ALIBABA CLOUD

Alibaba Cloud

E-MapReduce

JindoFS

Document Version: 20201027

Alibaba Cloud

Legal disclaimer

Alibaba Cloud reminds you to carefully read and fully understand the terms and conditions of this legal disclaimer before you read or use this document. If you have read or used this document, it shall be deemed as your total acceptance of this legal disclaimer.

1. You shall download and obtain this document from the Alibaba Cloud website or other Alibaba Cloud-authorized channels, and use this document for your own legal business activities only. The content of this document is considered confidential information of Alibaba Cloud. You shall strictly abide by the confidentiality obligations. No part of this document shall be disclosed or provided to any third party for use without the prior written consent of Alibaba Cloud.

2. No part of this document shall be excerpted, translated, reproduced, transmitted, or disseminated by any organization, company or individual in any form or by any means without the prior written consent of Alibaba Cloud.

3. The content of this document may be changed because of product version upgrade, adjustment, or other reasons. Alibaba Cloud reserves the right to modify the content of this document without notice and an updated version of this document will be released through Alibaba Cloud-authorized channels from time to time. You should pay attention to the version changes of this document as they occur and download and obtain the most up-to-date version of this document from Alibaba Cloud-authorized channels.

4. This document serves only as a reference guide for your use of Alibaba Cloud products and services. Alibaba Cloud provides this document based on the "status quo", "being defective", and "existing functions" of its products and services. Alibaba Cloud makes every effort to provide relevant operational guidance based on existing technologies. However, Alibaba Cloud hereby makes a clear statement that it in no way guarantees the accuracy, integrity, applicability, and reliability of the content of this document, either explicitly or implicitly. Alibaba Cloud shall not take legal responsibility for any errors or lost profits incurred by any organization, company, or individual arising from download, use, or trust in this document. Alibaba Cloud shall not, under any circumstances, take responsibility for any indirect, consequential, punitive, contingent, special, or punitive damages, including lost profits arising from the use or trust in this document (even if Alibaba Cloud has been notified of the possibility of such a loss).

5. By law, all the contents in Alibaba Cloud documents, including but not limited to pictures, architecture design, page layout, and text description, are intellectual property of Alibaba Cloud and/or its affiliates. This intellectual property includes, but is not limited to, trademark rights, patent rights, copyrights, and trade secrets. No part of this document shall be used, modified, reproduced, publicly transmitted, changed, disseminated, distributed, or published without the prior written consent of Alibaba Cloud and/or its affiliates. The names owned by Alibaba Cloud shall not be used, published, or reproduced for marketing, advertising, promotion, or other purposes without the prior written consent of Alibaba Cloud. The names owned by Alibaba Cloud include, but are not limited to, "Alibaba Cloud", "Aliyun", "HiChina", and other brands of Alibaba Cloud and/or its affiliates, which appear separately or in combination, as well as the auxiliary signs and patterns of the preceding brands, or anything similar to the company names, trade names, trademarks, product or service names, domain names, patterns, logos, marks, signs, or special descriptions that third parties identify as Alibaba Cloud and/or its affiliates.

6. Please directly contact Alibaba Cloud for any errors of this document.

Document conventions

Style	Description	Example
Danger	A danger notice indicates a situation that will cause major system changes, faults, physical injuries, and other adverse results.	**Danger:** Resetting will result in the loss of user configuration data.
Warning	A warning notice indicates a situation that may cause major system changes, faults, physical injuries, and other adverse results.	**Warning:** Restarting will cause business interruption. About 10 minutes are required to restart an instance.
Notice	A caution notice indicates warning information, supplementary instructions, and other content that the user must understand.	**Notice:** If the weight is set to 0, the server no longer receives new requests.
Note	A note indicates supplemental instructions, best practices, tips, and other content.	**Note:** You can use Ctrl + A to select all files.
>	Closing angle brackets are used to indicate a multi-level menu cascade.	Click **Settings**> **Network**> **Set network type**.
Bold	Bold formatting is used for buttons , menus, page names, and other UI elements.	Click **OK**.
`Courier font`	Courier font is used for commands	Run the `cd /d C:/window` command to enter the Windows system folder.
Italic	Italic formatting is used for parameters and variables.	`bae log list --instanceid` *Instance_ID*
[] or [a|b]	This format is used for an optional value, where only one item can be selected.	`ipconfig [-all|-t]`
{} or {a|b}	This format is used for a required value, where only one item can be selected.	`switch {active|stand}`

Table of Contents

1.Get started with JindoFS (earlier than EMR-3.27.0)

1.1. Use JindoFS in E-MapReduce V3.20.0 to V3.22.0 (V3.22.0 excluded)

This topic describes how to configure and use JindoFileSystem (JindoFS), and its scenarios.

Overview

JindoFS is a cloud-native file system that combines the advantages of Object Storage Service (OSS) and local storage. JindoFS is also the next-generation storage system that provides efficient and reliable storage services for cloud computing in E-MapReduce.

JindoFS supports the block storage mode and cache mode.

JindoFS adopts a heterogeneous multi-backup mechanism. Storage Service provides data storage capability. Data is stored in OSS to guarantee high reliability. Redundant backups are stored in the local cluster to accelerate read operations. Namespace Service manages metadata of JindoFS. In this case, metadata is queried from Namespace Service instead of OSS, which improves query performance. This query method of JindoFS is the same as that of Hadoop Distributed File System (HDFS).

> **Note**
>
> - E-MapReduce V3.20.0 and later support JindoFS. To use JindoFS, select the related services when you create an E-MapReduce cluster.
> - This topic describes how to use JindoFS in E-MapReduce V3.20.0 to V3.22.0 (V3.22.0 excluded). For more information about how to use JindoFS in E-MapReduce V3.22.0 or later, see Use JindoFS in E-MapReduce V3.22.0 or later.

Scenarios

E-MapReduce has three storage systems: E-MapReduce OssFileSystem, E-MapReduce HDFS, and E-MapReduce JindoFS. Among them, OssFileSystem and JindoFS store data in the cloud. The following table compares the features of three E-MapReduce storage systems and Hadoop support for Alibaba Cloud OSS.

Feature	Hadoop support for Alibaba Cloud OSS	E-MapReduce OssFileSystem	E-MapReduce HDFS	E-MapReduce JindoFS
Storage capacity	Tremendous	Tremendous	Depends on the E-MapReduce cluster scale	Tremendous
Reliability	High	High	High	High

Feature	Hadoop support for Alibaba Cloud OSS	E-MapReduce OssFileSystem	E-MapReduce HDFS	E-MapReduce JindoFS
Factor that affects throughput	Server	I/O performance of caches on disks in the E-MapReduce cluster	I/O performance of disks in the E-MapReduce cluster	I/O performance of disks in the E-MapReduce cluster
Metadata query efficiency	Low	Medium	High	High
Scale-out operation	Easy	Easy	Easy	Easy
Scale-in operation	Easy	Easy	Requires node decommission	Easy
Data locality	None	Weak	Strong	Medium

The block storage mode of JindoFS has the following features:

- JindoFS offers tremendous and scalable storage capacity by using OSS as the storage back end. The storage capacity is independent of the E-MapReduce cluster scale. The local cluster can be scaled in or out as required.
- JindoFS stores a certain amount of backup data in the local cluster to accelerate read operations. This improves the throughput by using limited local storage capacity, especially for Write Once Read Many (WORM) solutions.
- JindoFS provides efficient metadata query similar to HDFS. Compared with OssFileSystem, JindoFS saves much time in metadata query. In addition, JindoFS avoids system instability when data and metadata are frequently accessed.
- JindoFS moves computation as close as possible to data. This reduces the load on network transmission and improves the read performance.

Prepare the environment

- Create an E-MapReduce cluster

  Select a version from E-MapReduce V3.20.0 to V3.22.0 (V3.22.0 excluded). Select **SmartData** and **Bigboot** for Optional Services. For more information about how to create an E-MapReduce cluster, see Create a cluster. Bigboot provides distributed data management and component management services in E-MapReduce. Based on Bigboot, SmartData provides JindoFS for the application layer.

- Configure JindoFS

  JindoFS provided by SmartData uses OSS as the storage back end. Therefore, you need to set OSS-related parameters before using JindoFS. E-MapReduce provides two configuration methods. If you use the first configuration method, you need to create an E-MapReduce cluster, modify Bigboot-related parameters, and then restart SmartData for the configuration to take effect. If you use the second configuration method, you need to add custom configuration when you create an E-MapReduce cluster. In this case, the related services are restarted based on custom parameters after the E-MapReduce cluster is created.

- Initialize parameters after the E-MapReduce cluster is created

  You can set all JindoFS-related parameters in Bigboot. As shown in the following figure, the parameters framed in red are required. The `oss.access.bucket` parameter specifies the name of the OSS bucket. The `oss.data-dir` parameter specifies the directory of JindoFS in the OSS bucket. The directory only serves as the storage back end for JindoFS. The data generated in the directory cannot be damaged. The directory is automatically created when JindoFS writes data. You are not required to create the directory in advance. The `oss.access.endpoint` parameter specifies the region of the OSS bucket. The `oss.access.key` and `oss.access.secret` parameters specify the AccessKey ID and AccessKey secret used to access the OSS bucket, respectively. We recommend that you select an OSS bucket in the same region and under the same account as the storage back end of the E-MapReduce cluster for better performance and stability. In this case, the E-MapReduce cluster can access the OSS bucket without using the AccessKey ID and AccessKey secret.

  > **Note**
  > - The parameters framed in red in the preceding figure are required.
  > - JindoFS supports multiple namespaces. A namespace named test is used in this topic.

  Save and deploy the JindoFS configuration. Restart all components in SmartData to use JindoFS.

- Add custom configuration when creating an E-MapReduce cluster

  You can add custom configuration when creating an E-MapReduce cluster. For example, you want to create an E-MapReduce cluster in the same region as an OSS bucket to access OSS without using an AccessKey. As shown in the following figure, turn on **Custom Software Settings**. Add the following configuration including `oss.data-dir` and `oss.access.bucket` to the field in the Advanced Settings section:

  ```
  [
    {
    "ServiceName":"BIGBOOT",
    "FileName":"bigboot",
    "ConfigKey":"oss.data-dir",
    "ConfigValue":"jindoFS-1"
    },
    {
    "ServiceName":"BIGBOOT",
    "FileName":"bigboot",
    "ConfigKey":"oss.access.bucket",
    "ConfigValue":"oss-bucket-name"
    }
  ]
  ```

Use JindoFS

The use of JindoFS is similar to that of HDFS. JindoFS also provides a prefix. To use JindoFS, you only need to replace the hdfs prefix with the jfs prefix. Example:

```
hadoop fs -ls jfs:/// hadoop fs -mkdir jfs:///test-dirhadoop fs -put test.log jfs:///test-dir/
```

Data can be read from JindoFS only when Hadoop, Hive, and Spark jobs are running in the E-MapReduce cluster.

Control disk space usage

The back end of JindoFS is based on OSS that is capable of storing large amounts of data. However, the storage capacity of local disks is limited. Therefore, JindoFS releases data backups that are less frequently accessed. Alibaba Cloud uses node.data-dirs.watermark.high.ratio and node.data-dirs.watermark.low.ratio to adjust the space usage of local disks. The values of both parameters are in the range of 0 to 1 to indicate the percentage of space usage. JindoFS uses the total storage capacity of all data disks by default. The node.data-dirs.watermark.high.ratio parameter specifies the upper limit of space usage on each disk. Less frequently accessed data stored on a disk is released if the space used by JindoFS reaches the upper limit. The node.data-dirs.watermark.low.ratio parameter specifies the lower limit of space usage on each disk. After the space usage of a disk reaches the upper limit, less frequently accessed data is released until the space usage of the disk reaches the lower limit. You can set the upper limit and lower limit to adjust and assign disk space to JindoFS. Make sure that the upper limit is greater than the lower limit.

Configure the storage policy

JindoFS provides multiple storage policies to meet different storage needs. The following table lists four available storage policies for a directory.

Policy	Description
COLD	Data has only a backup in OSS but no backups in the local cluster. This policy is suitable for storing cold data.
WARM	The default storage policy. Data has a backup in OSS and a backup in the local cluster. The local backup can accelerate read operations.
HOT	Data has a backup in OSS and multiple backups in the local cluster. Local backups can accelerate read operations on hot data.
TEMP	Data has only a backup in the local cluster. This policy is suitable for storing temporary data. The local backup can accelerate read and write operations on the temporary data. However, this may lower data reliability.

JindoFS provides a command-line tool Admin to configure the storage policy of a directory. The default storage policy is WARM. New files are stored according to the storage policy configured for the parent directory. Run the following command to configure the storage policy:

```
jindo dfsadmin -R -setStoragePolicy [path] [policy]
```

Run the following command to obtain the storage policy configured for a directory:

```
jindo dfsadmin -getStoragePolicy [path]
```

> **Note** The *[path]* parameter specifies the directory. The *-R* option specifies that a recursive operation is performed to configure the same storage policy for all subdirectories of the directory.

The Admin tool provides the archive command to archive cold data.

This command allows you to explicitly evict local blocks. Assume that Hive partitions a table by the day. If the data generated a week ago in partitioned tables is infrequently accessed, you can regularly run the archive command on the directory that stores such data. Then, the backups stored in the local cluster are evicted, whereas the backups in OSS are retained.

Run the following archive command:

```
jindo dfsadmin -archive [path]
```

> **Note** The *[path]* parameter specifies the directory in which the data is to be archived.

1.2. Use JindoFS in E-MapReduce V3.22.0 or later

JindoFileSystem (JindoFS) is a cloud-native file system that combines the advantages of Object Storage Service (OSS) and local storage. JindoFS is also the next-generation storage system that provides efficient and reliable storage services for cloud computing in E-MapReduce. This topic describes how to configure and use JindoFS, and its scenarios.

Overview

JindoFS supports the block storage mode and cache mode.

JindoFS adopts a heterogeneous multi-backup mechanism. Storage Service provides data storage capability. Data is stored in OSS to guarantee high reliability. Redundant backups are stored in the local cluster to accelerate read operations. Namespace Service manages metadata of JindoFS. In this case, metadata is queried from Namespace Service instead of OSS, which improves query performance. This query method of JindoFS is the same as that of Hadoop Distributed File System (HDFS).

> **Note**
>
> - E-MapReduce V3.20.0 and later support JindoFS. To use JindoFS, select the related services when you create an E-MapReduce cluster.
> - This topic describes how to use JindoFS in E-MapReduce V3.22.0 or later. For more information about how to use JindoFS in E-MapReduce V3.20.0 to V3.22.0 (V3.22.0 excluded), see Use JindoFS in E-MapReduce V3.20.0 to V3.22.0 (V3.22.0 excluded).

Prepare the environment

- Create an E-MapReduce cluster

  Select E-MapReduce V3.22.0 or later. Select **SmartData** for Optional Services. For more information about how to create an E-MapReduce cluster, see Create a cluster.

