Cloud Object Storage

Data lake storage

Contents

Data lake storage
Cloud Native Datalake Storage
Getting Started

Last updated：2023-09-13 16:12:54

Feature Overview

Cloud Native Datalake Storage helps you quickly deploy a COS-based data lake storage service on TKE. Then, you can deploy big data and AI service applications required by various businesses in a TKE or EKS cluster. In addition, you can also use GooseFS to connect to massive distributed storage services.

Concepts and terms

The following lists some basic **concepts and terms** of Cloud Native Datalake Storage:

- **Environment**: It maintains the mappings between computing clusters and storage services. We recommend you uniformly manage computing clusters and storage services in an environment.

> **Note**
> If you need to delete a computing cluster, we recommend first clearing the Cloud Native Datalake Storage environment and then deleting the computing cluster in the container service console.

- **Computing cluster**: It is a container cluster for running various computing businesses. You can create a TKE or EKS cluster.
- **Storage service**: It refers to COS, which stores different types of data for computing.
- **Application market**: It houses the application components for diversified computing businesses, such as Flink and Spark. You can select an application as needed when creating an environment.

> **Note**
> When your container cluster is terminated, the deployed applications will also be destroyed. Please exercise caution when performing deletion operations.

- **GooseFS**: It manages different underlying buckets and caches frequently accessed data in your computing cluster to accelerate computing.

You can get some basic information in the following documents:

- COS: Getting Started with the Console describes how to create a bucket and upload/download files to/from it.
- TKE: Quickly Creating a Standard Cluster describes how to create a TKE or EKS cluster.
- Application market: Application Market describes how to create and deploy an application in a TKE cluster.
- Data Accelerator GooseFS: This service can be used to manage different underlying storage buckets and accelerate your business access.

Preparations

- Currently, Cloud Native Datalake Storage is provided through an allowlist. To use it, contact us for application.
- Cloud Native Datalake Storage relies on TKE and COS and requires **permissions to manipulate computing and storage services**. If you log in with a sub-account, make sure that the sub-account has at least the following permissions:
  - Permissions to manipulate COS buckets and files.
    - Permission to manipulate buckets: If you need to manage bucket configurations, get the corresponding permission from the root account. Generally, this permission doesn't affect data read/write and doesn't require extra configurations. It is sufficient to grant the read permission, such as the QcloudCOSBucketConfigRead policy.
    - Permission to manipulate files: Generally, computing jobs require reading/writing files from/to buckets. You can get full access from the root account, such as the QcloudCOSDataFullControl policy. Alternatively, the root account can grant the permission based on the principle of least privilege.
  - Permissions to manage container clusters:
    - Permission to manipulate clusters: Generally, you need to grant permissions to create and manipulate clusters. For detailed directions, see Using TKE Preset Policy Authorization.
    - Permission to manage clusters: TKE provides an authorization mode to connect to Kubernetes RBAC, so that sub-account access can be controlled in a refined manner. Sub-account operations are also subject to the TKE Kubernetes object-level permission control.
    - Permission to manipulate the application market: The application market relies on the operations of the TCR service. For detailed directions on how to authorize a sub-account, see TKE Image Registry Resource-level Permission Settings.

Instructions

The following details the steps, including environment creation, cluster association, computing application deployment, storage service association, and environment management:

1. Log in to the COS Console.
2. On the left sidebar, click **Cloud Native Datalake Storage**.
3. On the Cloud Native Datalake Storage page, you can see the capability overview and deployment guide.
    - The deployment guide is displayed by default, and you can click **Collapse Guide** in the top-right corner to disable it.
    - The environment list page allows for search. You can manipulate an existing environment as follows:
        - Click **Environment Name** to enter the environment details page and manage the environment.
        - Click **Associated Cluster** to enter the cluster details page in the TKE console.
        - Click **Associated Bucket** to enter the bucket page and view the file information.
4. Click **Create Environment** to enter the environment creation process. You need to select the corresponding container computing cluster and configure the following parameters:
    - **Environment Name**: It can contain up to 63 characters and must be globally unique.
    - **Region**: Select the region of the container cluster.
    - **Cluster Type**: It can be TKE or EKS. If there are no clusters in the current region, you can click **Create Container Cluster** to create one in the TKE console.
    - **Cluster**: It is the name of the cluster for deploying computing applications and running computing jobs based on the **specified region** and **specified cluster type** conditions.
    - **Computing application**: It indicates the application service required for running a computing job. Currently, Flink, k8s-big-data-suite, colocation, Airflow, PyTorch, and spark-operator applications are supported by default. You can select one or multiple applications as needed. If you need to deploy a custom application, you can go to the TKE console for deployment on your own.
5. Click **Next** to enter the **Bucket Configuration** page view. You can configure different storage buckets for the computing cluster on this page. By default, we provide GooseFS data accelerator service to manage different storage buckets and cache data to local nodes of the computing cluster for accelerating computing jobs. The required parameters are as follows:
    - Region: It is the region of the computing cluster by default and cannot be edited. If there are no available buckets for computing jobs in the region, you can click **Create Bucket** to create one.

- Bucket: You can select multiple buckets in the specified region. You can also mount only a specified file directory of a bucket.

  > **Note**
  > If you want to mount the entire storage bucket, there is no need to enter the second input box. If you need to specify a directory, you can do so by entering the directory name in the format `prefix/*`.

- Enable GooseFS: GooseFS accelerates computing jobs. It is enabled by default and cannot be modified. No extra fees will be incurred.

6. Click **Next** to enter the **GooseFS Application Configuration** page view.
   Since all computing tasks in the data lake environment need to access COS through GooseFS, you must configure the secretId and secretKey with permission to access the specified bucket for GooseFS.
7. Click **Next** and confirm the information.
8. To modify the configuration, click **Edit** to change the configuration information. After confirming the changes, click **Create Environment** to complete the environment creation. **Return to the environment list and refresh** to view the newly created Cloud Native Datalake Storage environment.
   If you need to **delete an environment**, click **Delete** in the environment list and confirm the deletion in the pop-up window.
9. Click the environment name in the list to enter the **Basic Information** page. We use three card views to describe environment information, computing cluster information, and storage bucket information.
   - Environment information: It displays the environment's name, region, associated computing cluster, storage service, and creation time.
   - Computing cluster information: It displays the computing cluster's name, number of nodes, and usage of CPU, memory, and GPU. You can click **View details** to enter the TKE console and view computing cluster details.
   - Bucket information: It displays the name, file URL, and GooseFS status of the bucket associated with the computing cluster. You can click **View details** to view the details of the storage service.

At this point, you have created a data lake environment.

Metadata Accelerator
Metadata Acceleration Overview

Last updated：2023-09-13 16:14:06

Metadata acceleration is a high-performance file system feature offered by COS. Metadata acceleration leverages the powerful metadata management feature of Cloud HDFS (CHDFS) at the underlying layer to allow you to access COS over file system semantics. The system design metrics can reach a bandwidth of up to 100 GB/s, over 100,000 queries per second (QPS), and a latency of milliseconds. Buckets with metadata acceleration enabled can be widely used in scenarios such as big data, high-performance computing, machine learning, and AI.
If you haven’t enabled metadata acceleration for a bucket, it cannot accelerate metadata access by default and does not support access with POSIX file semantics, making it less favorable in terms of file `LIST` and not support `RENAME` . After enabling metadata acceleration, you can use POSIX file semantics to access objects in buckets through native COS APIs or the Hadoop tool deployed on the client side, the console, or SDKs.

Note
Metadata acceleration can only be enabled upon bucket creation and cannot be disabled once enabled. Please **think twice** before enabling it.

Usage Limits

The following table compares the product feature support between enabling and not enabling metadata acceleration:

Metric	Metadata Acceleration Enabled	Metadata Acceleration Disabled
Creating/Querying/Deleting buckets	This feature is supported.	This feature is supported.
Uploading/Downloading/Deleting objects	This feature is supported.	This feature is supported.
Bucket permissions	This feature is supported.	This feature is supported.
Object Permissions	Inheriting bucket permissions by default	This feature is supported.

Intra-Region Disaster Recovery	This feature is supported.	This feature is supported.
Bucket encryption	N/A	This feature is supported.
Object encryption	N/A	This feature is supported.
CORS	N/A	This feature is supported.
Hotlink protection	N/A	This feature is supported.
Versioning	N/A	This feature is supported.
Cross-bucket replication	N/A	This feature is supported.
FAQs About Static Website	N/A	This feature is supported.
Origin-pull settings	N/A	This feature is supported.
Lifecycle	This feature is supported.	This feature is supported.
Inventory	N/A	This feature is supported.
Number of bucket tags	This feature is supported.	This feature is supported.
COS Select	N/A	This feature is supported.
COS Batch	N/A	This feature is supported.
CI	N/A	This feature is supported.

Billing description

Currently, metadata acceleration is still in beta testing and is not billed. If billing is required in the future, we will notify you via Message Center, email, or SMS. You can check your message center or see Billing Overview to keep up with the latest billing plans.

Data Access
Using Hadoop FileSystem API Code to Access COS Metadata Acceleration Bucket

Last updated：2023-09-13 16:16:44

Scenario

If metadata acceleration is enabled for a COS bucket, you can use Java code to access the bucket through Hadoop FileSystem API in addition to using the Hadoop command line and big data components. This document describes how to do so.

Preparations

- Make sure that metadata acceleration has been enabled, and the environment deployment and HDFS protocol configuration have been performed correctly. For more information, see Using HDFS to Access a Bucket with Metadata Acceleration Enabled.
- If there is a Hadoop environment, you can verify whether the Hadoop command line can be accessed correctly.

Instructions

1. Create a Maven project and add the following dependencies to `pom.xml` in Maven (set the versions of the `hadoop-common`, `hadoop-cos`, and `cos_api-bundle` packages based on your actual Hadoop version and environment).

```
<dependencies>
     <dependency>
         <groupId>org.apache.hadoop</groupId>
         <artifactId>hadoop-common</artifactId>
         <version>2.8.5</version>
         <scope>provided</scope>
     </dependency>
      <dependency>
         <groupId>com.qcloud.cos</groupId>
         <artifactId>hadoop-cos</artifactId>
         <version>xxx</version>
```

```
        </dependency>
        <dependency>
            <groupId>com.qcloud</groupId>
            <artifactId>cos_api-bundle</artifactId>
            <version>xxx</version>
        </dependency>

</dependencies>
```

2. Refer to the following Hadoop code for modifications. The configuration items can be modified according to the Configuration Item Description document. **Pay special attention to the explanations related to data persistence and visibility.**
The following list only includes some common file system operations. For other interfaces, please refer to the Hadoop FileSystem Interface Documentation.

```
package com.qcloud.cos.demo;

import org.apache.commons.io.IOUtils;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.FSDataInputStream;
import org.apache.hadoop.fs.FSDataOutputStream;
import org.apache.hadoop.fs.FileChecksum;
import org.apache.hadoop.fs.FileStatus;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;

import java.io.IOException;
import java.net.URI;
import java.nio.ByteBuffer;

public class Demo {
         private static FileSystem initFS() throws IOException {
             Configuration conf = new Configuration();
            // For configuration options, refer to
https://cloud.tencent.com/document/product/436/6884#.E4.B8.8B.E8.BD.BD
.E4.B8.8E.E5.AE.89.E8.A3.85
            // The following configurations are required
            conf.set("fs.cosn.impl",
"org.apache.hadoop.fs.CosFileSystem");
            conf.set("fs.AbstractFileSystem.cosn.impl",
"org.apache.hadoop.fs.CosN");
            conf.set("fs.cosn.userinfo.secretId", "xxxxxx");
            conf.set("fs.cosn.userinfo.secretKey", "xxxxxx");
            conf.set("fs.cosn.bucket.region", "xxxxxx");
            conf.set("fs.cosn.tmp.dir", "/data/chdfs_tmp_cache");

            // For configuration options, refer to
https://cloud.tencent.com/document/product/436/71550
            // Required configuration items for accessing via POSIX
(recommended method)
            conf.set("fs.cosn.trsf.fs.AbstractFileSystem.ofs.impl",
"com.qcloud.chdfs.fs.CHDFSDelegateFSAdapter");
            conf.set("fs.cosn.trsf.fs.ofs.impl",
"com.qcloud.chdfs.fs.CHDFSHadoopFileSystemAdapter");
            conf.set("fs.cosn.trsf.fs.ofs.tmp.cache.dir",
"/data/chdfs_tmp_cache");

            // Replace appid with the actual AppID
            conf.set("fs.cosn.trsf.fs.ofs.user.appid", "1250000000");
            // Replace 'region' with the actual region
            conf.set("fs.cosn.trsf.fs.ofs.bucket.region", "ap-
beijing");
            // For other optional configurations, refer to the
official documentation at
https://cloud.tencent.com/document/product/436/6884#.E4.B8.8B.E8.BD.BD
.E4.B8.8E.E5.AE.89.E8.A3.85
            // Enable CRC64 checksum, disabled by default, which means
the 'hadoop fs -checksum' command cannot be used to obtain the CRC64
checksum value of a file.
            conf.set("fs.cosn.crc64.checksum.enabled", "true");
            String cosHadoopFSUrl = "cosn://examplebucket-
12500000000/";
            return FileSystem.get(URI.create(cosHadoopFSUrl), conf);
        }

    private static void mkdir(FileSystem fs, Path filePath) throws
IOException {
```

```
                fs.mkdirs(filePath);
            }

     private static void createFile(FileSystem fs, Path filePath)
throws IOException {
                // Create a file (overwrite it if it already exists)
                // if the parent dir does not exist, fs will create it!
                FSDataOutputStream out = fs.create(filePath, true);
                try {
                        // Write a file
                        String content = "test write file";
                        out.write(content.getBytes());
                } finally {
                        // If close returns successfully, it indicates
data write success; if an exception is thrown, it indicates data write
failure.
                        out.close();
                }
            }

     private static void readFile(FileSystem fs, Path filePath) throws
IOException {
                FSDataInputStream in = fs.open(filePath);
                try {
                        byte[] buf = new byte[4096];
                        int readLen = -1;
                        do {
                                readLen = in.read(buf);
                        } while (readLen >= 0);
                } finally {
                        IOUtils.closeQuietly(in);
                }
            }


         private static void queryFileOrDirStatus(FileSystem fs, Path
path) throws IOException {
                FileStatus fileStatus = fs.getFileStatus(path);
                if (fileStatus.isDirectory()) {
                        System.out.printf("path %s is dir\n", path);
                        return;
                }

        long fileLen = fileStatus.getLen();
            long accessTime = fileStatus.getAccessTime();
            long modifyTime = fileStatus.getModificationTime();
            String owner = fileStatus.getOwner();
            String group = fileStatus.getGroup();


            System.out.printf("path %s is file, fileLen: %d,
accessTime: %d, modifyTime: %d, owner: %s, group: %s\n",
                        path, fileLen, accessTime, modifyTime, owner,
group);
        }


        // The default checksum type is COMPOSITE-CRC32C
        private static void getFileCheckSum(FileSystem fs, Path path)
throws IOException {
            FileChecksum checksum = fs.getFileChecksum(path);
            System.out.printf("path %s, checkSumType: %s,
checkSumCrcVal: %d\n",
                        path, checksum.getAlgorithmName(),
ByteBuffer.wrap(checksum.getBytes()).getInt());
        }


        private static void copyFileFromLocal(FileSystem fs, Path
chdfsPath, Path localPath) throws  IOException {
            fs.copyFromLocalFile(localPath, chdfsPath);
        }


        private static void copyFileToLocal(FileSystem fs, Path
chdfsPath, Path localPath) throws  IOException {
            fs.copyToLocalFile(chdfsPath, localPath);
        }
```

```
         private static void renamePath(FileSystem fs, Path oldPath,
Path newPath) throws IOException {
             fs.rename(oldPath, newPath);
         }


         private static void listDirPath(FileSystem fs, Path dirPath)
throws IOException {
             FileStatus[] dirMemberArray = fs.listStatus(dirPath);


             for (FileStatus dirMember : dirMemberArray) {
                     System.out.printf("dirMember path %s, fileLen:
%d\n", dirMember.getPath(), dirMember.getLen());
             }
         }


         // Recursive flag is used for deleting directories
         // If recursive is set to false and dir is not empty, the
operation will fail.
         private static void deleteFileOrDir(FileSystem fs, Path path,
boolean recursive) throws IOException {
             fs.delete(path, recursive);
         }


         private static void closeFileSystem(FileSystem fs) throws
IOException {
             fs.close();
         }


         public static void main(String[] args) throws IOException {
             // Initialize the file system
             FileSystem fs = initFS();


             // Create a file
```

```
            Path chdfsFilePath = new
Path("/folder/exampleobject.txt");
            createFile(fs, chdfsFilePath);


            // Read the file
            readFile(fs, chdfsFilePath);


            // Query file or directory
            queryFileOrDirStatus(fs, chdfsFilePath);


            // Obtain the file checksum
            getFileCheckSum(fs, chdfsFilePath);


            // Copy a file from local
            Path localFilePath = new
Path("file:///home/hadoop/cosn_demo/data/exampleobject.txt");
            copyFileFromLocal(fs, chdfsFilePath, localFilePath);


            // Retrieve the file to the local system
            Path localDownFilePath = new
Path("file:///home/hadoop/cosn_demo/data/exampleobject.txt");
            copyFileToLocal(fs, chdfsFilePath, localDownFilePath);


            // Renaming
            Path newPath = new Path("/doc/example.txt");
            renamePath(fs, chdfsFilePath, newPath);


            // Delete a file
            deleteFileOrDir(fs, newPath, false);


            // Create directory
            Path dirPath = new Path("/folder");
            mkdir(fs, dirPath);


            // Create a file in the directory
            Path subFilePath = new Path("/folder/exampleobject.txt");
            createFile(fs, subFilePath);


            // List the directory
            listDirPath(fs, dirPath);


            // Delete directory
            deleteFileOrDir(fs, dirPath, true);


            // Close the file system
            closeFileSystem(fs);
        }
}
```

3. Compile and run.

> **Note**
> - Before running the code, be sure to correctly set `classpath`, which must contain the paths of the Hadoop `common` package and the JAR package depended on by the metadata acceleration bucket.
> - For EMR environments, if you follow the steps in Using HDFS to Access a Bucket with Metadata Acceleration Enabled, the Hadoop common package is usually located in the `/usr/local/service/hadoop/share/hadoop/common/` directory, and the Jar packages required for metadata acceleration buckets are typically found in the `/usr/local/service/hadoop/share/hadoop/common/lib/` directory.

Using Hadoop Shell to Access COS Metadata-Accelerated Bucket

Last updated：2025-12-23 16:56:18

Scenarios

For relatively simple file operations, they can be performed directly using the command-line approach. This article guides you on how to access Metadata Acceleration-enabled buckets via Hadoop Shell using the command-line method.

Prerequisites

- Ensure that the metadata-acceleration service has been enabled, and that the environment has been properly deployed and the HDFS protocol configured. For specific deployment and configuration details, see Using the HDFS protocol to access a bucket with metadata acceleration enabled.
- Log in to any server that has completed environment deployment and HDFS protocol configuration, and go to the command-line interface.

Supported Operations

1. For detailed Hadoop command-line documentation, see Hadoop Shell Commands.
2. Below are some common file operations.

Traverse Directory (list)

```
hadoop fs -ls cosn://examplebucket-1250000000/
```

Create Directory (mkdir)

```
hadoop fs -mkdir -p cosn://examplebucket-1250000000/test_01/xxx
```

Uploading Files

```
hadoop fs -put ./len1m.txt cosn://examplebucket-
1250000000/test_01/xxx/len1m_1.txt
```

Downloading a File

```
hadoop fs -get cosn://examplebucket-1250000000/test_01/xxx/len1m_1.txt
./len1m_1.txt
```

Deletes files

```
hadoop fs -rm cosn://examplebucket-1250000000/test_01/xxx/len1m_1.txt
```

Deleting a Directory

```
hadoop fs -rm -r cosn://examplebucket-1250000000/test_01/xxx
```

Directory Statistics (du)

```
hadoop fs -du cosn://examplebucket-1250000000/test_01
```

File Checksum

```
hadoop fs -checksum cosn://examplebucket-
1250000000/test_01/xxx/len1m_1.txt
```

Big Data Security Ranger Overview

Last updated：2023-09-13 16:18:09

Example

COS Ranger Service is a big data permission control solution based on Tencent Cloud storage-computing separation. It features fine granularity, compatibility with Hadoop Ranger, and a pluggable architecture and helps you centrally manage big data components and manage the storage permissions in the cloud. For more information on the service and its architecture scheme, see COS Ranger Permission System Solution.
COS Ranger Service has been widely used since its launch. However, as customers have diverse businesses and a complex background, some questions have arisen. To answer them, this document describes the service as well as its version details and precautions.

Version Description

Components

Its components include Ranger-Plugin, COS Ranger Server, COS Ranger Client (i.e., Hadoop-Ranger-Client), and COSN Ranger Interface.

Ranger-Plugin

It provides a service definition plugin on the Ranger server based on the Ranger protocol (for more information, see Ranger Stacks - How to add a custom plugin?) and description of the COS service on the Ranger side. After this plugin is deployed, you can enter the COS permission policies such as path, bucket, user, and group access policies on the Ranger control page.

COS Ranger Server

It integrates the Ranger client, periodically syncs permission policies from the Ranger server, and verifies the permission locally after receiving an authentication request. In addition, it also provides `DelegationToken` generation and renewal APIs in Hadoop.
In the EMR environment, it is installed in `/usr/local/service/cosranger/lib` by default. For example, in the package name `cos-ranger-service-5.1.2-jar-with-dependencies.jar`, `5.1.2` is the version number of COS Ranger Server.

COS Ranger Client

The Hadoop SDK plugin loads COS Ranger Client dynamically according to the configuration in the `core-site.xml` file and forwards the permission verification requests to COS Ranger Server.
In the EMR environment, it is installed in `/usr/local/service/hadoop/share/hadoop/common/lib` by default. For example, in the package name `hadoop-ranger-client-for-hadoop-2.8.5-5.0.jar`, `2.8.5` is the Hadoop version number, and `5.0` is the package version number.

COSN Ranger Interface

This plugin consists of COS Ranger Server and COS Ranger Client, which share common data and interface definitions.
In the EMR environment, it is installed in `/usr/local/service/hadoop/share/hadoop/common/lib`. For example, in the package name `cosn-ranger-interface-1.0.4.jar`, `1.0.4` is the version number of COSN Ranger Interface.
You can get the above JAR packages from GitHub. For other components such as COS Ranger Client packages for Impala and Presto, contact the EMR team.
The above components will be installed automatically when you purchase the Ranger and COS Ranger components in the EMR console. You can also install them by yourself as instructed in CHDFS Ranger Permission System Solution.

Release notes

Versions can be divided into two types by core architecture: **ZooKeeper-dependent service registration and discovery** and **ZooKeeper-independent service registration and discovery**.
COS Ranger Server provides the service for COS Ranger Client to call.

- If COS Ranger Client discovers the service address of COS Ranger Server through ZooKeeper, you need to configure the ZooKeeper address, which is a feature specific to a **ZooKeeper-dependent version**.
- If COS Ranger Client doesn't depend on ZooKeeper to discover the COS Ranger Server service, you only need to configure `qcloud.object.storage.ranger.service.address` to directly specify the service address of COS Ranger Server, which is a feature specific to a **ZooKeeper-independent version**.

Version mappings

Component	ZooKeeper-Dependent Service Registration and Discovery	ZooKeeper-Independent Service Registration and Discovery
COS Ranger Server	v5.0.9 or earlier	v5.1.1 or later
COS Ranger Client	v3.9 or earlier	v4.1 or later

COSN Ranger Interface	v1.0.3	v1.0.4 or later

Note
If you use a version that **does not rely on Zookeeper service and discovery**, the COS Ranger Server must be v5.1.1 or later (recommended **v5.1.2**), the COS Ranger Client must be v4.1 or later (recommended **v5.0**), and the COSN Ranger Interface must be v1.0.4 or later.

Version compatibility

Versions can be mapped based on the above two types. However, as versions may not be divided into the two types due to diverse customers with a complex background, the compatibility relationships between components are as listed below. Components on the same row are compatible with each other.

COS Ranger Client Version	COS Ranger Server Version	COSN Ranger Interface Version	Dependency on ZooKeeper for Service Registration and Discovery
version ⩽ v3.9	version ⩽ v5.0.9	v1.0.3	Required
version ⩽ v3.9	version ⩾ v5.1.1	v1.0.4	Required
version ⩾ v4.1	version ⩾ v5.1.1	v1.0.4	Not required
version ⩾ v5.0	version ⩽ v5.0.9	v1.0.4	Required
version ⩾ v5.0	version ⩾ v5.1.1	v1.0.4	Not required

Note
- COS Ranger Client 5.0 is compatible with all versions of COS Ranger Server.
- COS Ranger Client 4.1 is compatible with only COS Ranger Server 5.1.1 or later.
- Though COS Ranger Client 3.x is compatible with all versions of COS Ranger Server, it can only depend on ZooKeeper to discover the COS Ranger Server service (the shortcomings of ZooKeeper-dependent versions are as detailed below).
- The COS Ranger Client and COSN Ranger Interface packages must be placed in the same directory for the Hadoop SDK plugin to load dynamically.

Notes

- Ranger verification needs to be enabled for the metadata acceleration bucket or CHDFS file system in the console.
- Use the **Zookeeper Service and Discovery Dependency** version, and configure the qcloud.object.storage.zk.address in core-site.xml. The value should be the Zookeeper address (separated by commas).
- If you use COS Ranger Server 5.1.1 or 5.1.2 but COS Ranger Client 3.x, you still need to depend on ZooKeeper for service registration and discovery, so you need to set `qcloud.object.storage.zk.address` to the ZooKeeper address (separate multiple addresses by comma such as `10.0.0.8:2181,10.0.0.9:2181,10.0.0.10:2181`) in `core-site.xml` .
- When using the **zookeeper-independent service discovery** version, you need to configure qcloud.object.storage.ranger.service.address in core-site.xml. The value should be the COS Ranger Server service address (separated by commas, for example, 127.0.0.1:9999,128.0.0.1:9999).
- To use the OFS protocol for access, you need to set `fs.ofs.ranger.enable.flag` to `true` in `core-site.xml` .
- To use the COSN protocol for access, you need to set `fs.cosn.credentials.provider` to `org.apache.hadoop.fs.auth.RangerCredentialsProvider` in `core-site.xml` .

Configuration Items

Configuration items	Note	Sample
qcloud.object.storage.zk.address	ZooKeeper address for COS Ranger Server registration	10.0.0.8:2181,10.0.0.9:2181,10.0.0.10:2181
qcloud.object.storage.ranger.service.address	COS Ranger Server RPC service address	127.0.0.1:9999,128.0.0.1:9999
fs.ofs.ranger.enable.flag	Ranger toggle when the OFS protocol is used	true
fs.cosn.credentials.provider	Ranger authentication class path when the COSN protocol is used	org.apache.hadoop.fs.auth.RangerCredentialsProvider
fs.cosn.posix.bucket.use_ofs_ranger.enabled	Whether to use CHDFS Ranger authentication	By default, it is set to false, which means COSN Ranger authentication. If set to true, it enables CHDFS Ranger authentication. Note: This configuration item is available in hadoop-cos v8.1.7 and later versions.

Configuration items

Component Version	Configuration items
cos ranger verser ⩽ v5.0.9 cos ranger client ⩽ v3.9 OR = v5.0 OFS Protocol Access	qcloud.object.storage.zk.address fs.ofs.ranger.enable.flag
cos ranger verser ⩽ v5.0.9 cos ranger client ⩽ v3.9 OR = v5.0 COSN Protocol Access	qcloud.object.storage.zk.address fs.cosn.credentials.provider

cos ranger verser = v5.1.1 OR = v5.1.2 cos ranger client ⩾ v4.1 OFS Protocol Access	qcloud.object.storage.ranger.service.address fs.ofs.ranger.enable.flag
cos ranger verser = v5.1.1 OR = v5.1.2 cos ranger client ⩾ v4.1 COSN Protocol Access	qcloud.object.storage.ranger.service.address fs.cosn.credentials.provider

Note

- To access the metadata acceleration bucket over the COSN protocol and use CHDFS Ranger for authentication, set `fs.cosn.posix.bucket.use_ofs_ranger.enabled` to `true` and make sure that Hadoop-COS is on v8.1.7 or later.
- If you add or modify the above configuration items, you need to restart big data components such as ResourceManager/NodeManager in YARN, HiveMetaStore/HiveServer2 in Hive, and applications in Impala and Presto.

Recommended versions

Component	Version No.
cos-ranger-server	>= v5.1.2
cos-ranger-client	>= v5.0
cosn-ranger-interface	>= v1.0.4

Recommendation description

The above versions are recommended for the following reasons:

- ZooKeeper is used only for master election but not for service registration and discovery, which greatly alleviates the pressure on ZooKeeper during big data jobs. As during each big data job, a large number of tasks access ZooKeeper to discover the COS Ranger Server service, bringing a huge pressure on ZooKeeper and affecting the stability of other big data components.
- The `hadoop-ranger-client` package on v5.0 is compatible with early versions of COS Ranger Server packages, which makes it easier for old users to upgrade COS Ranger Server.
- On COS Ranger Server versions earlier than 5.1.2, the obtained leader IP may be different from the IP in the leader latch. Moreover, the information that COS Ranger Server registers with ZooKeeper will be simplified to facilitate subsequent extension or upgrade.
- Several bugs are fixed.

Authentication FAQs

What should I do if the error `IOException: init fs.ofs.ranger.client.impl failed` is reported?

