB instance is available.**

   For example, if the system is faulty, a DB instance is abnormal, or a DB instance or a table is locked.

2. **(Common) Check whether the client connection is correct.**

   a. Download the database installation package from the official website. For details, see **How Can I Install a MongoDB Client?**

   b. Use a MongoDB client 4.0 or later to connect to the DB instance.

   c. If you connect to a DB instance over a private network, ensure that the DB instance and ECS are in the same region and VPC.

   d. If you connect to a DB instance over a public network, bind an EIP to the DB instance and then connect to the DB instance through the EIP.

3. **Check the connection method.**

   Run the following commands to enable and disable SSL on the console: Example:

   – Enable SSL: **./mongo --host<*DB_HOST*>--port<*DB_PORT*>-u<*DB_USER*>-p--authenticationDatabaseadmin--ssl --sslCAFile<*FILE_PATH*> --sslAllowInvalidHostnames**

   – Disable SSL: **./mongo --host<*DB_HOST*>--port<*DB_PORT*>-u<*DB_USER*>-p --authenticationDatabase admin**

4. **Check whether the parameters in the connection command are correct.**

   Check whether the following parameters are configured correctly: connection address, port number, username, password, and connection method.

5. **(Common) Check whether the network connectivity is normal.**

Private network connection

a. Ensure that the ECS and DDS instance are in the same region and VPC.

If the ECS and DB instance are in different VPCs, set up a **VPC peering connection** to enable network communication between the two VPCs.

b. Check security group rules.

i. To access a cluster instance in a different security group from the ECS, **add an inbound rule** for the security group.

c. On the ECS, check whether the DB instance port can be connected.

Public network connection

a. Check security group rules.

i. To access a cluster instance in a different security group from the ECS, **add an inbound rule** for the security group.

b. Check network ACL rules.

c. Ping the ECS to the DB instance in the same region.

Cross-CIDR block access (configuring IP address mapping)

If a client and a replica set instance are deployed in different CIDR blocks and the client is not in 192.168.0.0/16, 172.16.0.0/24, or 10.0.0.0/8, configure Cross-CIDR Access for the instance to communicate with the client.

a. Ensure that the source ECS can communicate with the instance node. If the network is abnormal, configure the network settings by referring to **VPC Peering Connection**.

b. For details about how to configure cross-CIDR block access, see **Configuring Cross-CIDR Access**.

6. **(Common) Check whether the number of connections to the DB instance reaches the upper limit.**

   If there is an excessive number of database connections, applications may fail to be connected.

7. **(Common) Check whether the disk is full.**

   If the instance disk usage is too high, the instance status is abnormal and the instance cannot be connected.

8. **Check whether the CPU usage is too high.**

   The CPU usage is high or close to 100%. As a result, data read/write processing is slow, connections cannot be obtained, and errors are reported, affecting service running.

9. **View the common connection error messages**

   Find corresponding solutions based on connection error messages.

Fault Locating

1. **Check whether the DB instance is available.**

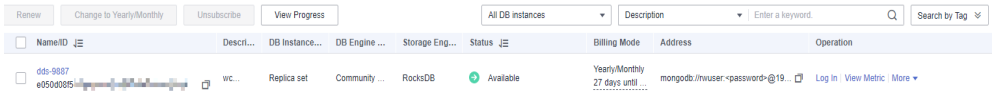
   Check method: Check whether the database instance status is **Available** on the console.

   Possible cause: The DDS system is faulty, the instance status is abnormal, or the instance or table is locked.

   Solution

- If the DDS instance status is abnormal, contact O&M personnel.
- If the system is faulty or the instance or table is locked, restart the instance. For details, see **Restarting an Instance**.

Figure 9-1 Checking the instance status



2. **Check whether the client connection is correct.**

   You are advised to use a MongoDB client of version 4.0 or later to connect to the instance. For details about how to install a MongoDB client, see **How Can I Install a MongoDB Client?**

   The following uses connecting to a cluster as an example:

   a. For details about how to connect to a DB instance over a private network, see **Connecting to a DB Instance Over Private Networks**.

   b. For details about how to connect to a DB instance over a public network, see **Connecting to a DB Instance Over Public Networks**.

Table 9-1 Connection methods

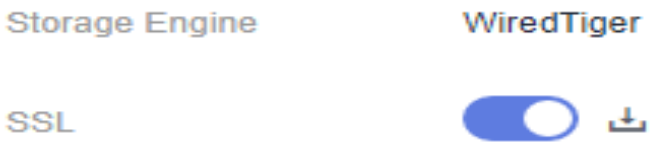
Method	Scenario
Private network	DDS provides a private IP address by default. If your applications are deployed on an ECS that is in the same region and VPC as the DB instance, you are advised to connect to the ECS and DB instance through a private IP address.
Public network	If you cannot access the DB instance over a private IP address, you are advised to bind an EIP to the DB instance and then connect to the DB instance through the EIP. For EIP pricing details, see **EIP billing details**.

3. **Check the connection method.**

   a. SSL connection is recommended. Enable SSL on the **Connections** page and upload the certificate to the ECS.

   ./mongo --host<*DB_HOST*>--port<*DB_PORT*>-u<*DB_USER*>-p--authenticationDatabaseadmin--ssl --sslCAFile<*FILE_PATH*> --sslAllowInvalidHostnames

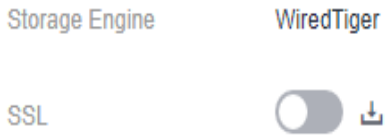
   Figure 9-2 Enabling SSL



b. Unencrypted connection: Disable SSL on the **Connections** page.

./mongo --host<*DB_HOST*>--port<*DB_PORT*>-u<*DB_USER*>-p --authenticationDatabase admin

Figure 9-3 Disabling SSL



4. **Check whether the parameters in the connection command are correct.**

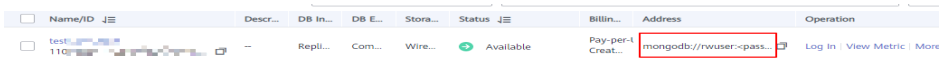
Ensure that the connection address, port, username and password are correct, and try to connect to the DB instance again in SSL mode.

SSL private network connection example:./mongo mongodb://rwuser:<password>@*<DB_HOST>*:*<DB_PORT>*,*<DB_HOST>*:*<DB_PORT>*/test?authSource=admin --ssl --sslCAFile *<FILE_PATH>* **--sslAllowInvalidHostnames**

- Connection address

  The connection information can be obtained in the **Address** column on the **Instances** page.

  Figure 9-4 Connection address



- Database port

  On the **Private Connection** tab of the **Connections** page, obtain the database port in the **Database Port** field.

- Username and password

  If the password of the root administrator is incorrect or you forget the password, **reset the administrator password**.

- Certificate

  Obtain the SSL certificate name from the directory where the command is executed.

SSL public network connection example:./mongo mongodb://rwuser:<password>@*<DB_HOST>*:*<DB_PORT>*/test?authSource=admin --ssl --sslCAFile *<FILE_PATH>* **--sslAllowInvalidHostnames**

- Connection address

  Click the instance name. On the **Public Connection** tab of the **Connections** page, obtain the public connection address in the **Public Network Connection Address** field.

- Database port

  On the **Public Connection** tab of the **Connections** page, obtain the database port in the **Database Port** field.

- Username and password

  If the password of the root administrator is incorrect or you forget the password, **reset the administrator password**.

- Certificate

  Obtain the SSL certificate name from the directory where the command is executed.

5. **Check whether the network connection is normal.**

   Private network connection

   a. Check whether the ECS and DDS instance are in the same region and VPC.

      - If the ECS and DB instance are in different regions, they cannot communicate with each other. Select a region near to your service area to reduce network latency and experience faster access.
      - If the ECS and DB instance are in different VPCs, set up a **VPC peering connection** to enable network communication between the two VPCs.

   b. Check security group rules.

      To access DB instances in a different security group from the ECS, **add an inbound rule** for the security group.

   c. On the ECS, check whether the DB instance port can be connected.

      ```
      telnet <DB instance address> {8635}
      ```

      - If the connection is normal, the network is normal.
      - If the connection fails, submit a to customer service for assistance.

   Public network connection

   a. Check security group rules.

      To access DB instances in a different security group from the ECS, **add an inbound rule** for the security group.

   b. Check network ACL rules.

      i. Go to the **Network ACLs**.

      ii. Check whether the NIC bound to the EIP is in the subnet associated with the network ACL.

      iii. Check whether the network ACL is enabled.

      If yes, **add an ICMP rule to allow traffic**.

      The default network ACL rule denies all inbound and outbound packets. After the network ACL is disabled, the default rule still takes effect.

   c. Ping the ECS to the DB instance in the same region.

      If you cannot ping the EIP on the original ECS, select another ECS in the same region and ping the EIP again. If the ping is successful, the network is normal. If the ping failed, .

   Cross-CIDR block access (configuring IP address mapping)

   If a client and a replica set instance are deployed in different CIDR blocks and the client is not in 192.168.0.0/16, 172.16.0.0/24, or 10.0.0.0/8, configure Cross-CIDR Access for the instance to communicate with the client.

   a. Ensure that the source ECS can communicate with the instance node. If the network is abnormal, configure the network settings by referring to **VPC Peering Connection**.

b. For details about how to configure cross-CIDR block access, see **Configuring Cross-CIDR Access**.