- Configure JindoFS

  JindoFS provided by SmartData uses OSS as the storage back end. Therefore, you need to set OSS-related parameters before using JindoFS. E-MapReduce provides two configuration methods. If you use the first configuration method, you need to create an E-MapReduce cluster, modify Bigboot-related parameters, and then restart SmartData for the configuration to take effect. If you use the second configuration method, you need to add custom configuration when you create an E-MapReduce cluster. In this case, the related services are restarted based on custom parameters after the E-MapReduce cluster is created.

  - Initialize parameters after the E-MapReduce cluster is created

    You can set all JindoFS-related parameters in Bigboot, as shown in the following figures.

    a. In the **Service Configuration** section, click **bigboot**.

    b. Click **Custom Configuration**.

    > **Note**
    > - The parameters framed in red in the preceding figures are required.
    > - JindoFS supports multiple namespaces. A namespace named test is used in this topic.

Parameter	Description	Example
jfs.namespaces	The namespace supported by JindoFS. Separate multiple namespaces with commas (,).	test
jfs.namespaces.test.uri	The storage back end of the test namespace.	*oss://oss-bucket/oss-dir* **Note** You can set the value to a directory in an OSS bucket. In this case, this directory serves as the root directory, in which the test namespace reads and writes data.

Parameter	Description	Example
jfs.namespaces.test.mode	The storage mode of the test namespace.	block **Note** JindoFS supports the block storage mode and cache mode.
jfs.namespaces.test.oss.access.key	The AccessKey ID used to access the OSS bucket that serves as the storage back end.	*xxxx* **Note** We recommend that you select an OSS bucket in the same region and under the same account as the storage back end of the E-MapReduce cluster for better performance and stability. In this case, the E-MapReduce cluster can access the OSS bucket without using the AccessKey ID and AccessKey secret.
jfs.namespaces.test.oss.access.secret	The AccessKey secret used to access the OSS bucket that serves as the storage back end.	

Save and deploy the JindoFS configuration. Restart all components in SmartData to use JindoFS.

- Add custom configuration when creating an E-MapReduce cluster

  You can add custom configuration when creating an E-MapReduce cluster. For example, you want to create an E-MapReduce cluster in the same region as an OSS bucket to access OSS without using an AccessKey. As shown in the following figure, turn on **Custom Software Settings**. Add the following configuration to the field in the Advanced Settings section to customize parameters for the test namespace:

```
[
{
"ServiceName":"BIGBOOT",
"FileName":"bigboot",
"ConfigKey":"jfs.namespaces","ConfigValue":"test"
},{
"ServiceName":"BIGBOOT",
"FileName":"bigboot",
"ConfigKey":"jfs.namespaces.test.uri",
"ConfigValue":"oss://oss-bucket/oss-dir"
},{
"ServiceName":"BIGBOOT",
"FileName":"bigboot",
"ConfigKey":"jfs.namespaces.test.mode",
"ConfigValue":"block"
}
]
```

Use JindoFS

The use of JindoFS is similar to that of HDFS. JindoFS also provides a prefix. To use JindoFS, you only need to replace the hdfs prefix with the jfs prefix.

Currently, JindoFS supports most of the computing components in the E-MapReduce cluster, including Hadoop, Hive, Spark, Flink, Presto, and Impala.

Examples:

- Run shell commands

```
hadoop fs -ls jfs://your-namespace/
hadoop fs -mkdir jfs://your-namespace/test-dir
hadoop fs -put test.log jfs://your-namespace/test-dir/
hadoop fs -get jfs://your-namespace/test-dir/test.log . /
```

- Run a MapReduce job

```
hadoop jar /usr/lib/hadoop-current/share/hadoop/mapreduce/hadoop-mapreduce-examples-2.8.5.jar ter
agen -Dmapred.map.tasks=1000 10737418240 jfs://your-namespace/terasort/input
hadoop jar /usr/lib/hadoop-current/share/hadoop/mapreduce/hadoop-mapreduce-examples-2.8.5.jar ter
asort -Dmapred.reduce.tasks=1000 jfs://your-namespace/terasort/input jfs://your-namespace/terasort/o
utput
```

- Run a Spark SQL test

```
CREATE EXTERNAL TABLE IF NOT EXISTS src_jfs (key INT, value STRING) location 'jfs://your-namespace/Spark
_sql_test/';
```

Control disk space usage

The back end of JindoFS is based on OSS that is capable of storing large amounts of data. However, the storage capacity of local disks is limited. Therefore, JindoFS releases data backups that are less frequently accessed. Alibaba Cloud uses node.data-dirs.watermark.high.ratio and node.data-dirs.watermark.low.ratio to adjust the space usage of local disks. The values of both parameters are in the range of 0 to 1 to indicate the percentage of space usage. JindoFS uses the total storage capacity of all data disks by default. The node.data-dirs.watermark.high.ratio parameter specifies the upper limit of space usage on each disk. Less frequently accessed data stored on a disk is released if the space used by JindoFS reaches the upper limit. The node.data-dirs.watermark.low.ratio parameter specifies the lower limit of space usage on each disk. After the space usage of a disk reaches the upper limit, less frequently accessed data is released until the space usage of the disk reaches the lower limit. You can set the upper limit and lower limit to adjust and assign disk space to JindoFS. Make sure that the upper limit is greater than the lower limit.

Configure the storage policy

JindoFS provides multiple storage policies to meet different storage needs. The following table lists four available storage policies for a directory.

Policy	Description
COLD	Data has only a backup in OSS but no backups in the local cluster. This policy is suitable for storing cold data.
WARM	The default storage policy. Data has a backup in OSS and a backup in the local cluster. The local backup can accelerate read operations.
HOT	Data has a backup in OSS and multiple backups in the local cluster. Local backups can accelerate read operations on hot data.
TEMP	Data has only a backup in the local cluster. This policy is suitable for storing temporary data. The local backup can accelerate read and write operations on the temporary data. However, this may lower data reliability.

JindoFS provides a command-line tool Admin to configure the storage policy of a directory. The default storage policy is WARM. New files are stored according to the storage policy configured for the parent directory. Run the following command to configure the storage policy:

```
jindo dfsadmin -R -setStoragePolicy [path] [policy]
```

Run the following command to obtain the storage policy configured for a directory:

```
jindo dfsadmin -getStoragePolicy [path]
```

> **Note** The *[path]* parameter specifies the directory. The *-R* option specifies that a recursive operation is performed to configure the same storage policy for all subdirectories of the directory.

Use the Admin tool

JindoFS supports the archive and jindo commands of the Admin tool.

- The Admin tool provides the archive command to archive cold data.

  This command allows you to explicitly evict local blocks. Assume that Hive partitions a table by the day. If the data generated a week ago in partitioned tables is infrequently accessed, you can regularly run the archive command on the directory that stores such data. Then, the backups stored in the local cluster are evicted, whereas the backups in OSS are retained.

  Run the following archive command:

  ```
  jindo dfsadmin -archive [path]
  ```

  > **Note** The *[path]* parameter specifies the directory in which the data is to be archived.

- The Admin tool provides the jindo commands to manage metadata of JindoFS for Namespace Service.

  ```
  jindo dfsadmin [-options]
  ```

  > **Note** You can run the `jindo dfsadmin --help` command to obtain help information.

The Admin tool provides the diff and sync commands for the cache mode.

- The diff command is used to display the difference between the data stored in the local cluster and that in OSS.

  ```
  jindo dfsadmin -R -diff [path]
  ```

  > **Note** By default, you can use the diff command to display the difference between the metadata stored in the local cluster and that in the subdirectories of the directory specified by the `[path]` parameter. The `-R` option specifies that a recursive operation is performed to compare the metadata stored in the local cluster with that stored in all subdirectories of the directory specified by the `[path]` parameter.

- The sync command is used to synchronize the metadata between the local cluster and OSS.

```
jindo dfsadmin -R -sync [path]
```

Note The `[path]` parameter specifies the directory in which the metadata is to be synchronized. By default, you can use the sync command to synchronize the metadata in the subdirectories of the directory specified by the `[path]` parameter to the local cluster. The `-R` option specifies that a recursive operation is performed to synchronize the metadata in all subdirectories of the directory specified by the `[path]` parameter.

1.3. Use the password-free feature of JindoFS SDK

This topic describes how to access JindoFS in password-free mode from an ECS instance (not an instance in E-MapReduce clusters) when you use JindoFS SDK.

Prerequisites

1. You have ECS instances other than those for E-MapReduce. 2. The Hadoop ecosystem is used. 3. JindoFS SDK for Java is obtained.

Context

Before you can use JindoFS SDK, you must remove packages related to Jindo from your environment, such as *jboot.jar* and *smartdata-aliyun-jfs-*.jar*. To use Spark, you must also remove the packages in */opt/apps/spark-current/jars/*.

Step 1: Create an instance RAM role

Perform the following operations to create an instance RAM role in the RAM console:

1. Log on to the RAM console with an Alibaba Cloud account.
2. In the left-side navigation pane, click **RAM Roles**.
3. On the RAM Roles page that appears, click **Create RAM Role**. In the pane that appears, select **Alibaba Cloud Service** for Trusted entity type.
4. Click **Next**.
5. Enter a role name in the RAM Role Name field and select **Elastic Compute Service** from the **Select Trusted Service** drop-down list.
6. Click **OK**.

Step 2: Grant permissions to the RAM role

Perform the following operations to grant system permissions or custom permissions to the RAM role:

1. Log on to the RAM console with an Alibaba Cloud account.
2. (Optional) If you do not use system permissions, you can create custom permissions. For more information, see the "(Optional) Create a custom authorization policy" section in Implement access control by using RAM.
3. In the left-side navigation pane, click **RAM Roles**.
4. Find the created RAM role and click **Input and Attach** in the **Actions** column.

5. In the pane that appears, select **System Policy** or **Custom Policy** for Type.
6. Enter the policy name.
7. Click **OK**.

Step 3: Bind the RAM role to an ECS instance

Perform the following operations:

1. Log on to the ECS console.
2. In the left-side navigation pane, choose **Instances & Images > Instances**.
3. In the top navigation bar, select a region.
4. Find the target ECS instance and choose **More > Instance Settings > Bind/Unbind RAM Role** in the Actions column.

5. In the Bind/Unbind RAM Role dialog box that appears, select the RAM role and click **OK**.

Step 4: Set environment variables on ECS

Run one of the following commands to set environment variables on ECS:

```
export CLASSPATH=/xx/xx/jindofs-2.5.0-sdk.jar
```

or

```
HADOOP_CLASSPATH=$HADOOP_CLASSPATH:/xx/xx/jindofs-2.5.0-sdk.jar
```

Step 5: Access JindoFS in password-free mode

1. Use Shell to access OSS.

   ```
   hdfs dfs -ls/-mkdir/-put/..... oss://<ossPath>
   ```

2. Access OSS by using the FileSystem interface of Hadoop.

   The following code is an example:

```
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.LocatedFileStatus;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.fs.RemoteIterator;

import java.net.URI;

public class test {
    public static void main(String[] args) throws Exception {
        FileSystem fs = FileSystem.get(new URI("ossPath"), new Configuration());
        RemoteIterator<LocatedFileStatus> iterator = fs.listFiles(new Path("ossPath"), false);
        while (iterator.hasNext()){
            LocatedFileStatus fileStatus = iterator.next();
            Path fullPath = fileStatus.getPath();
            System.out.println(fullPath);
        }
    }
}
```

1.4. Use the block storage mode

This topic describes the block storage mode of JindoFileSystem (JindoFS) and its scenarios.

Overview

Block storage is the most efficient mode to read and write data and query metadata. In addition, it supports Hadoop Distributed File System (HDFS) semantics related to data locality. JindoFS also provides an external client so that you can access JindoFS from the outside of an E-MapReduce cluster.

JindoFS uses Object Storage Service (OSS) as the storage back end. In block storage mode, JindoFS stores data as blocks in OSS and uses Namespace Service to maintain metadata. This guarantees high performance when you read and write data or query metadata.

Scenarios

E-MapReduce has three storage systems: E-MapReduce OssFileSystem, E-MapReduce HDFS, and E-MapReduce JindoFS. Among them, OssFileSystem and JindoFS store data in the cloud. The following table compares the features of three E-MapReduce storage systems and Hadoop support for Alibaba Cloud OSS.

Feature	Hadoop support for Alibaba Cloud OSS	E-MapReduce OssFileSystem	E-MapReduce HDFS	E-MapReduce JindoFS

Feature	Hadoop support for Alibaba Cloud OSS	E-MapReduce OssFileSystem	E-MapReduce HDFS	E-MapReduce JindoFS
Storage capacity	Tremendous	Tremendous	Depends on the E-MapReduce cluster scale	Tremendous
Reliability	High	High	High	High
Factor that affects throughput	Server	I/O performance of caches on disks in the E-MapReduce cluster	I/O performance of disks in the E-MapReduce cluster	I/O performance of disks in the E-MapReduce cluster
Metadata query efficiency	Low	Medium	High	High
Scale-out operation	Easy	Easy	Easy	Easy
Scale-in operation	Easy	Easy	Requires node decommission	Easy
Data locality	None	Weak	Strong	Medium

The block storage mode of JindoFS has the following features:

- JindoFS offers tremendous and scalable storage capacity by using OSS as the storage back end. The storage capacity is independent of the E-MapReduce cluster scale. The local cluster can be scaled in or out as required.
- JindoFS stores a certain amount of backup data in the local cluster to accelerate read operations. This improves the throughput by using limited local storage capacity, especially for Write Once Read Many (WORM) solutions.
- JindoFS provides efficient metadata query similar to HDFS. Compared with OssFileSystem, JindoFS saves much time in metadata query. In addition, JindoFS avoids system instability when data and metadata are frequently accessed.
- JindoFS moves computation as close as possible to data. This reduces the load on network transmission and improves the read performance.

Configure JindoFS

You can set all JindoFS related-parameters in Bigboot, as shown in the following figure.

> **Note**
>
> - The parameters framed in red in the preceding figure are required.
> - JindoFS supports multiple namespaces. A namespace named test is used in this topic.

Parameter	Description	Example
jfs.namespaces	The namespace supported by JindoFS. Separate multiple namespaces with commas (,).	test
jfs.namespaces.test.uri	The storage back end of the test namespace.	*oss://oss-bucket/oss-dir* **Note** You can set the value to a directory in an OSS bucket. In this case, this directory serves as the root directory, in which the test namespace reads and writes data.
jfs.namespaces.test.mode	The storage mode of the test namespace.	block
jfs.namespaces.test.oss.access.key	The AccessKey ID used to access the OSS bucket that serves as the storage back end.	*xxxx* **Note** We recommend that you select an OSS bucket in the same region and under the same account as the storage back end of the E-MapReduce cluster for better performance and stability. In this case, the E-MapReduce cluster can access the OSS bucket without using the AccessKey ID and AccessKey secret.
jfs.namespaces.test.oss.access.secret	The AccessKey secret used to access the OSS bucket that serves as the storage back end.	

Save and deploy the JindoFS configuration. Restart Namespace Service in SmartData to use JindoFS.

Configure the storage policy

JindoFS provides multiple storage policies to meet different storage needs. The following table lists four available storage policies for a directory.