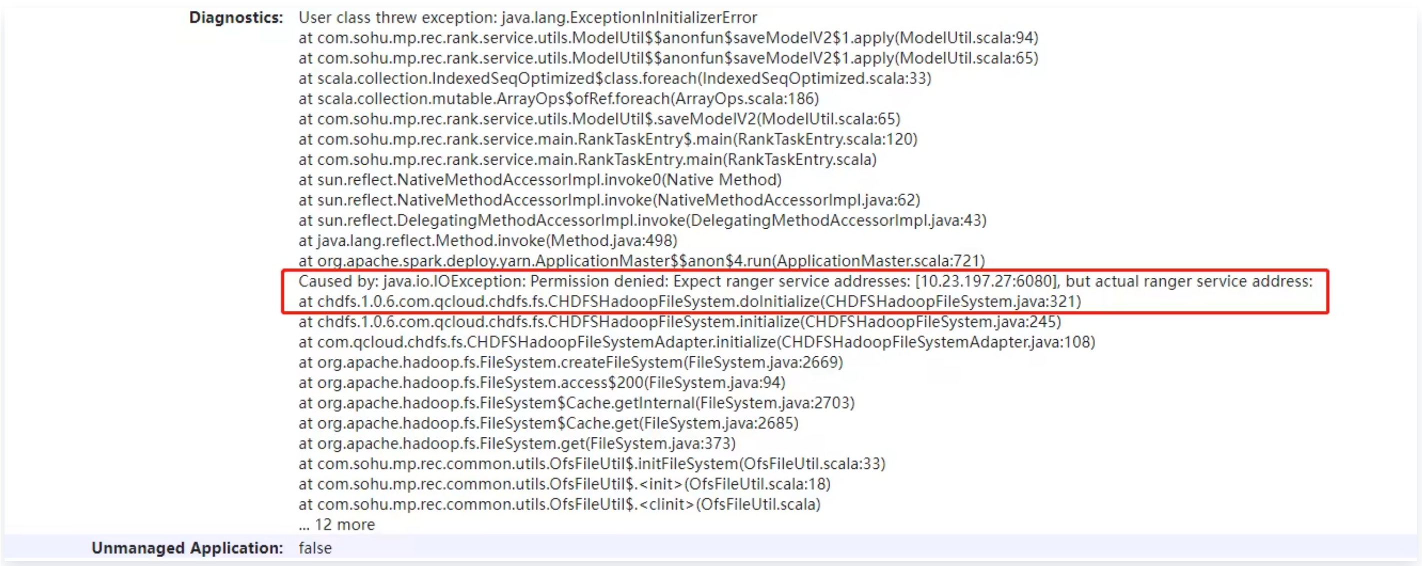
- If `Caused by: java.io.IOException: invalid zk address null` is displayed, you need to set `qcloud.object.storage.zk.address` to the ZooKeeper address (separate multiple addresses by comma such as `10.0.0.8:2181,10.0.0.9:2181,10.0.0.10:2181`) in `core-site.xml`.
- If `Caused by: java.io.IOException: ranger client is null, maybe ranger server for qcloud object storage is not deployed!` is displayed, refer to the next question.

What should I do if the error `ranger client is null, maybe ranger server for qcloud object storage is not deployed!` is reported?

Below are the error causes:

- If the `hadoop-ranger-client` package is on v3.8 or earlier, the ZooKeeper watch may be lost. In this case, we recommend you upgrade the package to v5.0.
- Check the configuration item `qcloud.object.storage.zk.address` or `qcloud.object.storage.ranger.service.address`.
- Check whether the COS Ranger Server service and process are normal.

What should I do if the error `Expect ranger service addresses: [127.0.0.1:6080,128.0.0.1:6080], but actual ranger service address` is reported?

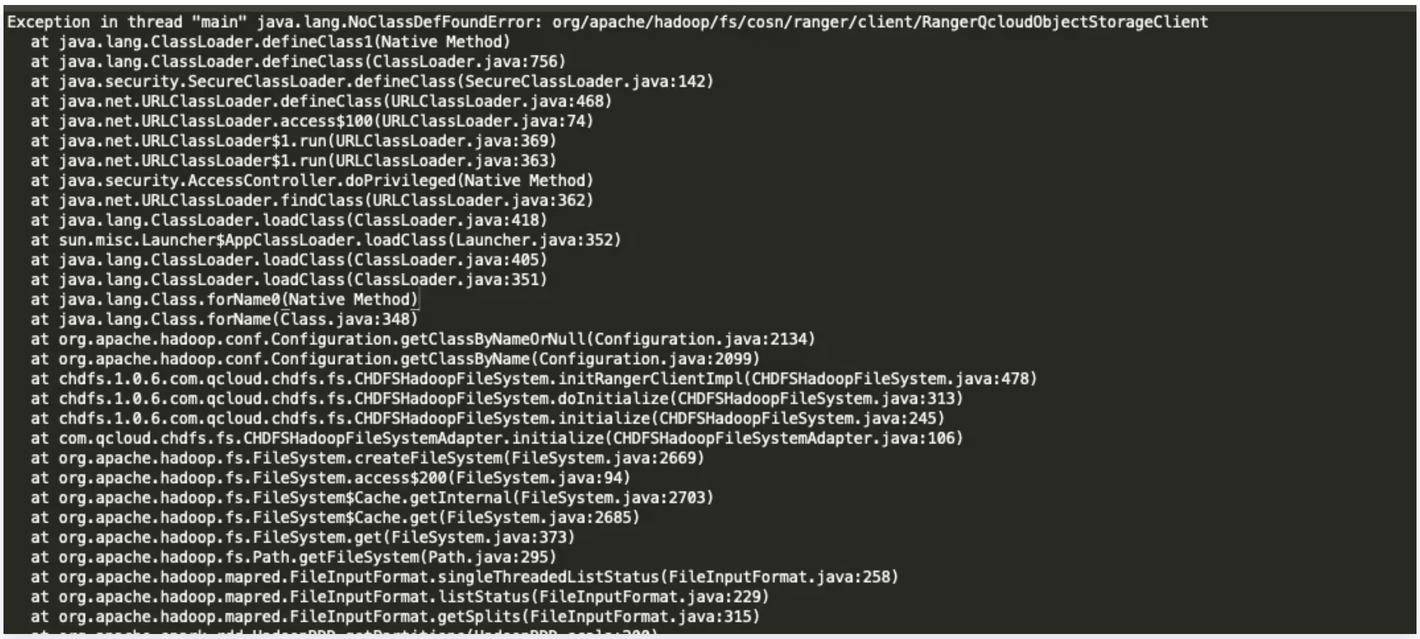


```
Diagnostics:  User class threw exception: java.lang.ExceptionInInitializerError
              at com.sohu.mp.rec.rank.service.utils.ModelUtil$$anonfun$saveModelV2$1.apply(ModelUtil.scala:94)
              at com.sohu.mp.rec.rank.service.utils.ModelUtil$$anonfun$saveModelV2$1.apply(ModelUtil.scala:65)
              at scala.collection.IndexedSeqOptimized$class.foreach(IndexedSeqOptimized.scala:33)
              at scala.collection.mutable.ArrayOps$ofRef.foreach(ArrayOps.scala:186)
              at com.sohu.mp.rec.rank.service.utils.ModelUtil$.saveModelV2(ModelUtil.scala:65)
              at com.sohu.mp.rec.rank.service.main.RankTaskEntry$.main(RankTaskEntry.scala:120)
              at com.sohu.mp.rec.rank.service.main.RankTaskEntry.main(RankTaskEntry.scala)
              at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
              at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)
              at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
              at java.lang.reflect.Method.invoke(Method.java:498)
              at org.apache.spark.deploy.yarn.ApplicationMaster$$anon$4.run(ApplicationMaster.scala:721)
              Caused by: java.io.IOException: Permission denied: Expect ranger service addresses: [10.23.197.27:6080], but actual ranger service address:
              at chdfs.1.0.6.com.qcloud.chdfs.fs.CHDFSHadoopFileSystem.doInitialize(CHDFSHadoopFileSystem.java:321)
              at chdfs.1.0.6.com.qcloud.chdfs.fs.CHDFSHadoopFileSystem.initialize(CHDFSHadoopFileSystem.java:245)
              at com.qcloud.chdfs.fs.CHDFSHadoopFileSystemAdapter.initialize(CHDFSHadoopFileSystemAdapter.java:108)
              at org.apache.hadoop.fs.FileSystem.createFileSystem(FileSystem.java:2669)
              at org.apache.hadoop.fs.FileSystem.access$200(FileSystem.java:94)
              at org.apache.hadoop.fs.FileSystem$Cache.getInternal(FileSystem.java:2703)
              at org.apache.hadoop.fs.FileSystem$Cache.get(FileSystem.java:2685)
              at org.apache.hadoop.fs.FileSystem.get(FileSystem.java:373)
              at com.sohu.mp.rec.common.utils.OfsFileUtil$.initFileSystem(OfsFileUtil.scala:33)
              at com.sohu.mp.rec.common.utils.OfsFileUtil$.<init>(OfsFileUtil.scala:18)
              at com.sohu.mp.rec.common.utils.OfsFileUtil$.<clinit>(OfsFileUtil.scala)
              ... 12 more
Unmanaged Application:  false
```

- This error occurs because Ranger verification is enabled for the metadata acceleration bucket or CHDFS in the console but not on the client.

- To use the OFS protocol for access, you need to set `fs.ofs.ranger.enable.flag` to `true` in `core-site.xml`.
- To use the COSN protocol for access, you need to set `fs.cosn.credentials.provider` to `org.apache.hadoop.fs.auth.RangerCredentialsProvider` in `core-site.xml`.

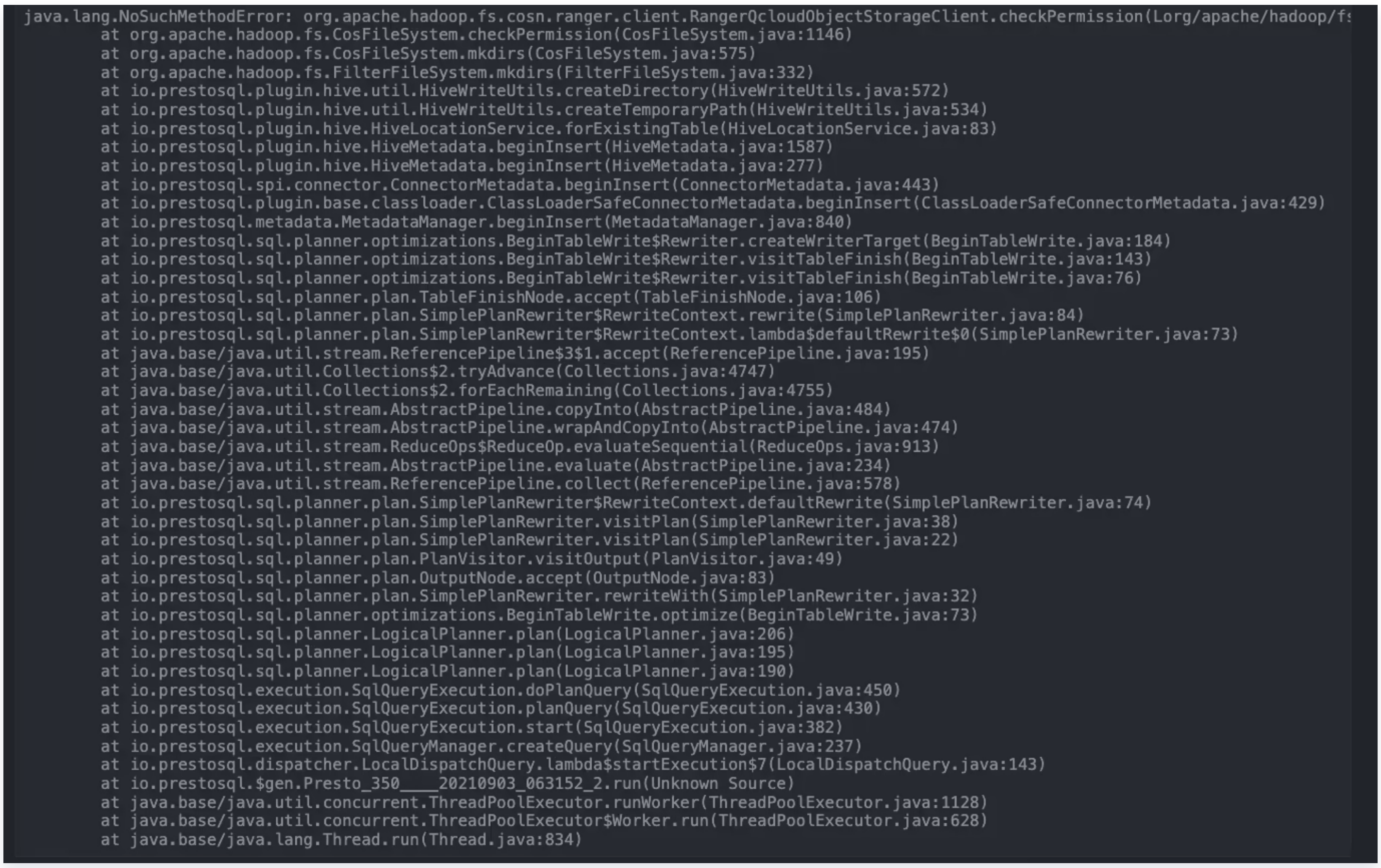
What should I do if the `RangerQcloudObjectStorageClient` class is not found?



```
Exception in thread "main" java.lang.NoClassDefFoundError: org/apache/hadoop/fs/cosn/ranger/client/RangerQcloudObjectStorageClient
    at java.lang.ClassLoader.defineClass1(Native Method)
    at java.lang.ClassLoader.defineClass(ClassLoader.java:756)
    at java.security.SecureClassLoader.defineClass(SecureClassLoader.java:142)
    at java.net.URLClassLoader.defineClass(URLClassLoader.java:468)
    at java.net.URLClassLoader.access$100(URLClassLoader.java:74)
    at java.net.URLClassLoader$1.run(URLClassLoader.java:369)
    at java.net.URLClassLoader$1.run(URLClassLoader.java:363)
    at java.security.AccessController.doPrivileged(Native Method)
    at java.net.URLClassLoader.findClass(URLClassLoader.java:362)
    at java.lang.ClassLoader.loadClass(ClassLoader.java:418)
    at sun.misc.Launcher$AppClassLoader.loadClass(Launcher.java:352)
    at java.lang.ClassLoader.loadClass(ClassLoader.java:405)
    at java.lang.ClassLoader.loadClass(ClassLoader.java:351)
    at java.lang.Class.forName0(Native Method)
    at java.lang.Class.forName(Class.java:348)
    at org.apache.hadoop.conf.Configuration.getClassByNameOrNull(Configuration.java:2134)
    at org.apache.hadoop.conf.Configuration.getClassByName(Configuration.java:2099)
    at chdfs.1.0.6.com.qcloud.chdfs.fs.CHDFSHadoopFileSystem.initRangerClientImpl(CHDFSHadoopFileSystem.java:478)
    at chdfs.1.0.6.com.qcloud.chdfs.fs.CHDFSHadoopFileSystem.doInitialize(CHDFSHadoopFileSystem.java:313)
    at chdfs.1.0.6.com.qcloud.chdfs.fs.CHDFSHadoopFileSystem.initialize(CHDFSHadoopFileSystem.java:245)
    at com.qcloud.chdfs.fs.CHDFSHadoopFileSystemAdapter.initialize(CHDFSHadoopFileSystemAdapter.java:106)
    at org.apache.hadoop.fs.FileSystem.createFileSystem(FileSystem.java:2669)
    at org.apache.hadoop.fs.FileSystem.access$200(FileSystem.java:94)
    at org.apache.hadoop.fs.FileSystem$Cache.getInternal(FileSystem.java:2703)
    at org.apache.hadoop.fs.FileSystem$Cache.get(FileSystem.java:2685)
    at org.apache.hadoop.fs.FileSystem.get(FileSystem.java:373)
    at org.apache.hadoop.fs.Path.getFileSystem(Path.java:295)
    at org.apache.hadoop.mapred.FileInputFormat.singleThreadedListStatus(FileInputFormat.java:258)
    at org.apache.hadoop.mapred.FileInputFormat.listStatus(FileInputFormat.java:229)
    at org.apache.hadoop.mapred.FileInputFormat.getSplits(FileInputFormat.java:315)
```

- If the cosn-ranger-interface package is missing, you can obtain it from the cosn-ranger-interface directory on Github.
- Other relevant classes are not found. You can check whether the `cosn-ranger-interface` and `hadoop-ranger-client` packages exist, their versions are matched, and they are placed in the correct path.
- If other relevant classes are not found, it may be caused by the package shade path. In this case, contact us for assistance.

What should I do if the error `NoSuchMethodError` is reported?



```
java.lang.NoSuchMethodError: org.apache.hadoop.fs.cosn.ranger.client.RangerQcloudObjectStorageClient.checkPermission(Lorg/apache/hadoop/f
        at org.apache.hadoop.fs.CosFileSystem.checkPermission(CosFileSystem.java:1146)
        at org.apache.hadoop.fs.CosFileSystem.mkdirs(CosFileSystem.java:575)
        at org.apache.hadoop.fs.FilterFileSystem.mkdirs(FilterFileSystem.java:332)
        at io.prestosql.plugin.hive.util.HiveWriteUtils.createDirectory(HiveWriteUtils.java:572)
        at io.prestosql.plugin.hive.util.HiveWriteUtils.createTemporaryPath(HiveWriteUtils.java:534)
        at io.prestosql.plugin.hive.HiveLocationService.forExistingTable(HiveLocationService.java:83)
        at io.prestosql.plugin.hive.HiveMetadata.beginInsert(HiveMetadata.java:1587)
        at io.prestosql.plugin.hive.HiveMetadata.beginInsert(HiveMetadata.java:277)
        at io.prestosql.spi.connector.ConnectorMetadata.beginInsert(ConnectorMetadata.java:443)
        at io.prestosql.plugin.base.classloader.ClassLoaderSafeConnectorMetadata.beginInsert(ClassLoaderSafeConnectorMetadata.java:429)
        at io.prestosql.metadata.MetadataManager.beginInsert(MetadataManager.java:840)
        at io.prestosql.sql.planner.optimizations.BeginTableWrite$Rewriter.createWriterTarget(BeginTableWrite.java:184)
        at io.prestosql.sql.planner.optimizations.BeginTableWrite$Rewriter.visitTableFinish(BeginTableWrite.java:143)
        at io.prestosql.sql.planner.optimizations.BeginTableWrite$Rewriter.visitTableFinish(BeginTableWrite.java:76)
        at io.prestosql.sql.planner.plan.TableFinishNode.accept(TableFinishNode.java:106)
        at io.prestosql.sql.planner.plan.SimplePlanRewriter$RewriteContext.rewrite(SimplePlanRewriter.java:84)
        at io.prestosql.sql.planner.plan.SimplePlanRewriter$RewriteContext.lambda$defaultRewrite$0(SimplePlanRewriter.java:73)
        at java.base/java.util.stream.ReferencePipeline$3$1.accept(ReferencePipeline.java:195)
        at java.base/java.util.Collections$2.tryAdvance(Collections.java:4747)
        at java.base/java.util.Collections$2.forEachRemaining(Collections.java:4755)
        at java.base/java.util.stream.AbstractPipeline.copyInto(AbstractPipeline.java:484)
        at java.base/java.util.stream.AbstractPipeline.wrapAndCopyInto(AbstractPipeline.java:474)
        at java.base/java.util.stream.ReduceOps$ReduceOp.evaluateSequential(ReduceOps.java:913)
        at java.base/java.util.stream.AbstractPipeline.evaluate(AbstractPipeline.java:234)
        at java.base/java.util.stream.ReferencePipeline.collect(ReferencePipeline.java:578)
        at io.prestosql.sql.planner.plan.SimplePlanRewriter$RewriteContext.defaultRewrite(SimplePlanRewriter.java:74)
        at io.prestosql.sql.planner.plan.SimplePlanRewriter.visitPlan(SimplePlanRewriter.java:38)
        at io.prestosql.sql.planner.plan.SimplePlanRewriter.visitPlan(SimplePlanRewriter.java:22)
        at io.prestosql.sql.planner.plan.PlanVisitor.visitOutput(PlanVisitor.java:49)
        at io.prestosql.sql.planner.plan.OutputNode.accept(OutputNode.java:83)
        at io.prestosql.sql.planner.plan.SimplePlanRewriter.rewriteWith(SimplePlanRewriter.java:32)
        at io.prestosql.sql.planner.optimizations.BeginTableWrite.optimize(BeginTableWrite.java:73)
        at io.prestosql.sql.planner.LogicalPlanner.plan(LogicalPlanner.java:206)
        at io.prestosql.sql.planner.LogicalPlanner.plan(LogicalPlanner.java:195)
        at io.prestosql.sql.planner.LogicalPlanner.plan(LogicalPlanner.java:190)
        at io.prestosql.execution.SqlQueryExecution.doPlanQuery(SqlQueryExecution.java:450)
        at io.prestosql.execution.SqlQueryExecution.planQuery(SqlQueryExecution.java:430)
        at io.prestosql.execution.SqlQueryExecution.start(SqlQueryExecution.java:382)
        at io.prestosql.execution.SqlQueryManager.createQuery(SqlQueryManager.java:237)
        at io.prestosql.dispatcher.LocalDispatchQuery.lambda$startExecution$7(LocalDispatchQuery.java:143)
        at io.prestosql.$gen.Presto_350____20210903_063152_2.run(Unknown Source)
        at java.base/java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1128)
        at java.base/java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:628)
        at java.base/java.lang.Thread.run(Thread.java:834)
```

This error occurs because the method originally exists but is missing in the loaded class, which may be caused by the following:

- The method is added to the package after version iteration but doesn't exist in earlier versions.
- The class with the same name in another package is loaded.

Run the following command in `/usr/local/service`:

```
find . -name "*.jar" -exec grep -Hls
"org/apache/hadoop/fs/cosn/ranger/client/RangerQcloudObjectStorageClient
Impl" {} \;
```

Find and delete the relevant package. For another class, simply modify the class path in the above command.

What should I do if the error `java.lang.ClassCastException org.apache.hadoop.fs.cosn.ranger.protocol.ClientQCloudObjectStorageProtocolProtos$GetSTSRequest cannot be cast to com.google.protobuf.Message` is reported?

Errors similar to this are generally caused by package contamination. The package on an earlier version exists on the server, and the Protobuf protocol is inconsistent, which is generally the same as the Alluxio package contamination below.
Run the following command in `/usr/local/service`:

```
find . -name "*.jar" -exec grep -Hls
"org/apache/hadoop/fs/cosn/ranger/protocol/ClientQCloudObjectStorageProt
ocolProtos" {} \;
```

Find and delete the relevant package. For another class, simply modify the class path in the above command.

What should I do if the modified Ranger policy doesn't take effect?

- For a CHDFS policy, decrease the value of `ranger.plugin.chdfs.policy.pollIntervalMs` in milliseconds in the COS Ranger Server configuration file `ranger-chdfs-security.xml`.
- For a COSN policy, decrease the value of `ranger.plugin.cos.policy.pollIntervalMs` in milliseconds in the COS Ranger Server configuration file `ranger-cos-security.xml`.

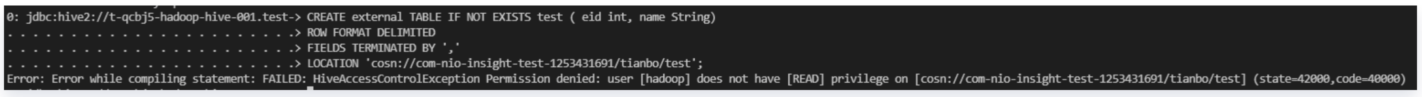
What should I do if the group configured for a Ranger policy doesn't take effect?

If the configured user takes effect, you need to contact the EMR team to check group sync. In other cases, contact us for assistance.

How do I configure a storage path policy rule in a Ranger policy?

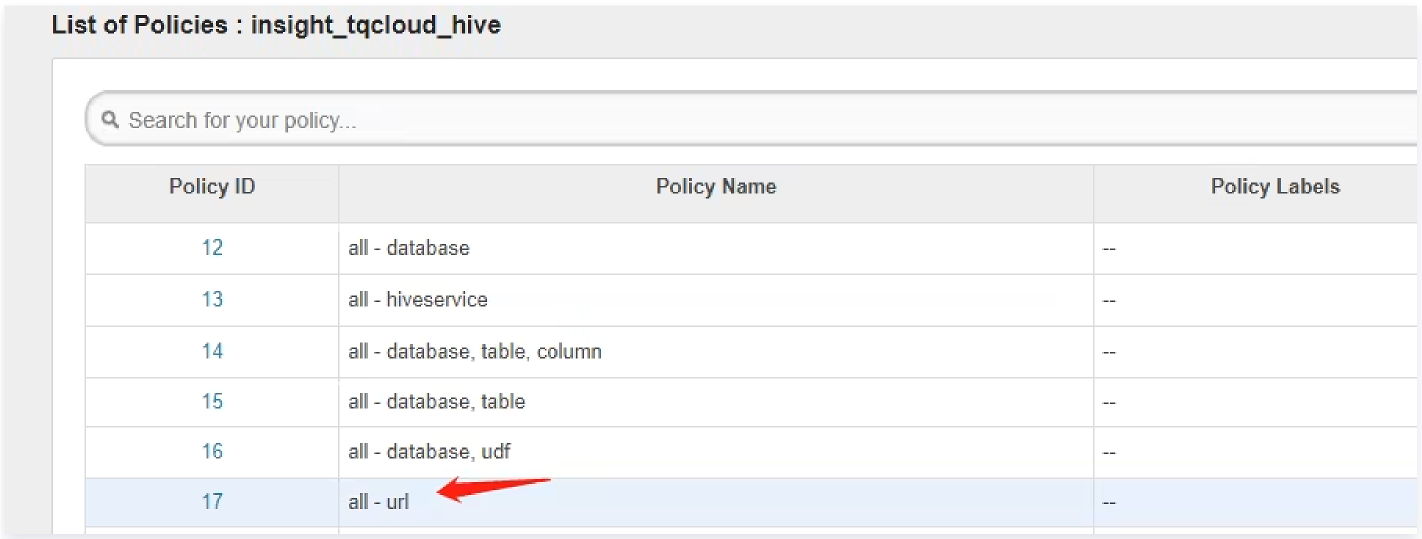
Ranger has a simple path verification rule, which is mainly based on string match. If the path rule of the policy is set to `**/a/** `for the file` /a/b/c`, the file cannot be accessed through **/a** or **/a/**, as the SDK will remove **/** at the end and the path will become **/a** in Ranger and cannot match **/a/**. If you access **/a/b** or **/a/b/c**, the prefix of the two paths can match the path rule **/a/** in the policy.

What should I do if the error `HiveAccessControlException` is reported when I specify the COSN or OFS path in Hive for table creation?



```
0: jdbc:hive2://t-qcbj5-hadoop-hive-001.test-> CREATE external TABLE IF NOT EXISTS test ( eid int, name String)
. . . . . . . . . . . . . . . . . . . . . . .> ROW FORMAT DELIMITED
. . . . . . . . . . . . . . . . . . . . . . .> FIELDS TERMINATED BY ','
. . . . . . . . . . . . . . . . . . . . . . .> LOCATION 'cosn://com-nio-insight-test-1253431691/tianbo/test';
Error: Error while compiling statement: FAILED: HiveAccessControlException Permission denied: user [hadoop] does not have [READ] privilege on [cosn://com-nio-insight-test-1253431691/tianbo/test] (state=42000,code=40000)
```

You need to disable URL verification in Hive by configuring the permission to allow URLs in Hive in the Ranger console:



List of Policies : insight_tqcloud_hive

Search for your policy...

Policy ID	Policy Name	Policy Labels
12	all - database	--
13	all - hiveservice	--
14	all - database, table, column	--
15	all - database, table	--
16	all - database, udf	--
17	all - url	--

> **Note**
> Note the error log format, which is highly likely to be reported by the Ranger service, and the error occurs because the permission configured by the Ranger admin is incorrect in most cases.

What should I do if the error `HiveAccessControlException` is reported when I submit a Spark job in an environment that uses Kerberos for authentication?

If an application needs to interact with other secure Hadoop file systems, the URI must be explicitly provided to Spark at startup. Configure the parameter spark.kerberos.access.hadoopFileSystems=cosn://bucket-appid,ofs://f4mxxxxxxxx-Xxxx.chdfs.ap-guangzhou.myqcloud.com. For more information, refer to Spark Official Documentation.

What should I do if a table isn't moved to the recycle bin after being dropped in Spark?

If the location is specified when you run CREATE TABLE in Spark, an external table will be created, and its data won't be deleted after it is dropped. For more information, see Upgrading from Spark SQL 1.6 to 2.0.

> **Note**
> The data can be deleted if you run DROP TABLE in Hive on MR.

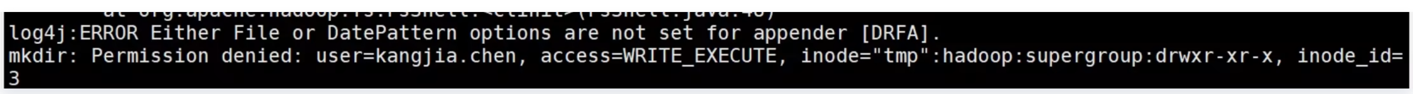
What should I do if the error `AccessControlException` is reported when Hive executes INSERT statements?

The default engine of Hive is MapReduce. Add the following configuration item to the `yarn-site.xml` file:

```
<property>
    <name>mapreduce.job.hdfs-servers</name>
    <value>
        ofs://f4mxxxxxxx-XXXX,cosn://bucketname-appid,${fs.defaultFS}
    </value>
</property>
```

If the Hive engine is Tez, add the `tez.job.fs-servers` configuration item to the `tez-site.xml` file and set it to the above value.
If you use Beeline to connect to Hive, restart HiveServer2 and load the new `yarn-site` configuration.