6. **Check whether the number of connections to the DB instance reaches the upper limit.**

   a. Check whether the number of connections to the DB instance has reached the upper limit. For details, see **How Do I Query and Limit the Number of Connections?**

      If the number of instance connections reaches the upper limit, see **What Can I Do If the Number of Connections of an Instance Reaches Its Maximum?**

   b. Check whether any metrics are abnormal and whether any alarms are generated on the Cloud Eye console. Cloud Eye monitors database metrics, such as the CPU usage, memory usage, storage space usage, and database connections, and allows you to set alarm policies to identify risks in advance if any alarms are generated.

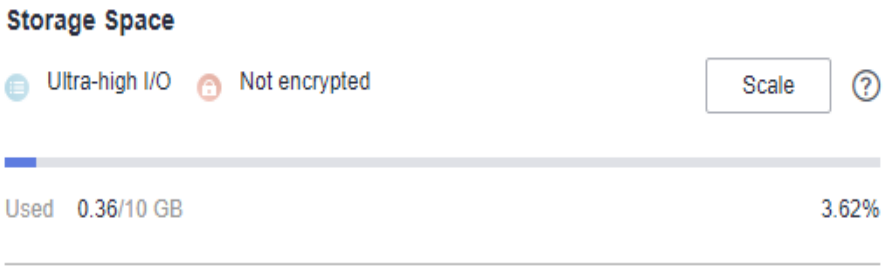
7. **Check whether the disk is full.**

   Check method: View the storage space usage on the management console or Cloud Eye.

   - On the management console

     Locate the target instance and click its name to go to the **Basic Information** page. In the **Storage Space** area, view the storage space usage.

     Figure 9-5 Storage space

   - On Cloud Eye

     Locate the target instance and click **View Metric** in the **Operation** column. On the displayed page, view the storage space usage.

   Possible cause: When the instance status is **Storage full**, the instance needs to preserve at least 15% of its capacity to work properly.

   Solution:

   - As your service data grows, the original storage space may be insufficient. You are advised to scale up storage space. For details, see **Scaling Up Storage Space**.
   - Process expired data files in a timely manner.
   - Check whether any metrics are abnormal and whether any alarms are generated on the Cloud Eye console. Cloud Eye monitors database metrics, such as the CPU usage, memory usage, storage space usage, and

database connections, and allows you to set alarm policies to identify risks in advance if any alarms are generated.

8. **Check whether the CPU usage is too high.**

   The CPU usage is high or close to 100%. As a result, data read/write processing is slow, connections cannot be obtained, and errors are reported, affecting service running.

   Solution: For details, see **What Should I Do If CPU Usage Is Unusually High?**

9. **View common connection error messages.**

   - Connection failure message: **network error while attempting to run command 'isMaster'**

     An error is reported when you run the following command to connect to a DDS DB instance:

     ./mongo --host 192.168.168.182 --port 8635 -u rwuser -p *xxxxxxxxxx* **--authenticationDatabase admin**

     Figure 9-6 Connection failed

     For details about how to rectify the fault, see **Connection Failure Message: network error while attempting to run command 'isMaster'**.

   - Connection failure messages: **No route to host** and **connection attempt failed**.

     An error is reported when you run the following command to connect to a DDS DB instance:

     mongo --host 192.168.1.6 --port 8635 -u rwuser -p *xxxxxxxxx* **--authenticationDatabase admin --ssl --sslCAFile /root/ca.crt --sslAllowInvalidHostnames**

     Error message:

     ```
     MongoDB shell version v3.4.17
     connecting to: mongodb://192.168.1.6:8635/
     2019-09-19T09:38:36.954+0800 W NETWORK  [thread1] Failed to connect to 192.168.1.6:8635, in(checking socket for error after poll), reason: No route to host
     2019-09-19T09:38:36.954+0800 E QUERY    [thread1] Error: couldn't connect to server 192.168.1.6:8635, connection attempt failed :
     connect@src/mongo/shell/mongo.js:240:13
     @(connect):1:6
     exception: connect failed
     ```

     For details about how to rectify the fault, see **Connection Failure Messages: "No route to host" and "connection attempt failed"**.

   - Connection failure messages: **No route to host** and **connection attempt failed**

     An error is reported when you run the following command to connect to a DDS DB instance:

     mongo --host 192.168.168.116 --port 8635 -u rwuser -p *xxxxxxxxx* **--authenticationDatabase admin --ssl --sslCAFile /root/ca.crt --sslAllowInvalidHostnames**

Error message:

```
MongoDB shell version v3.4.17
connecting to: mongodb://192.168.168.116:8635/
2019-09-19T09:39:24.306+0800 W NETWORK  [thread1] The server certificate does not match the host name. Hostname: 192.168.168.116 does not match CN: 172.16.2.65
MongoDB server version: 4.0.3
WARNING: shell and server versions do not match
2019-09-19T09:39:24.329+0800 E QUERY    [thread1] Error: Authentication failed. :
DB.prototype._authOrThrow@src/mongo/shell/db.js:1461:20
@(auth):6:1
@(auth):1:2
exception: login failed
```

For details about how to rectify the fault, see **Connection Failure Messages: "No route to host" and "connection attempt failed"**.

- Connection failure message: **couldn't connect to server**.

  An error is reported when you run the following command to connect to a DDS DB instance:

  mongo --host 192.168.64.201 --port 8635 -u rwuser -p *xxxxxxxxx* **--authenticationDatabase admin --ssl --sslCAFile /root/ca.crt --sslAllowInvalidHostnames**

  Error message:

  ```
  MongoDB shell version v3.4.17
  connecting to: mongodb://192.168.64.201:8635/
  2019-09-19T09:45:48.168+0800 W NETWORK  [thread1] Failed to connect to 192.168.64.201:8635 after 5000ms milliseconds, giving up.
  2019-09-19T09:45:48.168+0800 E QUERY    [thread1] Error: couldn't connect to server 192.168.64.201:8635, connection attempt failed :
  connect@src/mongo/shell/mongo.js:240:13
  @(connect):1:6
  exception: connect failed
  ```

  For details about how to rectify the fault, see **Connection Failure Message: "couldn't connect to server"**.

- Connection failure message: **cannot list multiple servers in URL without 'replicaSet' option**

  An error is reported when you run the following command to connect to a DDS replica set instance:

  ./mongo mongodb://rwuser:*xxxxxxxxxx*@192.168.168.116:8635,192.168.200.147:8635/test?authSource=admin&replicaSet=replica

  Error message:

  ```
  FailedToParse: Cannot list multiple servers in URL without 'replicaSet' option try 'mongo --help' for more information.
  ```

  An error is reported when you run the following command to connect to a DDS DB instance that is compatible with MongoDB 3.4:

  mongo "mongodb://rwuser:*xxxxxxxxx***@192.168.95.167:8635,192.168.92.43:8635/test?authSource=admin"**

  Error message:

  ```
  FailedToParse: Cannot list multiple servers in URL without 'replicaSet' option
  try 'mongo --help' for more information.
  ```

  For details, see **Connection Failure Message: cannot list multiple servers in URL without 'replicaSet' option**.

- Connection failure message: **Timeout while receiving message**.

An error is reported when you run the following command to connect to a DDS DB instance through Java driver:

Error message:

```
org.springframework.data.mongodb.UncategorizedMongoDbException: Timeout while receiving message; nested exception is com.mongodb.MongoSocketReadTimeoutException: Timeout while receiving message
```

For details about how to rectify the fault, see **Connection Failure Message: "Timeout while receiving message"**.

- Connection failure messages: **exception: login failed** and **U_STRINGPREP_PROHIBITED_ERROR**

  An error is reported when you run the following command to connect to a DDS DB instance:

  ./mongo "mongodb://rwuser:*xxxxxx*@192.168.0.45:8635,192.168.0.96:8635/test?authSource=admin&replicaSet=replica"

  Error message:

```
MongoDB shell version v4.0.3
connecting to: mongodb://192.168.0.45:8635,192.168.0.96:8635/test?authSource=admin&replicaSet=replica
2021-11-05T05:52:53.717+0000 I NETWORK  [js] Starting new replica set monitor for replica/192.168.0.45:8635,192.168.0.96:8635
2021-11-05T05:52:53.718+0000 I NETWORK  [ReplicaSetMonitor-TaskExecutor] Successfully connected to 192.168.0.45:8635 (1 connections now open to 192.168.0.45:8635 with a 5 second timeout)
2021-11-05T05:52:53.718+0000 I NETWORK  [js] Successfully connected to 192.168.0.96:8635 (1 connections now open to 192.168.0.96:8635 with a 5 second timeout)
Implicit session: session { "id" : UUID("5945d2a5-8275-4e3c-b06f-632f062a2ead") }
MongoDB server version: 4.0.3
2021-11-05T05:52:53.722+0000 I NETWORK  [js] Marking host 192.168.0.96:8635 as failed :: caused by :: Location50692: can't authenticate against replica set node 192.168.0.96:8635 :: caused by :: Error preflighting normalization: U_STRINGPREP_PROHIBITED_ERROR
2021-11-05T05:52:53.722+0000 I NETWORK  [js] Successfully connected to 192.168.0.45:8635 (1 connections now open to 192.168.0.45:8635 with a 0 second timeout)
2021-11-05T05:52:53.723+0000 I NETWORK  [js] Marking host 192.168.0.45:8635 as failed :: caused by :: Location50692: can't authenticate against replica set node 192.168.0.45:8635 :: caused by :: Error preflighting normalization: U_STRINGPREP_PROHIBITED_ERROR
2021-11-05T05:52:53.724+0000 I NETWORK  [js] Marking host 192.168.0.96:8635 as failed :: caused by :: Location50692: can't authenticate against replica set node 192.168.0.96:8635 :: caused by :: Error preflighting normalization: U_STRINGPREP_PROHIBITED_ERROR
2021-11-05T05:52:53.725+0000 I NETWORK  [js] Marking host 192.168.0.45:8635 as failed :: caused by :: Location50692: can't authenticate against replica set node 192.168.0.45:8635 :: caused by :: Error preflighting normalization: U_STRINGPREP_PROHIBITED_ERROR
2021-11-05T05:52:53.725+0000 E QUERY    [js] Error: can't authenticate against replica set node 192.168.0.45:8635 :: caused by :: Error preflighting normalization: U_STRINGPREP_PROHIBITED_ERROR :
DB.prototype._authOrThrow@src/mongo/shell/db.js:1685:20
@(auth):6:1
@(auth):1:2
exception: login failed
```

  For details about how to rectify the fault, see **Connection Failure Messages: exception: login failed and U_STRINGPREP_PROHIBITED_ERROR**.