Policy	Description
COLD	Data has only a backup in OSS but no backups in the local cluster. This policy is suitable for storing cold data.
WARM	The default storage policy. Data has a backup in OSS and a backup in the local cluster. The local backup can accelerate read operations.

Policy	Description
HOT	Data has a backup in OSS and multiple backups in the local cluster. Local backups can accelerate read operations on hot data.
TEMP	Data has only a backup in the local cluster. This policy is suitable for storing temporary data. The local backup can accelerate read and write operations on the temporary data. However, this may lower data reliability.

JindoFS provides a command-line tool Admin to configure the storage policy of a directory. The default storage policy is WARM. New files are stored according to the storage policy configured for the parent directory. Run the following command to configure the storage policy:

```
jindo dfsadmin -R -setStoragePolicy [path] [policy]
```

Run the following command to obtain the storage policy configured for a directory:

```
jindo dfsadmin -getStoragePolicy [path]
```

> **Note** The *[path]* parameter specifies the directory. The *-R* option specifies that a recursive operation is performed to configure the same storage policy for all subdirectories of the directory.

The Admin tool provides the archive command to archive cold data.

This command allows you to explicitly evict local blocks. Assume that Hive partitions a table by the day. If the data generated a week ago in partitioned tables is infrequently accessed, you can regularly run the archive command on the directory that stores such data. Then, the backups stored in the local cluster are evicted, whereas the backups in OSS are retained.

Run the following archive command:

```
jindo dfsadmin -archive [path]
```

> **Note** The *[path]* parameter specifies the directory in which the data is to be archived.

1.5. Use the cache mode

This topic describes the cache mode of JindoFileSystem (JindoFS) and its scenarios.

Overview

In cache mode, JindoFS stores data files as objects in Object Storage Service (OSS) and caches data and metadata of these files in the local cluster based on the requirements for accessing these files. This accelerates read and write operations on data and metadata. In addition, the cache mode provides multiple policies for you to synchronize metadata as required.

Scenarios

The cache mode is compatible with original OSS semantics. In cache mode, JindoFS stores data files as objects in OSS and caches data and metadata in the local cluster. This guarantees that JindoFS is compatible with the OSS client, E-MapReduce OssFileSystem, and other OSS interactive applications. You can also access data that exists in OSS before you configure JindoFS, with no need to migrate or convert data. In addition, local caches can be used to accelerate read and write operations on data and metadata.

Configure JindoFS

You can set all JindoFS related-parameters in Bigboot, as shown in the following figure.

> **Note**
> - The parameters framed in red in the preceding figure are required.
> - JindoFS supports multiple namespaces. A namespace named test is used in this topic.

Parameter	Description	Example
jfs.namespaces	The namespace supported by JindoFS. Separate multiple namespaces with commas (,).	test
jfs.namespaces.test.uri	The storage back end of the test namespace.	*oss://oss-bucket/* **Note** You can set the value to a directory in an OSS bucket. In this case, this directory serves as the root directory, in which the test namespace reads and writes data. Generally, you can set the value to an OSS bucket to guarantee that the path is the same as that in OSS.
jfs.namespaces.test.mode	The storage mode of the test namespace.	cache

Parameter	Description	Example
jfs.namespaces.test.oss.access.key	The AccessKey ID used to access the OSS bucket that serves as the storage back end.	*xxxx* **Note** We recommend that you select an OSS bucket in the same region and under the same account as the storage back end of the E-MapReduce cluster for better performance and stability. In this case, the E-MapReduce cluster can access the OSS bucket without using the AccessKey ID and AccessKey secret.
jfs.namespaces.test.oss.access.secret	The AccessKey secret used to access the OSS bucket that serves as the storage back end.	

Save and deploy the JindoFS configuration. Restart Namespace Service in SmartData to use JindoFS.

Configure the metadata synchronization policy

In cache mode, you may find that some data exists in OSS before you configure JindoFS. After JindoFS is configured, data and metadata are synchronized to JindoFS for future access. At the same time, you can configure the synchronization policy so that JindoFS caches data and metadata in the local cluster. Policies for synchronizing metadata include two types: the interval policy and the loading policy.

- Interval policy:

  You can set the **namespace.sync.interval** parameter to specify the synchronization interval. The default value is **-1**, which indicates that JindoFS does not synchronize metadata from OSS.

  - If you set this parameter to **0**, JindoFS synchronizes metadata from OSS each time data is accessed.
  - If you set this parameter to a value greater than **0**, JindoFS synchronizes metadata from OSS at intervals of the set value in units of seconds.

    Note For example, if you set this parameter to **5**, JindoFS synchronizes metadata from OSS every 5 seconds.

- Loading policy:

  You can set the **namespace.sync.loadtype** parameter to specify the loading policy. Valid values are never, once, and always. A value of **never** indicates that JindoFS never synchronizes metadata from OSS. A value of **once** indicates that JindoFS synchronizes metadata from OSS only once. This is the default value. A value of **always** indicates that JindoFS synchronizes metadata from OSS each time data is accessed.

  Note The **namespace.sync.loadtype** parameter only takes effect when you do not specify the **namespace.sync.interval** parameter.

1.6. Use the external client

This topic describes the external client of JindoFileSystem (JindoFS) and its scenarios.

Overview

JindoFS provides an external client so that you can access JindoFS from the outside of an E-MapReduce cluster. If you want to access JindoFS from the external client, make sure that JindoFS is in block storage mode. Currently, you cannot access JindoFS from the external client if JindoFS is in cache mode. To access JindoFS in cache mode from the outside of an E-MapReduce cluster, use the common Object Storage Service (OSS) client because the cache mode is compatible with original OSS semantics.

Scenarios

The external client of JindoFS is compatible with Hadoop Distributed File System (HDFS). To access data stored in JindoFS from the external client, make sure that your application is connected to Namespace Service of JindoFS. However, you cannot access cached data in the local cluster from the external client. In this case, the performance of data access from the external client is not as efficient as that from the inside of an E-MapReduce cluster.

Configure the external client

Make sure that the namespace supported by JindoFS in block storage mode is configured. For more information, see Use the block storage mode.

1. Obtain the Bigboot package.

   Access the */usr/lib/bigboot-current* directory in the E-MapReduce cluster to obtain the Bigboot package.

   > **Note** The Bigboot package is developed based on native code, which may be incompatible with your operating system. In this case, submit a ticket if relevant code needs to be compiled again.

2. Set up the environment.

   Set the **BIGBOOT_HOME** variable to the root directory for installing Bigboot on your device. Add the *ext* and *lib* directories in the root directory to the **classpath** parameter of your component for processing big data, such as Hadoop or Spark.

3. Copy the configuration file bigboot.cfg.external from the */usr/lib/bigboot-current/conf/* directory in the E-MapReduce cluster to the installation directory *conf/* on your device.

4. Configure Namespace Service.

   - `client.namespace.rpc.port` : the port for listening on Namespace Service.
   - `client.namespace.rpc.address` : the endpoint for listening on Namespace Service.

   > **Note** By default, E-MapReduce sets the preceding two parameters in the Bigboot configuration file.

5. Set data access parameters.

   - `client.namespaces.{YourNamespace}.oss.access.bucket` : the OSS bucket to be accessed.
   - `client.namespaces.{YourNamespace}.oss.access.endpoint` : the endpoint for accessing the OSS bucket.

- `client.namespaces.{YourNamespace}.oss.access.key`: the AccessKey ID used to access the OSS bucket.
- `client.namespaces.{YourNamespace}.oss.access.secret`: the AccessKey secret used to access the OSS bucket.

> **Note** In the preceding parameters, `{YourNamespace}` specifies the namespace that you want to access from the external client. In this topic, a namespace named test is used.

Configuration example:

```
client.namespace.rpc.port = 8101
client.namespace.rpc.address = {RPC_Address}
client.namespaces.test.oss.access.bucket = {YourOssBucket}
client.namespaces.test.oss.access.endpoint = {YourOssEndpoint}
client.namespaces.test.oss.access.key = {YourOssKey}
client.namespaces.test.oss.access.secret = {YourOssSecret}
```

Verify the configuration

- Run the following command to check whether the test namespace is configured correctly:

```
hdfs dfs -ls jfs://test/
```

- Run the following commands to check whether data can be uploaded to or downloaded from the test namespace:

```
hdfs dfs -put /etc/hosts  jfs://test/


hdfs dfs -get jfs://test/hosts
```

2.Get started with JindoFS (EMR-3.27.0 or later)

2.1. SmartData 2.6.x

SmartData 2.6.x is supported in EMR V3.26.3 and later. This SmartData version delivers new features and optimized performance. For example, Tablestore and Raft instances can be used as the metadata storage backends of Namespace Service. Namespace Service can be deployed in high availability (HA) mode. The cache mode, block storage mode, and read/write performance of JindoFS are optimized.

New metadata storage methods

In earlier versions, you can use only RocksDB as a metadata storage backend. In SmartData 2.6.x, you can also use Tablestore or Raft instances as metadata storage backends. JindoFS achieves complete cloud-native storage with metadata stored in Tablestore or Raft instances and data stored in OSS buckets. Storage modules are independent of EMR clusters. This facilitates data restoration when you re-create a cluster. You can delete a cluster anytime. If Tablestore or Raft instances are used, you can deploy Namespace Service in HA mode. RocksDB is still a simple and efficient metadata storage method for scenarios that do not require Namespace Service in HA mode or high performance of metadata storage. For example, when you use JindoFS in cache mode, we recommend that you use RocksDB to store metadata.

For more information about each metadata storage method, see the following topics:

- Use Tablestore instances to store metadata
- Use Raft-RocksDB-Tablestore to store metadata
- Use RocksDB to store metadata

Optimization of JindoFS usage modes

JindoFS supports the block storage mode and cache mode.

- The usage of the block storage mode is the same as that in versions earlier than EMR V3.26.3. For more information, see Use JindoFS in block storage mode.
- In cache mode, you can use the same method to access files as in block storage mode. You can also use the original file access method of OSS. For more information, see Use JindoFS in cache mode.

Supported permissions

To use JindoFS in block storage mode, you must be granted permissions on files. If you do not have permissions on a file, you are not allowed to read or write data from or to the file. You can run UNIX commands or use Ranger to manage permissions. UNIX allows you to grant the rwxrwxrwx permission on files. Ranger allows you to configure complex permissions. For example, you can use wildcards in paths when you configure a permission. For more information about permissions, see Manage JindoFS permissions.

2.2. Use JindoFS in block storage mode

The block storage mode ensures efficient read/write operations and high metadata accessibility. JindoFS stores data as blocks in OSS and caches data in local disks of clusters to accelerate data access. JindoFS uses Namespace Service to manage metadata and ensure high metadata accessibility. This topic describes how to use JindoFS in block storage mode.

Context

The block storage mode of JindoFS has the following features:

- JindoFS offers tremendous and scalable storage capacity by using OSS as the storage back end. The storage capacity is independent of the E-MapReduce cluster scale. The local cluster can be scaled in or out as required.
- JindoFS stores a certain amount of backup data in the local cluster to accelerate read operations. This improves the throughput by using limited local storage capacity, especially for Write Once Read Many (WORM) solutions.
- JindoFS provides efficient metadata query similar to HDFS. Compared with OssFileSystem, JindoFS saves much time in metadata query. In addition, JindoFS avoids system instability when data and metadata are frequently accessed.
- JindoFS moves computation as close as possible to data. This reduces the load on network transmission and improves the read performance.

Configure the block storage mode

1. Go to the SmartData service.
   i.
   ii. In the top navigation bar, select the region where your cluster resides. Select the resource group as required. By default, all resources of the account appear.
   iii. Click the **Cluster Management** tab.
   iv. On the **Cluster Management** page that appears, find the target cluster and click **Details** in the Actions column.
   v. In the left-side navigation pane, click **Cluster Service** and then **SmartData**.
2. Configure bigboot parameters.
   i. Click the **Configure** tab.
   ii. Click the **bigboot** tab in the Service Configuration section.

   
3. Configure required parameters.JindoFS allows you to configure multiple namespaces. A namespace named test is used in this topic.
   i. Set **jfs.namespaces** to **test**.If you configure multiple namespaces, separate them with commas (,).

ii. In the upper-right corner of the Service Configuration section, click **Custom Configuration**. In the **Add Configuration Item** dialog box, add the parameters described in the following table.

Parameter	Description	Example
jfs.namespaces.test.oss.uri	The storage backend of the test namespace.	*oss://<oss_bucket>/<oss_dir>/* **Note** We recommend that you set this parameter to a directory of an OSS bucket. The namespace stores data blocks in this directory.
jfs.namespaces.test.mode	The storage mode of the test namespace. Set this parameter to block.	block
jfs.namespaces.test.oss.access.key	The AccessKey ID of the OSS bucket that serves as the storage backend.	*xxxx* **Note** We recommend that you store data in an OSS bucket that is in the same region and under the same account as your EMR cluster. This ensures high performance and stability. In this case, you do not need to configure the AccessKey ID and AccessKey secret because the OSS bucket allows password-free access from the EMR cluster.
jfs.namespaces.test.oss.access.secret	The AccessKey secret of the OSS bucket that serves as the storage backend.	

iii. Click **OK**.

4. In the upper-right corner of the Service Configuration section, click **Save**.
5. Select **Restart Jindo Namespace Service** from the **Actions** drop-down list in the upper-right corner.After Namespace Service is restarted, you can use `jfs://test/<path_of_file>` to access files in JindoFS.

Control disk space usage

JindoFS uses OSS as the data storage backend, which allows you to store large volumes of data. However, the capacity of local disks is limited. JindoFS automatically deletes cold data in local disks. The `storage.watermark.high.ratio` and `storage.watermark.low.ratio` parameters are used to adjust the space usage of local disks. You can set the parameters to decimal numbers between 0 and 1.

1. Modify disk usage configurations.

   In the **Service Configuration** section for the SmartData service, click the **storage** tab and configure the parameters described in the following table.

Parameter	Description
storage.watermark.high.ratio	The upper limit of disk usage. When the disk usage of JindoFS data exceeds this limit, JindoFS automatically deletes data in the disk. Default value: 0.4.
storage.watermark.low.ratio	The lower limit of disk usage. After automatic data deletion is triggered, JindoFS starts to delete data until the disk usage of JindoFS data is reduced to this limit. Default value: 0.2.

   Note You can configure the upper limit and lower limit to adjust the disk space assigned to JindoFS. Make sure that the upper limit is greater than the lower limit.

2. Save the configurations.
   i. In the upper-right corner of the Service Configuration section, click **Save**.
   ii. In the **Confirm Changes** dialog box, specify Description and turn on **Auto-update Configuration**.
   iii. Click **OK**.
3. Restart Jindo Storage Service to apply the configurations.
   i. Select **Restart Jindo Storage Service** from the **Actions** drop-down list in the upper-right corner.
   ii. In the **Cluster Activities** dialog box, specify related parameters.
   iii. Click **OK**.
   iv. In the **Confirm** message, click **OK**.