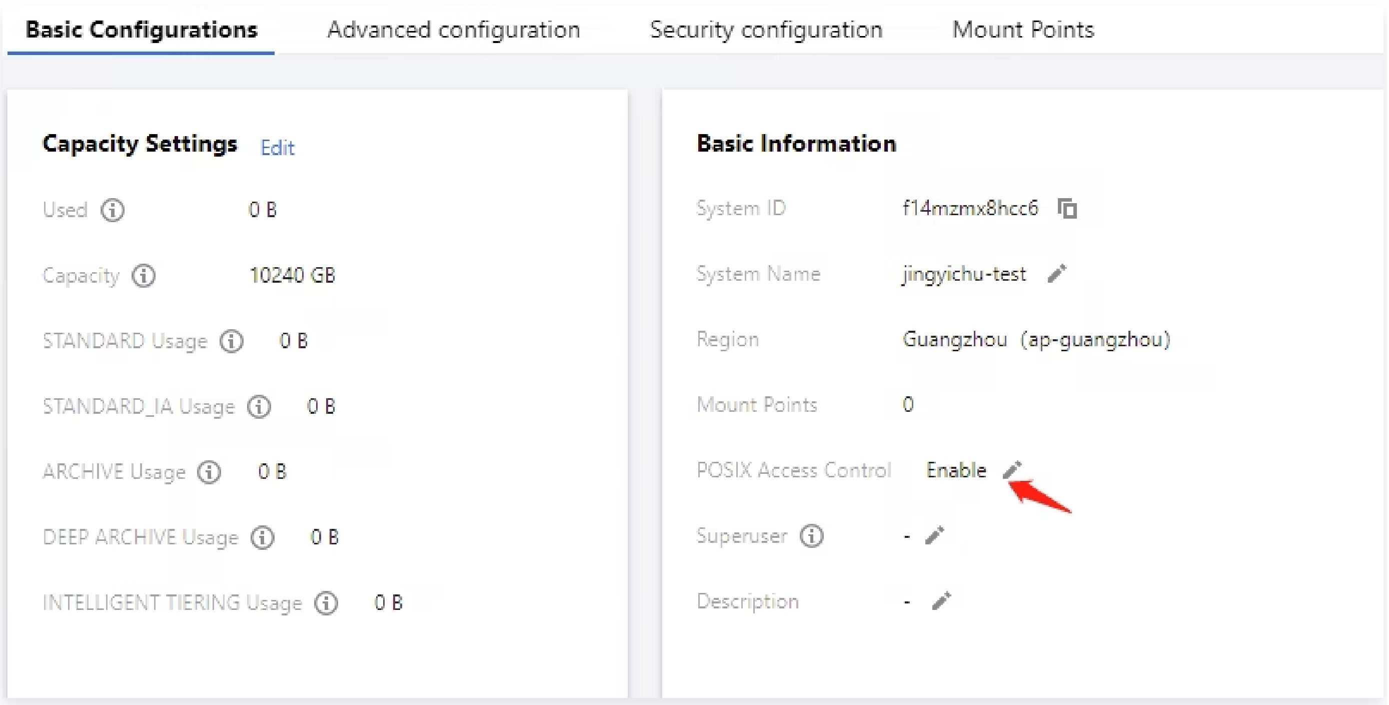
What should I do if an error occurs when I access OFS?



```
log4j:ERROR Either File or DatePattern options are not set for appender [DRFA].
mkdir: Permission denied: user=kangjia.chen, access=WRITE_EXECUTE, inode="tmp":hadoop:supergroup:drwxr-xr-x, inode_id=
3
```

The error is returned by the OFS backend. After Ranger is enabled, you need to disable POSIX in the following places:

- CHDFS Console



Capacity Settings

Used	0 B
Capacity	10240 GB
STANDARD Usage	0 B
STANDARD_IA Usage	0 B
ARCHIVE Usage	0 B
DEEP ARCHIVE Usage	0 B
INTELLIGENT TIERING Usage	0 B

Basic Information

System ID	f14mzmx8hcc6
System Name	jingyichu-test
Region	Guangzhou (ap-guangzhou)
Mount Points	0
POSIX Access Control	Enable
Superuser	-
Description	-

- COS Bucket Configuration

What should I do if the error `renew token failed` is reported when I submit a job through the YARN command?

The `-Dmapreduce.job.send-token-conf` parameter is required to execute the YARN command.

How do I build COS Ranger?

For more information, see COS Ranger Permission System Solution and CHDFS Ranger Permission System Solution.

How do I enable Ranger in EMR?

You can purchase the Ranger and COS Ranger components in the EMR console to eliminate the need of deployment on your own.

- For CHDFS, add the new configuration item `fs.ofs.ranger.enable.flag` in `core-site.xml` and set it to `true`.
- For COSN, add the new configuration item `fs.cosn.credentials.provider` in `core-site.xml` and set it to `org.apache.hadoop.fs.auth.RangerCredentialsProvider`.

What should I do if a NodeCache null pointer exception occurs?

Please confirm the hadoop-ranger-client version. If it is v3.8, we recommend upgrading to v5.0. For other situations, please contact us.
The reason for this error is that the concurrency level of big data jobs is high, causing high pressure on Zookeeper and resulting in lost Zookeeper watches.

What should I do if the error `java.lang.IllegalArgumentException: Failed to specify server's Kerberos principal name` is reported when I run a Hadoop FS command after COS Ranger is enabled?

- Add the following configuration item to `core-site.xml`: `qcloud.object.storage.kerberos.principal`.
- If the error is reported by the HDFS cluster, add the following configuration item to `core-site.xml`: `dfs.namenode.kerberos.principal`.

Server-Side Encryption Feature Introduction

Last updated：2025-11-13 15:38:40

Overview

A metadata acceleration bucket is a special type of bucket. Its support status, configuration, and method of use for server-side encryption differ from ordinary buckets. For a basic introduction to server-side encryption, please refer to Server-side Encryption Overview.
This document provides detailed information on the variations in server-side encryption between metadata acceleration buckets and ordinary buckets, as well as the method of use for server-side encryption in metadata acceleration buckets.

Server-Side Encryption Support Status

The support status for server-side encryption in buckets and metadata acceleration buckets is as shown in the table below:

Access Method	Server-Side Encryption Type	bucket	metadata acceleration bucket
S3 API	SSE-C	Supported	Not supported
	SSE-COS	Supported	Supported
	SSE-KMS	Supported	Not supported
HDFS API	SSE-C	Supported	Supported
	SSE-COS	Supported	Supported
	SSE-KMS	Supported	Supported

Note:
In addition to the differences listed in the above table, metadata acceleration buckets also have two additional usage limits:
- Bucket encryption is not supported (see bucket encryption overview), including console configuration and REST API configuration.

- Only supports the AES256 encryption algorithm. SM4 encryption algorithm is not supported.

Server-Side Encryption Method of Use

The server-side encryption configuration method for metadata acceleration buckets depends on the API type.

- S3 API access: The method to configure the encryption method is the same as for a bucket. See server-side encryption dedicated header to set request header parameters.
- HDFS API access: Select the configuration method based on the different clients used (OFS client or COSN client). For details about the two clients, see Client Introduction.

The configuration items listed in the following context need to be added or modified in the Hadoop core-site.xml file. The operation path varies depending on the environment.

- Tencent Cloud's EMR product: Operate via the EMR console. For the method to edit the core-site.xml configuration file, see configuration update.
- Self-built big data environment: The core-site.xml file is normally located under the $HADOOP_HOME/etc/hadoop directory, depending on the installation method and release version of the big data components.

OFS Client

When using the OFS client, the configuration items for various encryption methods are as follows:

SSE-C Encryption

Configuration Item	Configuration Description	Required
fs.ofs.sse.mode	Specify server encryption method, here is SSE-C	Yes
fs.ofs.sse.c.key	A Base64-encoded AES-256 key, for example MDEyMzQ1Njc4OUFCQ0RFRjAxMjM0NTY3ODlBQkNERUY=	Yes

SSE-COS Encryption

Configuration Item	Configuration Description	Required
fs.ofs.sse.mode	Specify server encryption method, here is SSE-COS	Yes

SSE-KMS Encryption

Configuration Item	Configuration Description	Required
fs.ofs.sse.mode	Specify server encryption method, here is SSE-KMS	Yes
fs.ofs.sse.kms.keyid	Specify the User Master Key (CMK) of KMS. If not specified, use the CMK created by COS by default. example value: 93866e69-9755-11ef-8e65-52540089bc41	No
fs.ofs.sse.kms.context	Specify the KMS encryption context, with the value being the Base64-encoded JSON-formatted encryption context key-value pair. example value: eyJrZXkxIjoidmFsdWUxIn0=	No

COSN Client

When using the COSN client, the configuration items for various encryption methods are as follows:

SSE-C Encryption

Configuration Item	Configuration Description	Required
fs.cosn.trsf.fs.ofs.sse.mode	Specify server encryption method, here is SSE-C	Yes
fs.cosn.trsf.fs.ofs.sse.c.key	A Base64-encoded AES-256 key, for example MDEyMzQ1Njc4OUFCQ0RFRjAxMjM0NTY3ODlBQkNERUY=	Yes

SSE-COS Encryption

Configuration Item	Configuration Description	Required
fs.cosn.trsf.fs.ofs.sse.mode	Specify server encryption method, here is SSE-COS	Yes

SSE-KMS Encryption

Configuration Item	Configuration Description	Required
fs.cosn.trsf.fs.ofs.sse.mode	Specify server encryption method, here is SSE-KMS	Yes
fs.cosn.trsf.fs.ofs.sse.kms.keyid	Specify the User Master Key (CMK) of KMS. If not specified, use the CMK created by COS by default. example value: 93866e69-9755-11ef-8e65-52540089bc41	No
fs.cosn.trsf.fs.ofs.sse.kms.context	Specify the KMS encryption context, with the value being the Base64-encoded JSON-formatted encryption context key-value pair. example value: eyJrZXkxIjoidmFsdWUxIn0=	No

Data Management
Metadata Acceleration Bucket Lifecycle

Last updated：2025-12-08 11:35:47

COS metadata-accelerated buckets are compatible with COS lifecycle capabilities and are suitable for scenarios such as data warehousing and cold and hot data tiering.The metadata acceleration bucket supports using the COS console and API to configure lifecycle rules. For specific usage, see: COS Lifecycle Overview.

Differences From COS Lifecycle

Metadata acceleration buckets have the following differences from COS bucket lifecycle. Please note when using:

1. Specification limits: The metadata acceleration bucket has a limit of 1,000 lifecycle rules per single bucket. If a unique scenario requires increasing this limit, please contact us.
2. Update time: The metadata acceleration bucket lifecycle uses the **latest file update time** as the basis for tiering or deletion. The **latest time** among the file's MTime, ATime, and CTime in the COS metadata acceleration backend is used as the file update time. Specifying a particular time (MTime, ATime, CTime) as the tiering basis is currently not supported. For example, if you create a file `text.txt` on June 1 and access it again on June 10, with no subsequent operations on this file, then configure a lifecycle rule on June 11 to tier `text.txt` to STANDARD_IA 10 days after the update. At this point, the lifecycle scan determines the file's MTime as June 1 and ATime as June 10, making the latest update time June 10. The file will be tiered to STANDARD_IA storage on June 20.

Note:
- To ensure the best read/write performance, COS metadata acceleration does not enable ATime tracing by default. If you need to use the ATime feature, contact us.
- To avoid lifecycle management being executed earlier than the specified time, please refer to Recycle Bin Clearing Mechanism before using lifecycle settlement or removing files (such as .Trash).

3. File prefix: The metadata acceleration bucket lifecycle supports prefix filtering, which only allows filling in as a **directory** and does not currently support the prefix exclusion feature. When configuring a prefix, there is no need to include `/` before and after the path, as the COS backend will default to treating this path as a directory. Additionally, since prefixes

only support directories, the path wildcard functionality in ordinary COS buckets is not currently supported. For example, your bucket has the following two paths:

- `user/hive/warehouse/test.db/test_table/`
- `user/hive/warehouse/test.db/test_table2/`

If you configure the file prefix in the lifecycle rule as `user/hive/warehouse/test.db/test_table` and match this path as a directory, only the first path will be hit, and it will not take effect on `test_table2`.

4. Currently, metadata acceleration buckets support one-way file settlement in the following sequence:
   - Single-AZ metadata acceleration bucket: STANDARD > STANDARD_IA > ARCHIVE > DEEP_ARCHIVE
   - Multi-AZ metadata acceleration bucket: MAZ_STANDARD > MAZ_STANDARD_IA

Usage

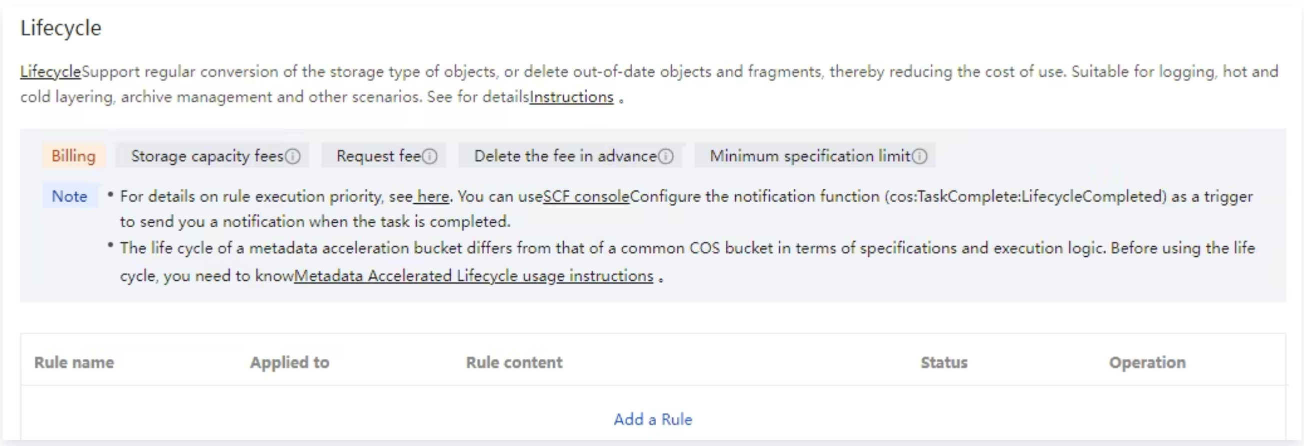
Metadata acceleration buckets currently support the use of COS console and S3 Lifecycle API to configure lifecycle rules.

Configuring in the Console

The process of configuring via console is as follows:

Creating Rules

1. Enter the COS console.
2. In the left sidebar, click **Bucket List** to enter.
3. Find the metadata acceleration bucket that requires enabling the lifecycle feature, click its bucket name, and enter the Bucket Details Page.
4. Click on the left **Basic Configuration > Lifecycle** configuration item.


Lifecycle

LifecycleSupport regular conversion of the storage type of objects, or delete out-of-date objects and fragments, thereby reducing the cost of use. Suitable for logging, hot and cold layering, archive management and other scenarios. See for detailsInstructions .

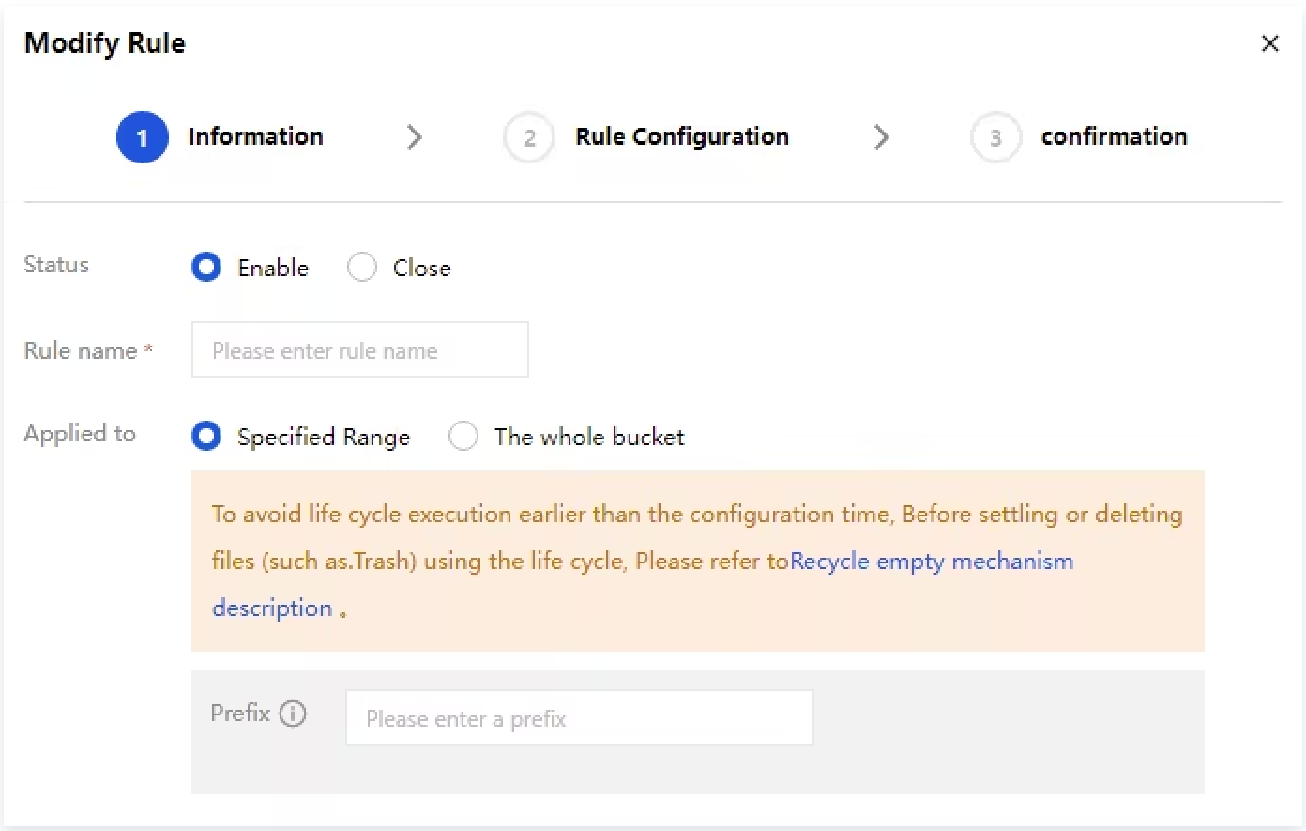
Billing Storage capacity fees Request fee Delete the fee in advance Minimum specification limit

Note
- For details on rule execution priority, see here. You can useSCF consoleConfigure the notification function (cos:TaskComplete:LifecycleCompleted) as a trigger to send you a notification when the task is completed.
- The life cycle of a metadata acceleration bucket differs from that of a common COS bucket in terms of specifications and execution logic. Before using the life cycle, you need to knowMetadata Accelerated Lifecycle usage instructions .

Rule name	Applied to	Rule content	Status	Operation

Add a Rule

5. Click **Add Rule.**


Modify Rule

1 Information > 2 Rule Configuration > 3 confirmation

Status [✓ Enable] Close

Rule name * Please enter rule name

Applied to [✓ Specified Range] The whole bucket

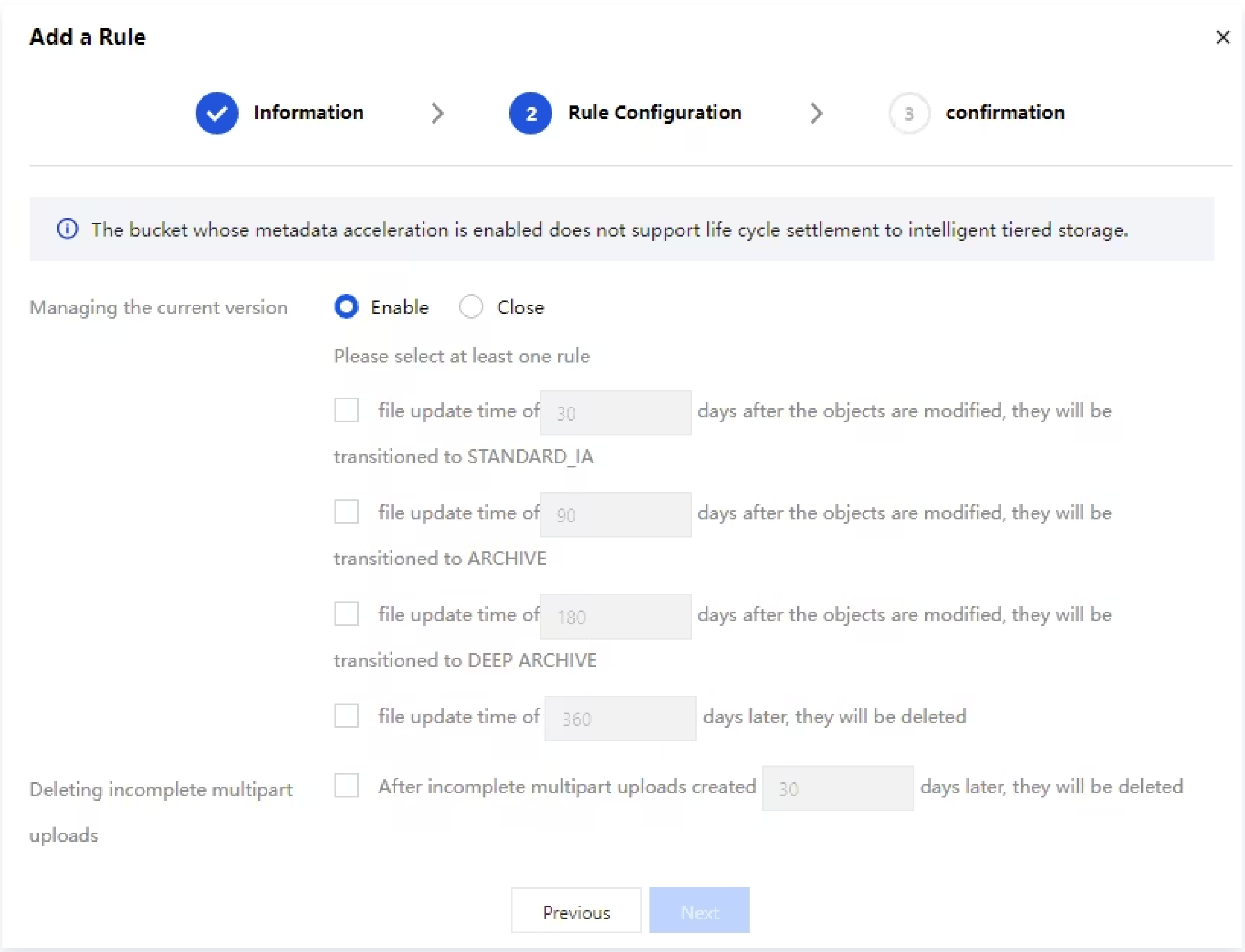
To avoid life cycle execution earlier than the configuration time, Before settling or deleting files (such as.Trash) using the life cycle, Please refer toRecycle empty mechanism description 。

Prefix Please enter a prefix

The basic information configuration items are described below.

- **Rule Name** : Enter a name for your lifecycle rule.
- **Scope:** This lifecycle rule can apply to the entire bucket or files within a specified prefix range.
  - When selecting a specified range:
    - **Object prefix:** Metadata acceleration buckets only support **directory** as the prefix and do not currently support the prefix exclusion feature. When configuring the prefix, there is no need to carry `/` before and after the path. The COS Backend will default to processing this path as a directory. Regular expressions are not supported.
  - When selecting the entire bucket: The rule takes effect on all files in the entire bucket.

6. Click **Next** and enter **Rule Configuration**.


Add a Rule

Information > 2 Rule Configuration > 3 confirmation

The bucket whose metadata acceleration is enabled does not support life cycle settlement to intelligent tiered storage.

Managing the current version [✓ Enable] Close

Please select at least one rule

☐ file update time of 30 days after the objects are modified, they will be transitioned to STANDARD_IA

☐ file update time of 90 days after the objects are modified, they will be transitioned to ARCHIVE

☐ file update time of 180 days after the objects are modified, they will be transitioned to DEEP ARCHIVE

☐ file update time of 360 days later, they will be deleted

Deleting incomplete multipart uploads ☐ After incomplete multipart uploads created 30 days later, they will be deleted

The configuration items are described as follows:

- **Manage current version files**: You can enable the option to transition or delete current version objects. Objects in the bucket can be transitioned from STANDARD (hot data) to STANDARD_IA (cold data) storage classes, and support deletion after expiration.
  - Storage types from hot to cold are: **STANDARD** > **STANDARD_IA** > **ARCHIVE** > **DEEP_ARCHIVE**. Storage type conversion can only go from hot to cold, not in reverse. For details on storage types and applicable regions, see Storage Class Overview .

    Note:
    For buckets with multi-AZ configuration enabled, the lifecycle transition order only supports **MAZ_STANDARD** > **MAZ_STANDARD_IA**.

  - Metadata acceleration buckets determine settlement or deletion **based on the most recent file update time.** The backend metadata acceleration in COS uses the **latest time** among MTime, ATime, and CTime as the file update time. Currently, a specified time (MTime, ATime, or CTime) cannot be used as the basis for settlement.

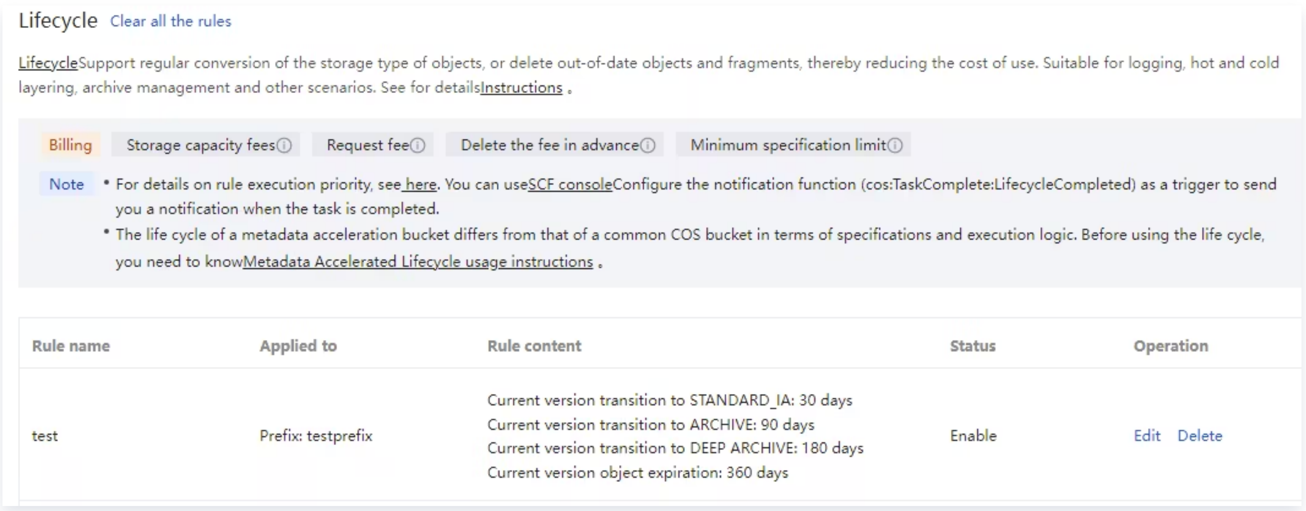
> **Note:**
> For example, if you create a file `text.txt` on June 1 and access it again on June 10 with no subsequent operations, then configure a lifecycle rule on June 11 specifying that `text.txt` will be tiered to STANDARD_IA 10 days after the update. At this point, the lifecycle scan determines the file's MTime as June 1 and ATime as June 10, making the latest update time June 10. The file will be tiered to STANDARD_IA on June 20.

- **Delete Incomplete Multipart Uploads**: When uploading a file, the upload fails due to various reasons, and only a part of it is transmitted. For such damaged files, you can set periodic deletion.

7. Click **Next**, confirm that everything is correct, then click **Confirm** to complete the lifecycle rule creation.

Viewing Rules

1. Enter the COS console.
2. In the left sidebar, click **Bucket List** to enter.
3. Find the metadata acceleration bucket that requires enabling the lifecycle feature, click its bucket name, and enter the Bucket Details Page.
4. Click on the left **Basic Configuration > Lifecycle** configuration item, and you can see ALL lifecycle rules for this bucket.


Lifecycle Clear all the rules

LifecycleSupport regular conversion of the storage type of objects, or delete out-of-date objects and fragments, thereby reducing the cost of use. Suitable for logging, hot and cold layering, archive management and other scenarios. See for detailsInstructions .

Billing Storage capacity fees Request fee Delete the fee in advance Minimum specification limit

Note
- For details on rule execution priority, see here. You can useSCF consoleConfigure the notification function (cos:TaskComplete:LifecycleCompleted) as a trigger to send you a notification when the task is completed.
- The life cycle of a metadata acceleration bucket differs from that of a common COS bucket in terms of specifications and execution logic. Before using the life cycle, you need to knowMetadata Accelerated Lifecycle usage instructions .

Rule name	Applied to	Rule content	Status	Operation
test	Prefix: testprefix	Current version transition to STANDARD_IA: 30 days Current version transition to ARCHIVE: 90 days Current version transition to DEEP ARCHIVE: 180 days Current version object expiration: 360 days	Enable	Edit Delete

Editing Rules

1. Enter the COS console.
2. In the left sidebar, click **Bucket List** to enter.
3. Find the metadata acceleration bucket that requires enabling the lifecycle feature, click its bucket name, and enter the Bucket Details Page.

4. Click on the left **Basic Configuration > Lifecycle** configuration item.
5. Locate the expected rule to edit, click the **Edit** button, then edit the lifecycle rule via the popup.

Deleting Rules

1. Enter the COS console.
2. In the left sidebar, click **Bucket List** to enter.
3. Find the metadata acceleration bucket that requires enabling the lifecycle feature, click its bucket name, and enter the Bucket Details Page.
4. Click on the left **Basic Configuration > Lifecycle** configuration item.
5. Locate the rule you expect to delete, click the **Delete** button to delete the lifecycle rule.

Note:
If you only need a short stop of rule execution with no need to delete the rule, click **Edit** and change the corresponding rule status to **Disabled**.