10. If the problem persists, to contact Huawei Cloud customer service for assistance.

9.2 What Can I Do If the Number of Connections of an Instance Reaches Its Maximum?

The number of connections indicates the number of applications that can be simultaneously connected to the database. The number of connections is irrelevant to the maximum number of users allowed by your applications or websites.

- For a cluster instance, the number of connections is the number of connections between the client and the dds mongos nodes.
- For a replica set instance, the number of connections is the number of connections between the client and the primary and secondary nodes.
- For a single-node instance, the number of connections is the number of connections between the client and the node.

When the number of connections to a DDS instance reaches the maximum supported, new connection requests cannot be responded to, and the connection attempt fails.

Symptom

Some common errors:

- If the following information is displayed when you use Mongo Shell to connect to an instance, no more connections can be established.

  Figure 9-7 Message displayed

  ```
  [root@623- ~]# mongo "mongodb://rwuser: @192.168.0.180:8635,192.168.0.50:8635/test?authSource=admin&replicaSet=replica"
  MongoDB shell version v3.4.17
  connecting to: mongodb://rwuser:623_Huawei@192.168.0.180:8635,192.168.0.50:8635/test?authSource=admin&replicaSet=replica
  2019-10-16T17:43:45.203+0800 I NETWORK  [thread1] Starting new replica set monitor for replica/192.168.0.180:8635,192.168.0.50:8635
  2019-10-16T17:43:45.205+0800 W NETWORK  [ReplicaSetMonitor-TaskExecutor-0] No primary detected for set replica
  2019-10-16T17:43:45.205+0800 I NETWORK  [ReplicaSetMonitor-TaskExecutor-0] All nodes for set replica are down. This has happened for 1 check
  s in a row.
  2019-10-16T17:43:45.708+0800 W NETWORK  [thread1] No primary detected for set replica
  2019-10-16T17:43:45.708+0800 I NETWORK  [thread1] All nodes for set replica are down. This has happened for 2 checks in a row.
  2019-10-16T17:43:46.210+0800 W NETWORK  [thread1] No primary detected for set replica
  2019-10-16T17:43:46.210+0800 I NETWORK  [thread1] All nodes for set replica are down. This has happened for 3 checks in a row.
  2019-10-16T17:43:46.712+0800 W NETWORK  [thread1] No primary detected for set replica
  2019-10-16T17:43:46.712+0800 I NETWORK  [thread1] All nodes for set replica are down. This has happened for 4 checks in a row.
  2019-10-16T17:43:47.215+0800 W NETWORK  [thread1] No primary detected for set replica
  2019-10-16T17:43:47.215+0800 I NETWORK  [thread1] All nodes for set replica are down. This has happened for 5 checks in a row.
  2019-10-16T17:43:47.717+0800 W NETWORK  [thread1] No primary detected for set replica
  2019-10-16T17:43:47.717+0800 I NETWORK  [thread1] All nodes for set replica are down. This has happened for 6 checks in a row.
  2019-10-16T17:43:48.218+0800 W NETWORK  [thread1] No primary detected for set replica
  2019-10-16T17:43:48.218+0800 I NETWORK  [thread1] All nodes for set replica are down. This has happened for 7 checks in a row.
  2019-10-16T17:43:48.721+0800 W NETWORK  [thread1] No primary detected for set replica
  2019-10-16T17:43:48.721+0800 I NETWORK  [thread1] All nodes for set replica are down. This has happened for 8 checks in a row.
  2019-10-16T17:43:49.222+0800 W NETWORK  [thread1] No primary detected for set replica
  2019-10-16T17:43:49.222+0800 I NETWORK  [thread1] All nodes for set replica are down. This has happened for 9 checks in a row.
  2019-10-16T17:43:49.724+0800 W NETWORK  [thread1] No primary detected for set replica
  2019-10-16T17:43:49.724+0800 I NETWORK  [thread1] All nodes for set replica are down. This has happened for 10 checks in a row.
  2019-10-16T17:43:50.226+0800 W NETWORK  [thread1] No primary detected for set replica
  2019-10-16T17:43:50.226+0800 I NETWORK  [thread1] All nodes for set replica are down. This has happened for 11 checks in a row.
  2019-10-16T17:43:50.727+0800 W NETWORK  [thread1] No primary detected for set replica
  2019-10-16T17:43:51.230+0800 W NETWORK  [thread1] No primary detected for set replica
  2019-10-16T17:43:51.731+0800 W NETWORK  [thread1] No primary detected for set replica
  ```

- If the following information is displayed when you use Python to connect to an instance, the number of connections reaches its maximum.

  pymongo.errors.ServerSelectionTimeoutError: connection closed, connection closed

Handling Method

1. Check what applications are connected, optimize the connections, and release any that are not necessary.

2. Check the value of the **net.maxIncomingConnections** parameter and the instance specifications. Change the parameter value or the database specifications.
3. Check for abnormal metrics and alarms on the Cloud Eye console. Cloud Eye monitors database metrics, such as the CPU usage, memory usage, storage space usage, and database connections, and allows you to set alarm policies to identify risks in advance if any alarms are generated. For details, see the *Cloud Eye User Guide*.

Solution

1. Release unnecessary connections.
   a. You can restart the instance to release all of the connections. For details, see **Restarting an Instance or a Node**.
   b. You can query the current number of connections on a node and the connection source, analyze the number of connections established between each client and the instance, and adjust the number of connections. For details, see **How Do I Query and Limit the Number of Connections?**
2. Change parameter values or database specifications.

   You can change the maximum number of connections of a DB instance by modifying the **net.maxIncomingConnections** parameter. For details about how to change parameter values, see **Editing a Parameter Template**.

   - If the value is **default**, the maximum number of connections is the default value and is related to the instance specifications. For details, see **Instance Specifications**.
   - If there are too many connections, the service may break down. In this case, you can increase the number of connections by changing the instance specifications. For details, see **Changing a Cluster Instance Class**.

   NOTE

   If a default parameter template is used, you cannot change its settings. You can create a parameter template and change the corresponding parameter values. After the change, associate the new parameter template with the instance.
3. Check whether there are **slow queries**. You can add indexes to improve queries.

9.3 How Do I Query and Limit the Number of Connections?

The following uses a replica set instance as an example to describe how to query the connection status and set the number of connections in the connection pool.

Querying the Number of Connections

The maximum number of connections varies according to the instance specifications.

> **NOTE**
>
> The maximum number of connections refers to the maximum number of connections of each node in an instance.

Example: If a replica set instance has two vCPUs and 4 GB memory for each node, the maximum number of connections of the primary and secondary nodes is 1000 respectively. The hidden node does not provide services because of its architecture features.

Use Mongo Shell to connect to the primary node, and run the **db.serverStatus().connections** command to query the number of connections on the node.

```
replica:PRIMARY> db.serverStatus().connections
{ "current" : 7, "available" : 398, "totalCreated" : 818364 }
```

Pay attention to the following parameters and their values:

- **current**: Existing connections
- **available**: Number of available connections.

Querying the Source of Connections

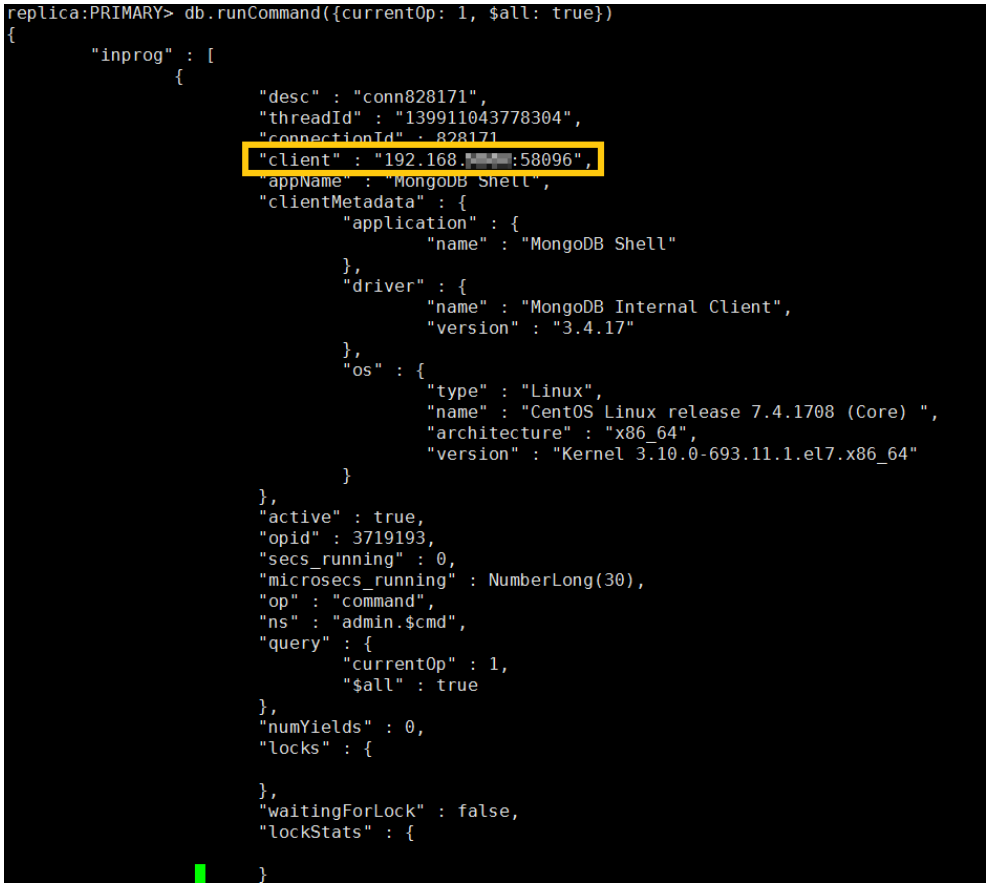
Step 1 Use Mongo Shell to connect to the primary node and switch to the **admin** database.

```
replica:PRIMARY> use admin
```