2.3. Use JindoFS in cache mode

When you use JindoFS in cache mode, files are stored as objects in Object Storage Service (OSS), and the frequently used files are cached in an EMR cluster to improve the data access efficiency. In cache mode, JindoFS can access files in OSS without the need to convert the file formats, and JindoFS is fully compatible with OSS clients. This topic describes how to use JindoFS in cache mode.

Context

In cache mode, JindoFS supports the object semantics of OSS and is fully compatible with various OSS clients. This ensures that you can access files in OSS without the need to migrate data or convert data formats. In cache mode, JindoFS caches frequently used files in an EMR cluster. This improves the read and write performance and relieves pressure on bandwidth.

Methods to access files in OSS

In cache mode, you can use one of the following methods to access files in OSS based on your business requirements:

- OSS Scheme

  For more information, see (Recommended) Configure OSS Scheme.

- JFS Scheme

  For more information, see Configure JFS Scheme.

(Recommended) Configure OSS Scheme

OSS Scheme refers to the original method of accessing files in OSS. You can use the `oss://<bucket_name>/<path_of_your_file>` command to access files in OSS. After you create an EMR cluster, you can use this method to access files in OSS without additional configurations. You can also run existing jobs to read or write data from or to OSS without the need to modify the configurations.

Configure JFS Scheme

1. Go to the SmartData service.
   i.
   ii. In the top navigation bar, select the region where your cluster resides. Select the resource group as required. By default, all resources of the account appear.
   iii. Click the **Cluster Management** tab.
   iv. On the **Cluster Management** page that appears, find the target cluster and click **Details** in the Actions column.
   v. In the left-side navigation pane, click **Cluster Service** and then **SmartData**.
2. Configure bigboot parameters.
   i. Click the **Configure** tab.
   ii. Click the **bigboot** tab in the Service Configuration section.

   
3. Configure the required parameters.JindoFS allows you to configure multiple namespaces. A namespace named test is used in this topic.
   i. Set **jfs.namespaces** to **test**.If you configure multiple namespaces, separate them with commas (,).

ii. Click **Custom Configuration**. In the **Add Configuration Item** dialog box, configure the parameters described in the following table and click OK.

Parameter	Description	Example
jfs.namespaces.test.oss.uri	The storage backend of the test namespace.	*oss://<oss_bucket>/<oss_dir>/* **Note** Set this parameter to a directory for a specific OSS bucket or the root directory.
jfs.namespaces.test.mode	The storage mode of the test namespace. Set this parameter to cache.	cache

4. In the upper-right corner of the Service Configuration section, click **Save**.
5. Select **Restart Jindo Namespace Service** from the **Actions** drop down list in the upper-right corner.After the service is restarted, you can use the `jfs://test/<path_to_your_file>` command to access files in OSS. The files for the test namespace are organized based on the setting of the **jfs.namespaces.test.oss.uri** parameter. For example, `jfs://test/hello.txt` corresponds to *oss://<oss_bucket>/<oss_dir>/hello.txt*.

Enable local cache

After you enable local cache, hot data blocks are cached on local disks. By default, this feature is disabled, and EMR directly reads data from OSS.

1. In the left-side navigation pane, click **Cluster Service** and then **SmartData**. On the SMARTDATA page, click the **Configure** tab. In the Service Configuration section, click the **client** tab.
2. Set **jfs.cache.data-cache.enable** to **1** to enable local cache.The configuration immediately takes effect on the client, without the need to restart the SmartData service.

After you enable local cache, Jindo automatically manages cached data. It clears cache based on the high and low watermarks that you configured. For more information about how to configure the watermarks, see Control disk space usage.

Control disk space usage

JindoFS uses OSS as the data storage backend, which allows you to store large volumes of data. However, the capacity of local disks is limited. JindoFS automatically deletes cold data in local disks. The `storage.watermark.high.ratio` and `storage.watermark.low.ratio` parameters are used to adjust the space usage of local disks. You can set the parameters to decimal numbers between 0 and 1.

1. Modify disk usage configurations.

   In the **Service Configuration** section for the SmartData service, click the **storage** tab and configure the parameters described in the following table.

Parameter	Description
storage.watermark.high.ratio	The upper limit of disk usage. When the disk usage of JindoFS data exceeds this limit, JindoFS automatically deletes data in the disk. Default value: 0.4.
storage.watermark.low.ratio	The lower limit of disk usage. After automatic data deletion is triggered, JindoFS starts to delete data until the disk usage of JindoFS data is reduced to this limit. Default value: 0.2.

Note You can configure the upper limit and lower limit to adjust the disk space assigned to JindoFS. Make sure that the upper limit is greater than the lower limit.

2. Save the configurations.
   i. In the upper-right corner of the Service Configuration section, click **Save**.
   ii. In the **Confirm Changes** dialog box, specify Description and turn on **Auto-update Configuration**.
   iii. Click **OK**.
3. Restart Jindo Storage Service to apply the configurations.
   i. Select **Restart Jindo Storage Service** from the **Actions** drop-down list in the upper-right corner.
   ii. In the **Cluster Activities** dialog box, specify related parameters.
   iii. Click **OK**.
   iv. In the **Confirm** message, click **OK**.

Access an OSS bucket

If you access an OSS bucket that is under the same Alibaba Cloud account and in the same region as your EMR cluster, you do not need to configure an AccessKey pair. In other cases, you must configure an AccessKey pair and an OSS bucket endpoint. Configure the parameters based on the method that you use to access files in OSS:

- OSS Scheme
  i. In the left-side navigation pane, click **Cluster Service** and then **SmartData**. On the SMARTDATA page, click the **Configure** tab. In the Service Configuration section, click the **smartdata-site** tab.
  ii. Click **Custom Configuration**. In the **Add Configuration Item** dialog box, configure the parameters described in the following table and click **OK**.

Parameter	Description
fs.jfs.cache.oss-accessKeyId	The AccessKey ID of the OSS bucket that serves as the storage backend.
fs.jfs.cache.oss-accessKeySecret	The AccessKey secret of the OSS bucket that serves as the storage backend.

Parameter	Description
fs.jfs.cache.oss-endpoint	The endpoint of the OSS bucket that serves as the storage backend.

- JFS Scheme
  i. In the left-side navigation pane, click **Cluster Service** and then **SmartData**. On the SMARTDATA page, click the **Configure** tab. In the Service Configuration section, click the **bigboot** tab.
  ii. Set **jfs.namespaces** to **test**.
  iii. Click **Custom Configuration**. In the **Add Configuration Item** dialog box, configure the parameters described in the following table and click **OK**.

Parameter	Description
jfs.namespaces.test.oss.uri	The storage backend of the test namespace. Example: *oss://<oss_bucket.endpoint>/<oss_dir>*. The OSS bucket endpoint is specified in this parameter.
jfs.namespaces.test.oss.access.key	The AccessKey ID of the OSS bucket that serves as the storage backend.
jfs.namespaces.test.oss.access.secret	The AccessKey secret of the OSS bucket that serves as the storage backend.

Advanced configurations

You can configure some advanced parameters to optimize cache performance. After you configure the parameters, you do not need to restart the SmartData service. The configurations take effect immediately on the client.

- In the **Service Configuration** section, click the **client** tab and configure the parameters described in the following table.

Parameter	Description
client.oss.upload.threads	The number of OSS upload threads for each data write stream. Default value: 4.
client.oss.upload.max.parallelism	The maximum number of concurrent OSS upload threads of a process. This parameter prevents upload threads from occupying an excessive amount of bandwidth and memory. Default value: 16.

- In the **Service Configuration** section, click the **smartdata-site** tab and configure the parameters described in the following table.

Parameter	Description
fs.jfs.cache.copy.simple.max.byte	The threshold for the size of a file that is renamed over a common copy interface. If the size of a file is smaller than this threshold, a common copy interface is used. If the size is larger than this threshold, the Multipart Copy interface is used to improve copy efficiency. **Note** If you have enabled the fast copy feature of OSS, set this parameter to **-1**. This value indicates that all files are renamed over a common copy interface. This way, you can obtain the optimal rename performance.
fs.jfs.cache.write.buffer.size	The buffer size of data write streams. Unit: bytes. You must set this parameter to a power of 2. The maximum value is 8388608 (8 MB). If too much memory is occupied by write streams, we recommend that you set this parameter to a small value. Default value: 1048576.
fs.oss.committer.magic.enabled	Specifies whether to enable Jindo Job Committer. This Job Committer does not require rename operations and improves job commit performance. Default value: true. **Note** In cache mode, the performance of renaming files in OSS is less than standard. We recommend that you use Jindo Job Committer.

2.4. JindoFS Namespace Service

2.4.1. Use Tablestore instances to store metadata

Namespace Service of JindoFS supports different metadata storage methods. This topic describes how to configure a Tablestore instance as the metadata storage backend.

Prerequisites

- An EMR cluster is created.

  For more information, see Create a cluster.
- A Tablestore instance is created. We recommend that you use a high-performance instance.

  For more information, see Create instances.

> **Note** You must enable the transaction feature.

Context

In SmartData 2.6.x and later, Namespace Service allows you to use Tablestore instances to store metadata. You can bind a Tablestore instance to each EMR JindoFS cluster. Namespace Service creates a Tablestore table for each namespace to manage and store metadata.

The following figure shows the structure of Namespace Service in HA mode with a Tablestore instance as the metadata storage backend.

Configure a Tablestore instance

To use a Tablestore instance as the metadata storage backend, you must first bind the Tablestore instance to Namespace Service. Perform the following steps:

1. Go to the SmartData service.
   i.
   ii. In the top navigation bar, select the region where your cluster resides. Select the resource group as required. By default, all resources of the account appear.
   iii. Click the **Cluster Management** tab.
   iv. On the **Cluster Management** page that appears, find the target cluster and click **Details** in the Actions column.
   v. In the left-side navigation pane, click **Cluster Service** and then **SmartData**.
2. Configure bigboot parameters.
   i. Click the **Configure** tab.
   ii. Click the **bigboot** tab in the Service Configuration section.

   
3. Configure the parameters described in the following table.For example, you have created a Tablestore instance named emr-jfs in the China (Hangzhou) region. Your EMR cluster is deployed in a VPC network. You have obtained the AccessKey ID and AccessKey secret that are used to access the Tablestore instance.

Parameter	Description	Example
namespace.backend.type	The backend storage type of Namespace Service. ○ rocksdb ○ ots ○ raft Default value: rocksdb. Set this parameter to ots.	ots
namespace.ots.instance	The name of the Tablestore instance.	emr-jfs

Parameter	Description	Example
namespace.ots.accessKey	The AccessKey ID that is used to access the Tablestore instance.	kkkkkk
namespace.ots.accessSecret	The AccessKey secret that is used to access the Tablestore instance.	XXXXXX
namespace.ots.endpoint	The endpoint of the Tablestore instance. We recommend that you use a VPC endpoint.	*http://emr-jfs.cn-hangzhou.vpc.tablestore.aliyuncs.com*

4. Save the configurations.
   i. In the upper-right corner of the Service Configuration section, click **Save**.
   ii. In the **Confirm Changes** dialog box, specify Description and turn on **Auto-update Configuration**.
   iii. Click **OK**.
5. Select **Restart Jindo Namespace Service** from the **Actions** drop-down list in the upper-right corner.

Configure a Tablestore instance (password-free access)

1. Configure the parameters described in the following table.

   If the target Tablestore instance allows password-free access from your EMR cluster, you do not need to specify the **namespaces.ots.accessKey** and **namespace.ots.accessSecret** parameters.

Parameter	Description	Example
namespace.ots.instance	The name of the Tablestore instance.	emr-jfs
namespace.ots.endpoint	The endpoint of the Tablestore instance. We recommend that you use a VPC endpoint.	*http://emr-jfs.cn-hangzhou.vpc.tablestore.aliyuncs.com*
namespace.backend.type	The backend storage type of Namespace Service. Valid values: rocksdb, ots, and raft. Default value: rocksdb. Set this parameter to ots.	ots

2. Save the configurations.
   i. In the upper-right corner of the Service Configuration section, click **Save**.
   ii. In the **Confirm Changes** dialog box, specify Description and turn on **Auto-update Configuration**.
   iii. Click **OK**.
3. Select **Restart Jindo Namespace Service** from the **Actions** drop-down list in the upper-right corner.

Configure a Tablestore instance (HA mode)

If your EMR cluster is deployed in high availability (HA) mode, we recommend that you deploy Namespace Service in HA mode.

In HA mode, Namespace Service supports automatic failover. If the active namespace fails, the client automatically switches services to the standby namespace.

1. Go to the bigboot tab for the SmartData service and perform the following operations:
   i. Change the value of **jfs.namespace.server.rpc-address** to **emr-header-1:8101,emr-header-2:8101**.
   ii. In the upper-right corner of the Service Configuration section, click **Custom Configuration**. In the Add Configuration Item dialog box, add **namespace.backend.ots.ha** as a key and set this parameter to **true**.
   iii. Click **OK**.
   iv. Save the configurations.
      a. In the upper-right corner of the Service Configuration section, click **Save**.
      b. In the **Confirm Changes** dialog box, specify Description and turn on **Auto-update Configuration**.
      c. Click **OK**.
2. Select **Restart Jindo Namespace Service** from the **Actions** drop-down list in the upper-right corner.
3. Select **Restart Jindo Storage Service** from the **Actions** drop-down list in the upper-right corner.

2.4.2. Use RocksDB to store metadata

Namespace Service of JindoFS supports different metadata storage backends. By default, RocksDB is used. This topic describes how to configure RocksDB as the metadata storage backend.

Context

RocksDB cannot be configured in high availability mode. If you need to configure a metadata storage backend in high availability mode, we recommend that you use Tablestore or Raft instances. For more information, see Use Tablestore instances to store metadata and Use Raft-RocksDB-Tablestore to store metadata.

The following figure shows the structure of a single RocksDB instance for Namespace Service.

Configure RocksDB as the metadata storage backend

1. Go to the SmartData service.
   i.
   ii. In the top navigation bar, select the region where your cluster resides. Select the resource group as required. By default, all resources of the account appear.
   iii. Click the **Cluster Management** tab.

iv. On the **Cluster Management** page that appears, find the target cluster and click **Details** in the Actions column.

v. In the left-side navigation pane, click **Cluster Service** and then **SmartData**.

2. Configure bigboot parameters.

i. Click the **Configure** tab.

ii. Click the **bigboot** tab in the Service Configuration section.

3. Set **namespace.backend.type** to **rocksdb**.
4. Save the configurations.

i. In the upper-right corner of the Service Configuration section, click **Save**.

ii. In the **Confirm Changes** dialog box, specify Description and turn on **Auto-update Configuration**.

iii. Click **OK**.

5. Select **Restart Jindo Namespace Service** from the **Actions** drop-down list in the upper-right corner.

2.4.3. Use Raft-RocksDB-Tablestore to store metadata

In EMR V3.27.0 and later, you can use Raft-RocksDB-Tablestore to store the metadata managed by Namespace Service of JindoFS. When you create an EMR JindoFS cluster, create three master nodes and deploy a Raft instance on the master nodes. Each peer node of the Raft instance uses the local Raft-RocksDB to store metadata.