Configuring Via API

Metadata-accelerated buckets reuse COS lifecycle capabilities and support using the COS Lifecycle interface to enable lifecycle configuration. Due to the feature of metadata acceleration buckets, some fields in the API differ from COS APIs. For specific calling methods, see the following API examples:

PUT Bucket Lifecycle

This interface is used to create a lifecycle rule in a bucket. For interface parameter description, refer to COS PUT Bucket Lifecycle API Documentation.
Due to the characteristics of metadata-accelerated buckets, some fields in the API differ from those in the COS lifecycle interface. The specific details are as follows:

- `Filter` node
  - Prefix exclusion feature is not supported, so the `PrefixNotEquals` field is unsupported.
  - Metadata acceleration bucket does not support file tags, so the `tag` field is unsupported for filtering.
  - File size filtering is not supported, so the `ObjectSizeGreaterThan` and `ObjectSizeLessThan` fields are unsupported.
- `AccessFrequency` node

  - Metadata acceleration bucket currently does not support specifying access time settlement, so the `AccessFrequency` and its child nodes parameters are unsupported.
- `NoncurrentVersion` related node
  - Metadata acceleration bucket does not support version control, so the parameters `NoncurrentVersionExpiration` , `NoncurrentVersionTransition` and their child nodes are unsupported.
- `StorageClass` node
  - The current metadata acceleration bucket supports the following input parameters: `STANDARD_IA` , `ARCHIVE` , `DEEP_ARCHIVE` .

Request

```
PUT /?lifecycle HTTP/1.1
Host: <BucketName-APPID>.cos.<Region>.myqcloud.com
Content-Length: length
Date: GMT Date
Authorization: Auth String
Content-MD5: MD5

<?xml version="1.0" encoding="UTF-8"?>
<LifecycleConfiguration>
    <Rule>
        <ID>test_rule_1</ID>
        <Filter>
            <Prefix>test_prefix</Prefix>
        </Filter>
        <Status>Enabled</Status>
        <Transition>
            <Days>30</Days>
            <StorageClass>STANDARD_IA</StorageClass>
        </Transition>
        <Transition>
            <Days>90</Days>
            <StorageClass>ARCHIVE</StorageClass>
        </Transition>
        <Transition>
            <Days>180</Days>
            <StorageClass>DEEP_ARCHIVE</StorageClass>
        </Transition>
        <Expiration>
```

```
            <Days>360</Days>
        </Expiration>
    </Rule>
    <Rule>
        <ID>test_rule_2</ID>
        <Filter/>
        <Status>Enabled</Status>
        <AbortIncompleteMultipartUpload>
            <DaysAfterInitiation>30</DaysAfterInitiation>
        </AbortIncompleteMultipartUpload>
    </Rule>
</LifecycleConfiguration>
```

Response

```
HTTP/1.1 200 OK
Content-Type: application/xml
Content-Length: 0
Date: Wed, 16 Aug 2020 11:59:33 GMT
Server: tencent-cos
```

GET Bucket Lifecycle

This interface is used to obtain the lifecycle rule in a bucket. For interface parameter description, refer to COS GET Bucket Lifecycle API Documentation.
Request

```
GET /?lifecycle HTTP/1.1
Host: <BucketName-APPID>.cos.<Region>.myqcloud.com
Date: GMT Date
Authorization: Auth String
```

Response

```
HTTP/1.1 200 OK
Content-Type: application/xml
Content-Length: 312
Date: Wed, 16 Aug 2017 12:23:54 GMT
Server: tencent-cos
```

```
x-cos-request-id: NTk5NDM5NWFfMjQ4OGY3Xzc3NGRf****

 <LifecycleConfiguration>
  <Rule>
   <ID>test_rule_1</ID>
   <Filter>
    <Prefix>test_prefix</Prefix>
   </Filter>
   <Status>Enabled</Status>
   <Transition>
    <Days>30</Days>
    <StorageClass>STANDARD_IA</StorageClass>
   </Transition>
   <Transition>
    <Days>90</Days>
    <StorageClass>ARCHIVE</StorageClass>
   </Transition>
   <Transition>
    <Days>180</Days>
    <StorageClass>DEEP_ARCHIVE</StorageClass>
   </Transition>
   <Expiration>
    <Days>360</Days>
   </Expiration>
  </Rule>
  <Rule>
   <ID>test_rule_2</ID>
   <Filter></Filter>
   <Status>Enabled</Status>
   <AbortIncompleteMultipartUpload>
    <DaysAfterInitiation>30</DaysAfterInitiation>
   </AbortIncompleteMultipartUpload>
  </Rule>
 </LifecycleConfiguration>
```

DELETE Bucket Lifecycle

This interface is used to delete the lifecycle rule in a bucket. For interface parameter description, refer to COS DELETE Bucket Lifecycle API Documentation.

Request

```
DELETE /?lifecycle HTTP/1.1
Host: <BucketName-APPID>.cos.<Region>.myqcloud.com
Date: GMT Date
Authorization: Auth String
```

Response

```
HTTP /1.1 204 No Content
Content-Type: application/xml
Date: Wed, 16 Aug 2017 12:59:09 GMT
Server: tencent-cos
x-cos-request-id: NTk5NDQxOWNfMjQ4OGY3Xzc3NGRf****
```

Metadata Acceleration Bucket Inventory Feature

Last updated: 2025-11-13 15:48:18

Metadata Acceleration Bucket reuses the COS Storage Bucket Inventory feature, providing file metadata export functionality similar to the Hadoop OIV tool. The metadata acceleration bucket can perform scheduled scans of files in the specified path within the user's bucket daily or weekly according to the user's inventory task configuration, and produce an inventory report stored in CSV format in the user-specified path. It is suitable for file analysis, audit, governance, and other scenarios.

Note:

- The Metadata Acceleration Bucket Inventory Feature allowlist is currently in public beta. This document is for reference during the beta test. To apply for enabling the inventory feature, please contact us.
- During the public beta period, the inventory feature is temporarily free of COS management feature fees. Subsequent pricing information will be notified via internal message and official documentation.

Use Limits

Metadata acceleration bucket inventory has the following limitations during the public beta period:

- Each COS metadata acceleration bucket supports a maximum of 5 configured inventory rules.

Note:

Typically, it is recommended to configure the bucket root path as the manifest scan path. If your usage scenario requires configuring more than 5 rules, please contact us.

- Inventory reports do not support cross-bucket delivery. The target path in task configuration must match the source bucket.
- Inventory does not support custom IDs. A task ID is automatically assigned upon task creation. This ID can be queried via the List API.
- Metadata acceleration buckets do not currently support instant inventory.

Inventory Report Delivery Instructions

After configuring an inventory task, COS will perform scheduled scans of files in the specified path of your bucket based on the configuration and output an inventory report in CSV format. The delivery path for the report can be specified in the task configuration. The specific path for the CSV file is as follows:

```
destination-prefix/YYYY-MM-DD-
HH/Data/8d1048acf48096d17502bd62efc9f8c1.csv
```

- destination-prefix: The shipping route specified by the user in the list delivery task.
- YYYY-MM-DD-HH: Timestamp, the year, month, day, and hour when the inventory task is executed.

Note:
The hour in the path indicates the task start execution time, which may differ from the actual delivery time. COS metadata acceleration buckets can only be configured for daily tasks. The initial configuration or delivery within 24 hours will generate one copy of the inventory report. It is unable to specify a fixed time to execute inventory tasks or deliver inventory reports.

- Inventory report: Delivered in CSV format, with filenames as random values generated by COS.

Manifest tasks scan files under the configured path according to rule configuration. The generated report excludes header info and prints field order as follows:

Field	Description
FileName	File name.
ATime	File access time **Note:** To ensure the best read/write performance, the metadata acceleration bucket does not enable ATime tracing by default. If you need the ATime feature, contact us .
MTime	Change time of file content
CTime	Change time of file metadata

Size	File size (Byte)
FileType	Print file status as directory or file, output format: DIR (directory) or FILE (file)
ACL	Print file permissions in digits, for example: 755
UserName	Username, for example: hadoop
GroupName	User group name, for example: supergroup
StorageClass	The current file storage type. Possible values: • STANDARD (standard storage) • DEGRADE (infrequent storage) • ARCHIVE (archive storage) • DEEP_ARCHIVE (deep archive storage) • ARCHIVE+STANDARD (archive storage restored to standard) • DEEP_ARCHIVE+STANDARD (deep archive storage restored to standard)

Manifest Configuration Method

Currently, metadata acceleration buckets in COS cannot be accessed through the console for inventory configuration and only supports using APIs to configure. Specific steps are as follows:

1. Create a COS role.
2. Bind permissions to the COS role.
3. Enable the inventory feature.

1. Creating a COS role

Create a COS role. For more information about the API, see CreateRole.
Here, roleName should be COS_QcsRole.
policyDocument should be:

```
{
    "version": "2.0",
    "statement": [{
        "action": "name/sts:AssumeRole",
        "effect": "allow",
        "principal": {
            "service": "cos.cloud.tencent.com"
        }
    }]
}
```

2. Binding Permissions to the COS Role

Bind permissions to the role. For more information about the API, see AttachRolePolicy. Here, policyName is QcloudCOSFullAccess, and roleName is the COS_QcsRole in step 1 or the roleID returned when roleName is created.

3. Enabling the Inventory Feature

Call the API to enable the inventory feature. Metadata acceleration buckets reuse the COS inventory capacity and support using the COS Inventory API to enable inventory configuration. Due to the metadata acceleration bucket feature, some fields in the API differ from the COS inventory API. For the specific calling method, please refer to the following API example:

PUT Bucket Inventory

This interface is used to create inventory tasks in the bucket. For the interface parameter description, refer to COS PUT Inventory API Documentation. Due to the metadata acceleration bucket feature, some fields in the API differ from the COS inventory interface. Details are as follows:

- Metadata acceleration bucket inventory does not support specifying an ID. When creating a task, ID information is not needed, as the ID is automatically generated by COS.
- Metadata acceleration buckets currently only support delivering inventory reports to the source bucket. `COSBucketDestination` needs to be configured as the source bucket.
- Inventory report encryption cannot be configured temporarily. The `Encryption` field cannot be configured when creating a task.
- Metadata acceleration buckets do not support version control capability, so the `IncludedObjectVersions` field is unsupported during configuration.
- Metadata Bucket cannot be specified to generate inventory report fields. For the `OptionalFields` field, refer to the example below for input parameters.

Request

```
PUT /?inventory HTTP/1.1
Host: <BucketName-APPID>.cos.<Region>.myqcloud.com
Date: GMT Date
Authorization: Auth String
```

```
Content-MD5: MD5

<InventoryConfiguration>
   <IsEnabled>true</IsEnabled>
   <Destination>
    <COSBucketDestination>
     <Format>CSV</Format>
     <AccountId>1000000000</AccountId>
     <Bucket>qcs::cos:ap-beijing::bucketname-100000000</Bucket>
     <Prefix>/testprefix</Prefix>
    </COSBucketDestination>
   </Destination>
   <Schedule>
    <Frequency>Daily</Frequency>
   </Schedule>
   <Filter>
    <Prefix>/my-inventory</Prefix>
   </Filter>
   <OptionalFields>
    <Field>ATime</Field>
    <Field>ACL</Field>
    <Field>MTime</Field>
    <Field>CTime</Field>
    <Field>Size</Field>
    <Field>UserName</Field>
    <Field>GroupName</Field>
    <Field>StorageClass</Field>
    <Field>FileType</Field>
   </OptionalFields>
</InventoryConfiguration>
```

Response

```
HTTP/1.1 200 OK
Content-Type: application/xml
Content-Length: 0
Date: Mon, 28 Aug 2018 02:53:38 GMT
Server: tencent-cos
x-cos-request-id: NTlhMzg1ZWVfMjQ4OGY3MGFfMWE1NF8****
```

GET Bucket Inventory

This interface is used to query specific inventory task information in the bucket. For the interface parameter description, refer to COS GET Inventory API Documentation. This interface requires the inventory ID parameter, which is automatically generated by the metadata acceleration bucket when creating a task. If needed, call the List Bucket Inventory Configurations API to obtain this ID.

Request

```
GET /?inventory&id=inventory-configuration-ID HTTP/1.1
Host: <BucketName-APPID>.cos.<Region>.myqcloud.com
Date: GMT Date
Authorization: Auth String
```

Response

```
HTTP/1.1 200 OK
Content-Type: application/xml
Content-Length: 331
Date: Mon, 28 Aug 2018 02:53:39 GMT
Server: tencent-cos
x-cos-request-id: NTlhMzg1ZWVfMjQ4OGY3MGFfMWE1NF84Y2M
<?xml version = "1.0" encoding = "UTF-8">
<InventoryConfiguration xmlns = "http://....">
   <Id>12761</Id>
   <IsEnabled>true</IsEnabled>
   <Destination>
    <COSBucketDestination>
     <Format>CSV</Format>
     <AccountId>1000000000</AccountId>
     <Bucket>qcs::cos:ap-beijing::bucketname-100000000</Bucket>
     <Prefix>/testprefix</Prefix>
    </COSBucketDestination>
   </Destination>
   <Schedule>
    <Frequency>Daily</Frequency>
   </Schedule>
   <Filter>
```

```
    <Prefix>/my-inventory</Prefix>
   </Filter>
   <OptionalFields>
    <Field>ATime</Field>
    <Field>ACL</Field>
    <Field>MTime</Field>
    <Field>CTime</Field>
    <Field>Size</Field>
    <Field>UserName</Field>
    <Field>GroupName</Field>
    <Field>StorageClass</Field>
    <Field>FileType</Field>
   </OptionalFields>
  </InventoryConfiguration>
```

List Bucket Inventory Configurations

This interface is used to request return all inventory tasks in a bucket. For the interface parameter description, refer to COS List Bucket Inventory Configurations API Documentation.

Request

```
GET /?inventory HTTP/1.1
Host: <BucketName-APPID>.cos.<Region>.myqcloud.com
Date: GMT Date
Authorization: Auth String
```

Response

```
HTTP/1.1 200 OK
Content-Type: application/xml
Content-Length: 331
Date: Mon, 28 Aug 2018 02:53:39 GMT
Server: tencent-cos
x-cos-request-id: NTlhMzg1ZWVfMjQ4OGY3MGFfMWE1NF8****
<?xml version = "1.0" encoding = "UTF-8">
<ListInventoryConfigurationResult xmlns = "http://....">
<InventoryConfiguration>
```

```
     <Id>12761</Id>
     <IsEnabled>true</IsEnabled>
     <Destination>
      <COSBucketDestination>
       <Format>CSV</Format>
       <AccountId>1000000000</AccountId>
       <Bucket>qcs::cos:ap-beijing::bucketname-100000000</Bucket>
       <Prefix>/testprefix</Prefix>
      </COSBucketDestination>
     </Destination>
     <Schedule>
      <Frequency>Daily</Frequency>
     </Schedule>
     <Filter>
      <Prefix>/my-inventory</Prefix>
     </Filter>
     <OptionalFields>
      <Field>ATime</Field>
      <Field>ACL</Field>
      <Field>MTime</Field>
      <Field>CTime</Field>
      <Field>Size</Field>
      <Field>UserName</Field>
      <Field>GroupName</Field>
      <Field>StorageClass</Field>
      <Field>FileType</Field>
     </OptionalFields>
    </InventoryConfiguration>
  <InventoryConfiguration>
     <Id>12762</Id>
     <IsEnabled>true</IsEnabled>
     <Destination>
      <COSBucketDestination>
       <Format>CSV</Format>
       <AccountId>1000000000</AccountId>
       <Bucket>qcs::cos:ap-beijing::bucketname-100000000</Bucket>
       <Prefix>/testprefix</Prefix>
      </COSBucketDestination>
     </Destination>
     <Schedule>
```

```
      <Frequency>Daily</Frequency>
     </Schedule>
     <Filter>
      <Prefix>/my-inventory2</Prefix>
     </Filter>
     <OptionalFields>
      <Field>ATime</Field>
      <Field>ACL</Field>
      <Field>MTime</Field>
      <Field>CTime</Field>
      <Field>Size</Field>
      <Field>UserName</Field>
      <Field>GroupName</Field>
      <Field>StorageClass</Field>
      <Field>FileType</Field>
     </OptionalFields>
    </InventoryConfiguration>
    <IsTruncated>false</IsTruncated>
   </ListInventoryConfigurationResult>
```

DELETE Bucket Inventory

This interface is used to delete specified inventory tasks in a bucket. For the interface parameter description, refer to COS DELETE Bucket Inventory API Documentation. This API requires the inventory ID parameter, which is automatically generated by the metadata acceleration bucket when creating a task. If needed, call the List Bucket Inventory Configurations API to obtain this ID.

Request

```
DELETE /?inventory&id=inventory-configuration-id HTTP/1.1
Host: <BucketName-APPID>.cos.<Region>.myqcloud.com
Date: GMT Date
Authorization: Auth String
```

Response

```
HTTP/1.1 204 No Content
Server: tencent-cos
Date: Mon, 28 Aug 2018 02:53:40 GMT
```

```
x-cos-id-2:0dfafa/DAPDIFdafdsfDdfSFFfdfKKJdafasiuKJK2
x-cos-request-id: NTlhM2I3M2JfMjQ4OGY3MGFfMWE1NF84****
```

Data migration and synchronization Migrating HDFS Data to Metadata Acceleration-Enabled Bucket

Last updated：2023-09-13 16:43:24

Feature Overview

COS offers the metadata acceleration feature to provide high-performance file system capabilities. Metadata acceleration leverages the powerful metadata management feature of Cloud HDFS (CHDFS) at the underlying layer to allow using file system semantics for COS access. The designed system metrics can reach a bandwidth of up to 100 GB/s, over 100,000 queries per second (QPS), and a latency of milliseconds. Buckets with metadata acceleration enabled can be widely used in scenarios such as big data, high-performance computing, machine learning, and AI. For more information on metadata acceleration, see Metadata Acceleration Overview.
COS provides the Hadoop semantics through the metadata acceleration service. Therefore, you can use COSDistCp to easily implement two-way data migration between COS and other Hadoop file systems. This document describes how to use COSDistCp to migrate files in the local HDFS to a metadata acceleration bucket in COS.

Environment Preparations Before Migration

Migration tools

1. Download the JAR packages of the tools as listed below and place them in the local directory on the node running the migration task in the cluster, such as `/data01/jars`.

Tencent Cloud EMR Environment
Installation notes

JAR Package Filename	Note	Download Address
cos-distcp-1.12-3.1.0.jar	COSDistCp package, whose data needs to be copied to COSN.	For more information, see COSDistCp Tool.
chdfs_hadoop_plugin_network-2.8.jar	OFS plugin	Download

Self-built Hadoop/CDH environments
Software dependency

Hadoop v2.6.0 or above, Hadoop-COS plugin v8.1.5 or above, and the cos_api-bundle plugin version corresponding to the Hadoop-COS version. For more information, please refer to COSN GitHub Releases.

Installation notes

Install the following plugins in the Hadoop environment:

JAR Package Filename	Note	Download Address
cos-distcp-1.12-3.1.0.jar	COSDistCp package, whose data needs to be copied to COSN.	For more information, see COSDistCp Tool.
chdfs_hadoop_plugin_network-2.8.jar	OFS plugin	Download
Hadoop-COS	Version >= 8.1.5	Please refer to Hadoop-COS Tools
cos_api-bundle	The version needs to match the Hadoop-COS version.	Download

Note

- Hadoop-COS supports access to metadata acceleration buckets in the format of `cosn://bucketname-appid/` starting from v8.1.5.
- The metadata acceleration feature can only be enabled during bucket creation and cannot be disabled once enabled. Therefore, carefully consider whether to enable it based on your business conditions. You should also note that legacy Hadoop-COS packages cannot access metadata acceleration buckets.

2. Create a metadata acceleration bucket and configure the HDFS protocol for it as instructed in "Creating Bucket and Configuring the HDFS Protocol" in Using HDFS to Access Metadata Acceleration-Enabled Bucket.
3. Modify the migration cluster's `core-site.xml` and distribute the configuration to all nodes. If only data needs to be migrated, you don't need to restart the big data component.

key	value	Configuration File	Note
fs.cosn.trsf.fs.ofs.impl	com.qcloud.chdfs.fs.CHDFSHadoopFileSystemAdapter	core-site.xml	COSN implementation class, which is required.
fs.cosn.trsf.fs.AbstractFileSystem.ofs.impl	com.qcloud.chdfs.fs.CHDFSDelegateFSAdapter	core-site.xml	COSN implementation class, which is required.
fs.cosn.trsf.fs.ofs.tmp.cache.dir	In the format of `/data/emr/hdfs/tmp/`	core-site.xml	Temporary directory, which is required. It will be created on all MRS nodes. You need to ensure that there are sufficient space and permissions.
fs.cosn.trsf.fs.ofs.user.appid	`appid` of your COS bucket	core-site.xml	Required
fs.cosn.trsf.fs.ofs.ranger.enable.flag	false	core-site.xml	This key is required. You need to check whether the value is `false`.
fs.cosn.trsf.fs.ofs.bucket.region	Bucket region	core-site.xml	This key is required. Valid values: eu-frankfurt (Frankfurt), ap-chengdu (Chengdu), and ap-singapore (Singapore).

4. You can verify the migration by accessing the metadata acceleration bucket over the private network as instructed in "Configuring Computing Cluster to Access COS" in Using HDFS to Access Metadata Acceleration-Enabled Bucket. Use the migration cluster submitter to verify whether COS can be accessed successfully.

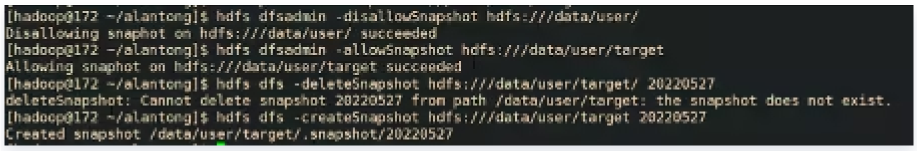
Existing Data Migration

1. Determine the migration directory

Generally, HDFS storage data will be first migrated. The directory to be migrated in the source HDFS cluster will be selected, and the target path needs to be the same as the source path. Suppose you need to migrate the HDFS directory `hdfs:///data/user/target` to `cosn://{bucketname-appid}/data/user/target`.
To ensure that the files in the source directory remain unchanged during migration, the snapshot feature of HDFS will be used to create a snapshot of the source directory (named the current date).

```
hdfs dfsadmin -disallowSnapshot hdfs:///data/user/
hdfs dfsadmin -allowSnapshot hdfs:///data/user/target
hdfs dfs -deleteSnapshot hdfs:///data/user/target {current date}
hdfs dfs -createSnapshot hdfs:///data/user/target {current date}
```

Refer to the successful example:



```
[hadoop@172 ~/alantong]$ hdfs dfsadmin -disallowSnapshot hdfs:///data/user/
Disallowing snaphot on hdfs:///data/user/ succeeded
[hadoop@172 ~/alantong]$ hdfs dfsadmin -allowSnapshot hdfs:///data/user/target
Allowing snaphot on hdfs:///data/user/target succeeded
[hadoop@172 ~/alantong]$ hdfs dfs -deleteSnapshot hdfs:///data/user/target/ 20220527
deleteSnapshot: Cannot delete snapshot 20220527 from path /data/user/target: the snapshot does not exist.
[hadoop@172 ~/alantong]$ hdfs dfs -createSnapshot hdfs:///data/user/target 20220527
Created snapshot /data/user/target/.snapshot/20220527
```

If you don't want to create a snapshot, you can directly migrate the target files in the source directory.

2. Migrating with COSDistCp

Start a COSDistCp task to copy files from the source HDFS to the target COS bucket. COSDistCp is a MapReduce task, and the success or failure of the MR task execution is indicated in the printed logs. If the task fails, you can check the YARN page and provide the logs or exception information to COS for troubleshooting. The migration process using COSDistCp consists of the following steps:
(1) Create a temporary directory
(2) Run the COSDistCp task
(3) Retry migration for failed files

(1) Create a temporary directory

```
hadoop fs -libjars /data01/jars/chdfs_hadoop_plugin_network-2.8.jar -
mkdir cosn://bucket-appid/distcp-tmp
```

(2) Run a COSDistCp task

```
nohup hadoop jar /data01/jars/cos-distcp-1.10-2.8.5.jar -libjars
/data01/jars/chdfs_hadoop_plugin_network-2.8.jar --
src=hdfs:///data/user/target/.snapshot/{current date}  --
dest=cosn://{bucket-appid}/data/user/target   --temp=cosn://bucket-
appid/distcp-tmp/ --preserveStatus=ugpt  --skipMode=length-checksum --
checkMode=length-checksum --cosChecksumType=CRC32C --taskNumber 6 --
workerNumber 32 --bandWidth 200 >> ./distcp.log &
```

The parameters are as detailed below. You can adjust their values as needed.

- --taskNumber=VALUE: Number of copy threads. Example: `--taskNumber=10` .
- --workerNumber=VALUE: Number of copy threads. COSDistCp will create a copy thread pool for each copy process based on this value set. Example: `workerNumber=4` .
- --bandWidth: Maximum bandwidth for reading each migrated file (in MB/s). Default value: `-1` , which indicates no limit on the read bandwidth. Example: `--bandWidth=10` .
- --cosChecksumType=CRC32C: CRC32C is used by default, but the HDFS cluster must be able to check `COMPOSITE_CRC32` . The Hadoop version must be 3.1.1 or later; otherwise, you need to change this parameter to `--cosChecksumType=CRC64` .

Note
The formula for calculating the total bandwidth limit of COSDistCp migration is: taskNumber * workerNumber * bandWidth. You can set `workerNumber` to `1` , use the

`taskNumber` parameter to control the number of concurrent migrations, and use the `bandWidth` parameter to control the bandwidth of a single concurrent migration.

Upon completion of the copy task, the task log will display statistical information about the file copy. The relevant counters are as follows:
Here, FILES_FAILED represents the number of failed files. If there is no FILES_FAILED counter, it indicates that all files have been successfully migrated.

```
CosDistCp Counters
        BYTES_EXPECTED=10198247
        BYTES_SKIPPED=10196880
        FILES_COPIED=1
        FILES_EXPECTED=7
        FILES_FAILED=1
        FILES_SKIPPED=5
```

The specific statistics items in the output result are as detailed below:

Statistics Item	Note
BYTES_EXPECTED	Total size (in bytes) to copy according to the source directory
FILES_EXPECTED	Number of files to copy according to the source directory, including the directory itself
BYTES_SKIPPED	Total size (in bytes) of files that can be skipped (same length or checksum value)
FILES_SKIPPED	Number of source files that can be skipped (same length or checksum value)
FILES_COPIED	Number of source files that are successfully copied
FILES_FAILED	Number of source files that failed to be copied
FOLDERS_COPIED	Number of directories that are successfully copied
FOLDERS_SKIPPED	Number of directories that are skipped

(3) Migrate failed files again

COSDistCp not only solves most problems of inefficient file migration but also allows you to use the `--delete` parameter to guarantee the complete consistency between the HDFS and

COS data.
When using the `--delete` parameter, you need to add the `--deleteOutput=/xxx(custom)` parameter but not the `--diffMode` parameter.

```
nohup hadoop jar /data01/jars/cos-distcp-1.10-2.8.5.jar -libjars
/data01/jars/chdfs_hadoop_plugin_network-2.8.jar --src=--
src=hdfs:///data/user/target/.snapshot/{current date} --
dest=cosn://{bucket-appid}/data/user/target --temp=cosn://bucket-
appid/distcp-tmp/ --preserveStatus=ugpt --skipMode=length-checksum --
checkMode=length-checksum --cosChecksumType=CRC32C --taskNumber 6 --
workerNumber 32 --bandWidth 200 --delete --deleteOutput=/dele-xx >>
./distcp.log &
```

After execution, the different data between HDFS and COS will be moved to the `trash` directory, and the list of moved files will be generated in the `/xxx/failed` directory. You can run `hadoop fs -rm URL` or `hadoop fs -rmr URL` to delete the data in the `trash` directory.

Incremental Migration

If any incremental data needs to be migrated afterwards, you only need to repeat the steps of full migration until all data has been migrated.

Best Practices
Client Best Practices

Last updated：2025-11-13 15:50:21

High-Throughput Practice

The client reads data from the bucket each time, and the data size is controlled by the configuration item `fs.ofs.block.memory.trunk.byte`.

Configuration Item	Configuration Item Content	Description
fs.ofs.block.memory.trunk.byte	1048576	Object block size in bytes. The default value is 1048576 (1 MB).

Note:
If you use COSN SDK, you need to add `fs.cosn.trsf` before the configuration item name, such as `fs.cosn.trsf.fs.ofs.prev.read.block.count`.

In high throughput scenarios, the client introduced a pre-read block logic, splitting a file into multiple pre-read blocks and caching them in memory to reduce read latency and enhance throughput. You can specify pre-read block parameters to meet throughput requirements in different scenarios.

Sequential Read Scenario

The client internally detects whether the current scenario is sequential read using a cursor. If it is a sequential read scenario, the pre-read logic will be enabled; otherwise, it will not be used. In the pre-read logic, the client fixes the number of pre-read blocks cached each time. The related configuration items are as follows. You can adjust related parameters based on the model configuration.

Configuration Item	Configuration Item Content	Description
fs.ofs.prev.read.block.count	16	Pre-read block count, default value: 16
fs.ofs.prev.read.block.release.enable	true	Whether to release read blocks from memory, default value: true

fs.ofs.block.max.read.memory.cache.mb	16	Single file available memory amount, default value: 16, unit: MB **Note:** To avoid OOM, you can refer to the following context for memory usage practice and control the global cache model.
fs.ofs.data.transfer.thread.count	32	Core thread count of the IO thread pool for prefetching blocks from the bucket
fs.ofs.data.transfer.max.thread.count	Integer.MAX_VALUE	Maximum number of threads in the IO thread pool

Random Read Scenario

As mentioned in the previous context, the client will detect the current scenario. If it is a random read, the prefetch logic will not trigger. Additionally, in this scenario, it is recommended to adjust the configuration item `fs.ofs.block.memory.trunk.byte` based on actual business scenarios, modify the data size read from the COS bucket each time, and avoid read amplification in random read scenarios.

Client Memory Usage Practice

To enhance performance, OFS SDK caches data during upload and download. The cache includes memory cache and disk cache, with different applications in upload and download.