Step 2 Run the **db.runCommand({currentOp: 1, $all: true})** command to query the connection source.

By analyzing the command output, you can query the source IP address of each connection. In this way, the number of connections between each client and DDS DB instance is obtained.

Figure 9-8 Command output



----End

Limiting the Number of Connections

DDS allows you to log in to the database using Connection String URI. When logging in to the database using Connection String URI, you can add **&maxPoolSize=<integer>** to the end of the URI to set the number of connections in the connection pool.

Example: When Mongo Shell is used to connect replica set instances, run the following command to set the number of connections in the connection pool to 10:

mongo "mongodb://
rwuser:*xxxxxxxxxx***@192.168.168.116:8635,192.168.200.147:8635/test?**
authSource=admin&replicaSet=replica&maxPoolSize=10"

Figure 9-9 Limiting the number of connections

NOTE

For details about how to limit the number of connection pools on **clients in different languages**, see the API documents of clients in different languages on the MongoDB official website.

9.4 What Should I Do If the ECS and DDS Are Deployed in Different VPCs and They Cannot Communicate with Each Other?

- Change the VPC hosting the ECS to the same as that hosting the DDS by referring to **Changing a VPC**.
- Create a VPC peering connection. For details, see **VPC Peering Connection Overview**.

9.5 Do Applications Need to Support Automatic Reconnecting to the DDS Database?

It is recommended that your applications support automatic reconnections to the database. After a database reboot, your applications will automatically reconnect to the database to increase service availability and continuity.

In addition, you are advised to set your applications to connect to the database using a long connection to reduce resource consumption and improve performance.

9.6 How Do I Create and Log In to an ECS?

For details on how to create and log in to an ECS, see **Purchasing and Logging In to a Windows ECS** and **Purchasing and Logging In to a Linux ECS**.

- The ECS to be created must be in the same VPC with the DDS DB instance to which it connects.
- When you create an ECS, select an OS, such as Red Hat 6.6, and bind an EIP to it.
- Configure the security group to enable the ECS to access the DB instance through the private IP address, that is, the node address in the **Private IP Address** column on the **Basic Information** page.

10 Installing a Client

10.1 How Can I Install a MongoDB Client?

MongoDB official website provides client installation packages for different OSs. Download the official package at **https://www.mongodb.com/try/download/community**.

NOTICE

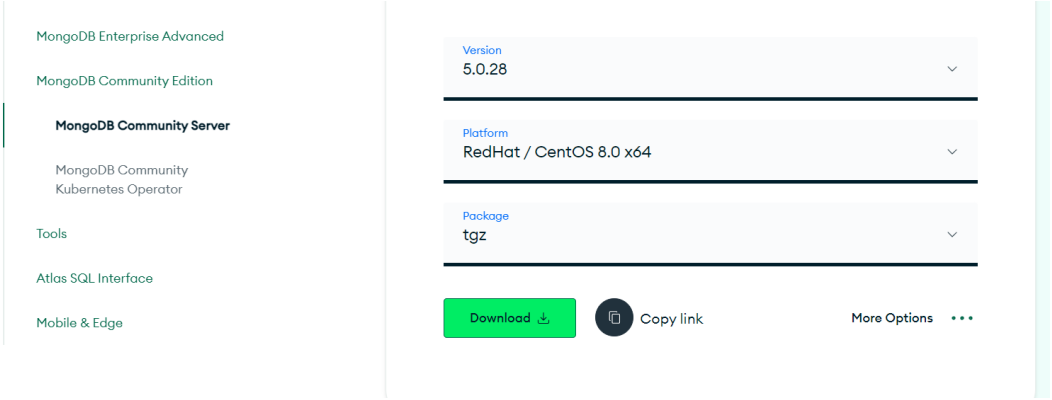
- The following uses **RedHat/CentOS 8.0 ×64** and MongoDB 5.0.28 as examples to describe how to obtain the required installation package and install the MongoDB client.
- During the installation, select a client version that matches the instance version based on the actual operating system.

Procedure

Step 1 Obtain the installation package.

1. Visit the **MongoDB official website**.
2. Select version **5.0.28**, platform **RedHat/CentOS 8.0 ×64**, and package **tgz**. **Figure 10-1** shows an example.

Figure 10-1 MongoDB official webpage



3. Use either of the following methods to upload the installation package to the ECS:

   NOTE

   For details about how to log in to an ECS, see **How Can I Create and Log In to an ECS?**

   - Click **Download** to obtain the binary installation package of version 5.0.28. The name of the installation package is **mongodb-linux-x86_64-rhel80-5.0.28.tgz**. Upload the installation package to the ECS.
   - Click **Copy link** to obtain the download address. Log in to the ECS and run the **wget** *copylink* command.

     NOTE

     Replace *copylink* with the actual download address.

Step 2 Decompress the installation package on the ECS.

tar zxvf mongodb-*linux-x86_64-rhel80-5.0.28***.tgz**

NOTE

Replace the installation package name with the actual one.

Step 3 Access the directory where the installation package is located.

cd mongodb-*linux-x86_64-rhel80-5.0.28***/bin**

NOTE

Replace the installation package name with the actual one.

The common tools are as follows:

- MongoDB client mongo
- Data export tool mongoexport
- Data import tool mongoimport

Step 4 Make the packages executable.

- Run the **chmod +x mongo** command to grant a client permission to connect to an instance.

- Run the **chmod +x mongoexport** command to grant a client permission to export data.
- Run the **chmod +x mongoimport** command to grant a client permission to import data.

Step 5 Connect to an instance from the client. For details, see **Connecting to a Cluster Instance**, **Connecting to a Replica Set Instance**, and **Connecting to a Single Node Instance**.

----End

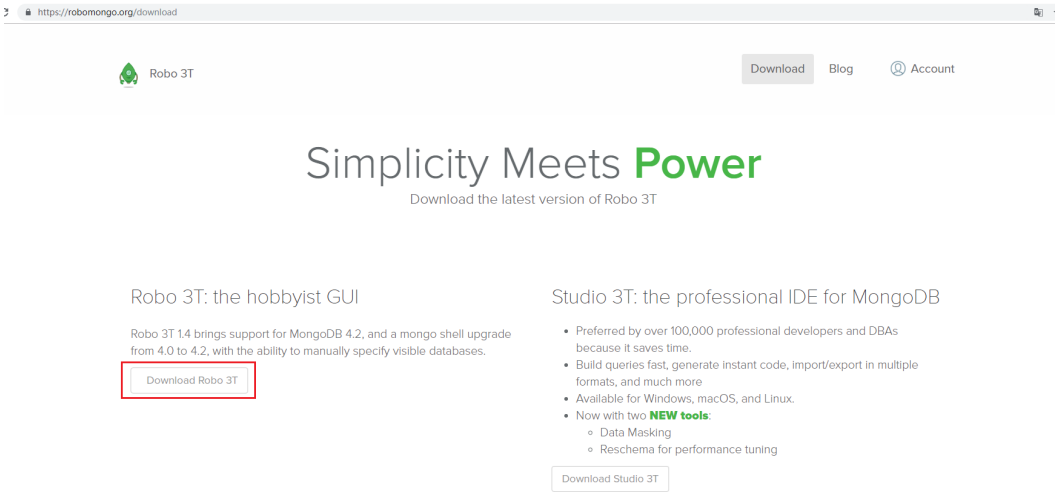
10.2 How Do I Install Robo 3T?

This section describes how to obtain the Robo 3T installation package and install Robo 3T.