Prerequisites

- A Tablestore instance is created. We recommend that you use a high-performance instance. For more information, see Create instances.

  > **Note** You must enable the transaction feature.

- An EMR cluster that has three master nodes is created. For more information, see Create a cluster.

  > **Note** If no deployment modes are available, submit a ticket and request the EMR technical support personnel to grant you the permissions to specify a deployment mode.

Context

RocksDB supports data replication among three nodes based on the Raft protocol. You can bind an EMR cluster to a Tablestore instance and use the Tablestore instance as an additional storage medium for Namespace Service of JindoFS. EMR asynchronously uploads metadata from the local RocksDB to the Tablestore instance in real time.

The following figure shows the architecture of high-availability Raft-RocksDB-Tablestore for Namespace Service.

Configure a Raft instance as the local storage backend

1. Suspend all the SmartData components of the target EMR cluster.
   i.
   ii. In the top navigation bar, select the region where your cluster resides. Select the resource group as required. By default, all resources of the account appear.
   iii. Click the **Cluster Management** tab.
   iv. On the **Cluster Management** page that appears, find the target cluster and click **Details** in the Actions column.
   v. In the left-side navigation pane, click **Cluster Service** and then **SmartData**.
   vi. Select **Stop All Components** from the **Actions** drop-down list in the upper-right corner.
2. Configure Namespace Service parameters based on your business requirements.
3. Go to the **bigboot** tab for the SmartData service.
   i. In the left-side navigation pane, click **Cluster Service** and then **SmartData**.
   ii. Click the **Configure** tab.
   iii. In the **Service Configuration** section, click the **bigboot** tab.
4. Configure the parameters described in the following table.

Parameter	Description	Example
namespace.backend.type	The backend storage type of Namespace Service. ○ rocksdb ○ ots ○ raft Default value: rocksdb. Set this parameter to raft.	raft
namespace.backend.raft.initial-conf	The addresses of the three master nodes where the Raft instance is deployed. The value is fixed.	emr-header-1:8103:0,emr-header-2:8103:0,emr-header-3:8103:0
jfs.namespace.server.rpc-address	The addresses of the three master nodes that are used for the client to access the Raft instance. The value is fixed.	emr-header-1:8101,emr-header-2:8101,emr-header-3:8101

> **Note** If you do not need to use Tablestore for remote storage, skip Step 5.

5. (Optional)Configure a Tablestore instance as the remote asynchronous storage backend.On the **bigboot** tab for the SmartData service, configure the parameters described in the following table.

Parameter	Description	Example
namespace.ots.instance	The name of the Tablestore instance.	emr-jfs
namespace.ots.accessKey	The AccessKey ID that is used to access the Tablestore instance.	-
namespace.ots.accessSecret	The AccessKey secret that is used to access the Tablestore instance.	-
namespace.ots.endpoint	The endpoint of the Tablestore instance. We recommend that you use a VPC endpoint.	*http://emr-jfs.cn-hangzhou.vpc.tablestore.aliyuncs.com*
namespace.backend.raft.async.ots.enabled	Specifies whether to enable asynchronous upload to Tablestore. Valid values: ◦ true ◦ false Set this parameter to true. Make sure that the initialization of the SmartData service is not completed. **Note** If the initialization is completed, the setting does not take effect because metadata has been generated in the local RocksDB.	true

6. Save the configurations.
   i. In the upper-right corner of the Service Configuration section, click **Save**.
   ii. In the **Confirm Changes** dialog box, specify Description and turn on **Auto-update Configuration**.
   iii. Click **OK**.
7. Select **Start All Components** from the **Actions** drop-down list in the upper-right corner.

Recover metadata from a Tablestore instance

If you have configured a Tablestore instance as the remote asynchronous storage backend for your EMR cluster, a complete replica of JindoFS metadata is stored in the Tablestore instance. After you stop or release the EMR cluster, you can recover JindoFS metadata from the Tablestore instance to a new EMR cluster. This way, you can access the original files from the new EMR cluster.

1. (Optional)Prepare for the recovery.

i. (Optional)Collect the metadata statistics of the original EMR cluster. The metadata statistics indicate the numbers of files and folders.

```
[hadoop@emr-header-1 ~]$ hadoop fs -count jfs://test/
      1596      1482809              25 jfs://test/
  (Number of folders) (Number of files)
```

ii. Stop the jobs that are running on the original EMR cluster. You may have to wait for 30 to 120 seconds for EMR to synchronize all metadata of the cluster to the Tablestore instance. Run the following command to view the metadata status. If the command output contains `_synced=1` for the LEADER node, the latest metadata is synchronized to the Tablestore instance.

```
jindo jfs -metaStatus -detail
```

iii. Stop or release the original EMR cluster and make sure that no clusters are accessing the Tablestore instance.

2. Create an EMR cluster.Create an EMR cluster that resides in the same region as the Tablestore instance. Suspend all SmartData components. For more information, see Step 1 in Configure a Raft instance as the local storage backend.
3. Configure parameters for metadata recovery.On the **bigboot** tab for the SmartData service, configure the parameters described in the following table.

Parameter	Description	Required value
namespace.backend.raft.async.ots.enabled	Specifies whether to enable asynchronous upload to Tablestore. Valid values: ○ true ○ false	false
namespace.backend.raft.recovery.mode	Specifies whether to enable metadata recovery from Tablestore. Valid values: ○ true ○ false	true

4. Save the configurations.
   i. In the upper-right corner of the Service Configuration section, click **Save**.
   ii. In the **Confirm Changes** dialog box, specify Description and turn on **Auto-update Configuration**.
   iii. Click **OK**.
5. Select **Start All Components** from the **Actions** drop-down list in the upper-right corner.
6. After the SmartData service of the new EMR cluster is activated, EMR automatically recovers metadata from the Tablestore instance to the local Raft-RocksDB. You can run the following command to view the recovery progress:

```
jindo jfs -metaStatus -detail
```

As shown in the following figure, if the state of the LEADER node is FINISH, the recovery is completed.

7. (Optional)Check whether the numbers of files and folders of the new EMR cluster are consistent with those of the original EMR cluster.The new EMR cluster is in recovery mode and is read-only.

```
# Count the numbers of files and folders for the new EMR cluster. The results are consistent with those
of the original EMR cluster.
[hadoop@emr-header-1 ~]$ hadoop fs -count jfs://test/
      1596      1482809              25 jfs://test/

# Use the CAT or GET command to read data from the files.
[hadoop@emr-header-1 ~]$ hadoop fs -cat jfs://test/testfile
this is a test file
# View file directories.
[hadoop@emr-header-1 ~]$ hadoop fs -ls jfs://test/
Found 3 items
drwxrwxr-x   - root   root          0 2020-03-25 14:54 jfs://test/emr-header-1.cluster-50087
-rw-r-----   1 hadoop hadoop        5 2020-03-25 14:50 jfs://test/haha-12096RANDOM.txt
-rw-r-----   1 hadoop hadoop       20 2020-03-25 15:07 jfs://test/testfile

# Remove a file. A read-only error is returned.
[hadoop@emr-header-1 ~]$ hadoop fs -rm jfs://test/testfile
java.io.IOException: ErrorCode : 25021 , ErrorMsg: Namespace is under recovery mode, and is read-only.
```

8. Enable asynchronous upload to Tablestore and disable recovery from Tablestore.On the **bigboot** tab for the SmartData service, configure the parameters described in the following table.

Parameter	Description	Required value
namespace.backend.raft.async.ots.enabled	Specifies whether to enable asynchronous upload to Tablestore. Valid values: ○ true ○ false	true
namespace.backend.raft.recovery.mode	Specifies whether to enable metadata recovery from Tablestore. Valid values: ○ true ○ false	false

9. Restart the new EMR cluster.
   i. Click the **Cluster Management** tab.
   ii. On the **Cluster Management** page, find the target cluster. In the Actions column of this cluster, click **More** and then click **Restart**.

2.5. Manage JindoFS permissions

This topic describes how to manage the permissions of JindoFS in block storage mode. You can run UNIX commands or use Ranger to manage permissions.

Context

UNIX allows you to grant the rwxrwxrwx permission on files and configure owners and groups of files. JindoFS authenticates a user based on the configurations. Ranger allows you to configure complex permissions. For example, you can use wildcards in paths when you configure a permission. To use Ranger to manage permissions, you must first configure permissions in the Apache Ranger component of EMR and activate the Ranger plug-in in JindoFS. Then, you can manage JindoFS permissions in Ranger by using the same method as you manage permissions on other components.

Enable UNIX-based permission management

1. Go to the SmartData service.
   i.
   ii. In the top navigation bar, select the region where your cluster resides. Select the resource group as required. By default, all resources of the account appear.
   iii. Click the **Cluster Management** tab.
   iv. On the **Cluster Management** page that appears, find the target cluster and click **Details** in the Actions column.
   v. In the left-side navigation pane, click **Cluster Service** and then **SmartData**.
2. Configure bigboot parameters.
   i. Click the **Configure** tab.
   ii. Click the **bigboot** tab in the Service Configuration section.

3. Click **Custom Configuration**. In the **Add Configuration Item** dialog box, set Key to **jfs.namespaces.<namespace>.permission.method** and Value to **unix** and click **OK**.
4. Save the configurations.
   i. In the upper-right corner of the Service Configuration section, click **Save**.
   ii. In the **Confirm Changes** dialog box, specify Description and turn on **Auto-update Configuration**.
   iii. Click **OK**.
5. Select **Restart Jindo Namespace Service** from the **Actions** drop-down list in the upper-right corner.After the service is restarted, you can run UNIX commands to manage JindoFS permissions by using the same method as you manage HDFS permissions. You can use the following commands:

```
hadoop fs -chmod 777 jfs://{namespace_name}/dir1/file1
hadoop fs -chown john:staff jfs://{namespace_name}/dir1/file1
```

If a user does not have permissions on a file, the following error is returned:

Enable Ranger-based permission management

1. Configure Ranger as a permission management method in JindoFS.
   i. On the **bigboot** tab for the SmartData service, click **Custom Configuration**.
   ii. In the **Add Configuration Item** dialog box, set Key to **jfs.namespaces.<namespace>.permission.method** and Value to **ranger** and click OK.
   iii. Save the configurations.
      a. In the upper-right corner of the Service Configuration section, click **Save**.
      b. In the **Confirm Changes** dialog box, specify Description and turn on **Auto-update Configuration**.
      c. Click **OK**.
   iv. Select **Restart Jindo Namespace Service** from the **Actions** drop-down list in the upper-right corner.
2. Add the HDFS service on the web UI of Ranger and configure required parameters.
   i. Log on to the Ranger web UI.For more information, see Overview.
   ii. Add the HDFS service on the web UI of Ranger.

iii. Configure required parameters.

Parameter	Description
Service Name	Set this parameter in the format of jfs-{*namespace_name*}.
Username	Customize a username.
Password	Customize a password.
Namenode URL	Set this parameter in the format of *jfs://{namespace_name}*.
Authorization Enabled	Retain the default value No.
Authentication Type	Retain the default value Simple.
dfs.datanode.kerberos.principal	Leave these parameters empty.
dfs.namenode.kerberos.principal	
dfs.secondary.namenode.kerberos.principal	
Add New Configurations	Leave this parameter empty.

iv. Click **Add**.

Enable synchronization of user groups from an LDAP server in JindoFS

If you have enabled synchronization of user groups from an LDAP server in Ranger Usersync, you also need to enable this feature in JindoFS. Otherwise, JindoFS cannot obtain the information of user groups synchronized from the LDAP server and cannot verify the permissions of the user groups.

1. On the **bigboot** tab for the SmartData service, click **Custom Configuration**.
2. In the **Add Configuration Item** dialog box, configure the LDAP parameters listed in the following table and click **OK**.

Parameter	Example
hadoop.security.group.mapping	org.apache.hadoop.security.CompositeGroupsMapping
hadoop.security.group.mapping.providers	shell4services,ad4users
hadoop.security.group.mapping.providers.combined	true
hadoop.security.group.mapping.provider.shell4services	org.apache.hadoop.security.ShellBasedUnixGroupsMapping

Parameter	Example
hadoop.security.group.mapping.provider.ad4users	org.apache.hadoop.security.LdapGroupsMapping
hadoop.security.group.mapping.ldap.url	ldap://emr-header-1:10389
hadoop.security.group.mapping.ldap.search.filter.user	(&(objectClass=person)(uid={0}))
hadoop.security.group.mapping.ldap.search.filter.group	(objectClass=groupOfNames)
hadoop.security.group.mapping.ldap.base	o=emr

> **Note** Configure the parameters based on the configurations in open source HDFS.

3. Save the configurations.
   i. In the upper-right corner of the Service Configuration section, click **Save**.
   ii. In the **Confirm Changes** dialog box, specify Description and turn on **Auto-update Configuration**.
   iii. Click **OK**.
4. Select **Restart All Components** from the **Actions** drop-down list in the upper-right corner.
5. Log on to the emr-header-1 node of the EMR cluster in SSH mode and connect Ranger Usersync to the LDAP server.For more information, see Integrate Ranger UserSync with an LDAP server.

2.6. Use Jindo Job Committer

This topic describes how to use Jindo Job Committer, which is also called JindoOssCommitter.

Context

Job Committer is a basic component of distributed computing frameworks such as MapReduce and Spark. It is used to handle data consistency issues of write operations for distributed tasks. MapReduce and Spark jobs use FileOutputCommitter by default. When FileOutputCommitter is used, each task writes data to a temporary directory for the task. After a task is completed, the file is moved to a temporary directory for the job. After all tasks are completed, MapReduce Application Master or Spark Driver moves the files to the final output directory. The files are moved based on rename operations. In HDFS, rename operations are cost-effective and secure. However, in Object Storage Service (OSS), to complete a rename operation, the system needs to copy and then delete data. As a result, the time taken to commit a job increases linearly with the output data volume for a distributed task. Therefore, FileOutputCommitter is not recommended when large volumes of data are stored in OSS.

Jindo Job Committer is an effective Job Committer developed by the Alibaba Cloud EMR team. It is dedicated for job commits in the OSS scenario. Jindo Job Committer is combined with the multipart upload and file system customization features of OSS. If Jindo Job Committer is used, the output data of tasks is directly written in the final output directory without the need to perform rename operations. Before the job commit is completed, intermediate data is invisible to users. This ensures data consistency.

> **Notice**
>
> - The OSS bandwidth and the enabling of some advanced features may affect the data copy performance of OSS. Therefore, the data copy performance for different users or buckets may vary. If you have any question, contact the technical support personnel of OSS.
> - After all tasks are completed, MapReduce Application Master or Spark Driver commits the job. If a node fails or some other issues occur during the commit process, a task may fail. As a result, data consistency cannot be ensured. This triggers a short time window in which only a part of the result files appear in the final output directory. The length of the time window is positively correlated with the number of files. You can set `fs.oss.committer.threads` to a larger value to speed up concurrent processing.
> - Hive and Presto jobs use data consistency mechanisms that are independent of Hadoop Job Committer. As a result, you cannot use Jindo Job Committer to optimize Hive and Presto job commits.
> - In EMR clusters, Jindo Job Committer is enabled by default.