- Upload: Use memory cache + disk cache
- Download: Use memory cache only

Among them, memory cache blocks are allocated on demand, with priority given to memory cache blocks. When memory cache blocks are insufficient, disk cache blocks will be allocated. Disk cache uses off-heap memory during data writing to reduce copy operations between heap memory and kernel-space memory, thereby improving data writing performance.

To avoid multiple files affecting each other (for example, when a certain file is not closed or resources are not released, causing new reading and writing operations to fail), the SDK's default principle is to control the number of caches used by a **single file**. Meanwhile, to prevent

OOM issues, the client provides a **global (file system granularity)** cache configuration item. You can adjust the configuration based on the client environment to achieve the best performance.

Single File Cache Control Model

OFS SDK uses the following two configuration items to control the cache usage of a single file:

Configuration Item	Configuration Item Content	Description
fs.ofs.block.max.memory.cache.mb	16	Memory cache usage, default value: 16, unit: MB
fs.ofs.block.max.file.cache.mb	256	Disk cache usage, default value: 256, unit: MB

Note:
If you use COSN SDK, you need to add `fs.cosn.trsf` before the configuration item name, such as `fs.cosn.trsf.fs.ofs.block.max.memory.cache.mb`.

Global (File System Granularity) Cache Control Model

The SDK provides global (file system granularity) cache control for read and write requests to avoid OOM issues. Details are as follows:

Uploading

OFS SDK provides three configuration items used to control global upload memory.

Configuration Item	Configuration Item Content	Description
fs.ofs.block.total.memory.cache.mb	0	Maximum upload memory usage, default value 0 (no control), MB
fs.ofs.block.total.memory.cache.percent	100	Maximum upload memory usage ratio, default value 100, unit: percentage
fs.ofs.block.total.memory.jvm.heap.percent	0	Maximum JVM memory usage ratio, default value 0 (no control), unit: percentage

The SDK offers two global controls:

- Rule 1: Control the maximum memory usage for uploads via `fs.ofs.block.total.memory.cache.mb` and `fs.ofs.block.total.memory.cache.percent`. After configuration, the maximum memory size used is: `fs.ofs.block.total.memory.cache.mb` * `fs.ofs.block.total.memory.cache.percent` / 100. Once configured, the SDK will calculate the number of files that can be written concurrently based on the global memory cache size / the maximum size of a single file `fs.ofs.block.max.memory.cache.mb`, and allocate semaphores accordingly. When opening a new file, it will apply for one semaphore. If the application fails, it will forcibly use disk cache. When the file is closed, the semaphore will be returned.
- Rule 2: Control the maximum JVM memory via fs.ofs.block.total.memory.jvm.heap.percent. After configuration, the SDK will obtain the JVM maximum memory `(ManagementFactory.getMemoryMXBean().getHeapMemoryUsage().getMax())` * `fs.ofs.block.total.memory.jvm.heap.percent` / 100 to determine.

By default, `fs.ofs.block.total.memory.cache.mb` and `fs.ofs.block.total.memory.jvm.heap.percent` are set to 0, meaning no memory control is performed. If both configuration items are non-zero, rule 1 (maximum memory usage) has a higher priority than rule 2 (maximum JVM memory usage).

Download

CHDFS SDK provides three configuration items used to control global download memory.

Configuration Item	Configuration Item Content	Description
fs.ofs.block.total.read.memory.cache.mb	0	Maximum download memory usage, default value 0 (no control), MB
fs.ofs.block.total.read.memory.cache.percent	100	Maximum download memory usage ratio, default value 100, unit: percentage

The SDK controls the maximum memory usage for downloads via `fs.ofs.block.total.read.memory.cache.mb` and `fs.ofs.block.total.read.memory.cache.percent`. After configuration, the maximum memory used is: `fs.ofs.block.total.read.memory.cache.mb` * `fs.ofs.block.total.read.memory.cache.percent` / 100. Once configured, the SDK calculates the number of files that can be written simultaneously based on the global memory cache size divided by the maximum size of a single file `fs.ofs.block.max.memory.cache.mb`, thereby allocating signals. When opening a new file, it requests one semaphore. If the request fails, it forcibly uses disk cache. When a file is closed, the semaphore is returned. The request implements a blocking mechanism through queuing. When semaphores are insufficient, it waits for other files to close and release semaphores.

Accessing COS over HDFS in CDH Cluster

Last updated: 2023-09-20 12:16:58

Feature Overview

CDH (Cloudera's distribution, including Apache Hadoop) is one of the most popular Hadoop distributions in the industry. This document describes how to access a COS bucket over the HDFS protocol, a flexible, cost-effective big-data solution, in a CDH environment to separate big data computing from storage.

Note
To access a COS bucket via the HDFS protocol, you must first enable metadata acceleration capabilities.

Currently, the support for big data modules by COS is as follows:

Module	Supported by CHDFS	Service Module to Restart
Yarn	This feature is supported.	NodeManager
Yarn	This feature is supported.	NodeManager
Hive	This feature is supported.	HiveServer and HiveMetastore
Spark	This feature is supported.	NodeManager
Sqoop	This feature is supported.	NodeManager
Presto	This feature is supported.	HiveServer, HiveMetastore, and Presto
Flink	This feature is supported.	Not required
Impala	This feature is supported.	Not required

EMR	This feature is supported.	Not required
Self-built components	Will be supported later.	-
HBase	Not recommended.	-

Versions

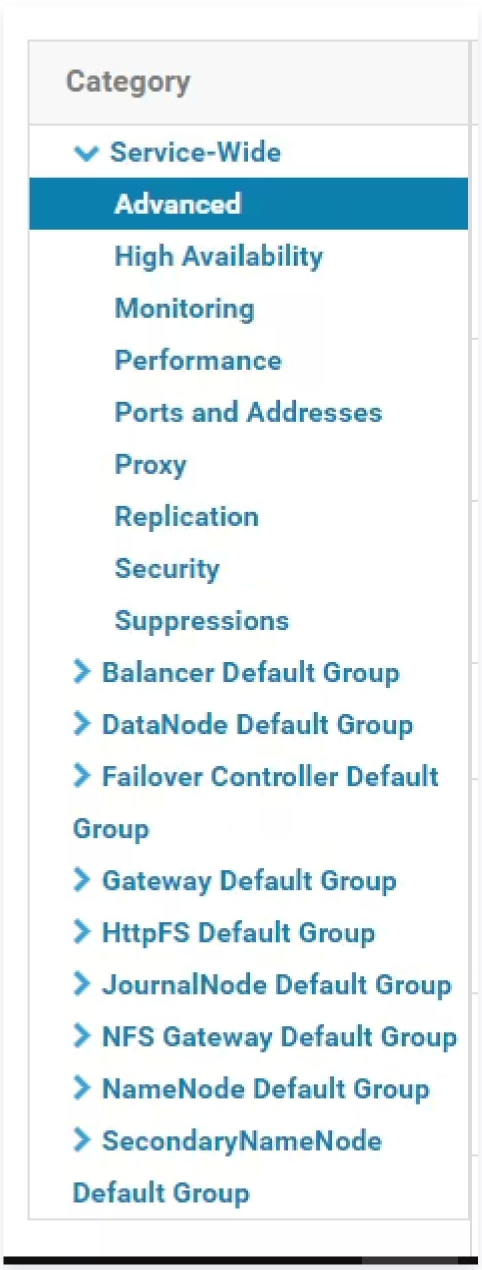
This example uses software versions as follows:

- CDH 5.16.1
- Hadoop 2.6.0

How to Use

Configuring storage environment

1. Log in to the CDH management page.
2. On the system homepage, select **Configuration** > **Service-Wide** > **Advanced** to access the advanced configuration code snippet page, as shown below:



3. Specify your COS settings in the configuration snippet `Cluster-wide Advanced Configuration Snippet(Safety Valve) for core-site.xml`.

```
<property>
<name>fs.AbstractFileSystem.ofs.impl</name>
<value>com.qcloud.chdfs.fs.CHDFSDelegateFSAdapter</value>
</property>
<property>
<name>fs.ofs.impl</name>
<value>com.qcloud.chdfs.fs.CHDFSHadoopFileSystemAdapter</value>
</property>
```

```
<!--Temporary directory of the local cache. For data read/write, data
will be written to the local disk when the memory cache is
insufficient. This path will be created automatically if it does not
exist-->
<property>
<name>fs.ofs.tmp.cache.dir</name>
<value>/data/emr/hdfs/tmp/chdfs/</value>
</property>
<!--appId-->
<property>
<name>fs.ofs.user.appid</name>
<value>1250000000</value>
</property>
```

The following are required configuration items (to be added to core-site.xml). For other configurations, please refer to Mounting COS Buckets in a Computing Cluster.

Configuration items	Value	Description
fs.ofs.user.appid	1250000000	User `appid`
fs.ofs.tmp.cache.dir	/data/emr/hdfs/tmp/chdfs/	Temporary directory of the local cache
fs.ofs.impl	com.qcloud.chdfs.fs.CHDFSHadoopFileSystemAdapter	The implementation class of CHDFS for `FileSystem` is fixed at `com.qcloud.chdfs.fs.CHDFSHadoopFileSystemAdapter`.
fs.AbstractFileSystem.ofs.impl	com.qcloud.chdfs.fs.CHDFSDelegateFSAdapter	The implementation class of CHDFS for `AbstractFileSystem` is fixed at `com.qcloud.chdfs.fs.CHDFSDelegateFSAdapter`.

4. Take action on your HDFS service by clicking. Now, the core-site.xml settings above will apply to servers in the cluster.
5. Place the latest client installation package in the path of the JAR package of the CDH HDFS service and replace the relevant information with the actual value as shown below:

```
cp chdfs_hadoop_plugin_network-2.0.jar /opt/cloudera/parcels/CDH-
5.16.1-1.cdh5.16.1.p0.3/lib/hadoop-hdfs/
```

> **Note**
> The SDK package must be placed in the same location on each machine within the cluster.

Data Migration

Use Hadoop Distcp to migrate your data from CDH HDFS to a COS bucket. For details, see Migrating Data Between HDFS and COS.

Big Data Suite using CHDFS

MapReduce

Operation Steps

1. Configure HDFS as instructed in Data migration and put the client installation package of COS in the correct HDFS directory.
2. On the Cloudera Manager homepage, find YARN and restart the NodeManager service (recommended). You can choose not to restart it for the TeraGen command, but must restart it for the TeraSort command because of the internal business logic.

Example

The example below shows TeraGen and TeraSort in Hadoop standard test:

```
hadoop jar ./hadoop-mapreduce-examples-2.7.3.jar teragen -
Dmapred.map.tasks=4 1099 ofs://examplebucket-1250000000/teragen_5/

hadoop jar ./hadoop-mapreduce-examples-2.7.3.jar terasort  -
Dmapred.map.tasks=4 ofs://examplebucket-1250000000/teragen_5/
ofs://examplebucket-1250000000/result14
```

Note
Please replace the path after the `ofs:// schema` with the user's CHDFS mount point path.

Hive

MR engine

Operation Steps

1. Configure HDFS as instructed in Data migration and put the client installation package of COS in the correct HDFS directory.
2. On the Cloudera Manager homepage, find HIVE and restart the Hiveserver2 and HiverMetastore roles.

Example

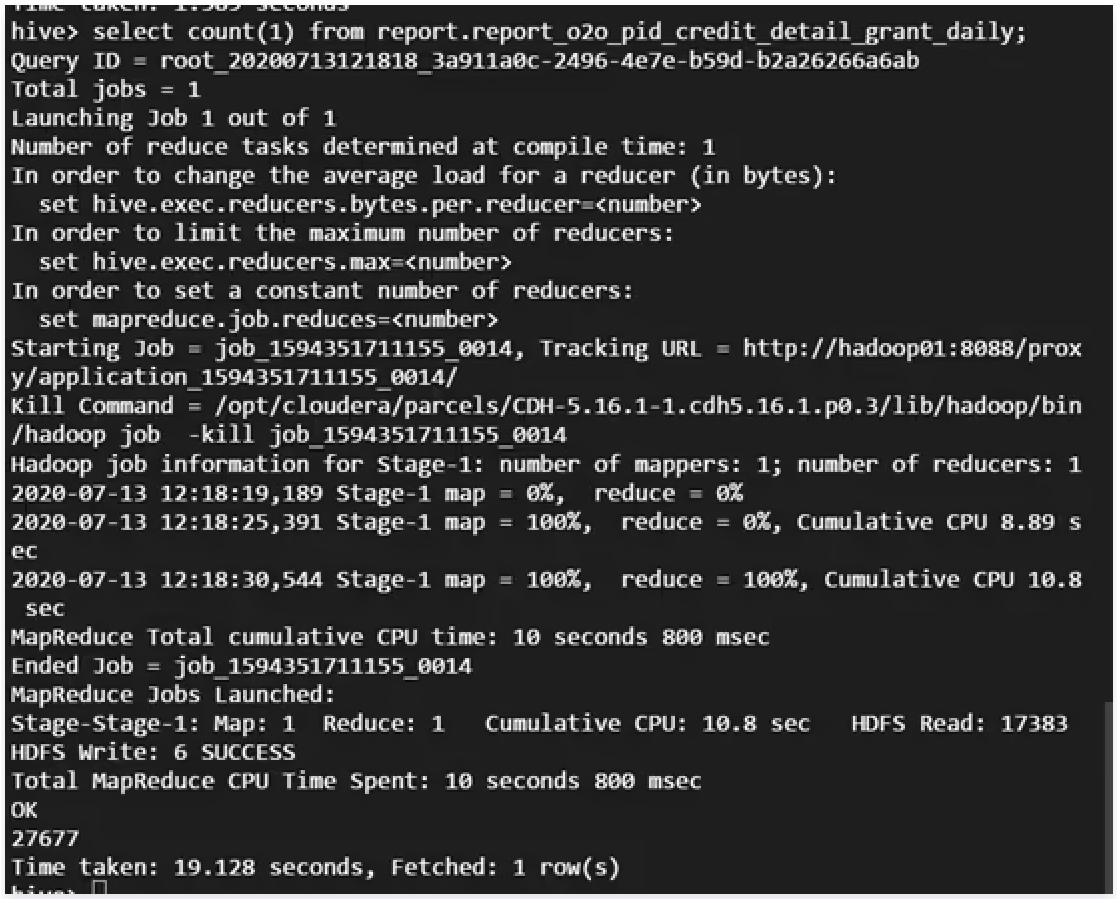
To query your actual business data, use the Hive command line to create a location as a partitioned table on CHDFS:

```
CREATE TABLE report.report_o2o_pid_credit_detail_grant_daily(
   cal_dt string,
   change_time string,
   merchant_id bigint,
   store_id bigint,
   store_name string,
   wid string,
   member_id bigint,
   meber_card string,
   nickname string,
   name string,
   gender string,
   birthday string,
   city string,
   mobile string,
   credit_grant bigint,
   change_reason string,
   available_point bigint,
   date_time string,
   channel_type bigint,
   point_flow_id bigint)
PARTITIONED BY (
   topicdate string)
ROW FORMAT SERDE
   'org.apache.hadoop.hive.ql.io.orc.OrcSerde'
STORED AS INPUTFORMAT
   'org.apache.hadoop.hive.ql.io.orc.OrcInputFormat'
     OUTPUTFORMAT
   'org.apache.hadoop.hive.ql.io.orc.OrcOutputFormat'
LOCATION
   'ofs://examplebucket-
1250000000/user/hive/warehouse/report.db/report_o2o_pid_credit_detail_gr
ant_daily'
TBLPROPERTIES (
   'last_modified_by'='work',
   'last_modified_time'='1589310646',
   'transient_lastDdlTime'='1589310646')
```

Perform a SQL query:

```
select count(1) from report.report_o2o_pid_credit_detail_grant_daily;
```

The observed results are as follows:



```
hive> select count(1) from report.report_o2o_pid_credit_detail_grant_daily;
Query ID = root_20200713121818_3a911a0c-2496-4e7e-b59d-b2a26266a6ab
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1594351711155_0014, Tracking URL = http://hadoop01:8088/prox
y/application_1594351711155_0014/
Kill Command = /opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/lib/hadoop/bin
/hadoop job  -kill job_1594351711155_0014
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2020-07-13 12:18:19,189 Stage-1 map = 0%,  reduce = 0%
2020-07-13 12:18:25,391 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 8.89 s
ec
2020-07-13 12:18:30,544 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 10.8
 sec
MapReduce Total cumulative CPU time: 10 seconds 800 msec
Ended Job = job_1594351711155_0014
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 10.8 sec   HDFS Read: 17383
HDFS Write: 6 SUCCESS
Total MapReduce CPU Time Spent: 10 seconds 800 msec
OK
27677
Time taken: 19.128 seconds, Fetched: 1 row(s)
```

Tez engine

You need to import the client installation package of COS as part of a Tez tar.gz file. The following example uses apache-tez.0.8.5:

Operation Steps

1. Locate and decompress the Tez tar.gz file installed in the CDH cluster, e.g., /usr/local/service/tez/tez-0.8.5.tar.gz.
2. Put the client installation package of COS in the directory generated after decompression and recompress it to output a compressed package.
3. Upload this new file to the path as specified by tez.lib.uris, or simply replace the existing file with the same name.
4. On the Cloudera Manager homepage, find HIVE and restart hiveserver and hivemetastore.

Spark

Operation Steps

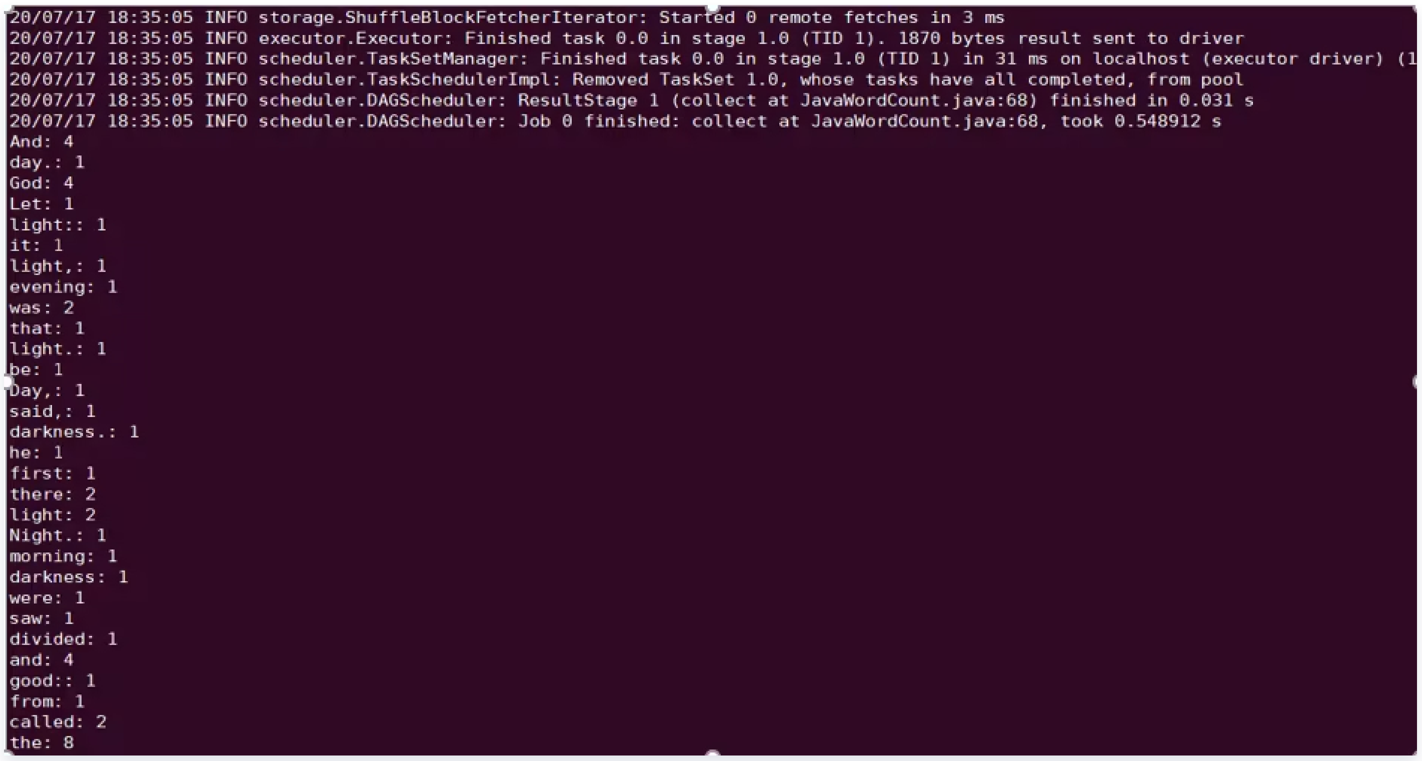
1. Configure HDFS as instructed in Data migration and put the client installation package of COS in the correct HDFS directory.
2. Restart NodeManager.

Example

The following uses the conducted `Spark example word count` test as an example.

```
spark-submit  --class org.apache.spark.examples.JavaWordCount --
executor-memory 4g --executor-cores 4  ./spark-examples-1.6.0-cdh5.16.1-
hadoop2.6.0-cdh5.16.1.jar ofs://examplebucket-1250000000/wordcount
```

The execution result is as follows:



```
20/07/17 18:35:05 INFO storage.ShuffleBlockFetcherIterator: Started 0 remote fetches in 3 ms
20/07/17 18:35:05 INFO executor.Executor: Finished task 0.0 in stage 1.0 (TID 1). 1870 bytes result sent to driver
20/07/17 18:35:05 INFO scheduler.TaskSetManager: Finished task 0.0 in stage 1.0 (TID 1) in 31 ms on localhost (executor driver) (1
20/07/17 18:35:05 INFO scheduler.TaskSchedulerImpl: Removed TaskSet 1.0, whose tasks have all completed, from pool
20/07/17 18:35:05 INFO scheduler.DAGScheduler: ResultStage 1 (collect at JavaWordCount.java:68) finished in 0.031 s
20/07/17 18:35:05 INFO scheduler.DAGScheduler: Job 0 finished: collect at JavaWordCount.java:68, took 0.548912 s
And: 4
day.: 1
God: 4
Let: 1
light:: 1
it: 1
light,: 1
evening: 1
was: 2
that: 1
light.: 1
be: 1
Day,: 1
said,: 1
darkness.: 1
he: 1
first: 1
there: 2
light: 2
Night.: 1
morning: 1
darkness: 1
were: 1
saw: 1
divided: 1
and: 4
good:: 1
from: 1
called: 2
the: 8
```

Sqoop

Operation Steps

1. Configure HDFS as instructed in Data migration and put the client installation package of COS in the correct HDFS directory.
2. Put the client installation package of COS in the `sqoop` directory, for example, `/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/lib/sqoop/`.
3. Restart NodeManager.

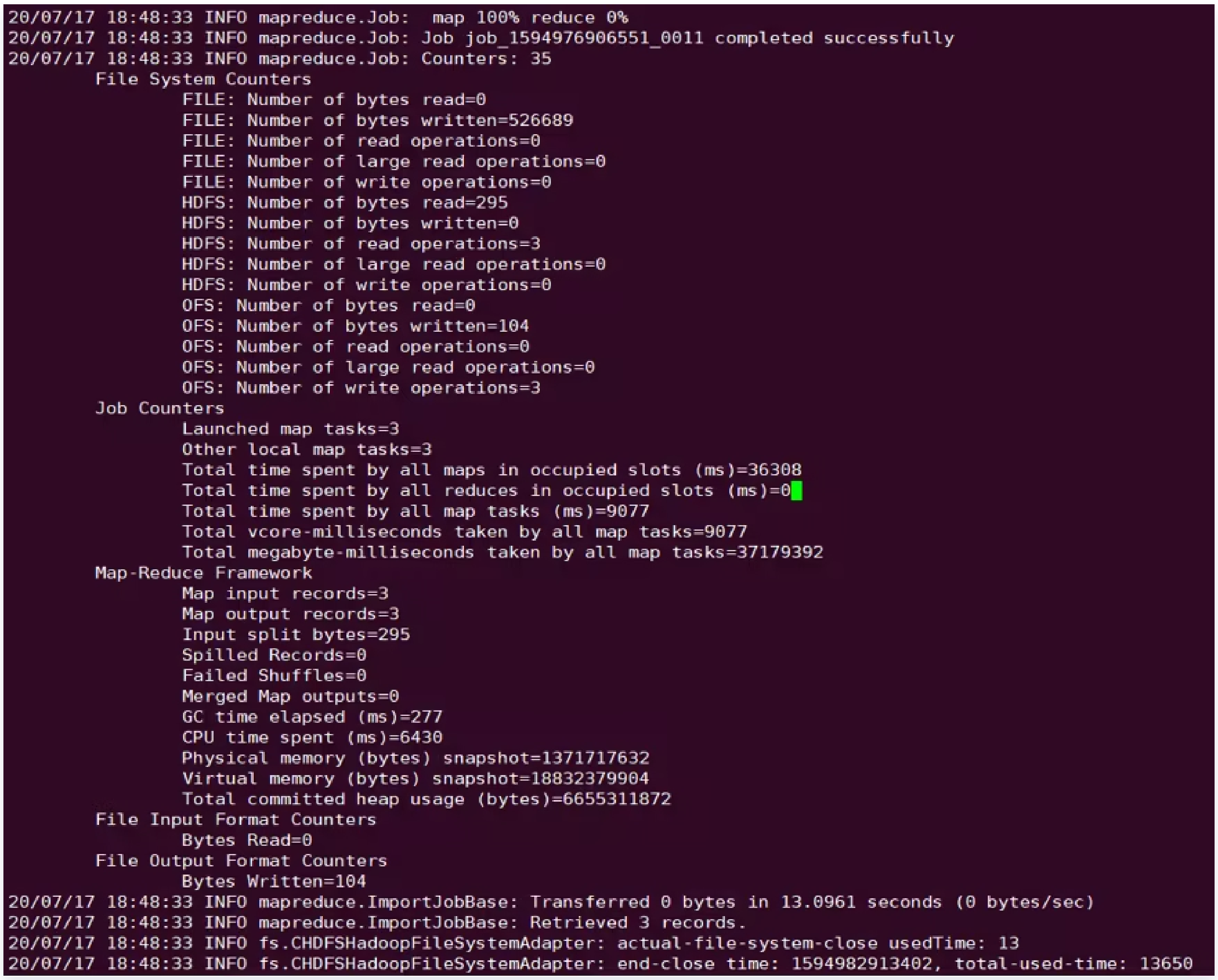
Example

For example, to export MySQL tables to COS, refer to Import/Export of Relational Database and HDFS.

```
sqoop import --connect "jdbc:mysql://IP:PORT/mysql" --table sqoop_test -
-username root --password 123  --target-dir ofs://examplebucket-
```

```
1250000000/sqoop_test
```

The execution result is as follows:



```
20/07/17 18:48:33 INFO mapreduce.Job:  map 100% reduce 0%
20/07/17 18:48:33 INFO mapreduce.Job: Job job_1594976906551_0011 completed successfully
20/07/17 18:48:33 INFO mapreduce.Job: Counters: 35
        File System Counters
                FILE: Number of bytes read=0
                FILE: Number of bytes written=526689
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=295
                HDFS: Number of bytes written=0
                HDFS: Number of read operations=3
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=0
                OFS: Number of bytes read=0
                OFS: Number of bytes written=104
                OFS: Number of read operations=0
                OFS: Number of large read operations=0
                OFS: Number of write operations=3
        Job Counters
                Launched map tasks=3
                Other local map tasks=3
                Total time spent by all maps in occupied slots (ms)=36308
                Total time spent by all reduces in occupied slots (ms)=0
                Total time spent by all map tasks (ms)=9077
                Total vcore-milliseconds taken by all map tasks=9077
                Total megabyte-milliseconds taken by all map tasks=37179392
        Map-Reduce Framework
                Map input records=3
                Map output records=3
                Input split bytes=295
                Spilled Records=0
                Failed Shuffles=0
                Merged Map outputs=0
                GC time elapsed (ms)=277
                CPU time spent (ms)=6430
                Physical memory (bytes) snapshot=1371717632
                Virtual memory (bytes) snapshot=18832379904
                Total committed heap usage (bytes)=6655311872
        File Input Format Counters
                Bytes Read=0
        File Output Format Counters
                Bytes Written=104
20/07/17 18:48:33 INFO mapreduce.ImportJobBase: Transferred 0 bytes in 13.0961 seconds (0 bytes/sec)
20/07/17 18:48:33 INFO mapreduce.ImportJobBase: Retrieved 3 records.
20/07/17 18:48:33 INFO fs.CHDFSHadoopFileSystemAdapter: actual-file-system-close usedTime: 13
20/07/17 18:48:33 INFO fs.CHDFSHadoopFileSystemAdapter: end-close time: 1594982913402, total-used-time: 13650
```

Presto

Operation Steps

1. Configure HDFS as instructed in Data migration and put the client installation package of COS in the correct HDFS directory.
2. Put the client installation package of COS in the `presto` directory, for example, `/usr/local/services/cos_presto/plugin/hive-hadoop2`.
3. Since Presto does not load gson-2...jar from the Hadoop common directory, you need to place gson-2...jar in the Presto directory as well (e.g., /usr/local/services/cos_presto/plugin/hive-hadoop2, as only COS depends on gson).
4. Restart HiveServer, HiveMetaStore, and Presto.

Example

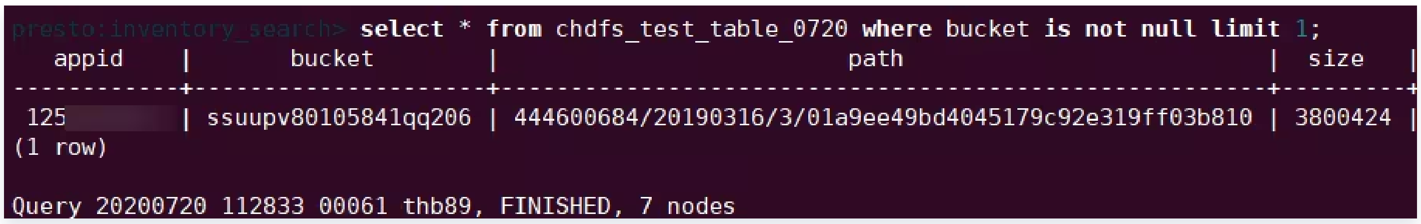
The example below queries the COS scheme table as a HIVE-created Location:

```
select * from chdfs_test_table where bucket is not null limit 1;
```

Note

`chdfs_test_table` is a table with a location using the `ofs` scheme.