Procedure

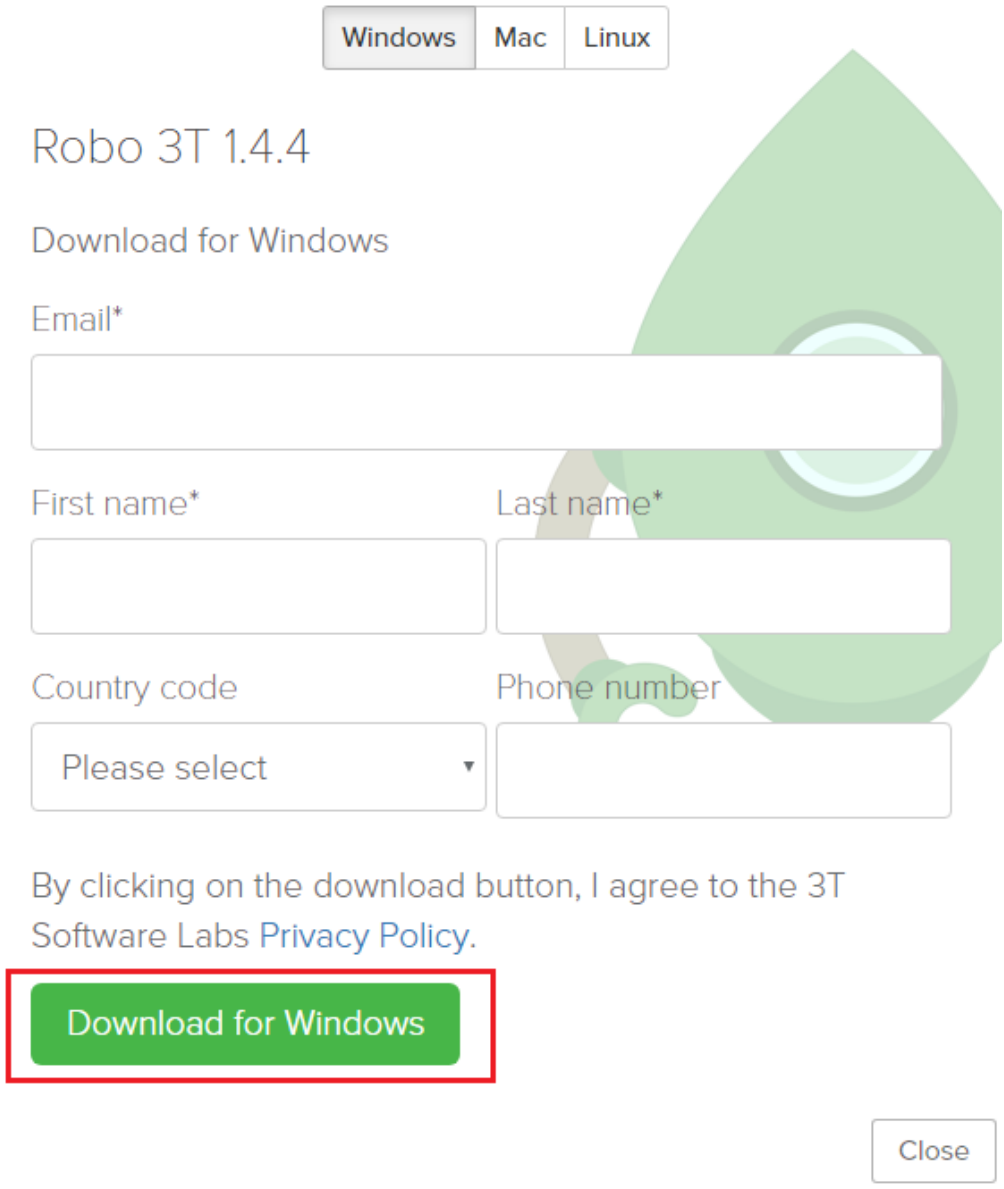
Step 1 Access the Robo 3T download address **https://robomongo.org/download** and click **Download Robo 3T**.

Figure 10-2 Downloading page



Step 2 In the displayed dialog box, enter the required information and click **Download for Windows** to download the **robo3t-1.4.4-windows-x86_64-e6ac9ec5.zip** package.

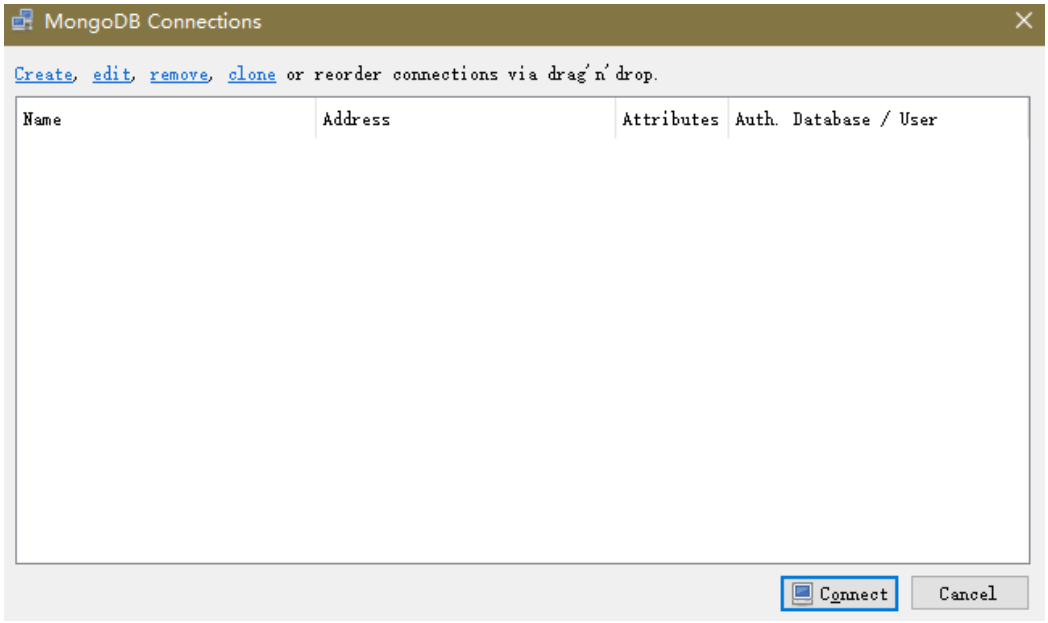
Figure 10-3 Downloading Robo 3T



Step 3 Decompress the downloaded package obtained in **Step 2** and double-click the **robo3t.exe** file in the decompressed directory to start the installation.

Step 4 After the installation is complete, start the tool.

Figure 10-4 Main window



Step 5 Connect to an instance using the tool over a public network. For details, see "Connecting to an Instance" in *Getting Started with Document Database Service*

----End

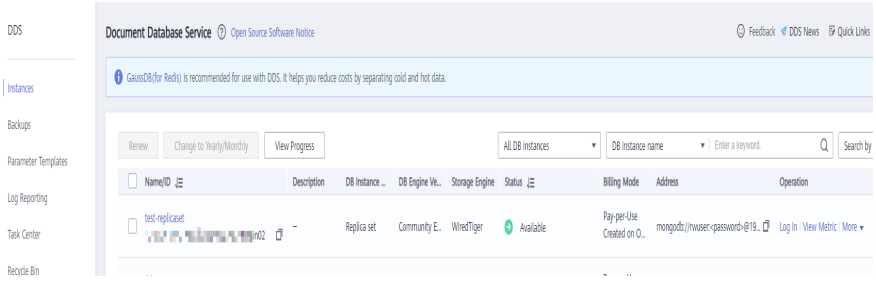
11 Database Usage

11.1 How Do I View the Primary/Standby Nodes of a Replica Set Instance?

You can view node information for the current instance on the DDS console.

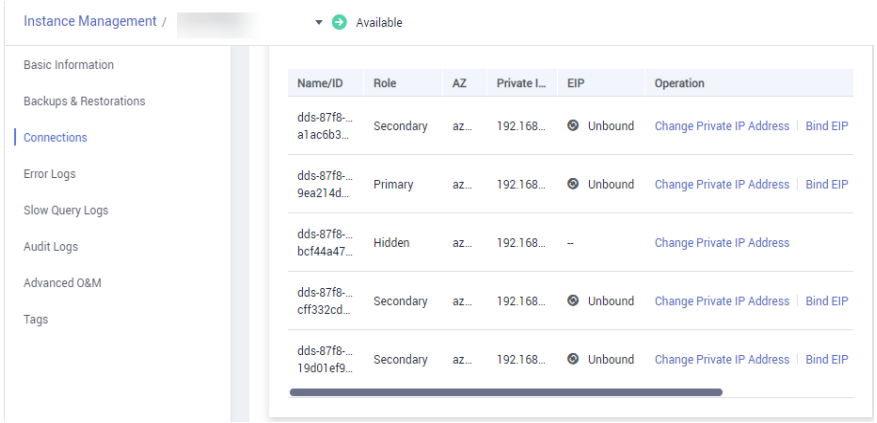
1. Logging In to the DDS Console
2. On the **Instances** page, click the instance name.

Figure 11-1 Instances



3. In the navigation pane on the left, choose **Connections**. The node information is displayed in the right area.

Figure 11-2 Connections



11.2 Can I Delete an Index That Fails to Be Created in DDS?

Run the **db.**<*collection*>**.stats()** statement to query the indexes that fail to be created in the database and then delete them.

NOTE

Set *collection* as required.

11.3 Does DDS Delete Expired Data Using TTL Indexes?

You can delete expired data using time-to-live (TTL) indexes. For details, see the **official guide**.

11.4 How Do I Use MapReduce Commands?

Overview

MapReduce commands are used for executing map-reduce operations on big data.

How Do I Enable MapReduce Commands?

The MapReduce command is controlled by the DDS parameter **security.javascriptEnabled**. The default value is **false**, indicating that the **mapReduce** and **group** commands are unavailable. If you need to use the MapReduce command, change the parameter value to **true**. Then, restart the DB instance for the change to take effect.

- For a cluster instance, change the parameter values in the parameter template associated with all shard nodes and restart the instance for the change to take effect.
- For a replica set or single-node instance, change the parameter values in the parameter template associated with the instance and restart the instance for the change to take effect.