Use Jindo Job Committer in MapReduce jobs

1. Go to the **mapred-site** tab for the YARN service.
   i.
   ii. In the top navigation bar, select the region where your cluster resides. Select the resource group as required. By default, all resources of the account appear.
   iii. Click the **Cluster Management** tab.
   iv. On the **Cluster Management** page that appears, find the target cluster and click **Details** in the Actions column.
   v. In the left-side navigation pane, click **Cluster Service** and then **YARN**.
   vi. Click the **Configure** tab.
   vii. In the **Service Configuration** section, click the **mapred-site** tab.
2. Configure the Job Committer parameter based on your Hadoop version:
   - Hadoop 2.x

     On the **mapred-site** tab, set **mapreduce.outputcommitter.class** to **com.aliyun.emr.fs.oss.commit.JindoOssCommitter**.
   - Hadoop 3.x

     On the **mapred-site** tab, set **mapreduce.outputcommitter.factory.scheme.oss** to **com.aliyun.emr.fs.oss.commit.JindoOssCommitterFactory**.
3. Save the configuration.
   i. In the upper-right corner of the Service Configuration section, click **Save**.
   ii. In the **Confirm Changes** dialog box, specify Description and turn on **Auto-update Configuration**.
   iii. Click **OK**.
4. Go to the **smartdata-site** tab for the SmartData service.
   i. In the left-side navigation pane, click **Cluster Service** and then **SmartData**.

ii. Click the **Configure** tab.

iii. In the **Service Configuration** section, click the **smartdata-site** tab.

5. On the **smartdata-site** tab, set **fs.oss.committer.magic.enabled** to **true**.
6. Save the configuration.
   i. In the upper-right corner of the Service Configuration section, click **Save**.
   ii. In the **Confirm Changes** dialog box, specify Description and turn on **Auto-update Configuration**.
   iii. Click **OK**.

> **Note** After you set **mapreduce.outputcommitter.class** to **com.aliyun.emr.fs.oss.commit.JindoOssCommitter**, you can use the **fs.oss.committer.magic.enabled** parameter to determine which Job Committer is used. If you set this parameter to true, MapReduce jobs use Jindo Oss Magic Committer, which does not require rename operations. If you set this parameter to false, JindoOssCommitter functions the same as FileOutputCommitter.

Use Jindo Job Committer in Spark jobs

1. Go to the **spark-defaults** tab for the Spark service.
   i. In the left-side navigation pane, click **Cluster Service** and then **Spark**.
   ii. Click the **Configure** tab.
   iii. In the **Service Configuration** section, click the **spark-defaults** tab.
2. On the **spark-defaults** tab, configure the parameters listed in the following table.

Parameter	Description
spark.sql.sources.outputCommitterClass	com.aliyun.emr.fs.oss.commit.JindoOssCommitter
spark.sql.parquet.output.committer.class	com.aliyun.emr.fs.oss.commit.JindoOssCommitter
spark.sql.hive.outputCommitterClass	com.aliyun.emr.fs.oss.commit.JindoOssCommitter

   These parameters specify the Job Committers used to write data to Spark data source tables, Spark Parquet data source tables, and Hive tables. Configure these parameters based on your business requirements.
3. Save the configuration.
   i. In the upper-right corner of the Service Configuration section, click **Save**.
   ii. In the **Confirm Changes** dialog box, specify Description and turn on **Auto-update Configuration**.
   iii. Click **OK**.
4. Go to the **smartdata-site** tab for the SmartData service.
   i. In the left-side navigation pane, click **Cluster Service** and then **SmartData**.
   ii. Click the **Configure** tab.
   iii. In the **Service Configuration** section, click the **smartdata-site** tab.

5. On the **smartdata-site** tab, set **fs.oss.committer.magic.enabled** to **true**.

   > **Note** You can use the `fs.oss.committer.magic.enabled` parameter to determine which Job Committer is used. If you set this parameter to true, Spark jobs use Jindo Oss Magic Committer, which does not require rename operations. If you set this parameter to false, JindoOssCommitter functions the same as FileOutputCommitter.

6. Save the configuration.
   i. In the upper-right corner of the Service Configuration section, click **Save**.
   ii. In the **Confirm Changes** dialog box, specify Description and turn on **Auto-update Configuration**.
   iii. Click **OK**.

Optimize Jindo Job Committer

If your MapReduce or Spark tasks write a large number of files, you can adjust the number of threads that can be concurrently executed for job commit-related tasks to improve job commit performance.

1. Go to the **smartdata-site** tab for the SmartData service.
   i. In the left-side navigation pane, click **Cluster Service** and then **SmartData**.
   ii. Click the **Configure** tab.
   iii. In the **Service Configuration** section, click the **smartdata-site** tab.
2. On the **smartdata-site** tab, set **fs.oss.committer.threads** to **8**.The default value is 8.
3. Save the configuration.
   i. In the upper-right corner of the Service Configuration section, click **Save**.
   ii. In the **Confirm Changes** dialog box, specify Description and turn on **Auto-update Configuration**.
   iii. Click **OK**.

3.Basic JindoFS (EMR-3.28.0 or later)

3.1. Use Jindo DistCp

This topic describes how to use the data copy tool Jindo DistCp.

Prerequisites

An EMR cluster of the 3.28.0 version is created. For information about how to create a cluster, see Create a cluster.

Use Jindo DistCp

Run the following command to obtain the help information:

```
[root@emr-header-1 opt]# jindo distcp --help
```

The following information is returned:

```
--help  -  Print help text
--src=VALUE  -  Directory to copy files from
--dest=VALUE  -  Directory to copy files to
--parallelism=VALUE  -  Copy task parallelism
--outputManifest=VALUE  -  The name of the manifest file
--previousManifest=VALUE  -  The path to an existing manifest file
--requirePreviousManifest=VALUE  -  Require that a previous manifest is present if specified
--copyFromManifest  -  Copy from a manifest instead of listing a directory
--srcPrefixesFile=VALUE  -  File containing a list of source URI prefixes
--srcPattern=VALUE  -  Include only source files matching this pattern
--deleteOnSuccess  -  Delete input files after a successful copy
--outputCodec=VALUE  -  Compression codec for output files
--groupBy=VALUE  -  Pattern to group input files by
--targetSize=VALUE  -  Target size for output files
--enableBalancePlan  -  Enable plan copy task to make balance
--enableDynamicPlan  -  Enable plan copy task dynamically
--enableTransaction  -  Enable transation on Job explicitly
--diff  -  show the difference between src and dest filelist
```

When you use Jindo DistCp, you can specify the following options to copy data:

- --src and --dest
- --parallelism
- --srcPattern

- --deleteOnSuccess
- --outputCodec
- --outputManifest and --requirePreviousManifest
- --outputManifest and --previousManifest
- --copyFromManifest
- --srcPrefixesFile
- --groupBy and -targetSize
- --enableBalancePlan
- --enableDynamicPlan
- --enableTransaction
- --diff

--src and --dest

`--src` specifies the source directory. `--dest` specifies the destination directory.

By default, Jindo DistCp copies all files in the directory specified by `--src` to the directory specified by `--dest`. If you do not specify a root directory, Jindo DistCp automatically creates a root directory.

For example, you can run the following command to copy files in the */opt/tmp* directory of HDFS to an OSS bucket:

```
jindo distcp --src /opt/tmp --dest oss://yang-hhht/tmp
```

--parallelism

`--parallelism` specifies the mapreduce.job.reduces parameter for the MapReduce job that is executed to copy files. The default value is 7. You can customize `--parallelism` based on your available cluster resources. This allows you to determine how many reduce tasks can be run in parallel.

For example, you can run the following command to copy files in the */opt/tmp* directory of HDFS to an OSS bucket:

```
jindo distcp --src /opt/tmp --dest oss://yang-hhht/tmp --parallelism 20
```

--srcPattern

`--srcPattern` specifies a regular expression that filters files for the copy operation. The regular expression must match a full path.

For example, if you need to copy all log files in the */data/incoming/hourly_table/2017-02-01/03* directory, set `--srcPattern` to .*\.log.

Run the following command to view the files in the */data/incoming/hourly_table/2017-02-01/03* directory.

```
[root@emr-header-1 opt]# hdfs dfs -ls /data/incoming/hourly_table/2017-02-01/03
```

The following information is returned:

```
Found 6 items
-rw-r-----   2 root hadoop      2252 2020-04-17 20:42 /data/incoming/hourly_table/2017-02-01/03/000151.sst
-rw-r-----   2 root hadoop      4891 2020-04-17 20:47 /data/incoming/hourly_table/2017-02-01/03/1.log
-rw-r-----   2 root hadoop      4891 2020-04-17 20:47 /data/incoming/hourly_table/2017-02-01/03/2.log
-rw-r-----   2 root hadoop      4891 2020-04-17 20:42 /data/incoming/hourly_table/2017-02-01/03/OPTIONS-000109
-rw-r-----   2 root hadoop      1016 2020-04-17 20:47 /data/incoming/hourly_table/2017-02-01/03/emp01.txt
-rw-r-----   2 root hadoop      1016 2020-04-17 20:47 /data/incoming/hourly_table/2017-02-01/03/emp06.txt
```

Run the following command to copy the log files:

```
jindo distcp --src /data/incoming/hourly_table --dest oss://yang-hhht/hourly_table --srcPattern .*\.log --parallelism 20
```

Run the following command to view files in the destination OSS bucket:

```
[root@emr-header-1 opt]# hdfs dfs -ls oss://yang-hhht/hourly_table/2017-02-01/03
```

The following information is returned. Only log files in the source directory are copied.

```
Found 2 items
-rw-rw-rw-   1      4891 2020-04-17 20:52 oss://yang-hhht/hourly_table/2017-02-01/03/1.log
-rw-rw-rw-   1      4891 2020-04-17 20:52 oss://yang-hhht/hourly_table/2017-02-01/03/2.log
```

--deleteOnSuccess

`--deleteOnSuccess` enables Jindo DistCp to delete the copied files from the source directory after a copy operation succeeds.

For example, you can run the following command to copy files in */data/incoming/hourly_table* to an OSS bucket and delete the copied files from the source directory:

```
jindo distcp --src /data/incoming/hourly_table --dest oss://yang-hhht/hourly_table --deleteOnSuccess --parallelism 20
```

--outputCodec

`--outputCodec` specifies the compression codec that is used to compress copied files online. Example:

```
jindo distcp --src /data/incoming/hourly_table --dest oss://yang-hhht/hourly_table --outputCodec=gz --parallelism 20
```

Run the following command to view files in the destination directory:

```
[root@emr-header-1 opt]# hdfs dfs -ls oss://yang-hhht/hourly_table/2017-02-01/03
```

The following information is returned. Files in the destination directory are compressed in the GZ format.

```
Found 6 items
-rw-rw-rw-   1      938 2020-04-17 20:58 oss://yang-hhht/hourly_table/2017-02-01/03/000151.sst.gz
-rw-rw-rw-   1     1956 2020-04-17 20:58 oss://yang-hhht/hourly_table/2017-02-01/03/1.log.gz
-rw-rw-rw-   1     1956 2020-04-17 20:58 oss://yang-hhht/hourly_table/2017-02-01/03/2.log.gz
-rw-rw-rw-   1     1956 2020-04-17 20:58 oss://yang-hhht/hourly_table/2017-02-01/03/OPTIONS-000109.gz
-rw-rw-rw-   1      506 2020-04-17 20:58 oss://yang-hhht/hourly_table/2017-02-01/03/emp01.txt.gz
-rw-rw-rw-   1      506 2020-04-17 20:58 oss://yang-hhht/hourly_table/2017-02-01/03/emp06.txt.gz
```

You can set this parameter to gzip, gz, lzo, lzop, snappy, none, or keep. Default value: keep.

- none: Jindo DistCp does not compress copied files. If the files have been compressed, Jindo DistCp decompresses them.
- keep: Jindo DistCp copies files with no change in the compression.

> **Note** If you want to use the LZO codec in an open source Hadoop cluster, you must install the native library of gplcompression and the hadoop-lzo package.

--outputManifest and --requirePreviousManifest

`--outputManifest` generates a manifest file that contains the information about all the files copied by Jindo DistCp. The information includes the destination files, source files, and file sizes.

If you want to generate a manifest file, set `--requirePreviousManifest` to `false`. By default, the file is compressed in the GZ format. This is the only supported format.

```
jindo distcp --src /data/incoming/hourly_table --dest oss://yang-hhht/hourly_table --outputManifest=manif
est-2020-04-17.gz --requirePreviousManifest=false --parallelism 20
```

Run the following command to view the content of the file:

```
[root@emr-header-1 opt]# hadoop fs -text oss://yang-hhht/hourly_table/manifest-2020-04-17.gz > before.
lst
[root@emr-header-1 opt]# cat before.lst
```

The following information is returned:

{"path":"oss://yang-hhht/hourly_table/2017-02-01/03/000151.sst","baseName":"2017-02-01/03/000151.sst","srcDir":"oss://yang-hhht/hourly_table","size":2252}
{"path":"oss://yang-hhht/hourly_table/2017-02-01/03/1.log","baseName":"2017-02-01/03/1.log","srcDir":"oss://yang-hhht/hourly_table","size":4891}
{"path":"oss://yang-hhht/hourly_table/2017-02-01/03/2.log","baseName":"2017-02-01/03/2.log","srcDir":"oss://yang-hhht/hourly_table","size":4891}
{"path":"oss://yang-hhht/hourly_table/2017-02-01/03/OPTIONS-000109","baseName":"2017-02-01/03/OPTIONS-000109","srcDir":"oss://yang-hhht/hourly_table","size":4891}
{"path":"oss://yang-hhht/hourly_table/2017-02-01/03/emp01.txt","baseName":"2017-02-01/03/emp01.txt","srcDir":"oss://yang-hhht/hourly_table","size":1016}
{"path":"oss://yang-hhht/hourly_table/2017-02-01/03/emp06.txt","baseName":"2017-02-01/03/emp06.txt","srcDir":"oss://yang-hhht/hourly_table","size":1016}

--outputManifest and --previousManifest

`--outputManifest` generates a manifest file that contains a list of both previously and newly copied files. `--previousManifest` generates a manifest file that contains a list of previously copied files. This way, you can recreate the full history of operations and see what files are copied by the current job:

For example, two files are added to the source directory. Run the following command to copy the newly added files:

```
jindo distcp --src /data/incoming/hourly_table --dest oss://yang-hhht/hourly_table --outputManifest=manifest-2020-04-18.gz --previousManifest=oss://yang-hhht/hourly_table/manifest-2020-04-17.gz --parallelism 20
```

Run the following command to view the copied files:

```
[root@emr-header-1 opt]# hadoop fs -text oss://yang-hhht/hourly_table/manifest-2020-04-18.gz > current.lst
[root@emr-header-1 opt]# diff before.lst current.lst
```

The following information is returned:

```
3a4,5
> {"path":"oss://yang-hhht/hourly_table/2017-02-01/03/5.log","baseName":"2017-02-01/03/5.log","srcDir":"oss://yang-hhht/hourly_table","size":4891}
> {"path":"oss://yang-hhht/hourly_table/2017-02-01/03/6.log","baseName":"2017-02-01/03/6.log","srcDir":"oss://yang-hhht/hourly_table","size":4891}
```

--copyFromManifest

You can use `--copyFromManifest` to specify a manifest file that was previously generated by `--outputManifest` and copy the files listed in the manifest file to the destination directory. Example:

```
jindo distcp --src /data/incoming/hourly_table --dest oss://yang-hhht/hourly_table --previousManifest=oss://yang-hhht/hourly_table/manifest-2020-04-17.gz --copyFromManifest --parallelism 20
```

--srcPrefixesFile

`--srcPrefixesFile` enables Jindo DistCp to copy files in multiple folders at a time.