The query results are as follows:



```
presto:inventory_search> select * from chdfs_test_table_0720 where bucket is not null limit 1;
   appid    |        bucket        |                        path                            |  size   |
------------+----------------------+--------------------------------------------------------+---------+
 125[redacted] | ssuupv80105841qq206 | 444600684/20190316/3/01a9ee49bd4045179c92e319ff03b810 | 3800424 |
(1 row)

Query 20200720 112833 00061 thb89, FINISHED, 7 nodes
```

Private Network Access to Metadata-Accelerated Bucket

Last updated：2026-03-24 12:02:49

Overview

This document details how to access Tencent Cloud COS Metadata Acceleration Buckets via Direct Connect using VPC Endpoint Service. The Metadata Acceleration feature is a high-performance file system feature launched by COS for big data services. Its underlying layer adopts excellent metadata management capabilities, supporting accessing object storage services via HDFS semantics, which can achieve hundreds of GBs-level bandwidth, hundreds of thousands of QPS, and millisecond-level latency.

Prerequisites

Before starting the configuration, ensure that the following conditions are met:

- **Network Environment:** A cloud infrastructure supporting VPC access has been configured, managed by the Network and VPC team.
- **Endpoint Service:** The Endpoint Service for COS and COS Metadata Acceleration has been created, managed by the COS Metadata-Acceleration team.
- **Allowlist Authorization:** Your account UIN has been added to the Endpoint Service allowlist, managed by the COS Metadata-Acceleration team.
- **Metadata-Accelerated Bucket:** A bucket with the metadata-acceleration feature enabled has been created.

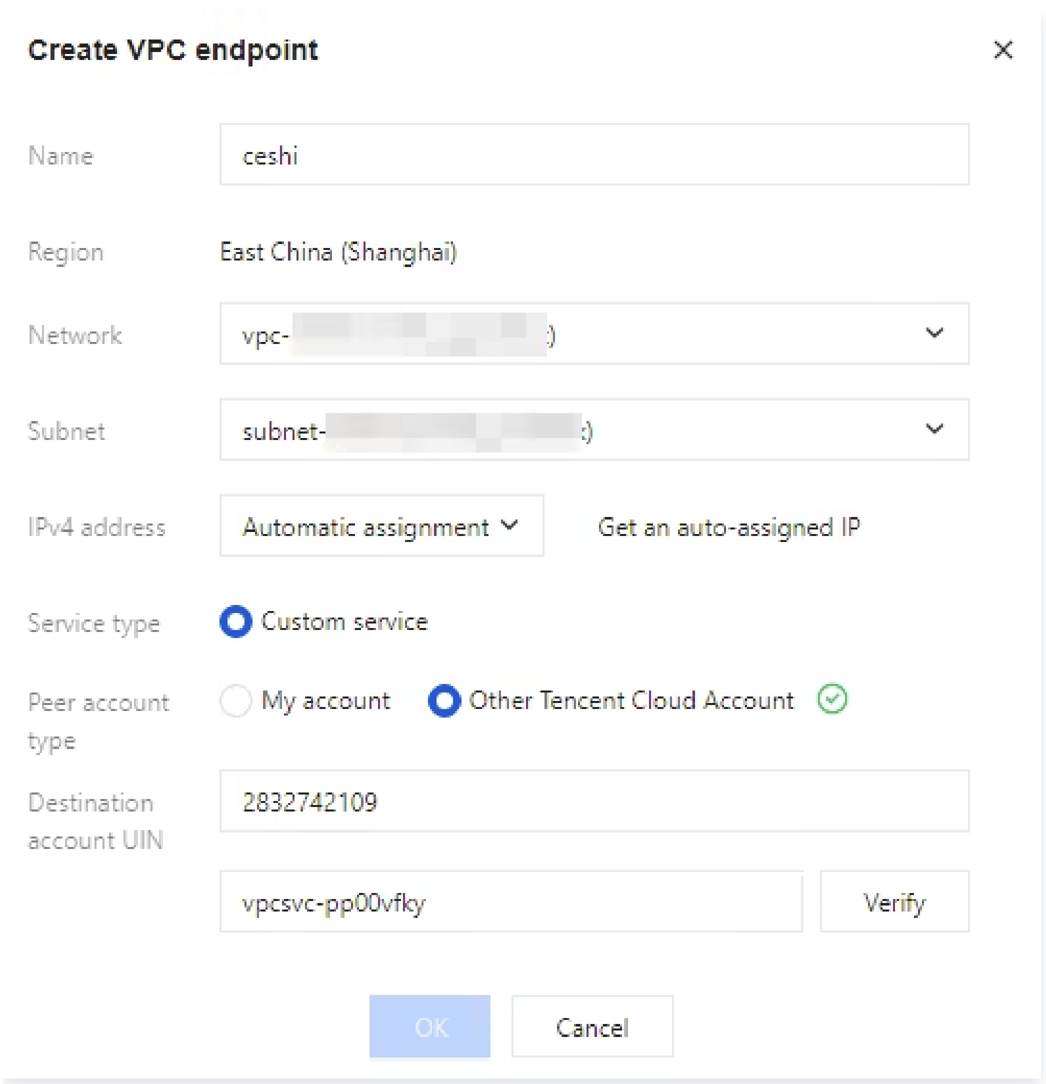
Fee Instructions

Using Endpoint Service will incur additional fees charged by the VPC product. For detailed billing descriptions and examples, see Billing Overview.

Operation Steps

Create COS Endpoint

1. Log in to the VPC Console.
2. Switch to the region where the Metadata Acceleration Bucket you want to access is located. Click **New** and configure the following parameters to create the endpoint.



Create VPC endpoint

Name: ceshi

Region: East China (Shanghai)

Network: vpc-[redacted])

Subnet: subnet-[redacted])

IPv4 address: Automatic assignment — Get an auto-assigned IP

Service type: [✓ Custom service]

Peer account type: My account [✓ Other Tencent Cloud Account]

Destination account UIN: 2832742109

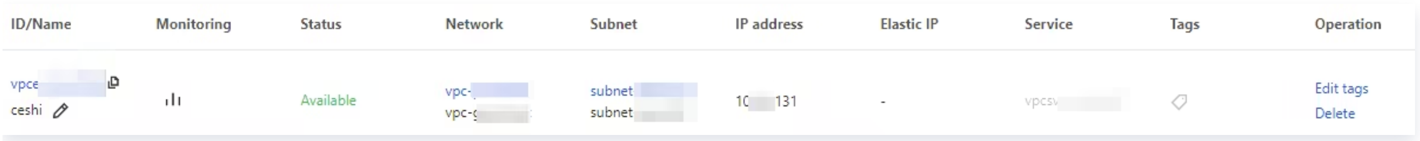
vpcsvc-pp00vfky

- **Name**: Custom endpoint name.
- **Region**: Select the same region as your bucket.
- **Affiliated Network**: Select your VPC network.
- **Subnet**: Select your subnet (the VIP created by the endpoint will be within this subnet range).
- **IP Address**: Select **Automatic Assignment**.
- **Service Type**: Select **Private Service**.
- **Peer Account Type**: Select **Other Account**.
- **Peer Account UIN**: Enter 2832742109. This is the UIN used by the COS endpoint service and remains unchanged.
- **Endpoint Service ID**: Enter the endpoint service ID corresponding to the region where your bucket is located. The regions where COS has deployed endpoints and their corresponding service IDs are as shown in the following table:

Region	Endpoint Service	Region	ID of the Endpoint

	ID		Service
Beijing	vpcsvc-j7f6u7y8	Beijing Finance	vpcsvc-hsuy2wpj
Shanghai	vpcsvc-pp00vfky	Shanghai Finance	vpcsvc-pny969zd
Guangzhou	vpcsvc-okzzl947	Thailand	vpcsvc-i0q3t2v6
Chengdu	vpcsvc-la3dnfti	East US	vpcsvc-fdpg3nm1
Chongqing	vpcsvc-1snqqeih	Brazil	vpcsvc-5edkepjj
Nanjing	vpcsvc-mmoajwvt	West US	vpcsvc-l6fn15sr
Hong Kong (China)	vpcsvc-jejpq2xo	Germany	vpcsvc-r70hxf29
Japan	vpcsvc-q8mwrk38	South Korea	vpcsvc-ocuykgps
Singapore	vpcsvc-71louv91	Indonesia	vpcsvc-8nrq4era
Shanghai Autonomous Driving	vpcsvc-mu3yl0ov	-	-

3. Verify and create. Click **Validate** to verify. After verification passes, click **Confirm** to create the endpoint. The system will automatically generate a VIP address. Please record this VIP for subsequent configuration.



ID/Name	Monitoring	Status	Network	Subnet	IP address	Elastic IP	Service	Tags	Operation
vpce[redacted] ceshi		Available	vpc-[redacted] vpc-[redacted]	subnet[redacted] subnet[redacted]	10[redacted]131	-	vpcsv[redacted]		Edit tags Delete

Create an Endpoint for the COS Metadata Acceleration Service

1. Log in to the VPC Console.
2. Switch to the region where the metadata accelerated bucket you want to access is located, and create a new endpoint. For configuration instructions, refer to Create a COS Endpoint. Note that the COS metadata acceleration service uses different endpoint services. The regions where endpoints have been deployed and their corresponding service IDs are shown in the table below:

Region	ID of the Endpoint Service

Beijing	vpcsvc-dabvvd8k

Note:
If the region where your Metadata Accelerated Bucket is located does not have a corresponding endpoint service, you can contact us to add it.

3. Verify and create. Click **Validate** to verify. After verification passes, click **Confirm** to create the endpoint. The system will automatically generate a VIP address. Please record this VIP for subsequent configuration.

DNS Configuration

After completing the creation of the endpoint, you next need to configure DNS resolution in your local environment. In total, you need to configure resolution for two domain names:

1. **DNS Resolution for the COS Metadata Acceleration Service**
- Domain name format: *.chdfs.<region>.myqcloud.com.
- Target address: the VIP address generated by Create COS Metadata Acceleration Service Endpoint.

2. **DNS Resolution for the Standard COS Object Storage Service**
- Format of the domain name: *.cos.<region>.myqcloud.com.
- Target address: the VIP address generated by Create COS Endpoint.

Note:
The local hosts file does not support wildcard DNS resolution. If you are configuring via hosts, replace the "*" in the wildcard domain name with the specific bucket-appid. Example: .
- 10.0.0.12 examplebucket-1250000000.chdfs.ap-beijing.myqcloud.com
- 10.0.0.13 examplebucket-1250000000.cos.ap-beijing.myqcloud.com

Verification and Testing

1. **Basic Connectivity Test**

Verify the port connectivity of the endpoint VIP using the telnet command:

```
telnet <VIP> 80     # Verify the HTTP port
telnet <VIP> 443    # Verify the HTTPS port
```

2. **Verification of Private Network Access**

On your compute node, use the nslookup command to resolve the COS domain name and the domain name for metadata acceleration. If it returns the VIP of the endpoint, then it indicates that domain name resolution is successful.

3. **Feature Test**

Refer to the content of Mount Metadata-Accelerated Bucket to access the metadata-accelerated bucket from the compute node and check whether the feature is functioning properly.

Mounting a COS Bucket in a Computing Cluster

Last updated：2024-11-21 09:54:14

Feature Overview

In COS, you can enable metadata acceleration to gain the HDFS protocol access capability. After metadata acceleration is enabled, COS generates a mount point for your bucket. Then you can download the HDFS client and use the mount point in the client to mount COS. This document introduces how to mount a bucket with metadata acceleration enabled to a computing cluster.

> **Note**
> - Starting from version 8.1.5, Hadoop-cos supports accessing metadata-accelerated buckets using the `cosn://bucketname-appid/` format.
> - The metadata acceleration feature can only be enabled during bucket creation and cannot be disabled once enabled. Therefore, **carefully consider** whether to enable it based on your business conditions. You should also note that legacy Hadoop-COS packages cannot access metadata acceleration buckets.

Preparations

- Java 1.8 available at Java Downloads has been installed on the server or container that needs to be mounted in the computing cluster.
- The computing cluster's machine or container to be mounted is authorized with the access permission. You need to specify the VPC and IP that can be accessed in the HDFS permission configuration.
- Descriptions of JAR dependency packages:
  - chdfs_hadoop_plugin_network-2.8.jar version ≥ 2.7.
  - cos_api-bundle.jar version ≥ 5.6.69.
  - Hadoop-cos version ≥ 8.1.5.
  - ofs-java-sdk.jar v1.0.4 or later, which can be automatically pulled without installation required. You can run `hadoop fs ls` to check whether the versions meet the requirements in the directory configured by `fs.cosn.trsf.fs.ofs.tmp.cache.dir`.

Instructions

1. Download the Hadoop client tool installation package from GitHub.

2. Download the POSIX Hadoop client tool installation package from GitHub.
3. Download the COS JAVA SDK installation package.
4. Place the installation package in the classpath of each node to ensure proper loading during task startup, such as in the `$HADOOP_HOME/share/hadoop/common/lib/` directory.

> **Note**
>
> In the EMR environment, the required dependency JAR packages are pre-installed, so you can directly access metadata-accelerated buckets using POSIX semantics without additional installation. If you need to access the buckets using the S3 protocol, modify the fs.cosn.posix_bucket.fs.impl configuration item. For more information, please refer to the following sections.

5. Edit the `core-site.xml` file by adding the following basic configuration items:

> **Note**
>
> - We recommend you use a sub-account key or temporary key instead of a permanent key to improve the business security. When authorizing a sub-account, follow the Notes on Principle of Least Privilege to avoid unexpected data leakage.
> - If you must use a permanent key, we recommend you follow the Notes on Principle of Least Privilege to limit the scope of permission, including allowed operations, scope of resources, and conditions such as access IP, on the permanent key so as to improve the usage security.

```
<!--API key information for your account. You can view the Cloud API
key by logging into the [Access Management Console]
(https://console.cloud.tencent.com/capi).-->
<!--We recommend using sub-account keys or temporary keys for
configuration to enhance security. When granting permissions to sub-
accounts, please follow the [principle of least privilege]
(https://cloud.tencent.com/document/product/436/38618).-->
<property>
         <name>fs.cosn.userinfo.secretId/secretKey</name>
         <value>AKIDxxxxxxxxxxxxxxxxxxxxx</value>
</property>

<!-- COSN implementation class -->
<property>
```

```
        <name>fs.AbstractFileSystem.cosn.impl</name>
        <value>org.apache.hadoop.fs.CosN</value>
</property>

<!-- COSN implementation class -->
<property>
        <name>fs.cosn.impl</name>
        <value>org.apache.hadoop.fs.CosFileSystem</value>
</property>

<!-- Bucket region in the format of `ap-guangzhou` -->
<property>
        <name>fs.cosn.bucket.region</name>
        <value>ap-guangzhou</value>
</property>

<!-- Local temporary directory, which is used to store temporary files
generated during execution. ->
<property>
        <name>fs.cosn.tmp.dir</name>
        <value>/tmp/hadoop_cos</value>
</property>
```

6. Sync `core-site.xml` to all `hadoop` nodes.

> **Note**
> For an EMR cluster, you only need to modify the HDFS configuration in component management in the EMR console for the above steps 3 and 4.

7. Run the `hadoop fs -ls cosn://${bucketname-appid}/` command on the `hadoop fs` command line, where `bucketname-appid` is the mount address, i.e., the bucket name. If the file list is displayed normally, the COS bucket has been successfully mounted.
8. Users can also use other `Hadoop` configuration options or run `MapReduce` tasks on COS buckets with metadata acceleration enabled. For `MapReduce` tasks, you can change the default input/output `FileSystem` for the task to the corresponding bucket by using `-Dfs.defaultFS=ofs://${bucketname-appid}/`.

Configuration Item Description

Note
You can access metadata acceleration buckets through POSIX semantics or S3 protocol. We recommend you use POSIX for better performance.

1. Required Common Configuration Items

Note
No matter how you access a metadata acceleration bucket, you must set the following common configuration items.

Configuration items	Content	Note
fs.cosn.userinfo.secretId/secretKey	A value in the format of `AKIDxxxxxxxxxxxxxxxxxxxxx`	Specifies the API key for your account. Log in to the CAM console to view the key.
fs.cosn.impl	org.apache.hadoop.fs.CosFileSystem	COSN implementation class for FileSystem; fixed as `org.apache.hadoop.fs.CosFileSystem` .
fs.AbstractFileSystem.cosn.impl	org.apache.hadoop.fs.CosN	The COSN implementation class for `AbstractFileSystem` , which is fixed at `org.apache.hadoop.fs.CosN` .
fs.cosn.bucket.region	A value in the format of `ap-beijing`	Please provide the region information for the bucket you want to access. For a list of region abbreviations, refer to Regions and Access Domain Names , such as ap-beijing, ap-guangzhou, etc. This is compatible with the original configuration: fs.cosn.userinfo.region.
fs.cosn.tmp.dir	`/tmp/hadoop_cos` by default	Set an existing local directory, where temporary files generated during execution will be placed. Meanwhile, be sure to configure sufficient space and permissions for this directory on each node.

2. Required Configuration for POSIX Access Method (Recommended)

Note
- In addition to the general configuration items, the POSIX access method requires the following configurations. To use the other optional configuration items for the POSIX access method with metadata-accelerated buckets, simply add the "fs.cosn.trsf." prefix.
- Note that the configuration items related to legacy Hadoop-COS are no longer applicable.

Configuration items	Content	Note
fs.cosn.trsf.fs.AbstractFileSystem.ofs.impl	com.qcloud.chdfs.fs.CHDFSDelegateFSAdapter	Implementation class for metadata acceleration bucket access.
fs.cosn.trsf.fs.ofs.impl	com.qcloud.chdfs.fs.CHDFSHadoopFileSystemAdapter	Implementation class for metadata acceleration bucket access.
fs.cosn.trsf.fs.ofs.tmp.cache.dir	A value in the format of `/data/emr/hdfs/tmp/posix-cosn/`	Set an existing local directory such as `"/data/emr/hdfs/tmp/posix-cosn/"`, where temporary files generated during execution will be placed. Meanwhile, be sure to configure sufficient space and permissions for this directory on each node.
fs.cosn.trsf.fs.ofs.user.appid	A value in the format of `12500000000`	Your `appid`, which is required.
fs.cosn.trsf.fs.ofs.bucket.region	A value in the format of `ap-beijing`	Your bucket region, which is required.

3. Required Configuration Items for S3 Protocol Access

The following configuration items are required for the S3 access mode. For other optional parameters, see Hadoop.

Configuration items	Content	Note
fs.cosn.posix_bucket.fs.impl	org.apache.hadoop.fs.CosNFileSystem	This parameter is fixed at `com.qcloud.chdfs.fs.CHDFSHadoopFileSystemAdapter` for the POSIX access mode (default mode) or `org.apache.hadoop.fs.CosNFileSystem` for the S3 access mode, respectively.

5. Points for Attention

1. You cannot use a legacy Hadoop-COS JAR package to access metadata acceleration buckets.
2. To access metadata acceleration buckets in POSIX mode of Hadoop-COS 8.1.5 or earlier, you need to disable ranger verification in the console.

Using HDFS to Access Metadata Acceleration-Enabled Bucket

Last updated：2023-09-13 16:14:54

Metadata Acceleration Overview

COS offers the metadata acceleration feature to provide high-performance file system capabilities. Metadata acceleration leverages the powerful metadata management feature of Cloud HDFS (CHDFS) at the underlying layer to allow using file system semantics for COS access. The designed system metrics can reach a bandwidth of up to 100 GB/s, over 100,000 queries per second (QPS), and a latency of milliseconds. Buckets with metadata acceleration enabled can be widely used in scenarios such as big data, high-performance computing, machine learning, and AI. For more information on metadata acceleration, see Metadata Acceleration Overview.

Benefits of HDFS Access

In the past, big data access based on COS was mainly implemented through the Hadoop-COS tool. The tool internally adapts Hadoop Compatible File System (HCFS) APIs to COS RESTful APIs to access data in COS. The difference in metadata organization between COS and file systems leads to varying metadata operation performance, which affects the big data analysis performance. Buckets with metadata acceleration enabled are fully compatible with the HCFS protocol and can be accessed directly by using native HDFS APIs. This saves the overheads of converting the HDFS protocol to COS protocol and provides native HDFS features such as efficient directory rename (as an atomic operation), file atime, and mtime update, efficient directory DU statistics, and POSIX ACL permission support.

Creating Bucket and Configuring the HDFS Protocol

1. Create a COS bucket and enable metadata acceleration.
   After the bucket is created, go to the **File List** page of the bucket, where you can perform file upload and download operations.
2. In the left sidebar, click **Performance Configuration > Metadata Acceleration Capability** to see that metadata acceleration is enabled. If you are creating a bucket with **metadata acceleration enabled** for the first time, follow the prompts to perform the necessary **authorization** operation.

   After clicking to authorize, the HDFS protocol will be automatically enabled, and you will see the default bucket mount point information.

> **Note**
> If the system indicates that the corresponding HDFS file system is not found, submit a ticket.

3. In the HDFS Permission Configuration section, click **Add Permission Configuration**.
4. Select the VPC network address where the compute cluster is located in the VPC Network Name column, enter the IP address or IP range that needs to be allowed in the Node IP Address column under the VPC subnet, choose the access type as Read/Write or Read-Only, and click **Save** to complete the configuration.

> **Note**
> **HDFS Permission Configuration** differs from the native COS permission system. When using the HDFS protocol for access, it is recommended to configure HDFS permissions to authorize specific machines within a VPC to access COS buckets, ensuring a consistent permission experience with native HDFS.

Configuring Computing Cluster to Access COS

Environment Dependencies

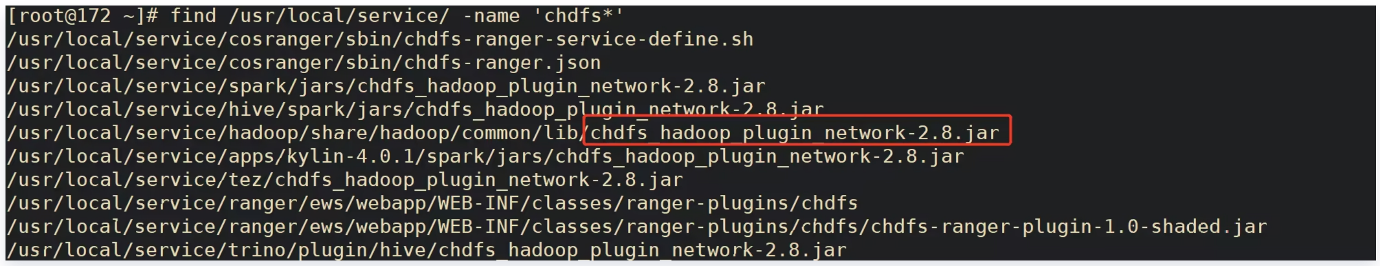
Dependency	chdfs-hadoop-plugin	COSN（hadoop-cos）	cos_api-bundle
Version	≥ v2.7	≥ v8.1.5	Make sure that the version matches the COSN version as listed in tencentyun/hadoop-cos.
Download address	GitHub	GitHub	GitHub

EMR environment

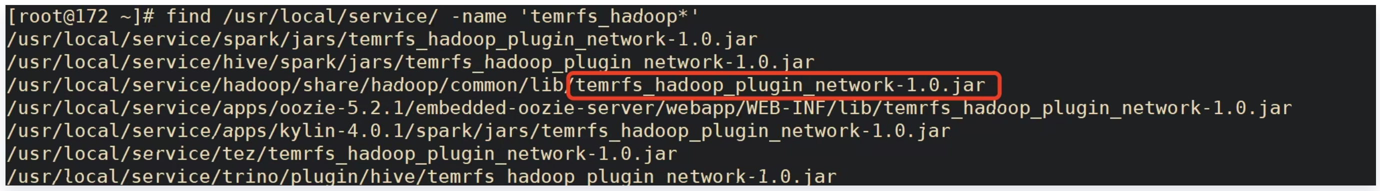
Tencent Cloud EMR environment has seamlessly integrated COS. You only need to complete the following steps:

1. Find an EMR server and run the following command on it to check whether the package versions in the folders of the services required by the EMR environment meet the requirements for environment dependencies.

```
find / -name "chdfs*"
find / -name "temrfs_hadoop*"
```



```
[root@172 ~]# find /usr/local/service/ -name 'chdfs*'
/usr/local/service/cosranger/sbin/chdfs-ranger-service-define.sh
/usr/local/service/cosranger/sbin/chdfs-ranger.json
/usr/local/service/spark/jars/chdfs_hadoop_plugin_network-2.8.jar
/usr/local/service/hive/spark/jars/chdfs_hadoop_plugin_network-2.8.jar
/usr/local/service/hadoop/share/hadoop/common/lib/chdfs_hadoop_plugin_network-2.8.jar
/usr/local/service/apps/kylin-4.0.1/spark/jars/chdfs_hadoop_plugin_network-2.8.jar
/usr/local/service/tez/chdfs_hadoop_plugin_network-2.8.jar
/usr/local/service/ranger/ews/webapp/WEB-INF/classes/ranger-plugins/chdfs
/usr/local/service/ranger/ews/webapp/WEB-INF/classes/ranger-plugins/chdfs/chdfs-ranger-plugin-1.0-shaded.jar
/usr/local/service/trino/plugin/hive/chdfs_hadoop_plugin_network-2.8.jar
```



```
[root@172 ~]# find /usr/local/service/ -name 'temrfs_hadoop*'
/usr/local/service/spark/jars/temrfs_hadoop_plugin_network-1.0.jar
/usr/local/service/hive/spark/jars/temrfs_hadoop_plugin_network-1.0.jar
/usr/local/service/hadoop/share/hadoop/common/lib/temrfs_hadoop_plugin_network-1.0.jar
/usr/local/service/apps/oozie-5.2.1/embedded-oozie-server/webapp/WEB-INF/lib/temrfs_hadoop_plugin_network-1.0.jar
/usr/local/service/apps/kylin-4.0.1/spark/jars/temrfs_hadoop_plugin_network-1.0.jar
/usr/local/service/tez/temrfs_hadoop_plugin_network-1.0.jar
/usr/local/service/trino/plugin/hive/temrfs_hadoop_plugin_network-1.0.jar
```

Make sure that the versions of two JAR packages in the search results meet the above requirements for environment dependencies.

2. If the chdfs-hadoop-plugin needs to be updated, perform the following steps:

   Download the script files of the updated JAR package at:
   - update_cos_jar.sh
   - update_cos_jar_common.sh

   Run the following command to put the two scripts in the `/root` directory of the server to add the execution permission to `update_cos_jar.sh`:

   ```
   sh update_cos_jar.sh  https://hadoop-jar-beijing-1259378398.cos.ap-beijing.myqcloud.com/hadoop_plugin_network/2.7
   ```

   Replace the parameter with the corresponding regional bucket, for example, if the region is Guangzhou, replace it with `https://hadoop-jar-guangzhou-1259378398.cos.ap-guangzhou.myqcloud.com/hadoop_plugin_network/2.7`.
   Perform the above steps on each EMR node until the .jar packages on all machines are replaced.

3. If the hadoop-cos package or cos_api-bundle package needs to be updated, perform the following steps:
   - Replace `/usr/local/service/hadoop/share/hadoop/common/lib/hadoop-temrfs-1.0.5.jar` with `temrfs_hadoop_plugin_network-1.1.jar`.
   - Add the following configuration items to `core-site.xml`:
     - emr.temrfs.download.md5=822c4378e0366a5cc26c23c88b604b11
     - emr.temrfs.download.version=2.7.5-8.1.5-1.0.6 (Replace `2.7.5` with your Hadoop version and replace `8.1.5` with the version of the required hadoop-cos

package (which must be 8.1.5 or later). The cos_api-bundle version will be automatically adapted.)
        - emr.temrfs.download.region=sh
        - emr.temrfs.tmp.cache.dir=/data/emr/hdfs/tmp/temrfs
    - Modify the configuration item `fs.cosn.impl=com.qcloud.emr.fs.TemrfsHadoopFileSystemAdapter` in `core-site.xml`.
4. In the EMR Console, configure core-site.xml and add the configuration items `fs.cosn.bucket.region` and `fs.cosn.trsf.fs.ofs.bucket.region`. These parameters are used to specify the COS region where the bucket is located, such as `ap-shanghai`.

> **Note**
> `fs.cosn.bucket.region` and `fs.cosn.trsf.fs.ofs.bucket.region` are required to specify the COS region where the bucket resides, such as `ap-shanghai`.