For details about how to change parameter values, see **Editing a Parameter Template**.

Common Errors and Handling Methods for MapReduce Command Execution Failures

Error information: cannot run map reduce without the js engine or map is not defined

Figure 11-3 Case 1

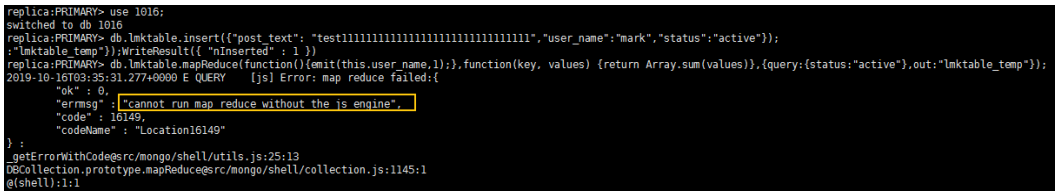
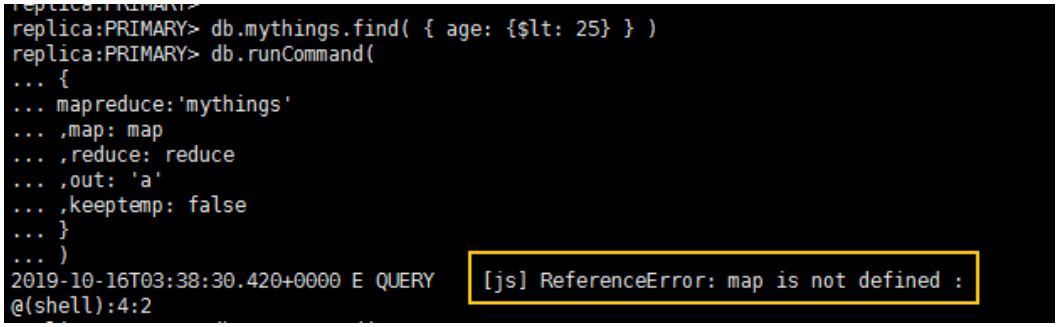


Figure 11-4 Case 2



Possible cause: MapReduce commands are restricted and cannot be used.

Solution: Change the value of **security.javascriptEnabled** in the instance parameter template to **true** and restart the instance for the change to take effect.

NOTE

If a parameter template is a default parameter template, you are not allowed to change its parameter values. You can create a parameter template and change the corresponding parameter values. After the change, associate the new parameter template with the instance. For details, see **Changing the Associated Parameter Template**.

11.5 Does DDS Support the $round Function?

DDS supports the $round function. For details, see **Compatible Aggregate Functions**.

11.6 How Do I Manage the Balancer?

Scenario

The balancer occupies system resources of nodes in a cluster to migrate chunks, which may deteriorate the read and write performance of the cluster. If you have enabled data sharding and have high requirements on instance performance, or the service traffic has peak and off-peak differences, to avoid negative impact on your businesses, perform the following operations:

- **Enable the balancer** during off-peak hours.
- **Disable the balancer** during peak hours.
- **Schedule a balancing window** so that the balancer can work during off-peak hours.

Constraints

- The balancer applies only to cluster instances.
- DDS enables the balancer by default.

Enabling the Balancer

Step 1 **Connect to a cluster instance using Mongo Shell**.

Step 2 After connecting to a dds mongos node, run the following command to switch to the config database:

```
use config
```

Step 3 Run the following command to enable the balancer:

```
sh.setBalancerState(true)
```

----End

Disabling the Balancer

Step 1 **Connect to a cluster instance using Mongo Shell**.

Step 2 After connecting to a dds mongos node, run the following command to switch to the config database:

```
use config
```

Step 3 Run the following command to check the status of the balancer:

```
while( sh.isBalancerRunning() ) {
        print("waiting...");
        sleep(500);
}
```

- If the command output is empty, the balancer is not executing any task. In this case, you can proceed to disable the balancer.
- If the command output is **waiting**, the balancer is migrating chunks. In this case, you cannot disable the balancer. Otherwise, data may become inconsistent.

Step 4 If the command output in **3** is empty, run the following command to disable the balancer:

```
sh.stopBalancer()
```

----End

Scheduling a Balancing Window

To prevent chunk migration from affecting your businesses, you can perform the following operations to schedule a balancing window.

NOTE

Before scheduling a balancing window, ensure that the balancer is enabled by referring to **Enabling the Balancer**.

Step 1 **Connect to a cluster instance using Mongo Shell**.

Step 2 After connecting to a dds mongos node, run the following command to switch to the config database:

```
use config
```

Step 3 Run the following commands to schedule a balancing window:

```
db.settings.update(
  { _id: "balancer" },
  { $set: { activeWindow : { start : "<start-time>", stop : "<stop-time>" } } },
  { upsert: true }
)
```

NOTE

- **<start-time>**: indicates the beginning of the balancing window. Specify the time in the HH:MM format (UTC time). For **HH** values, use hour values ranging from 00 to 23. For **MM** value, use minute values ranging from 00 to 59.
- **<stop-time>**: indicates the end of the balancing window. Specify the time in the HH:MM format (UTC time). For **HH** values, use hour values ranging from 00 to 23. For **MM** value, use minute values ranging from 00 to 59.

You can run the **sh.status()** command to check a balancing window schedule. The following command output in **Figure 11-5** shows that the balancing window is 22:00 to 24:00.

Figure 11-5 Checking a balancing window schedule

```
mongos> sh.status()
--- Sharding Status ---
  sharding version: {
        "_id" : 1,
        "minCompatibleVersion" : 5,
        "currentVersion" : 6,
        "clusterId" : ObjectId("64478ef7d201d17f7beb47c7")
  }
  shards:
        {  "_id" : "shard_1",  "host" : "shard_1/172.16.29.203:8637,172.16.61.17:8637",  "state" : 1 }
        {  "_id" : "shard_2",  "host" : "shard_2/172.16.27.2:8637,172.16.47.171:8637",  "state" : 1 }
  active mongoses:
        "4.0.3" : 2
  autosplit:
        Currently enabled: yes
  balancer:
        Currently enabled:  no
        Currently running:  no
                Balancer active window is set between 22:00 and 24:00 server local time
```

----End

Removing a Balancing Window Schedule

If you have set the balancing window and wish to remove the schedule so that the balancer is always running, run the following command:

```
db.settings.update({ _id : "balancer" }, { $unset : { activeWindow : true } })
```

12 Database Migration

12.1 Does DDS Support Cross-Region Migration?

You can use the Data Replication Service (DRS) to migrate databases across regions on the cloud.

For details, see **Migrating Data Using DRS**.

12.2 How Do I Migrate DDS Databases Between Different Accounts?

If you want to migrate DDS instance A of account A to DDS-instance B of account B, you can use DRS to replicate data from instance A to instance B. For details, see **Does DRS Support Cross-Account Cloud Database Migration?**

12.3 How Do I Evaluate the Compatibility When Migrating MongoDB from a Later Version to an Earlier Version?

DDS of a later version is compatible with that of an earlier version. In a few cases, if a function is terminated, official information will be summarized and released. When DDS of a later version is migrated to an earlier version, new functions of the later version will not be supported in the earlier version. You need to evaluate whether these new functions are needed in your businesses.

The evaluation methods include estimation suggestions and drill suggestions.

- Estimation suggestions: By querying the version release of the community document, you can view the new features when a new version is released and analyze the features one by one to determine whether they are used in the service code.
- Drill suggestions: Use **Data Replication Service (DRS)** or **mongodump and mongorestore** provided by the community to migrate data to an earlier

version, test the function simulation, and check whether the migration is feasible.

If your businesses are complex, contact Huawei technical experts for professional evaluation support.

13 Database Storage

13.1 How Does DDS Store Data?

EVS is used to store DDS instance data. For details about EVS, see *Elastic Volume Service User Guide*.

DDS instance backup data is stored in OBS and does not count towards your DDS storage quota. For details on the DDS instance storage configuration, see the *Object Storage Service User Guide*.

13.2 What Should I Do If My Data Exceeds the Database Storage Space of a DDS Instance?

For details about the files that occupy the storage space, see **Which Items Occupy the Storage Space of DDS Instances?**

If the storage required by your applications exceeds the maximum allocated capacity, you can **scale up the storage**.

For a cluster instance, you can also add shard nodes to increase storage. For details, see **Adding Nodes**.

13.3 Why Does a DDS DB Instance Become Read-only?

To ensure that a DB instance can still be used if the storage space is about to be used up, databases running on the instance are set to read-only, and data cannot be modified. If this happens, you can **add more storage** to restore the database to read/write status.

Detailed rules are as follows:

- If the storage space you purchased exceeds 600 GB and the remaining storage space is 18 GB, the DB instance becomes **Read-only**.
- If the storage space you purchased is less than 600 GB and the storage space usage reaches 97%, the DB instance becomes **Read-only**.

- If the storage space you purchased exceeds 600 GB and the remaining available storage space is greater than 90 GB, the read-only state is automatically canceled.
- If the storage space you purchased is less than or equal to 600 GB and the used storage space is less than 85% of the total storage space, the read-only state is automatically canceled.

You are advised to remove unnecessary resources or expand the storage space.

14 Database Parameters

14.1 Can I Change the Time Zone of a DDS Instance?

The server time is in UTC format and cannot be changed. You are advised to control the data insertion time on the application.