Run the following command to view the files under *hourly_table*:

```
[root@emr-header-1 opt]# hdfs dfs -ls oss://yang-hhht/hourly_table
```

The following information is returned:

```
Found 4 items
drwxrwxrwx   -          0 1970-01-01 08:00 oss://yang-hhht/hourly_table/2017-02-01
drwxrwxrwx   -          0 1970-01-01 08:00 oss://yang-hhht/hourly_table/2017-02-02
```

Run the following command to copy the files under *hourly_table* to the *folders.txt* file:

```
jindo distcp --src /data/incoming/hourly_table --dest oss://yang-hhht/hourly_table --srcPrefixesFile file:///opt/folders.txt --parallelism 20
```

Run the following command to view the content of the *folders.txt* file:

```
[root@emr-header-1 opt]# cat folders.txt
```

The following information is returned:

```
hdfs://emr-header-1.cluster-50466:9000/data/incoming/hourly_table/2017-02-01
hdfs://emr-header-1.cluster-50466:9000/data/incoming/hourly_table/2017-02-02
```

--groupBy and -targetSize

Reading a large number of small files from HDFS affects the data processing performance. Therefore, we recommend that you use Jindo DistCp to merge small files into large files of a specified size. This optimizes analysis performance and reduces costs.

Run the following command to view the files in the specified folder:

```
[root@emr-header-1 opt]# hdfs dfs -ls /data/incoming/hourly_table/2017-02-01/03
```

The following information is returned:

```
Found 8 items
-rw-r-----   2 root hadoop      2252 2020-04-17 20:42 /data/incoming/hourly_table/2017-02-01/03/000151.sst
-rw-r-----   2 root hadoop      4891 2020-04-17 20:47 /data/incoming/hourly_table/2017-02-01/03/1.log
-rw-r-----   2 root hadoop      4891 2020-04-17 20:47 /data/incoming/hourly_table/2017-02-01/03/2.log
-rw-r-----   2 root hadoop      4891 2020-04-17 21:08 /data/incoming/hourly_table/2017-02-01/03/5.log
-rw-r-----   2 root hadoop      4891 2020-04-17 21:08 /data/incoming/hourly_table/2017-02-01/03/6.log
-rw-r-----   2 root hadoop      4891 2020-04-17 20:42 /data/incoming/hourly_table/2017-02-01/03/OPTIONS-000109
-rw-r-----   2 root hadoop      1016 2020-04-17 20:47 /data/incoming/hourly_table/2017-02-01/03/emp01.txt
-rw-r-----   2 root hadoop      1016 2020-04-17 20:47 /data/incoming/hourly_table/2017-02-01/03/emp06.txt
```

Run the following command to merge the TXT files in the folder into files each with a size of no more than 10 MB:

```
jindo distcp --src /data/incoming/hourly_table --dest oss://yang-hhht/hourly_table --targetSize=10 --groupBy='.*/([a-z]+). *.txt' --parallelism 20
```

Run the following command to view the files in the destination directory. The two TXT files are merged into one.

```
[root@emr-header-1 opt]# hdfs dfs -ls oss://yang-hhht/hourly_table/2017-02-01/03/
Found 1 items
-rw-rw-rw-   1      2032 2020-04-17 21:18 oss://yang-hhht/hourly_table/2017-02-01/03/emp2
```

--enableBalancePlan

If both small and large files are to be copied but the file sizes are not significantly different among small files and among large files, you can use `--enableBalancePlan` to optimize the job allocation plan. This improves the copy performance of Jindo DistCp. If you do not specify an application plan, files are randomly allocated.

```
jindo distcp --src /data/incoming/hourly_table --dest oss://yang-hhht/hourly_table --enableBalancePlan --parallelism 20
```

> **Note** You cannot use this option in the same command as `--groupby` or `--targetSize`.

--enableDynamicPlan

If the files to be copied are significantly different in size and most of the files are small files, you can use `--enableDynamicPlan` to optimize the job allocation plan. This improves the copy performance of Jindo DistCp.

```
jindo distcp --src /data/incoming/hourly_table --dest oss://yang-hhht/hourly_table --enableDynamicPlan --p
arallelism 20
```

> **Note** You cannot use this option in the same command as `--groupby` or `--targetSize`.

--enableTransaction

`--enableTransaction` ensures the integrity of job levels and transaction support among jobs. Example:

```
jindo distcp --src /data/incoming/hourly_table --dest oss://yang-hhht/hourly_table --enableTransaction --p
arallelism 20
```

--diff

After files are copied, you can use `--diff` to check the differences between the source and destination directories.

Example:

```
jindo distcp --src /data/incoming/hourly_table --dest oss://yang-hhht/hourly_table --diff
```

If all files are copied, the following information is returned:

```
INFO distcp.JindoDistCp: distcp has been done completely
```

If some files are not copied, a *manifest* file that contains a list of these files is generated in the destination directory. Then, you can use `--copyFromManifest` and `--previousManifest` to copy the files in the list to the destination directory. This way, the data volume and file quantity are verified. If Jindo DistCp has performed compression or decompression operations during the copy process, `--diff` does not return accurate file size differences.

```
jindo distcp --src /data/incoming/hourly_table --dest oss://yang-hhht/hourly_table --dest oss://yang-hhht/
hourly_table --previousManifest=file:///opt/manifest-2020-04-17.gz --copyFromManifest --parallelism 20
```

> **Note** If your destination directory is an HDFS directory, you must specify `--dest` in the format of */path*, *hdfs://hostname:ip/path*, or *hdfs://headerIp:ip/path*. Other formats, such as *hdfs:///path* and *hdfs:/path*, are not supported.

Check DistCp Counters

Run the following commands to check DistCp Counters in the counter information of MapReduce:

```
Distcp Counters
        Bytes Destination Copied=11010048000
        Bytes Source Read=11010048000
        Files Copied=1001

Shuffle Errors
        BAD_ID=0
        CONNECTION=0
        IO_ERROR=0
        WRONG_LENGTH=0
        WRONG_MAP=0
        WRONG_REDUCE=0
```

Note If Jindo DistCp has performed compression or decompression operations during the copy process, the values of `Bytes Destination Copied` and `Bytes Source Read` may be different.

4.Use JindoFS in EMR-3.29.X

4.1. JindoFS Namespace Service

5.JindoFS ecosystem

5.1. Migrate data from HDFS to JindoFS

This topic describes how to migrate data from Hadoop Distributed File System (HDFS) to JindoFileSystem (JindoFS) that stores data in Object Storage Service (OSS).

Migrate data

- Use Hadoop FS shell commands

  You can use File System (FS) shell commands to migrate a small amount of data:

  ```
  hadoop dfs -cp  hdfs://emr-cluster/README.md  jfs://emr-jfs/
  ```

  ```
  hadoop dfs -cp  oss://oss_bucket/README.md  jfs://emr-jfs/
  ```

- Use Hadoop DistCp

  You can use DistCp, a built-in tool of Hadoop, to migrate a large amount of data:

  ```
  hadoop distcp  hdfs://emr-cluster/files  jfs://emr-jfs/output/
  ```

  ```
  hadoop distcp  oss://oss_bucket/files  jfs://emr-jfs/output/
  ```

  > **Note** For more information about DistCp parameters, see DistCp Version2 Guide.

Use the cache mode

In cache mode, JindoFS stores data files as objects in OSS without changing the metadata and data. When you access these OSS objects, JindoFS can cache data and metadata of these OSS objects in the local cluster so that you can quickly access them next time. For more information, see Use the cache mode.

5.2. Use MapReduce to process data in JindoFS

This topic describes how to use MapReduce to read and write data in JindoFileSystem (JindoFS).

JindoFS configuration

For example, a namespace named emr-jfs is created with the following configuration:

- jfs.namespaces=emr-jfs
- jfs.namespaces.emr-jfs.uri=oss://oss-bucket/oss-dir
- jfs.namespaces.emr-jfs.mode=block

MapReduce introduction

Generally, Hadoop MapReduce jobs read and write data through Hadoop Distributed File System (HDFS). JindoFS is compatible with HDFS. You can set the input and output directories of MapReduce jobs to directories in JindoFS. In this case, MapReduce jobs can read and write data in JindoFS.

Hadoop MapReduce is a software framework for easily writing applications. Applications based on Hadoop MapReduce can process multi-terabyte datasets in parallel in large clusters that consist of thousands of nodes in a reliable and fault-tolerant manner. A MapReduce job usually splits the input dataset into independent blocks that are processed by map tasks in parallel. The MapReduce framework sorts the output of map tasks and writes the sorted output to reduce tasks. Both the input and the output of the job are stored in a file system. The MapReduce framework is responsible for scheduling tasks, monitoring tasks, and rerunning failed tasks.

Job input and output

In applications, the input and output directories of MapReduce jobs are specified. Map and reduce functions are implemented based on appropriate methods or abstract classes. The Hadoop job client submits MapReduce jobs and their configuration to ResourceManager. Then, ResourceManager schedules tasks. In this case, you can modify the input and output directories of MapReduce jobs to directories in JindoFS to read and write data in JindoFS.

Examples

The following examples show how to modify the input and output directories of a MapReduce job to read and write data in JindoFS.

- Teragen

  In this MapReduce job, use Teragen to generate data in a specified number of rows in the specified directory:

  ```
  hadoop jar /usr/lib/hadoop-current/share/hadoop/mapreduce/hadoop-mapreduce-examples-*.jar teragen <num rows> <output dir>
  ```

  Replace the output directory with a directory in JindoFS to write data to JindoFS:

  ```
  hadoop jar /usr/lib/hadoop-current/share/hadoop/mapreduce/hadoop-mapreduce-examples-*.jar teragen 100000 jfs://emr-jfs/teragen_data_0
  ```

- Terasort

  In this MapReduce job, use Terasort to sort data in the specified input directory and write the sorted data to the specified output directory:

  ```
  hadoop jar /usr/lib/hadoop-current/share/hadoop/mapreduce/hadoop-mapreduce-examples-*.jar terasort <in> <out>
  ```

  Replace the input and output directories with directories in JindoFS to process data in JindoFS:

  ```
  hadoop jar /usr/lib/hadoop-current/share/hadoop/mapreduce/hadoop-mapreduce-examples-*.jar terasort jfs://emr-jfs/teragen_data_0/ jfs://emr-jfs/terasort_data_0
  ```

5.3. Use Hive to query data in JindoFS

Apache Hive is a distributed SQL query engine widely used in the Hadoop ecosystem. Hive manages data in databases, tables, and partitions. You can specify the storage location to query data at the storage backend.

JindoFS configuration

For example, a namespace named emr-jfs is created with the following configuration:

- jfs.namespaces=emr-jfs
- jfs.namespaces.emr-jfs.uri=oss://oss-bucket/oss-dir
- jfs.namespaces.emr-jfs.mode=block

Specify the storage location for data warehouses, databases, tables, or partitions

- Specify the storage location for a data warehouse

  The hive-site configuration file contains the hive.metastore.warehouse.dir parameter. This parameter specifies the directory in which a Hive data warehouse stores data. For example, set this parameter to `jfs://emr-jfs/user/hive/warehouse`.

- Specify the storage location for a database

  A Hive database has a storage location. The location is also used as the default storage location of tables in the database. The parameter that specifies the storage location is optional when you create a database. By default, the storage location of a database is the value of the hive.metastore.warehouse.dir parameter in the hive-site file added with the database name. You can run the following SQL statements to specify the storage location.

  - Set the storage location to a directory in JindoFileSystem (JindoFS) when you create a database:

    ```
    CREATE DATABASE database_name
    LOCATION
    'jfs://namespace/database_dir';
    ```

    For example, to create a Hive database named database_on_jindofs and set the storage location to `jfs://emr-jfs/warehouse/database_on_jindofs`, run the following statement:

    ```
    CREATE DATABASE database_on_jindofs
    LOCATION
    'jfs://emr-jfs/hive/warehouse/database_on_jindofs';
    ```

- Modify the storage location of a database to a directory in JindoFS:
  a. Run the following SHOW CREATE statement to query the storage location of a database:

     ```
     SHOW CREATE DATABASE database_name;
     ```

  b. View the returned storage location. By default, the storage location of a database is that of its data warehouse added with the database name.

     ```
     CREATE DATABASE `database_name`
     LOCATION
     'hdfs://emr-jfs/user/hive/warehouse/database_name.db'
     ```

  c. Modify the storage location to a directory in JindoFS. This operation does not affect existing tables. If you do not specify the storage location for a new table, the modified location is used as the default storage location of the table.

     ```
     ALTER DATABASE database_name SET LOCATION jfs_path;
     ```

- Specify the storage location for a table or partition

  Similar to the storage location of a database, that of a table or partition is also specified based on the upper-level storage location. Data in a non-partitioned table is stored in the storage location of the table. Data in a partitioned table is stored in the storage location of respective partitions. You can run the following SQL statements to specify the storage location.

  - Set the storage location to a directory in JindoFS when you create a database:

    ```
    CREATE DATABASE database_name
    LOCATION
    'jfs://namespace/database_dir';
    ```

    For example, to create a Hive database named database_on_jindofs and set the storage location to `jfs://emr-jfs/warehouse/database_on_jindofs`, run the following statement:

    ```
    CREATE DATABASE database_on_jindofs
    LOCATION
    'jfs://emr-jfs/hive/warehouse/database_on_jindofs';
    ```

- Modify the storage location of a database to a directory in JindoFS:
  a. Run the following SHOW CREATE statement to query the storage location of a database:

     ```
     SHOW CREATE DATABASE database_name;
     ```

  b. View the returned storage location. By default, the storage location of a database is that of its data warehouse added with the database name.

     ```
     CREATE DATABASE `database_name`
     LOCATION
     'hdfs://emr-jfs/user/hive/warehouse/database_name.db'
     ```

  c. Modify the storage location to a directory in JindoFS. This operation does not affect existing tables. If you do not specify the storage location for a new table, the modified location is used as the default storage location of the table.