5. Restart resident services such as YARN, Hive, Presto, and Impala.

Self-built Hadoop/CDH environments

1. For self-built environments, download the three required JAR packages that meet the version requirements from the environment dependencies.
2. Place the installation packages in the `classpath` path of each server in the Hadoop cluster, such as `/usr/local/service/hadoop/share/hadoop/common/lib/` (which may vary by component).
3. Navigate to the `$HADOOP_HOME/etc/hadoop` directory and edit the hadoop-env.sh file. Add the following content to include the COSN-related JAR files in the Hadoop environment variables:

```
for f in $HADOOP_HOME/share/hadoop/tools/lib/*.jar; do
  if [ "$HADOOP_CLASSPATH" ]; then
    export HADOOP_CLASSPATH=$HADOOP_CLASSPATH:$f
  else
    export HADOOP_CLASSPATH=$f
  fi
done
```

4. Add the following configuration items to `core-site.xml` in the computing cluster:

```
<!-- COSN implementation class -->
<property>
```

```
        <name>fs.cosn.impl</name>
        <value>org.apache.hadoop.fs.CosFileSystem</value>
</property>

<!-- Bucket region in the format of `ap-guangzhou` -->
<property>
        <name>fs.cosn.bucket.region</name>
        <value>ap-guangzhou</value>
</property>

<!-- Bucket region in the format of `ap-guangzhou` -->
<property>
        <name>fs.cosn.trsf.fs.ofs.bucket.region</name>
        <value>ap-guangzhou</value>
</property>

<!-- Configure how to get `SecretId` and `SecretKey` -->
<property>
        <name>fs.cosn.credentials.provider</name>

<value>org.apache.hadoop.fs.auth.SimpleCredentialProvider</value>
</property>

<!--API key information for your account. You can view the Cloud API
key by logging into the [Access Management Console]
(https://console.cloud.tencent.com/capi).-->
<property>
        <name>fs.cosn.userinfo.secretId</name>
        <value>XXXXXXXXXXXXXXXXXXXXXXXX</value>
</property>

<!--API key information for your account. You can view the Cloud API
key by logging into the [Access Management Console]
(https://console.cloud.tencent.com/capi).-->
<property>
        <name>fs.cosn.userinfo.secretKey</name>
        <value>XXXXXXXXXXXXXXXXXXXXXXXX</value>
</property>

<!-- Configure the account's `appid` -->
```

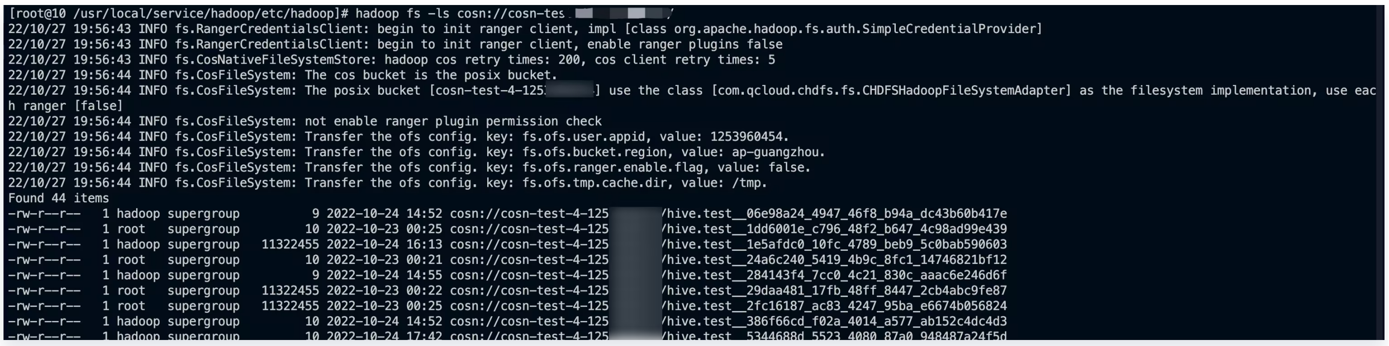
```
<property>
        <name>fs.cosn.trsf.fs.ofs.user.appid</name>
        <value>125XXXXXX</value>
</property>

<!-- Local temporary directory, which is used to store temporary files
generated during execution. -->
<property>
        <name>fs.cosn.trsf.fs.ofs.tmp.cache.dir</name>
        <value>/tmp</value>
</property>
```

5. Restart resident services such as YARN, Hive, Presto, and Impala.

Verifying environment

After all environment configuration steps are completed, you can verify the environment in the following ways:

- Verify the successful mounting on the client using the Hadoop command line, as shown in the following figure:



```
[root@10 /usr/local/service/hadoop/etc/hadoop]# hadoop fs -ls cosn://cosn-tes[redacted]/
22/10/27 19:56:43 INFO fs.RangerCredentialsClient: begin to init ranger client, impl [class org.apache.hadoop.fs.auth.SimpleCredentialProvider]
22/10/27 19:56:43 INFO fs.RangerCredentialsClient: begin to init ranger client, enable ranger plugins false
22/10/27 19:56:43 INFO fs.CosNativeFileSystemStore: hadoop cos retry times: 200, cos client retry times: 5
22/10/27 19:56:44 INFO fs.CosFileSystem: The cos bucket is the posix bucket.
22/10/27 19:56:44 INFO fs.CosFileSystem: The posix bucket [cosn-test-4-125[redacted]] use the class [com.qcloud.chdfs.fs.CHDFSHadoopFileSystemAdapter] as the filesystem implementation, use eac
h ranger [false]
22/10/27 19:56:44 INFO fs.CosFileSystem: not enable ranger plugin permission check
22/10/27 19:56:44 INFO fs.CosFileSystem: Transfer the ofs config. key: fs.ofs.user.appid, value: 1253960454.
22/10/27 19:56:44 INFO fs.CosFileSystem: Transfer the ofs config. key: fs.ofs.bucket.region, value: ap-guangzhou.
22/10/27 19:56:44 INFO fs.CosFileSystem: Transfer the ofs config. key: fs.ofs.ranger.enable.flag, value: false.
22/10/27 19:56:44 INFO fs.CosFileSystem: Transfer the ofs config. key: fs.ofs.tmp.cache.dir, value: /tmp.
Found 44 items
-rw-r--r--   1 hadoop supergroup          9 2022-10-24 14:52 cosn://cosn-test-4-125[redacted]/hive.test__06e98a24_4947_46f8_b94a_dc43b60b417e
-rw-r--r--   1 root   supergroup         10 2022-10-23 00:25 cosn://cosn-test-4-125/hive.test__1dd6001e_c796_48f2_b647_4c98ad99e439
-rw-r--r--   1 hadoop supergroup   11322455 2022-10-24 16:13 cosn://cosn-test-4-125/hive.test__1e5afdc0_10fc_4789_beb9_5c0bab590603
-rw-r--r--   1 root   supergroup         10 2022-10-23 00:21 cosn://cosn-test-4-125/hive.test__24a6c240_5419_4b9c_8fc1_14746821bf12
-rw-r--r--   1 hadoop supergroup          9 2022-10-24 14:55 cosn://cosn-test-4-125/hive.test__284143f4_7cc0_4c21_830c_aaac6e246d6f
-rw-r--r--   1 root   supergroup   11322455 2022-10-23 00:22 cosn://cosn-test-4-125/hive.test__29daa481_17fb_48ff_8447_2cb4abc9fe87
-rw-r--r--   1 root   supergroup   11322455 2022-10-23 00:25 cosn://cosn-test-4-125/hive.test__2fc16187_ac83_4247_95ba_e6674b056824
-rw-r--r--   1 hadoop supergroup         10 2022-10-24 14:52 cosn://cosn-test-4-125/hive.test__386f66cd_f02a_4014_a577_ab152c4dc4d3
-rw-r--r--   1 hadoop supergroup         10 2022-10-24 17:42 cosn://cosn-test-4-125/hive.test__5344688d_5523_4080_87a0_948487a24f5d
```

- Log in to the COS console to check whether the files and directories in the bucket file list are consistent.

Ranger Permission Configuration

By default, the native POSIX ACL mode is adopted for authentication for the HDFS protocol. If you need to use Ranger authentication, configure as follows:

EMR environment

1. The EMR environment has integrated the COS Ranger Service, so you can select it when purchasing an EMR cluster.
2. Under the **HDFS Authentication Mode** of the HDFS protocol, select **Ranger Authentication Mode** and configure the corresponding Ranger address information:

- Add the new configuration item `fs.cosn.credentials.provider` and set it to `org.apache.hadoop.fs.auth.RangerCredentialsProvider`.
- If you have any questions about Ranger, see COS Ranger Permission System Solution.

Self-built Hadoop/CDH environments

1. Configure the Ranger service to access COS over the HDFS protocol. For more information, see COS Ranger Permission System Solution.
2. Under the **HDFS Authentication Mode** of the HDFS protocol, select **Ranger Authentication Mode** and configure the corresponding Ranger address information:
   - Add the new configuration item `fs.cosn.credentials.provider` and set it to `org.apache.hadoop.fs.auth.RangerCredentialsProvider`.
   - If you have any questions about Ranger, see COS Ranger Permission System Solution.

Others

In a big data scenario, you can access a bucket with metadata acceleration enabled over the HDFS protocol in the following steps:

1. Configure the HDFS mount target information in `core-stie.xml` as instructed in Creating Bucket and Configuring HDFS Protocol.
2. Access the bucket using components such as Hive, MR, and Spark. For more information, see Mounting a COS Bucket in a Computing Cluster.
3. By default, native `POSIX ACL` is used for authentication. If you need to use `Ranger authentication`, please refer to the COS Ranger Permission System Solution.

GooseFS Release Notes

Last updated：2023-09-13 16:19:38

The GooseFS version update list is as follows. If you have any questions or suggestions, please feel free to contact us.

Version No.	Updated On	Changes	Download Link
1.4.2	May 1, 2023	• Bug fixes Fixed the issue where deadlock occurred after writing large files exhausted client resources.	• GooseFS: Click to Download • GooseFS (KonaJDK Version): Click to Download • GooseFS Client: Click to download • GooseFS Client (KonaJDK version): Click to download • GooseFS FUSE Client: Click to download • GooseFS FUSE Client (KonaJDK Version): Click to Download
1.4.1	March 1, 2023	• Feature Enhancement ○ Support for cleaning and viewing incomplete file lists ○ Optimized the locking granularity for the recursive metadata loading operation (loadmetadata -R). • Bug fixes	• GooseFS: Click to Download • GooseFS (KonaJDK Version): Click to Download • GooseFS Client: Click to download • GooseFS Client (KonaJDK

		○ Fixed the issue of unified authentication flow and data flow status in Flume write scenarios. ○ Fixed the issue where deadlock occurred after writing large files exhausted client resources.	version): Click to download • GooseFS FUSE Client: Click to download • GooseFS FUSE Client (KonaJDK Version): Click to Download
1.4.0	November 11, 2022	• New features: ○ File decompression feature provided (Beta, available only in Shanghai Auto-Driving Zone) ○ Supported the temporary key management feature. ○ Supported hierarchical traversal for `distributed Load` . ○ Supported downgraded read in GooseFS-FUSE. • Feature Enhancement ○ GooseFS distributedLoad supports hierarchical traversal capabilities, allowing recursive retrieval of metadata information from specified directories. ○ Improved the sequential read performance of GooseFS-FUSE for large files. ○ Added the percentile duration monitoring for master query and RocksDB update to improve the monitoring sensitivity of the metadata service.	• GooseFS: Click to Download • GooseFS (KonaJDK Version): Click to Download • GooseFS Client: Click to download • GooseFS Client (KonaJDK version): Click to download • GooseFS FUSE Client: Click to download • GooseFS FUSE Client (KonaJDK Version): Click to Download

		◦ Reduced the cluster recovery time of GooseFS in HA mode to improve the cluster availability. ◦ Updated the COSN dependency version. You can access buckets with metadata acceleration enabled over the native HDFS protocol, which improves the file operation performance in big data scenarios. ◦ Simplified and optimized GooseFS configuration by removing unnecessary configuration items. ◦ Simplified and optimized `listInfo` to improve the file listing performance. • Bug fixes ◦ Fixed the issue where the Worker received a large number of invalid async block requests. ◦ Optimized the processing logic of orphan blocks during worker reporting.	
1.3.0	July 25, 2022	• New features: ◦ Kerberos authentication is supported. ◦ Supports accessing buckets with metadata acceleration capabilities enabled, allowing native POSIX semantics for object storage services. ◦ Supported the LRU algorithm for the master node cache elimination	• GooseFS: Click to Download • GooseFS (KonaJDK Version): Click to Download • GooseFS Client: Click to download • GooseFS Client (KonaJDK version): Click to download

		policy to avoid frequent cache replacement. ○ Supported expired metadata cleanup for the master node. • Feature Enhancement ○ GooseFS Master node has optimized the lock bottleneck issue, significantly improving Master QPS and resulting in approximately 35% performance enhancement. ○ Supported concurrent formatting to improve the performance of GooseFS worker nodes. ○ Supported overwrite operations for the GooseFS-FUSE client. ○ GooseFS Fuse client has optimized the memory usage issue of the 'ls' command. ○ Optimized the performance of ListNamespace in the GooseFS HDFS client. • Bug fixes ○ Fixed RocksDB leak and Core issues, preventing memory leaks. ○ Fixed the issue where ZooKeeper Curator mistakenly printed logs. ○ Fixed the issue of inaccurate UFS bandwidth reading. ○ Fixed the LostWorker issue caused by excessive log printing during DistributedLoad.	• GooseFS FUSE Client: Click to download • GooseFS FUSE Client (KonaJDK Version): Click to Download

1.2.0	January 25, 2022	• New features: ◦ GooseFS supports reporting logs to the Cloud Log system. ◦ Supported separated configuration of RocksDB by using Inode and Block metadata. ◦ Added the hot switch capability in the GooseFS client to improve disaster recovery measures. ◦ Added the infrastructure for certain full-linkage logs. • Feature Enhancement ◦ Optimized the issue of high delay during Raft leader switch. ◦ Optimized the memory usage of the GooseFS HCFS client. • Bug fixes ◦ Fixed the issue of Journal disorder. ◦ Fixed the gRPC issue caused by Ratis deadlock. ◦ Fixed the incorrect HDFSUnderFileSystemFactory loading position. ◦ Fixed the Log4j2 vulnerabilities. ◦ Fixed ufsPath prefix check errors.	• GooseFS: Click to Download • GooseFS (KonaJDK Version): Click to Download • GooseFS FUSE Client: Click to download
1.1.0	September 1, 2021	• New Features: ◦ Supports Ranger authentication. ◦ Supported concurrent listing.	This version is no longer maintained. Download the latest version.

		◦ Supported DistributedLoad directory atomicity. ◦ Supported the namespace cache allowlist. ◦ Supported imperceptible OFS scheme acceleration. • Feature Enhancements: ◦ CLI adds help command for each subcmd. ◦ Optimized the logic to check for namespace existence during table mounting. ◦ Improved the monitoring of job workers and worker metrics. • Bug fixes: ◦ Fixed the issue of DistributedLoad read bloat.	
1.0.0	June 1, 2021	1. Namespace-based read/write policy and TTL management. 2. Prefetch at Hive table and table partition levels. 3. Compatibility with paths of existing COSN users to achieve imperceptible acceleration. 4. Fluid integration with GooseFS. 5. CHDFS support integration.	This version is no longer maintained. Download the latest version.

Key Features
Caching

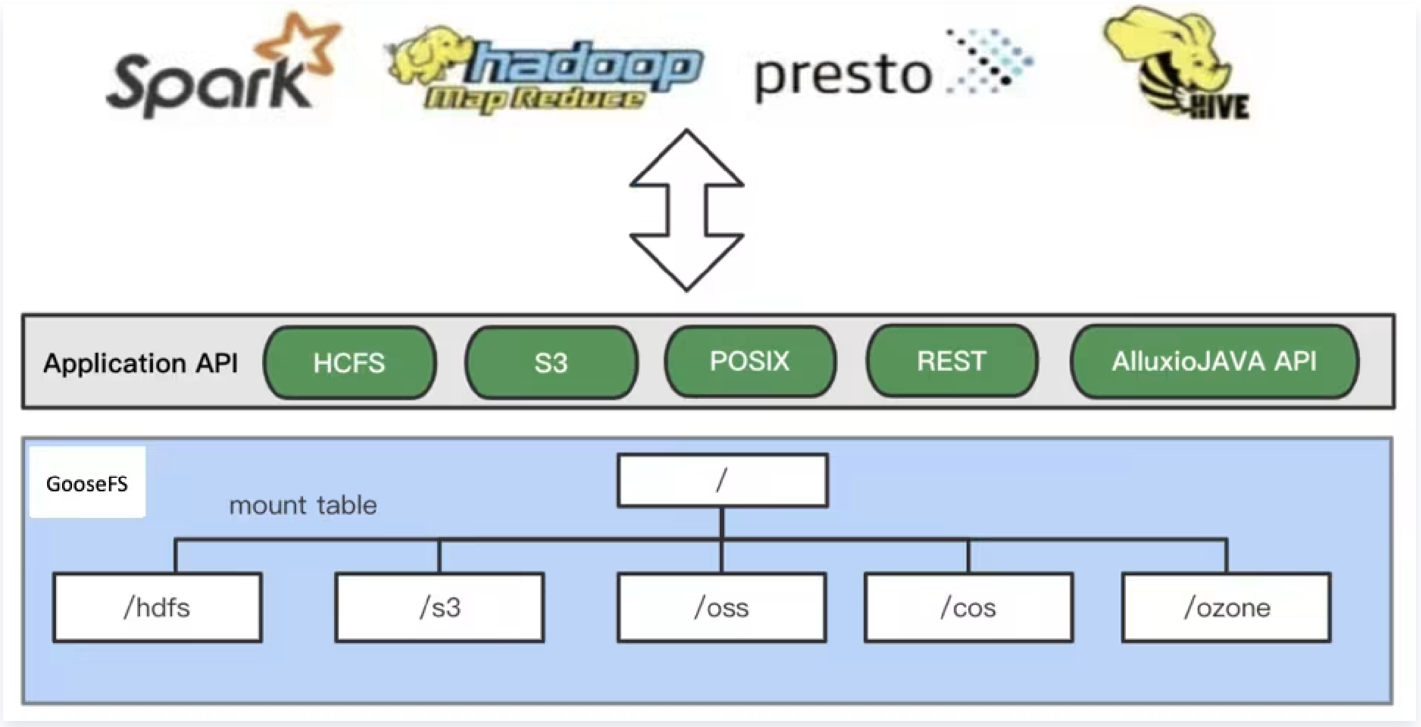
Last updated：2024-11-21 09:56:52

Overview

Based on the open-source Alluxio framework, GooseFS provides a powerful distributed cache ability to unify cross-platform data and improve the overall I/O throughput.
It can be implemented as follows:

- 1. GooseFS NameSpace remote storage: The NS storage space is based on external storage, spanning across COSN and HDFS. It can connect one or multiple NS to the same underlying storage, allowing for the persistence of massive data. By adopting the COSN storage-compute separation solution, GooseFS NameSpace can provide elastic expansion and high availability capabilities for big data storage.
- GooseFS local cache: GooseFS leverages the distributed cache ability of the master and workers to store the cached and temporary data generated at the application layer in the memory and disk of the local application node. Each local node is configurable. The remote persistent storage layer connected to namespaces will use the same client to serve users' reads/writes.

You can configure the cache policy to decide whether to use the local cache or remote storage for namespaces.



GooseFS supports local cache and remote storage for namespaces to provide an all-around storage separated solution and create data affinity for application nodes.

GooseFS Cache Configuration

Users can view the GooseFS cache configuration in the goosefs-site.properties file. Currently, GooseFS cache settings can be understood from aspects such as cache hierarchy and cache eviction policies. Cache hierarchy mainly includes single-tier and multi-tier storage, while cache eviction policies are primarily divided into LRU (Least Recently Used) and LRFU (Least Recently/Frequently Used). The following sections provide a detailed explanation.

1. Cache Layers

GooseFS provides single-level and multi-level caches. Single-level cache uses only one memory device, while multi-level cache uses various storage devices to meet the needs of different business loads and provide satisfactory I/O performance. By default, GooseFS uses single-level storage. If there are multiple storage devices, cache replacement will affect performance and thus **single-level storage** is recommended. You can choose a memory device such as MEM, SSD, or HDD according to your I/O requirements.
You can modify `levels` in the `goosefs-site.properties` configuration file to change the cache level, where `1` indicates using only one level of storage and `3` indicates 3 levels.

```
goosefs.worker.tieredstore.levels=1
```

You can also modify `alias` in `goosefs-site.properties` to change the storage device used for a cache level.

```
goosefs.worker.tieredstore.level{x}.alias=MEM
```

Note
"x" in level{x} is the index of the storage level. For example, `level0` indicates single-level storage.

1.1 Single-level storage

Common configurations for single-level storage are as follows:

```
goosefs.worker.tieredstore.levels=1
goosefs.worker.tieredstore.level0.alias=SDD
goosefs.worker.ramdisk.size=16GB
goosefs.worker.memory.size=100GB
goosefs.worker.tieredstore.level0.dirs.path=/data/GooseFSWorker,/data1/G
ooseFSWorker,/data2/GooseFSWorker,/data3/GooseFSWorker,/data4/GooseFSWor
ker
goosefs.worker.tieredstore.level0.dirs.mediumtype=MEM,MEM,MEM,SSD,SSD
goosefs.worker.tieredstore.level0.dirs.quota=16GB,16GB,16GB,100GB,100GB
```

Configuration items:

- ramdisk.size: memory size for workers. The value must be smaller than `memory`. GooseFS will allocate memory from the total memory for each worker according to the specified value.
- memory.size: total memory of the GooseFS system. The specified size of memory will be automatically used from the storage device. The value must be smaller than the size of the physical memory of the storage device.
- dirs.path: directories that GooseFS allocates storage devices for. This parameter must be used together with `dirs.mediumtype`. In the example above, `/data/GooseFSWorker` is attached to the MEM storage device, and `/data4/GooseFSWorker` is attached to an SSD. Note that the directory sequence must correspond to the storage device sequence.
- dirs.mediumtype: storage device used for the specified directories. This parameter must be used together with `dirs.path`. By default, `MEM` and `SSD` are valid values. If other storage devices such as HDD are attached, you can configure them as needed.
- dirs.quota: This configuration item is used to specify the pre-allocated space for each directory. After configuration, GooseFS allocates the space for the specified directory, which needs to correspond to the directory order. The example above demonstrates allocating 16GB MEM space for directories such as /data/GooseFSWorker, /data1/GooseFSWorker, /data2/GooseFSWorker, and 100GB SSD space for /data3/GooseFSWorker, /data4/GooseFSWorker directories.

1.2 Multi-level storage

Multi-level storage reads and writes data blocks differently from single-level storage. In single-level mode, data reads/writes use the same storage device. However, in multi-level mode, data is first written to the highest level of storage device. If any data needs to be read, it will be moved to the highest level of storage device. Details are described below.

- **Data writes**: New data blocks will be written to the highest level of storage device by default. If the highest level of storage device is used up, data will be moved to the next storage level. If storage capacities of all storage devices are used up, older data will be replaced according to the cache replacement policies you set. If expired data cannot be replaced and the available memory is used up, data cannot be written.
- **Data reads**: In multi-level storage mode, cold data will be moved to the lower-level of storage device imperceptibly. If the data is read again, it will be moved back to the highest

storage level.

Note

- GooseFS will clear a specified amount of data according to the cache replacement policies configured. The amount can be specified with `goosefs.worker.tieredstore.free.ahead.bytes`, and the default value is `0`.
- In multi-level storage mode, data reads may cause data movement from a lower storage level to the highest one and compromise performance. Therefore, **you are advised to use single-level storage** in most cases.

Common configuration items:

```
goosefs.worker.tieredstore.levels
goosefs.worker.tieredstore.level{x}.alias
goosefs.worker.tieredstore.level{x}.dirs.path
goosefs.worker.tieredstore.level{x}.dirs.mediumtype
goosefs.worker.tieredstore.level{x}.dirs.quota
```

In the example above, `x` indicates the index of the cache level. In most cases, you can use three levels (corresponding MEM, SSD, and HDD). The example below uses two levels (MEM and SSD) and allocates 100 GB for each level:

```
goosefs.worker.tieredstore.levels=2
goosefs.worker.tieredstore.level0.alias=MEM
goosefs.worker.tieredstore.level0.dirs.path=/data/GooseFSWorker
goosefs.worker.tieredstore.level0.dirs.mediumtype=MEM
goosefs.worker.tieredstore.level0.dirs.quota=100GB
goosefs.worker.tieredstore.level1.alias=SSD
goosefs.worker.tieredstore.level1.dirs.path=/data1/GooseFSWorker
goosefs.worker.tieredstore.level1.dirs.mediumtype=SSD
goosefs.worker.tieredstore.level1.dirs.quota=100GB
```

You can configure an unlimited number of storage levels with GooseFS, but their names (alias) must be unique. If you use three levels, it will be easy to distinguish them if you name them MEM, SSD, and HDD.

2. Cache Eviction Policy

GooseFS supports two cache replacement policies:

- LRUAnnotator: least-recently-used (LRU) (default)
- LRFUAnnotator: combines the least-recently-used (LRU) and least-frequently-used (LFU) schemes. You can set weights of LRU and LFU using `goosefs.worker.block.annotator.lrfu.step.factor` and `goosefs.worker.block.annotator.lrfu.attenuation.factor` to decide how these two policies should take effect together.
  - If the weight of least-recently-used is set the maximum value, it will function in the same way as `LRUAnnotator`.

You can use `goosefs.worker.block.annotator.class` to specify the cache replacement policies, where the policy names should be correctly set as follows:

```
goosefs.worker.block.annotator.LRUAnnotator
goosefs.worker.block.annotator.LRFUAnnotator
```

Data Lifecycle

The lifecycle described here is for data in GooseFS rather than data in the remote UFS. You can manage data lifecycle with the following four operations:

- free: deletes the specified directories or files from GooseFS (data in the UFS will be intact). This operation is mainly used to free GooseFS cache space for other hotter data.
- load: loads directories/files in the UFS to GooseFS. This operation is mainly used to restore cold data for higher I/O performance.
- persist: writes GooseFS data to the UFS to store data persistently and avoid data loss.
- Time to Live (TTL): sets the lifecycle for directories/files in GooseFS. If data has expired, it will be deleted from GooseFS and the UFS. You can also delete data only in GooseFS or only free the disk space.

1. Releasing data from GooseFS

Run the `free` command to free data from GooseFS. In the example below, the data status becomes 0% in GooseFS after the freeing operation.

```
$ goosefs fs free /data/test.txt
/data/test.txt was successfully freed from GooseFS space.
$ goosefs fs ls /data/test.txt
-rw-rw-rw-  hadoop         hadoop                      14
PERSISTED 03-11-2021 11:46:15:000 0% /data/test.txt
```

`/data/test.txt` in the example can be replaced with any valid GooseFS path. If data is stored in the remote UFS, you can run the `load` command to reload the data.

Note
You are advised to set cache replacement policies to automatically clear GooseFS historical data.

2. Loading Data into GooseFS

By using the load command, data can be loaded from GooseFS. The following example demonstrates the data status in GooseFS becoming 100% after releasing the data:

```
$ goosefs fs load /data/test.txt
/data/test.txt loaded
$ goosefs fs ls /data/test.txt
-rw-rw-rw-  hadoop         hadoop                       14
PERSISTED 03-11-2021 11:46:15:000 100% /data/test.txt
```

`/data/test.txt` in the example can be replaced with any valid GooseFS path. To load data from a local file system, use the `copyFromLocal` command. Note that data loaded from a local file system will not be stored to the remote UFS. By default, data will be cached to GooseFS at the first access. Therefore, you don’t need to run the `load` command in most cases. However, if you need your data loaded into the cache in advance, you can run this command.

3. Persisting data in GooseFS

Data can be persisted to the remote storage system UFS using the "persist" command:

```
$ goosefs fs persist /data/test.txt
$ goosefs fs ls /data/test.txt
-rw-rw-rw-  hadoop         hadoop                       14
PERSISTED 03-11-2021 11:46:15:000 100% /data/test.txt
```

`/data/test.txt` in the example above can be replaced with any valid GooseFS path. You are advised to configure cache replacement policies to automatically store data persistently.

4. Configuring Data TTL Attributes

GooseFS allows you to add a time to live (TTL) value for any directory or file in a namespace. TTL ensures that your data can be deleted as scheduled to free space for new files and make full use of the local disk. The TTL values of GooseFS files and directories can be the metadata so that these TTL values will still take effect even if the cluster is restarted. The checker

program in the background will check TTL values periodically and clear expired data automatically.
The checker’s interval and action can be set in `goosefs-site.preperties`. In the example below, the checker’s interval is set to 10 minutes, and the action is set to `FREE`.

```
goosefs.master.ttl.checker.interval=10m
goosefs.user.file.create.ttl.action=FREE
```

After the configuration, GooseFS background threads will inspect expired files every 10 minutes. If any expired file is found during an inspection cycle, the file’s cache will be freed when the next cycle completes. For example, if a file is found expired in the cycle that ends at 00:00:00, the file’s cache will be cleared at 00:10:00.

Note
The default unit of the interval is ms (millisecond). You can use a unit such as s (second), m (minute), or h (hour) when you set the input parameter. For more information, please see the configuration description.

Data Replication

Data replication is classified into active replication and passive replication.

1. Passive Replication

In GooseFS, each file has one or more data blocks spread across the cluster. By default, GooseFS automatically adjusts the number of replicas for data blocks according to the load and capacity. Passive replication is performed when:

- When multiple clients read the same block simultaneously, it results in the same block being present on different workers at the same time.
- When local reads are prioritized, but the data is not found from the local environment, remote reads will be initialized, and the data will be saved to the local worker.
- If the number of replicas generated by passive replication is greater than the configured quantity, the excess will be deleted asynchronously, which is unperceivable to users.

2. Active Replication

Active replication is implemented by setting the number of replicas for a file. If the number of blocks is smaller than the configured quantity, blocks will be made up asynchronously. If there are excessive blocks, excessive replicas will be deleted. You can run the following command to configure active replication:

```
$ goosefs fs setReplication  [-R] [--max | --min ] <path>
```

Parameters are described as follows:

- max: the maximum number of replicas (default value: `-1`, indicating no upper limit). If this parameter is set to `0`, the file will not be cached in GooseFS. Usually, you can set this parameter to a positive integer. After the configuration, GooseFS will check the replica quantity and delete the excessive ones.
- min: the minimum number of replicas (default value: `0`, indicating no replica will be retained when the data expires). Usually, you can set this parameter to a positive integer, which must be **smaller than** `max`. After the configuration, GooseFS will check the replica quantity and make up the quantity automatically if the replica quantity is smaller than the configured value.
- path: a directory or a file path
- R: recursively replicates all files and sub-directories of a directory (if `path` is set to a directory) following `min` and `max`.