14.2 What DB Instance Parameters Do I Need to Pay Attention To?

Related parameters are described as follows:

- For details on parameter descriptions, visit **MongoDB official website**.
- The default value of the **net.maxIncomingConnections** parameter varies according to DB instance specifications. Therefore, this parameter is set to **default** before being specified.
- **disableJavaScriptJIT** and **security.javascriptEnabled** are used together to set the statistical function.
  - **disableJavaScriptJIT**: The default value is **true**, indicating that the JavaScriptJIT compiler is disabled.
  - **security.javascriptEnabled**: The default value is **false**, indicating that JavaScript cannot be executed on mongod and the mapReduce and group commands cannot be used.

14.3 How Do I Enable JavaScript?

For security purposes, DDS restricts the execution of JavaScript code. If you need to execute JavaScript, you can change the value of **security.javascriptEnabled** to **true** on the DDS console. For details, see **Editing a Parameter Template**.

14.4 Does DDS Support Majority Read Concern?

Write concern and read concern respectively specify the write and read policies for MongoDB.

If read concern is set to majority, data read by users has been written to a majority of nodes and will not be rolled back to avoid dirty reads.

DDS does not support majority read concern.

To meet this requirement, set write concern to majority. In this way, data is written to a majority of nodes, ensuring data consistency. Then, by reading data from a single node, it can be ensured that data read by the user can be written to a majority of nodes, and such data is not rolled back to avoid dirty read.

15 Backup and Restoration

15.1 Will I Be Charged for Manual Backups of Deleted DB Instances?

Yes. Manual backups are retained by default after your DB instances are deleted. The manual backups are billed based on the OBS pricing.

15.2 How Do I Back Up DDS Databases to an ECS?

You can store DDS backup data on an ECS using **mongoexport**. However, you are not advised to use an ECS as backup space.

For the best reliability and service assurance, **store backups** in Object Storage Service (OBS) buckets.

15.3 How Long Does DDS Store Backup Data For?

The automated backup retention period is 7 days by default. You can set a backup retention period from 1 to 732 days. There is no limit on the manual backup retention period. You can delete manual backup files as needed.

15.4 How Do I Retrieve Lost DDS Backup Data?

If Cloud Trace Service (CTS) is not enabled, you cannot query operation records.

To query backup deletion records, you can enable CTS. For details, see **Querying Traces**.

15.5 What Files Are Contained in the Backup Space?

The backup space contains automated backup files generated during automated backup, manual backup files, and incremental log backup files.

15.6 How Is DDS Backup Data Billed?

All the DDS full and incremental backups are stored on OBS without occupying the storage of your DB instances. DDS provides free backup space of the same size as your purchased storage.

The lifecycle of automated backups is the same as that of the DB instance. If you delete a DB instance, its automated backups are also deleted, but manual backups are not. For details, see **Deleting a Manual Backup**.

For example, if you purchase a DB instance with 200 GB of storage, you can get an additional 200 GB of backup space and will only be billed for backups in excess of 200 GB. The first 200 GB of backup data is free. When the 200 GB storage is used up, the backups will be billed on a pay-per-use basis. For pricing details, see **Product Pricing Details**.

NOTICE

If your storage is frozen, it is no longer billed and the free backup space is also unavailable.

If your DB instance is frozen, no free backup space is available. As a result, the original backups of the DB instance will be billed.

- If you unfreeze the DB instance, the free backup space will be restored.
- If you delete the frozen DB instance, the original automated backups will also be deleted. You need to manually delete the original manual backups. After the deletion, the backup space will no longer be billed.

16 Network Security

16.1 What Security Protection Policies Does DDS Have?

DDS allows you to set the VPC which DDS DB instances belong to, ensuring that the DDS DB instances are isolated from other services. In addition, the IAM service is provided, achieving access control over DDS resources.

16.2 Do I Need to Use DDS in a VPC?

A VPC allows you to create virtual network environment in a private and isolated network to control the private IP address range, subnets, route tables, and network gateways. The VPC also allows you to define the virtual network topology and network configuration to make the network similar to the traditional IP network you are operating in the data center.

You may need to use DDS in the VPC in the following cases:

You want to run Internet-oriented web applications and retain the backend server that the public cannot access. To do so, you can create an ECS and a DDS DB instance in the same VPC, allocate a public IP address for the ECS, and deploy a web server on the ECS.

16.3 How Do I Ensure the Security of DDS in a VPC?

The VPC security group helps ensure the security of DDS in a VPC. In addition, ACL can be used to allow or reject I/O network traffic for each subnet. Use the internal security infrastructure (including the network firewall, intrusion detection, and protection system) to monitor all IPsec VPN connection-based input and output network traffic for VPC.

16.4 Does DDS Support IPv6 Subnets?

If you selected an IPv6 subnet, you cannot associate it with your instance. Select an IPv4 subnet.

16.5 How Can I Import the Root Certificate to a Windows or Linux OS?

Importing the Root Certificate to the Windows OS

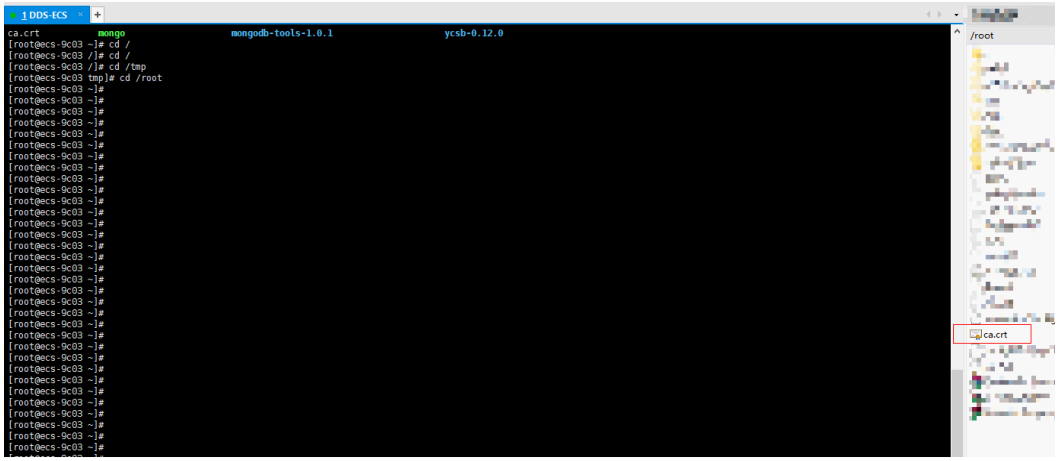
1. Click **Start** and choose **Run**. In the displayed **Run** dialog box, enter **MMC** and press **Enter**.
2. On the displayed console, choose **File** > **Add/Remove Snap-in**.
3. In the left **Available snap-ins** pane of the displayed dialog box, select **Certificates**. Click **Add** to add the certificate.
4. In the displayed **Certificates snap-in** dialog box, select **Computer account** and click **Next**.
5. In the displayed **Select Computer** dialog box, click **Finish**.
6. In the **Add or Remove Snap-ins** dialog box, click **OK**.
7. On the console, double-click **Certificates**.
8. Right-click **Trusted Root Certification Authorities** and choose **All Tasks** > **Import**.
9. Click **Next**.
10. Click **Browse**.
11. Locate the downloaded root certificate (a **ca.crt** file) and click **Open**. Then, click **Next**.
12. Click **Next**.
13. Click **Finish**.
14. Click **OK** to complete the import of the root certificate.

Importing the Root Certificate to the Linux OS

You can use a connection tool (such as WinSCP or PuTTY) to upload the certificate to any directory on a Linux OS.

Example:

Figure 16-1 Importing a certificate



17 Monitoring and Alarm

17.1 What DB Instance Monitoring Metrics Do I Need to Pay Attention To?

You need to focus on the CPU, memory, and storage space usage.

For more information about monitoring indicators, see **DDS Metrics**.

You can configure DDS to report alarms as needed. If an alarm is reported, you can take proper measures to clear it.

Configuration examples:

- Configure DDS to report alarms to Cloud Eye if CPU utilization reaching or exceeding a certain value (90% for example) for multiple times (3 for example) within a period of time (5 minutes for example).
- Configure DDS to report alarms to Cloud Eye if its memory utilization reaches or exceeds a specific value (for example, 90%) multiple times (for example, 4 times) within a set period (for example, 5 minutes).
- Configure DDS to report alarms to Cloud Eye if its storage utilization reaches or exceeds a specific value (for example, 85%) multiple times (for example, 5 times) within a set period (for example, 5 minutes).

NOTE

For details on Cloud Eye alarm configuration, see "Alarm Rule Management" in *Cloud Eye Service User Guide*.

If the CPU and memory usage stay high for a long time and the disk capacity cannot be expanded, you can change the CPU and memory specifications.

Measures:

If a storage usage alarm is reported, perform either of the following operations:

- Check the storage space consumption and see whether any space can be freed up by deleting data from DB instances or dumping the data to another system.
- Scale up the storage space. For details, see **Scaling Up Storage Space**.

17.2 What Should I Do If I Cannot Find Common Monitoring Items When Configuring DDS Alarm Rules?

Symptom

When a DDS alarm rule is configured, some common monitoring metrics, such as the CPU usage, memory usage, and storage space usage, are not displayed.

Possible Causes

When you configure alarm rules on the Cloud Eye console, the configured monitoring dimensions are inconsistent with the dimensions supported by the monitoring metric. As a result, the monitoring metrics are not displayed.

For example, if the objects whose CPU usage you want to monitor are dds mongos nodes or primary/standby nodes, the monitoring dimension you configure for an alarm rule on the Cloud Eye console must be at the node-level.