     ```
     ALTER DATABASE database_name SET LOCATION jfs_path;
     ```

Query data in the scratch directory

Hive stores temporary output files and job plans to the scratch directory. You can set the hive.exec.scratchdir parameter in the hive-site configuration file to a directory in JindoFS. You can also set the parameter by running the following commands:

```
bin/hive --hiveconf hive.exec.scratchdir=jfs://emr-jfs/scratch_dir
```

or

```
set hive.exec.scratchdir=jfs://emr-jfs/scratch_dir;
```

5.4. Use Spark to process data in JindoFS

Spark processes data in JindoFileSystem (JindoFS) by calling methods or using Spark SQL to read data from tables stored in JindoFS.

JindoFS configuration

For example, a namespace named emr-jfs is created with the following configuration:

- jfs.namespaces=emr-jfs
- jfs.namespaces.emr-jfs.uri=oss://oss-bucket/oss-dir
- jfs.namespaces.emr-jfs.mode=block

Process data in JindoFS

- Call methods

  The read and write operations performed by Spark in JindoFS are similar to those in other file systems. For example, to access data in JindoFS, use a directory with the jfs prefix in the following resilient distributed dataset (RDD) operation:

```
val a = sc.textFile("jfs://emr-jfs/README.md")
```

To write data to JindoFS, call the following method:

```
scala> a.collect().saveAsTextFile("jfs://emr-jfs/output")
```

- Use Spark SQL

  Set the parameter that specifies the storage location to a directory in JindoFS when you create databases, tables, and partitions. For more information, see Use Hive to query data in JindoFS. Then, you can query data from tables stored in JindoFS.

5.5. Use Flink to process data in JindoFS

This topic describes how to use Flink to process data in JindoFileSystem (JindoFS).

JindoFS configuration

For example, a namespace named emr-jfs is created with the following configuration:

- jfs.namespaces=emr-jfs
- jfs.namespaces.emr-jfs.uri=oss://oss-bucket/oss-dir
- jfs.namespaces.emr-jfs.mode=block

Use JindoFS

You can set the input and output directories of Flink jobs to directories in a namespace supported by JindoFS. In this case, Flink jobs can read and write data in JindoFS.

For example, to store job data in Hadoop Distributed File System (HDFS), run the following command:

```
flink run -m yarn-cluster -yD taskmanager.network.memory.fraction=0.4 -yD akka.ask.timeout=60s -yjm 2048 -ytm 2048 -ys 4 -yn 14 -c xxx.xxx.FlinkWordCount -p 56 XXX.jar --input hdfs:///test//large-input-flink --output hdfs:///runjob/test/large-output-flink"
```

To store job data in JindoFS, run the following command:

```
flink run -m yarn-cluster -yD taskmanager.network.memory.fraction=0.4 -yD akka.ask.timeout=60s -yjm 2048 -ytm 2048 -ys 4 -yn 14 -c xxx.xxx.FlinkWordCount -p 56 XXX.jar --input jfs://emr-jfs/test/large-input-flink --output jfs://emr-jfs/test/large-output-flink"
```

5.6. Use Impala or Presto to query data in JindoFS

This topic describes how to use Impala or Presto to query data in JindoFileSystem (JindoFS).

JindoFS configuration

For example, a namespace named emr-jfs is created with the following configuration:

- jfs.namespaces=emr-jfs
- jfs.namespaces.emr-jfs.uri=oss://oss-bucket/oss-dir
- jfs.namespaces.emr-jfs.mode=block

Use JindoFS

Currently, Impala, and Presto services for E-MapReduce V3.22.0 and later can read Hive metadata from Hive tables stored in JindoFS.

To store table data in JindoFS, you can set the parameter that specifies the storage location in the CREATE TABLE statement to a directory in JindoFS.

For example, to store table data in Hadoop Distributed File System (HDFS), use the following CREATE TABLE statement:

```
Create external table lineitem (L_ORDERKEY INT, L_PARTKEY INT, L_SUPPKEY INT, L_LINENUMBER INT, L_QUANTIT
Y DOUBLE, L_EXTENDEDPRICE DOUBLE, L_DISCOUNT DOUBLE, L_TAX DOUBLE, L_RETURNFLAG STRING, L_LINESTAT
US STRING, L_SHIPDATE STRING, L_COMMITDATE STRING, L_RECEIPTDATE STRING, L_SHIPINSTRUCT STRING, L_SHI
PMODE STRING, L_COMMENT STRING) ROW FORMAT DELIMITED FIELDS TERMINATED BY '|' STORED AS TEXTFILE LO
CATION 'hdfs:///tpch_impala/lineitem';
```

To store table data in JindoFS, use the following CREATE TABLE statement:

```
Create external table lineitem (L_ORDERKEY INT, L_PARTKEY INT, L_SUPPKEY INT, L_LINENUMBER INT, L_QUANTIT
Y DOUBLE, L_EXTENDEDPRICE DOUBLE, L_DISCOUNT DOUBLE, L_TAX DOUBLE, L_RETURNFLAG STRING, L_LINESTAT
US STRING, L_SHIPDATE STRING, L_COMMITDATE STRING, L_RECEIPTDATE STRING, L_SHIPINSTRUCT STRING, L_SHI
PMODE STRING, L_COMMENT STRING) ROW FORMAT DELIMITED FIELDS TERMINATED BY '|' STORED AS TEXTFILE LO
CATION 'jfs://emr-jfs/tpch_impala/lineitem';
```

5.7. Use JindoFS as the storage back end of HBase

This topic describes how to use JindoFileSystem (JindoFS) as the storage back end of HBase.

Overview

In the Hadoop ecosystem, HBase is a database with high write performance that is suitable for real-time data query. HBase clusters of E-MapReduce V3.22.0 or later support JindoFS or JindoFS-based OssFileSystem as the storage back end. Compared with Hadoop Distributed File System (HDFS), JindoFS and JindoFS-based OssFileSystem are more flexible.

JindoFS configuration

For example, a namespace named emr-jfs is created with the following configuration:

- jfs.namespaces=emr-jfs
- jfs.namespaces.emr-jfs.uri=oss://oss-bucket/oss-dir

- jfs.namespaces.emr-jfs.mode=block

Specify the storage location for an HBase cluster

JindoFS and JindoFS-based OssFileSystem in E-MapReduce V3.22.0 do not support data synchronization. You can set the hbase.rootdir parameter in the hbase-site file to a directory in JindoFS or JindoFS-based OssFileSystem, and the hbase.wal.dir parameter to a local directory in HDFS. Then, you can store write-ahead logging (WAL) files through HDFS in an HBase cluster. To release an HBase cluster, disable tables first and make sure that the updates in WAL files have been written to HFile.

Configuration file	Parameter	Description	Example
hbase-site	hbase.rootdir	The root directory of the HBase cluster in JindoFS.	*jfs://emr-jfs/hbase-root-dir*
	hbase.wal.dir	The local directory in which the HBase cluster stores WAL files in HDFS.	*hdfs://emr-cluster/hbase*

Create an E-MapReduce cluster

You can add custom configuration when creating an E-MapReduce cluster, as shown in the following figure.

config_sel

Use JindoFS

For example, to use JindoFS as the storage back end of an HBase cluster, use the following custom configuration and replace the Object Storage Service (OSS) bucket and relevant directories:

```
[
 {
    "ServiceName":"BIGBOOT",
    "FileName":"bigboot",
    "ConfigKey":"jfs.namespaces",
    "ConfigValue":"emr-jfs"
 },
 {
    "ServiceName":"BIGBOOT",
    "FileName":"bigboot",
    "ConfigKey":"jfs.namespaces.emr-jfs.uri",
    "ConfigValue":"oss://oss-bucket/jindoFS"
 },
 {
    "ServiceName":"BIGBOOT",
    "FileName":"bigboot",
    "ConfigKey":"jfs.namespaces.emr-jfs.mode",
    "ConfigValue":"block"
 },
 {
    "ServiceName":"HBASE",
    "FileName":"hbase-site",
    "ConfigKey":"hbase.rootdir",
    "ConfigValue":"jfs://emr-jfs/hbase-root-dir"
 },
 {
    "ServiceName":"HBASE",
    "FileName":"hbase-site",
    "ConfigKey":"hbase.wal.dir",
    "ConfigValue":"hdfs://emr-cluster/hbase"
 }
]
```

5.8. Store the logs of YARN MapReduce and Spark jobs

This topic describes how to store the logs of MapReduce and Spark jobs to JindoFileSystem (JindoFS) or JindoFS-based OssFileSystem.

Overview

E-MapReduce clusters support the pay-as-you-go and subscription billing methods to meet different needs. Pay-as-you-go clusters can be released at any time. Hadoop clusters store logs in Hadoop Distributed File System (HDFS) by default. If a pay-as-you-go cluster is released, users cannot query job logs of the cluster. In this case, users may have difficulty in troubleshooting job problems. This topic describes how to store the logs of MapReduce and Spark jobs to JindoFS or JindoFS-based OssFileSystem so that you can query the previous logs of jobs.

Configure JindoFS, YARN Container logs, and Spark History Server

- JindoFS configuration

Configuration file	Parameter	Description	Example
bigboot	jfs.namespaces	The namespace supported by JindoFS. Separate multiple namespaces with commas (,).	emr-jfs
	jfs.namespaces.emr-jfs.uri	The storage back end of the emr-jfs namespace.	*oss://oss-bucket/oss-dir*
	jfs.namespaces.test.mode	The storage mode of the emr-jfs namespace.	block **Note** JindoFS supports the block storage mode and cache mode.

- Configuration for YARN Container logs

Configuration file	Parameter	Description	Example
yarn-site	yarn.nodemanager.remote-app-log-dir	The directory in which YARN aggregates and stores logs after your application stops running. The log aggregation feature of YARN is enabled by default.	*jfs://emr-jfs/emr-cluster-log/yarn-apps-logs* or *oss://${oss-bucket}/emr-cluster-log/yarn-apps-logs*
mapred-site	mapreduce.jobhistory.done-dir	The directory in which JobHistory stores the logs of Hadoop jobs that are completed.	*jfs://emr-jfs/emr-cluster-log/jobhistory/done* or *oss://${oss-bucket}/emr-cluster-log/jobhistory/done*

Configuration file	Parameter	Description	Example
	mapreduce.jobhistory.intermediate-done-dir	The directory in which JobHistory stores the logs that are not archived for Hadoop jobs.	*jfs://emr-jfs/emr-cluster-log/jobhistory/done_intermediate* or *oss://${oss-bucket}/emr-cluster-log/jobhistory/done_intermediate*

- Configuration for Spark History Server

Configuration file	Parameter	Description	Example
spark-defaults	spark_eventlog_dir	The directory in which Spark History Server stores the logs of Spark jobs.	*jfs://emr-jfs/emr-cluster-log/spark-history* or *oss://${oss-bucket}/emr-cluster-log/spark-history*

Create an E-MapReduce cluster

You can add custom configuration when creating an E-MapReduce cluster, as shown in the following figure.

config_sel

Custom configuration example

For example, to store logs in JindoFS, use the following custom configuration and replace the Object Storage Service (OSS) bucket and relevant directories:

```
[
 {
    "ServiceName":"BIGBOOT",
    "FileName":"bigboot",
    "ConfigKey":"jfs.namespaces",
    "ConfigValue":"emr-jfs"
 },
 {
    "ServiceName":"BIGBOOT",
    "FileName":"bigboot",
    "ConfigKey":"jfs.namespaces.emr-jfs.uri",
    "ConfigValue":"oss://oss-bucket/jindoFS"
 },
 {
    "ServiceName":"BIGBOOT",
    "FileName":"bigboot",
    "ConfigKey":"jfs.namespaces.emr-jfs.mode",
    "ConfigValue":"block"
```

```
  },
  {
    "ServiceName":"YARN",
    "FileName":"mapred-site",
    "ConfigKey":"mapreduce.jobhistory.done-dir",
    "ConfigValue":"jfs://emr-jfs/emr-cluster-log/jobhistory/done"
  },
  {
    "ServiceName":"YARN",
    "FileName":"mapred-site",
    "ConfigKey":"mapreduce.jobhistory.intermediate-done-dir",
    "ConfigValue":"jfs://emr-jfs/emr-cluster-log/jobhistory/done_intermediate"
  },
  {
    "ServiceName":"YARN",
    "FileName":"yarn-site",
    "ConfigKey":"yarn.nodemanager.remote-app-log-dir",
    "ConfigValue":"jfs://emr-jfs/emr-cluster-log/yarn-apps-logs"
  },
  {
    "ServiceName":"SPARK",
    "FileName":"spark-defaults",
    "ConfigKey":"spark_eventlog_dir",
    "ConfigValue":"jfs://emr-jfs/emr-cluster-log/spark-history"
  }
]
```

For example, to store logs in JindoFS-based OssFileSystem, use the following custom configuration and replace the OSS bucket and relevant directories:

```
[
 {
    "ServiceName":"YARN",
    "FileName":"mapred-site",
    "ConfigKey":"mapreduce.jobhistory.done-dir",
    "ConfigValue":"oss://oss_bucket/emr-cluster-log/jobhistory/done"
 },
 {
    "ServiceName":"YARN",
    "FileName":"mapred-site",
    "ConfigKey":"mapreduce.jobhistory.intermediate-done-dir",
    "ConfigValue":"oss://oss_bucket/emr-cluster-log/jobhistory/done_intermediate"
 },
 {
    "ServiceName":"YARN",
    "FileName":"yarn-site",
    "ConfigKey":"yarn.nodemanager.remote-app-log-dir",
    "ConfigValue":"oss://oss_bucket/emr-cluster-log/yarn-apps-logs"
 },
 {
    "ServiceName":"SPARK",
    "FileName":"spark-defaults",
    "ConfigKey":"spark_eventlog_dir",
    "ConfigValue":"oss://oss_bucket/emr-cluster-log/spark-history"
  }
]
```

5.9. Import data from Kafka to JindoFS

When you collect logs and aggregate monitoring data, you can use Apache Kafka to process offline data and streaming data and analyze data in real time. This topic describes how to import data from Kafka to JindoFileSystem (JindoFS).

Import methods

- Use Flume

  Apache Flume is a system used for moving data into Hadoop Distributed File System (HDFS). We recommend that you use Flume to import data from Kafka to JindoFS. To implement this feature, set the jfs.type parameter to hdfs and the jfs.hdfs.path parameter to a directory in JindoFS:

```
a1.sinks = emr-jfs


...


a1.sinks.emr-jfs.type = hdfs
a1.sinks.emr-jfs.hdfs.path = jfs://emr-jfs/kafka/%{topic}/%y-%m-%d
a1.sinks.emr-jfs.hdfs.rollInterval = 10
a1.sinks.emr-jfs.hdfs.rollSize = 0
a1.sinks.emr-jfs.hdfs.rollCount = 0
a1.sinks.emr-jfs.hdfs.fileType = DataStream
```

- Call a Kafka API

  Some engines like MapReduce and Spark call a Kafka API to export data from Kafka to HDFS. You only need to reference HDFS and set the export path to a directory in JindoFS.

- Use Kafka HDFS Connector

  You can also use Kafka HDFS Connector to export data from Kafka to HDFS by setting the sink export path to a directory in JindoFS.