You can run the `stat` command to view the replication of a file. In the example below, `replicationMax` of the `/data/test.txt` file is set to `-1`, which means that the file will be deleted once it expires.

```
$ goosefs fs stat /data/test.txt
/data/test.txt is a file path.
FileInfo{fileId=50331647, fileIdentifier=null, name=test.txt,
path=/data/test.txt, ufsPath=hdfs://172.16.16.16:4007/data/test.txt,
length=0, blockSizeBytes=134217728, creationTimeMs=1618193473555,
completed=true, folder=false, pinned=false, pinnedlocation=[],
cacheable=true, persisted=true, blockIds=[], inMemoryPercentage=100,
lastModificationTimesMs=1616763603692, ttl=-1,
lastAccessTimesMs=1616763603692, ttlAction=DELETE, owner=hadoop,
group=supergroup, mode=420, persistenceState=PERSISTED,
mountPoint=false, replicationMax=-1, replicationMin=0, fileBlockInfos=
[], mountId=1, inGooseFSPercentage=100, ufsFingerprint=TYPE|FILE
UFS|hdfs OWNER|hadoop GROUP|supergroup MODE|420 CONTENT_HASH|
(len:0,_modtime:1616763603692) , acl=user::rw-,group::r--,other::r--,
defaultAcl=}
This file does not contain any blocks.
```

Querying Cache Usage

GooseFS will record the local cache and storage usage. You can run the commands below to view the running status of GooseFS for local cache management and maintenance.

- View the cache usage, in bytes:

```
$ goosefs fs getUsedBytes
Used Bytes: 0
```

- View the total cache capacity, in bytes:

```
$ goosefs fs getCapacityBytes
Capacity Bytes: 1610612736000
```

- View the cache usage report:

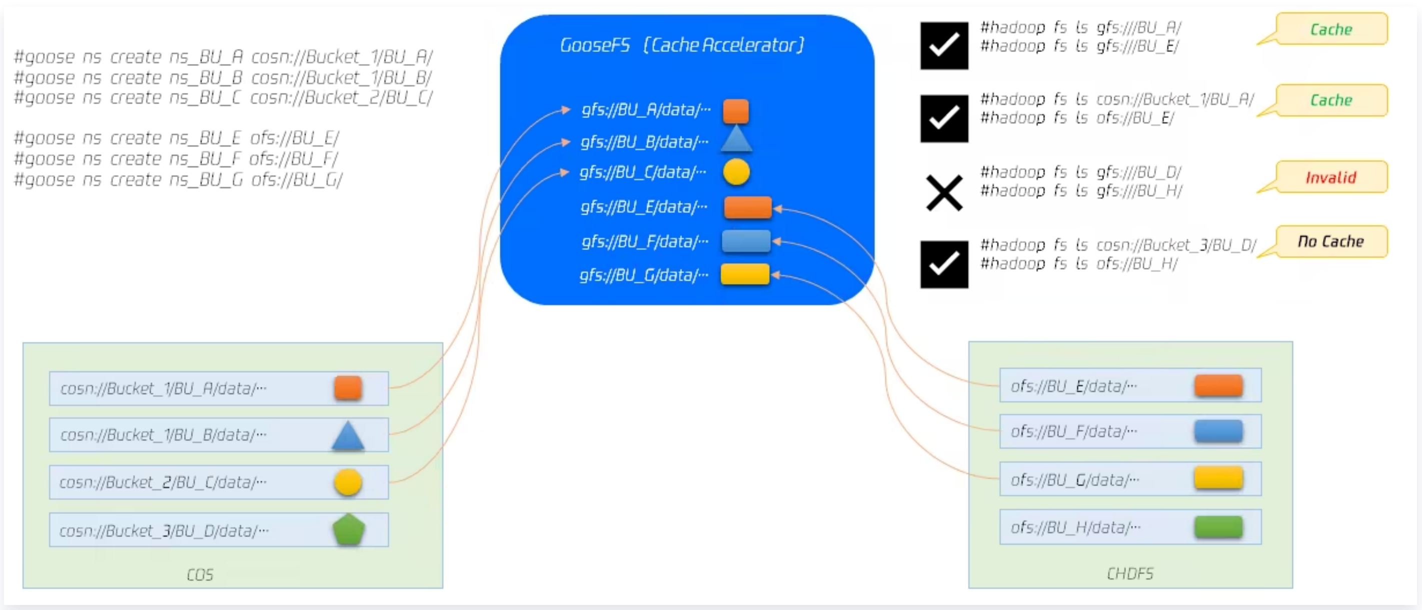
```
$ goosefs fsadmin report
GooseFS cluster summary:
    Master Address: 172.16.16.16:19998
    Web Port: 19999
    Rpc Port: 19998
    Started: 04-12-2021 10:52:05:255
    Uptime: 0 day(s), 1 hour(s), 28 minute(s), and 57 second(s)
    Version: 2.5.0-SNAPSHOT
    Safe Mode: false
    Zookeeper Enabled: false
    Live Workers: 3
    Lost Workers: 0
    Total Capacity: 1500.00GB
        Tier: HDD  Size: 1500.00GB
    Used Capacity: 0B
        Tier: HDD  Size: 0B
    Free Capacity: 1500.00GB
```

Unified Namespace Capability

Last updated：2023-09-13 17:07:24

Overview

The unified namespace capability of GooseFS integrates the access semantics of different underlying storage systems through a transparent naming mechanism, providing users with a unified interactive view for data management.
Leveraging this capability, GooseFS connects and communicates with different underlying storage systems, such as local file systems, Tencent Cloud Object Storage (COS), and Tencent Cloud HDFS (CHDFS), and provides unified access APIs and file protocols for upper-layer businesses. In this way, the business side only needs to call the access APIs provided by GooseFS to access data stored in different underlying storage systems.



```
#goose ns create ns_BU_A cosn://Bucket_1/BU_A/
#goose ns create ns_BU_B cosn://Bucket_1/BU_B/
#goose ns create ns_BU_C cosn://Bucket_2/BU_C/

#goose ns create ns_BU_E ofs://BU_E/
#goose ns create ns_BU_F ofs://BU_F/
#goose ns create ns_BU_G ofs://BU_G/
```

```
#hadoop fs ls gfs:///BU_A/
#hadoop fs ls gfs:///BU_E/
```

```
#hadoop fs ls cosn://Bucket_1/BU_A/
#hadoop fs ls ofs://BU_E/
```

```
#hadoop fs ls gfs:///BU_D/
#hadoop fs ls gfs:///BU_H/
```

```
#hadoop fs ls cosn://Bucket_3/BU_D/
#hadoop fs ls ofs://BU_H/
```

The figure above shows the working principle of the unified namespace. You can use GooseFS's namespace creation instruction `create ns` to mount specified file directories of COS and CHDFS to GooseFS, and then use the `gfs://` unified schema to access data.
Details are as follows:

- COS contains 3 buckets in total: bucket-1, bucket-2, and bucket-3. bucket-1 contains the BU_A and BU_B directories, and both bucket-1 and bucket-2 are mounted to GooseFS.
- CHDFS contains 4 directories: BU_E, BU_F, BU_G, and BU_H. All of them are mounted to GooseFS, except BU_H.
- Among GooseFS file operations, the BU_A and BU_E directories can be accessed using the unified schema `gfs://`, and the files are cached in the local file system of GooseFS.
- The BU_A and BU_E directories stored in the underlying file systems (COS and HDFS) have been mounted to GooseFS. If files are cached on GooseFS, they can be accessed

through the unified schema `gfs://` (for example, `hadoop fs ls gfs://BU_A`), and they can also be accessed through the namespace of each remote file system (for example, `hadoop fs ls cosn://bucket-1/BU_A`).

- If the files are not cached on GooseFS, they cannot be accessed through the unified schema `gfs://` because the files are not cached in the local file system of GooseFS, but they can still be accessed through the namespace of the underlying storage systems.

Using the Unified Namespace Capability

You can use the `create ns` instruction to create a namespace in GooseFS and map underlying storage systems to GooseFS. Currently supported underlying storage systems include COS, CHDFS, and local HDFS. The procedure for creating a namespace is similar to that for mounting a file volume disk in a Linux file system. With the namespace created, GooseFS can provide clients with a file system with uniform access semantics. The current operation instruction set for GooseFS namespaces is as follows:

Note

- We recommend that you avoid using permanent keys in the configuration. Configuring sub-account keys or temporary keys can help improve your business security. When authorizing a sub-account, grant only the permissions of the operations and resources that the sub-account needs, which helps avoid unexpected data leakage.
- If you must use a permanent key, we recommend that you limit ITS permission scope. This can improve the security by limiting its executable operations, resource scope, and conditions (access IP, etc.).

```
$ goosefs ns
Usage: goosefs ns [generic options]
         [create <namespace> <CosN/Chdfs path> <--wPolicy <1-6>> <--
rPolicy <1-5>> [--readonly] [--shared] [--secret
fs.cosn.userinfo.secretId=<AKIDxxxxxxx>] [--secret
fs.cosn.userinfo.secretKey=<xxxxxxxxxx>] [--attribute
fs.ofs.userinfo.appid=1200000000][--attribute fs.cosn.bucket.region=<ap-
xxx>/fs.cosn.bucket.endpoint_suffix=<cos.ap-xxx.myqcloud.com>]]
         [delete <namespace>]
         [help [<command>]]
         [ls [-r|--sort=option|--timestamp=option]]
         [setPolicy [--wPolicy <1-6>] [--rPolicy <1-5>] <namespace>]
         [setTtl [--action delete|free] <namespace> <time to live>]
```

```
        [stat <namespace>]
        [unsetPolicy <namespace>]
        [unsetTtl <namespace>]
```

The commands are described as follows:

Instruction	Note
create	Creates a namespace and maps a remote storage system (UFS) to the namespace. When creating the namespace, you can set cache read and write policies. You need to pass in an authorized key (`secretId` and `secretKey`).
delete	Deletes a specified namespace.
ls	Lists the detailed information of a specified namespace, including the UFS path, creation time, cache policy, and TTL information.
setPolicy	Sets the cache policy of a specified namespace.
setTtl	Sets TTL for a specified namespace.
stat	Provides the description of a specified namespace, including the mount point, UFS path, creation time, cache policy, TTL information, persistence status, user group, ACL, last access time, and modification time.
unsetPolicy	Resets the cache policy of a specified namespace.
unsetTtl	Resets the TTL of a specified namespace.

Creating or Deleting Namespaces

By creating a namespace in GooseFS, you can cache frequently accessed hot data from a remote storage system to a local high-performance storage node to provide high-performance data access for local computing businesses. The following shows how to map the COS bucket `example-bucket` , the `example-prefix` directory in the bucket, and the CHDFS to the `test_cos` , `test_cos_prefix` , and `test_chdfs` namespaces respectively.

```
# Map COS bucket example-bucket to the test_cos namespace
$ goosefs ns create test_cos cosn://example-bucket-1250000000/ --wPolicy
1 --rPolicy 1 --secret fs.cosn.userinfo.secretId=AKIDxxxxxxx --secret
```

```
fs.cosn.userinfo.secretKey=xxxxxxxxxx --attribute
fs.cosn.bucket.region=ap-guangzhou --attribute
fs.cosn.bucket.endpoint_suffix=cos.ap-guangzhou.myqcloud.com

# Map the example-prefix directory of the COS bucket example-bucket to
the test_cos_prefix namespace
$ goosefs ns create test_cos_prefix cosn://example-bucket-
1250000000/example-prefix/ --wPolicy 1 --rPolicy 1 --secret
fs.cosn.userinfo.secretId=AKIDxxxxxxx --secret
fs.cosn.userinfo.secretKey=xxxxxxxxxx --attribute
fs.cosn.bucket.region=ap-guangzhou --attribute
fs.cosn.bucket.endpoint_suffix=cos.ap-guangzhou.myqcloud.com

# Map CHDFS file system f4ma0l3qabc-Xy3 to the test_chdfs namespace
$ goosefs ns create test_chdfs ofs://f4ma0l3qabc-Xy3/ --wPolicy 1 --
rPolicy 1 --attribute fs.ofs.userinfo.appid=1250000000
```

After successful creation, you can use the `goosefs fs ls` instruction to view the directory details:

```
$ goosefs fs ls /test_cos
```

You can use the `delete` instruction to delete unwanted namespaces:

```
$ goosefs ns delete test_cos
Delete the namespace: test_cos
```

Setting the cache policy

You can use `setPolicy` and `unsetPolicy` to set the cache policy of a namespace. The instruction set is as follows:

```
$goosefs ns setPolicy [--wPolicy <1-6>] [--rPolicy <1-5>] <namespace>
```

The parameters are described as follows:

- wPolicy: write cache policy. Up to 6 write cache policies are supported.
- rPolicy: read cache policy. Up to 5 read cache policies are supported.
- namespace: specified namespace

The read and write cache policies currently supported by GooseFS are as follows:

Write cache policies

Policy Name	Behavior	Corresponding Write_Type	Data Security	Write Efficiency
MUST_CACHE (1)	Data is stored only in GooseFS and is not written to the remote storage system.	MUST_CACHE	Unreliable	Height
TRY_CACHE (2)	If the cache has space, data is written to GooseFS. Otherwise, data is written directly to underlying storage systems.	TRY_CACHE	Unreliable	Medium
CACHE_THROUGH (3)	Data is cached as much as possible and simultaneously written to remote storage systems.	CACHE_THROUGH	Reliable	Low
THROUGH (4)	Data is not stored in GooseFS, but written directly to the remote storage system.	THROUGH	Reliable	Medium
ASYNC_THROUGH (5)	Data is written to GooseFS and asynchronously purged to remote storage systems.	ASYNC_THROUGH	Weak reliability	Height

Note

- `Write_Type` indicates the file cache policy specified when the user calls the SDK or API to write data to GooseFS. It takes effect only for a single file.
- If you want to change the configured write cache policy, you need to carefully evaluate the importance of the cached data. If the data is important, we recommend that you ensure that the cached data has been persisted; otherwise, it may be lost. For example, after the write cache is changed from `MUST_CACHE` to `CACHE_THROUGH`, if the `persist` command is not called to persist the data, the data that is about to be eliminated cannot be written to the underlying layer and will be lost.

Read cache policies

Policy Name	Behavior	Metadata Sync	Corresponding Read_Type	Data Consistency	Read Efficiency	Whether to Cache Data
NO_CACHE (1)	Data is not cached and is directly read from remote storage systems instead.	NO	NO_CACHE	Strong consistency	Low	Not required
CACHE (2)	• Metadata access behavior: If the cache is hit, the metadata in the Master will be used, and metadata will not be actively synchronized from the	Once	CACHE	Weak	Hit: High Not hit: Low	Required

	underlying storage. • Data access behavior: The ReadType of the data stream adopts the CACHE policy.					
CACHE_PROMOTE (3)	• Metadata access behavior: Same as CACHE mode. • Data access behavior: The ReadType of the data stream	Once	CACHE_PROMOTE	Weak	Hit: High Not hit: Low	Required

	adopts the CACHE_PROMOTE strategy.					
CACHE_CONSISTENT_PROMOTE (4)	• Metadata behavior: Before each read operation, the metadata on the remote storage system UFS is synchronized. If the data does not exist in UFS, a "Not Exists" exception is thrown.	Always	CACHE	Strong consistency	Hit: Medium Not hit: Low	Required

	• Data access behavior: The ReadType of the data stream adopts the CACHE_PROMOTE strategy. Upon a cache hit, the data is cached in the hottest storage medium.					
CACHE_CONSISTENT (5)	• Metadata behavior: Same as CACHE_CO	Always	CACHE_PROMOTE	Strong consistency	Hit: Medium Not hit: Low	Required

		NSISTENT_PROMOTE. • Data access behavior: The ReadType of the data stream adopts the CACHE strategy, meaning that when the CACHE is hit, data will not be moved between different storage layers.				

Note

Read_Type: Refers to the file caching policy specified by the user when calling SDK or API to read data from GooseFS, which is effective for individual files.

Based on current big data business practices, we recommend the following combinations of read and write cache policies:

Write Cache Policy	Read Cache Policy	Policy Combination Performance
CACHE_THROUGH（3）	CACHE_CONSISTENT（5）	Strong data consistency between the cache and remote storage systems
CACHE_THROUGH（3）	CACHE（2）	Write: strong consistency; read: eventual consistency
ASYNC_THROUGH（5）	CACHE_CONSISTENT（5）	Write: eventual consistency; read: strong consistency
ASYNC_THROUGH（5）	CACHE（2）	Read/Write: eventual consistency
MUST_CACHE（1）	CACHE（2）	Data is read from the cache only.

The following example shows how to set the read and write cache policies of the `test_cos` namespace to `CACHE_THROUGH` and CACHE_CONSISTENT` respectively:

```
$ goosefs ns setPolicy --wPolicy 3 --rPolicy 5 test_cos
```

Note
In addition to specifying cache policies when creating namespaces, you can also configure global cache policies by setting `Read_Type` or `Write_Type` for specific files when reading or writing files, or by using the `Properties` configuration file. If multiple policies exist at the same time, their priority order is as follows: custom priority > namespace read and write policies > global cache policy configured in the configuration file. For the read policy, the combination of the custom `Read_Type` and the namespace's `DirReadPolicy` takes effect. That is, the custom `Read_Type` is used as the data stream read policy, and the namespace policy is used for metadata.
For example, GooseFS contains a COSN namespace whose read policy is `CACHE_CONSISTENT` and the namespace contains a `test.txt` file. When the client reads the `test.`

> `txt` file, `Read_Type` is specified as `CACHE_PROMOTE` . Then the entire read behavior is to sync metadata and perform `CACHE_PROMOTE` .

To reset the read and write cache policies, you can use the `unsetPolicy` instruction. The following shows how to reset the read and write cache policies for the `test_cos` namespace:

```
$ goosefs ns unsetPolicy test_cos
```

Setting TTL

Time to Live (TTL) is used to manage data cached on the local nodes of GooseFS. Setting TTL allows a specified operation, such as `delete` or `free` , to be performed on the cached data after a specified period of time. The instruction for setting TTL is as follows:

```
$ goosefs ns setTtl [--action delete|free] <namespace> <time to live>
```

The parameters are described as follows:

- action: operation performed after the cache duration expires. Currently, `delete` and `free` are supported. The `delete` operation deletes data from the cache and UFS, while the `free` operation deletes data only from the cache.
- namespace: specified namespace
- time to live: data cache duration, in milliseconds

The following example shows how to set the policy of the `test_cos` namespace to delete data only from the cache after 60 seconds:

```
$ goosefs ns setTtl --action free test_cos 60000
```

Metadata management

This section describes how GooseFS manages metadata, including metadata synchronization and updates. GooseFS provides users with unified namespace capability. Users can access files on different underlying storage systems using a unified `gfs://` path. You only need to specify the paths of the underlying storage systems. We recommend that you use GooseFS as a unified data access layer to uniformly read and write data from GooseFS to ensure metadata consistency.

Metadata synchronization overview

You can configure the metadata synchronization interval in the `conf/goosefs-site.properties` configuration file:

```
goosefs.user.file.metadata.sync.interval=<INTERVAL>
```

The metadata synchronization interval parameter supports 3 types of input values:

- -1: metadata is not updated after it is first loaded into GooseFS.
- 0: GooseFS updates metadata after each read/write operation.
- Positive integer: GooseFS periodically updates metadata at a specified interval.

You can choose an appropriate synchronization interval based on your number of nodes, the I/O distance between your GooseFS cluster and the underlying storage system, and the type of the underlying storage system. Usually:

- The greater the number of nodes in a GooseFS cluster, the greater the metadata synchronization delay.
- The greater the distance between the GooseFS cluster and the underlying storage system, the greater the metadata synchronization delay.
- The impact of the underlying storage system on the metadata synchronization delay depends on the QPS load. The higher the QPS load, the less the synchronization delay.

Metadata synchronization management

Configuration methods

1. Configuring via Command Line
   You can set the metadata information synchronization period using the command line:

   ```
   goosefs fs ls -R -Dgoosefs.user.file.metadata.sync.interval=0 <path to
   sync>
   ```

2. Using Configuration File
   For large-scale GooseFS clusters, you can configure the metadata synchronization period for Master nodes in the cluster in bulk through the goosefs-site.properties configuration file. The synchronization period for other nodes will be executed according to the default value of this period.

   ```
   goosefs.user.file.metadata.sync.interval=1m
   ```

   Note
   Many businesses choose to differentiate data usage by directory, and the access frequency of data in different directories may vary. Metadata synchronization intervals can be set to different values for different directories. For frequently

changing directories, the metadata synchronization interval can be set to a shorter time (e.g., 5 minutes). For rarely changing or static directories, the synchronization interval can be set to −1, so GooseFS will not automatically synchronize the metadata of that directory.

Recommended configuration

You can set different metadata synchronization intervals based on business access modes:

Access Mode		Metadata Synchronization Interval	Note
All file requests go through GooseFS		−1	–
Most file requests go through GooseFS	HDFS is used as UFS	Hot update or update by path is recommended	If the HDFS updates frequently, you are advised to set the update interval to `-1` to prohibit updates.
	Using COS as UFS	Configuring update intervals by path is recommended	Configuring different update intervals for different directories can alleviate the pressure of metadata synchronization.
File upload requests generally do not go through GooseFS	HDFS is used as UFS	Configuring update intervals by path is recommended	
	Using COS as UFS	Configuring update intervals by path is recommended	

Operations Manual
Monitoring Guide
Getting GooseFS Monitoring Metrics

Last updated：2023-09-13 16:27:42

GooseFS records monitoring data based on the Coda Hale Metrics Library. It allows you to obtain metric data in different ways such as the CLI, console, and files. GooseFS currently supports the following metric obtaining methods:

- MetricsServlet: monitoring metrics are provided to users in JSON format.
- CsvSink: monitoring metrics are provided in CSV files. After configuration, monitoring metrics are periodically generated in CSV files.
- PrometheusMetricsServlet: monitoring metrics are provided to users in the format defined by Prometheus.

The configuration of the aforementioned monitoring metrics can be specified through a configuration file. The default file path for GooseFS monitoring metrics configuration is `$ GooseFS_HOME/conf/metrics.properties`. You can specify a custom monitoring configuration file using goosefs.metrics.conf.file. GooseFS provides a default template, metrics.properties.template, which includes all configurable properties.

Getting Monitoring Metrics

The following describes three basic methods for obtaining monitoring metrics:

1. Fetch monitoring metrics in JSON format

GooseFS adopts the method of pulling monitoring metrics in JSON format by default, which corresponds to the MetricsServlet configuration item. You can use the CLI to initiate an HTTP request to the leading master node of GooseFS to pull all desired monitoring metrics. The metrics port is 9201 for the master node and 9204 for a worker node. The request format is as follows:

```
$ curl <LEADING_MASTER_HOSTNAME>:<MASTER_WEB_PORT>/metrics/json
```

In the example above, <LEADING_MASTER_HOSTNAME> must be a valid IP of the MASTER node, and <MASTER_WEB_PORT> must be an enabled port.
To get the monitoring metrics of a specific worker node, use the following command:

```
$ curl <WORKER_HOSTNAME>:<WORKER_WEB_PORT>/metrics/json
```

2. Obtain monitoring metrics via CSV files

GooseFS allows you to export data to CSV files. You can use this capability to get monitoring metrics. First, you need to prepare a directory to store monitoring metrics:

```
$ mkdir /tmp/goosefs-metrics
```

After the storage path is prepared, modify the `conf/metrics.properties` configuration file to enable the CsvSink capability:

```
sink.csv.class=goosefs.metrics.sink.CsvSink # Enable the CsvSink
capability

sink.csv.period=1 # Set the monitoring metric export period
sink.csv.unit=senconds # Set the unit of the monitoring metric export
period

sink.csv.directory=/tmp/goosefs-metrics # Set the monitoring metric
export path
```

After the configuration, you need to restart the node for the configuration to take effect. After the configuration takes effect, the monitoring metrics will be periodically exported to CSV files and stored to the specified path.

Note
- GooseFS has prepared a monitoring configuration template, which can be found in the conf/metrics.properties.template file.
- If GooseFS is deployed on a cluster, ensure that the specified metric storage path can be read by all nodes.

3. Retrieve Prometheus monitoring metrics

You can run the following commands to view Prometheus monitoring metrics on the GooseFS master node (metrics port: 9201) and worker node (metrics port: 9204):

```
curl <LEADING_MASTER_HOSTNAME>:<MASTER_WEB_PORT>/metrics/prometheus/
```

```
# HELP Master_CreateFileOps Generated from Dropwizard metric import
(metric=Master.CreateFileOps, type=com.codahale.metrics.Counter)
...

curl <WORKER_IP>:<WOKER_PORT>/metrics/prometheus/
# HELP pools_Code_Cache_max Generated from Dropwizard metric import
(metric=pools.Code-Cache.max,
type=com.codahale.metrics.jvm.MemoryUsageGaugeSet$$Lambda$51/137460818)
...
```

Monitoring GooseFS Based on Prometheus

Last updated：2025-07-11 15:27:17

GooseFS can output metric data to various monitoring systems, with Prometheus being one of them. Prometheus is an open-source monitoring framework, and Tencent Cloud's Observability Platform has already integrated it. This article will focus on GooseFS monitoring metrics and the process of reporting these metrics to both self-built Prometheus and cloud-based Prometheus instances.

Prerequisites

Before setting up the Prometheus monitoring system, you need to:

- Configure a GooseFS cluster.
- Download the Prometheus installation package (official or Tencent Cloud version).
- Download and configure Grafana.

Enabling GooseFS Metric Report

1. Edit the GooseFS configuration file conf/goosefs-site.properties, add the following configuration items, use goosefs copyDir conf/ to copy to all worker nodes, and restart the cluster with `./bin/goosefs-start.sh all`.

```
goosefs.user.metrics.collection.enabled=true
goosefs.user.metrics.heartbeat.interval=10s
```

2. You can run the following command to view the Prometheus master metrics (port: 9201) and worker metrics (port: 9204):

```
curl <LEADING_MASTER_HOSTNAME>:<MASTER_WEB_PORT>/metrics/prometheus/
# HELP Master_CreateFileOps Generated from Dropwizard metric import
(metric=Master.CreateFileOps, type=com.codahale.metrics.Counter)
...

curl <WORKER_IP>:<WOKER_PORT>/metrics/prometheus/
# HELP pools_Code_Cache_max Generated from Dropwizard metric import
(metric=pools.Code-Cache.max,
type=com.codahale.metrics.jvm.MemoryUsageGaugeSet$$Lambda$51/137460818)
```

```
...
```

Reporting Metrics to Self-Built Prometheus

1. Download the Prometheus installation package, extract it, and modify the prometheus.yml file:

```
# prometheus.yml
global:
 scrape_interval:     10s
 evaluation_interval: 10s
scrape_configs:
 - job_name: 'goosefs masters'
     metrics_path: /metrics/prometheus
     file_sd_configs:
     - refresh_interval: 1m
     files:
     - "targets/cluster/masters/*.yml"
 - job_name: 'goosefs workers'
     metrics_path: /metrics/prometheus
     file_sd_configs:
     - refresh_interval: 1m
     files:
     - "targets/cluster/workers/*.yml"
```

2. Create `targets/cluster/masters/masters.yml` and add the IP and port of the master:

```
 - targets:
    - "<TARGERTS_MASTER_IP>:<TARGERTS_MASTER_PORT>"
```

3. Create `targets/cluster/workers/workers.yml` and add the IP and port of the worker:

```
 - targets:
    - "<TARGERTS_WORKER_IP>:<TARGERTS_WORKER_PORT>"
```

4. Start Prometheus. `--web.listen-address` specifies the Prometheus listen address. The default port number is 9090.

```
nohup ./prometheus --config.file=prometheus.yml --web.listen-address="
<LISTEN_IP>:<LISTEN_PORT>" > prometheus.log 2>&1 &
```

5. View the GUI:

```
http://<PROMETHEUS_BI_IP>:<PROMETHEUS_BI_PORT>
```

6. View the target instance:

```
http://<PROMETHEUS_BI_IP>:<PROMETHEUS_BI_PORT>/targets
```

Reporting Metrics to Tencent Cloud Prometheus

1. Install the Prometheus agent on the master as instructed in the installation guide.

```
wget https://rig-1258344699.cos.ap-guangzhou.myqcloud.com/prometheus-
agent/agent_install && chmod +x agent_install && ./agent_install prom-
12kqy0mw agent-grt164ii ap-guangzhou <secret_id> <secret_key>
```

2. Configure jobs for the master and worker:

Method 1:

```
job_name: goosefs-masters
honor_timestamps: true
metrics_path: /metrics/prometheus
scheme: http
file_sd_configs:
- files:
  - /usr/local/services/prometheus/targets/cluster/masters/*.yml
  refresh_interval: 1m
job_name: goosefs-workers
honor_timestamps: true
metrics_path: /metrics/prometheus
scheme: http
file_sd_configs:
 - files:
  - /usr/local/services/prometheus/targets/cluster/workers/*.yml
  refresh_interval: 1m
```

Note
The job_name in the cluster does not contain spaces, while the job_name in a standalone Prometheus can include spaces.

Method 2:

```
job_name: goosefs masters
honor_timestamps: true
metrics_path: /metrics/prometheus
scheme: http
static_configs:
- targets:
 - "<TARGERTS_MASTER_IP>:<TARGERTS_MASTER_PORT>"
 refresh_interval: 1m

job_name: goosefs workers
honor_timestamps: true
metrics_path: /metrics/prometheus
scheme: http
static_configs:
- targets:
 - "<TARGERTS_WORKER_IP>:<TARGERTS_WORKER_PORT>"
 refresh_interval: 1m
```

Note
If you configure the scrape task using Method 2, there is no need to create masters.yml and workers.yml files under the targets/cluster/masters/ path.

Using Grafana to View Metrics

1. Start Grafana.

```
nohup ./bin/grafana-server web > grafana.log 2>&1 &
```

2. Open the login page at http://<GRAFANA_IP>:<GRAFANA_PORT>. Grafana's default port is 3000, and both the username and password are "admin". You can change the password

after the first login.

3. Create a Prometheus data source.

```
<PROMETHEUS_IP>:<PROMETHEUS_PORT>
```

4. Import Grafana dashboards for GooseFS. Select JSON import (Download JSON) and use it for the data source created above.

> **Note**
> When purchasing cloud-based Prometheus, you need to set a password. The configuration of the cloud-based Grafana visualization monitoring interface is similar to the above, but make sure the job_name is consistent.

5. You can export a modified dashboard.