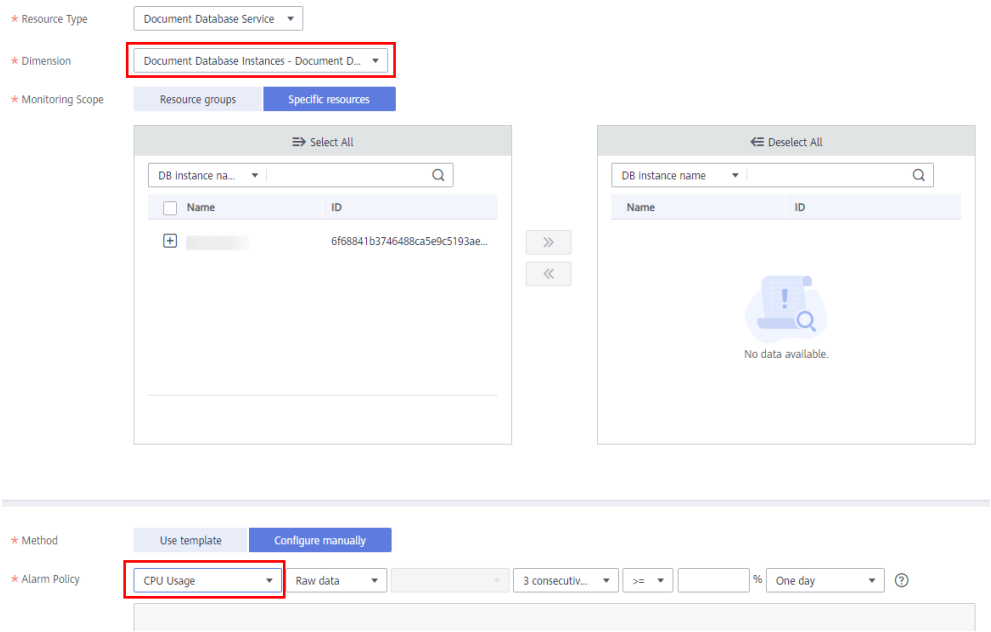
Solution

DDS supports instance-level and node-level metric monitoring. Set dimensions supported by monitoring metrics on the alarm rule page. Different metrics have different dimensions. For details, see **Supported DDS Metrics**.

For example, the monitored objects of CPU usage can be dds mongos nodes of a DDS cluster instance, or the primary and standby nodes of a DDS DB instance.

When configuring alarm rules on the Cloud Eye console, select **Document Database Instances - Document Database Node** for **Dimension**. Then, you can view the CPU usage in the alarm policy.

Figure 17-1 Configuring monitoring dimension



17.3 How Often Does DDS Collect Metrics?

The default monitoring interval is 1 minute. To improve the instantaneous accuracy of monitoring metrics, you can set the monitoring interval to 5 seconds.

For details, see **Configuring Monitoring by Seconds**.

17.4 How Do I Clear the Alarm that the Dirty Data in WiredTiger Cache Is Too High

Possible Causes

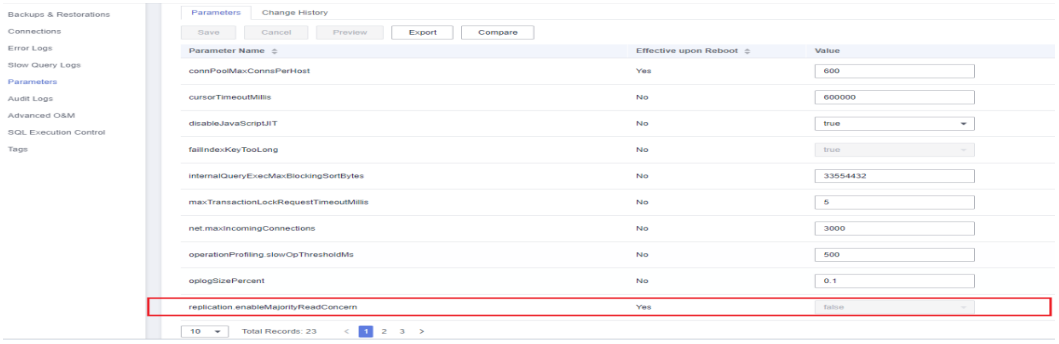
The dirty data in WiredTiger cache is too high in the following scenarios:

1. After the parameter **enableMajorityReadConcern** of MongoDB is set to **true**, the service write rate is too high. As a result, the secondary node cannot synchronize oplogs from the primary node in a timely manner.
2. After the parameter **enableMajorityReadConcern** of MongoDB is set to **false**, the CPU and memory specifications are too small to bear the write rate of the current service.

Fault Locating

- Check the running parameters of MongoDB. Log in to the console, click the instance name, and choose **Parameters** to view the value of **enableMajorityReadConcern**.

  Figure 17-2 Checking the value of enableMajorityReadConcern

- Log in to the management console, choose **Cloud Eye** > **Cloud Service Monitoring** > **Document Database Service**, and view the node metrics of WiredTiger Dirty Data Cache Percentage and Tracked Dirty Bytes in WiredTiger Cache.

Figure 17-3 Tracked Dirty Bytes in WiredTiger Cache

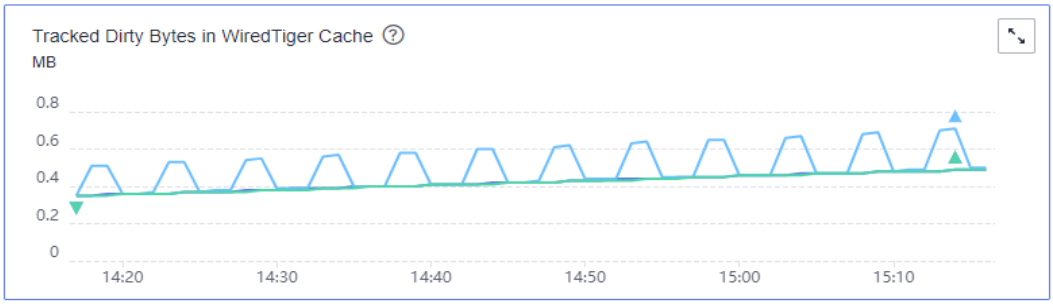
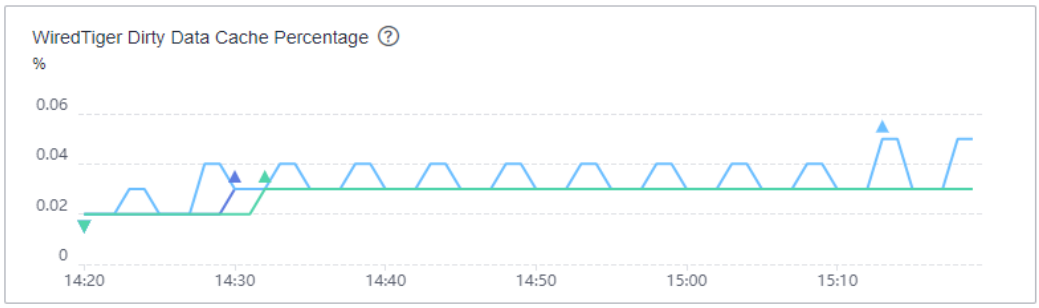


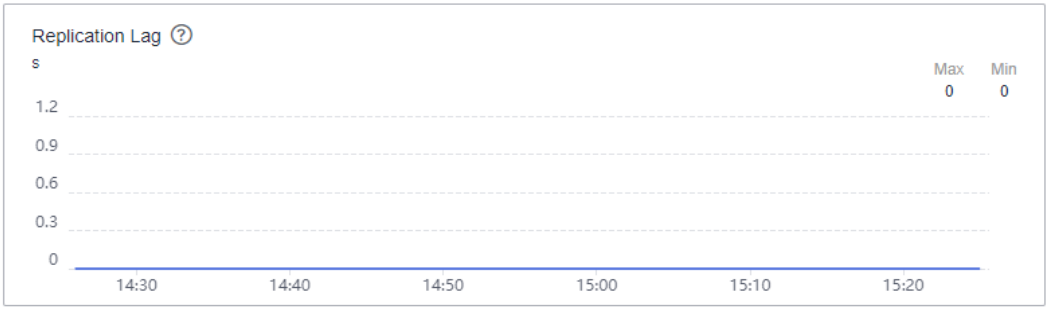
Figure 17-4 WiredTiger Dirty Data Cache Percentage



- Common scenarios:

  If the primary/standby replication latency keeps increasing, log in to the management console and choose **Cloud Eye** > **Secondary** to view the replication latency.

Figure 17-5 Replication Lag



Solution

Log in to the DDS console, click the instance name, and choose **Parameters** to view the value of enableMajorityReadConcern.

- If enableMajorityReadConcern is set to **false**, log in to the management console and choose **Cloud Eye** to view the metrics of Replication Lag, Tracked Dirty Bytes in WiredTiger Cache, WiredTiger Dirty Data Cache Percentage, CPU, and Memory. If the value of WiredTiger Dirty Data Cache Percentage exceeds 20% and keeps increasing, and the values of CPU and Memory exceed 90%, limit the client write rate in a timely manner and evaluate whether the specifications need to be upgraded.

- enableMajorityReadConcern is set to **true**.
  - If read consistency is not required, contact Huawei engineers to set this parameter to **false**.
  - If read consistency is required, the value of WiredTiger Dirty Data Cache Percentage exceeds 20% and keeps increasing, and the values of CPU and Memory exceed 90%, limit the client write rate in a timely manner and evaluate whether the specifications need to be upgraded.

