ALIBABA CLOUD

阿里云

E-MapReduce

JindoFS

文档版本：20201027

阿里云

法律声明

阿里云提醒您在阅读或使用本文档之前仔细阅读、充分理解本法律声明各条款的内容。如果您阅读或使用本文档，您的阅读或使用行为将被视为对本声明全部内容的认可。

1. 您应当通过阿里云网站或阿里云提供的其他授权通道下载、获取本文档，且仅能用于自身的合法合规的业务活动。本文档的内容视为阿里云的保密信息，您应当严格遵守保密义务；未经阿里云事先书面同意，您不得向任何第三方披露本手册内容或提供给任何第三方使用。

2. 未经阿里云事先书面许可，任何单位、公司或个人不得擅自摘抄、翻译、复制本文档内容的部分或全部，不得以任何方式或途径进行传播和宣传。

3. 由于产品版本升级、调整或其他原因，本文档内容有可能变更。阿里云保留在没有任何通知或者提示下对本文档的内容进行修改的权利，并在阿里云授权通道中不时发布更新后的用户文档。您应当实时关注用户文档的版本变更并通过阿里云授权渠道下载、获取最新版的用户文档。

4. 本文档仅作为用户使用阿里云产品及服务的参考性指引，阿里云以产品及服务的“现状”、“有缺陷”和“当前功能”的状态提供本文档。阿里云在现有技术的基础上尽最大努力提供相应的介绍及操作指引，但阿里云在此明确声明对本文档内容的准确性、完整性、适用性、可靠性等不作任何明示或暗示的保证。任何单位、公司或个人因为下载、使用或信赖本文档而发生任何差错或经济损失的，阿里云不承担任何法律责任。在任何情况下，阿里云均不对任何间接性、后果性、惩戒性、偶然性、特殊性或刑罚性的损害，包括用户使用或信赖本文档而遭受的利润损失，承担责任（即使阿里云已被告知该等损失的可能性）。

5. 阿里云网站上所有内容，包括但不限于著作、产品、图片、档案、资讯、资料、网站架构、网站画面的安排、网页设计，均由阿里云和/或其关联公司依法拥有其知识产权，包括但不限于商标权、专利权、著作权、商业秘密等。非经阿里云和/或其关联公司书面同意，任何人不得擅自使用、修改、复制、公开传播、改变、散布、发行或公开发表阿里云网站、产品程序或内容。此外，未经阿里云事先书面同意，任何人不得为了任何营销、广告、促销或其他目的使用、公布或复制阿里云的名称（包括但不限于单独为或以组合形式包含“阿里云”、“Aliyun”、“万网”等阿里云和/或其关联公司品牌，上述品牌的附属标志及图案或任何类似公司名称、商号、商标、产品或服务名称、域名、图案标示、标志、标识或通过特定描述使第三方能够识别阿里云和/或其关联公司）。

6. 如若发现本文档存在任何错误，请与阿里云取得直接联系。

通用约定

格式	说明	样例
危险	该类警示信息将导致系统重大变更甚至故障，或者导致人身伤害等结果。	危险 重置操作将丢失用户配置数据。
警告	该类警示信息可能会导致系统重大变更甚至故障，或者导致人身伤害等结果。	警告 重启操作将导致业务中断，恢复业务时间约十分钟。
注意	用于警示信息、补充说明等，是用户必须了解的内容。	注意 权重设置为0，该服务器不会再接受新请求。
说明	用于补充说明、最佳实践、窍门等，不是用户必须了解的内容。	说明 您也可以通过按Ctrl+A选中全部文件。
>	多级菜单递进。	单击**设置**> **网络**> **设置网络类型**。
粗体	表示按键、菜单、页面名称等UI元素。	在**结果确认**页面，单击**确定**。
Courier字体	命令或代码。	执行 `cd /d C:/window` 命令，进入Windows系统文件夹。
斜体	表示参数、变量。	`bae log list --instanceid` *Instance_ID*
[] 或者 [a|b]	表示可选项，至多选择一个。	`ipconfig [-all|-t]`
{} 或者 {a|b}	表示必选项，至多选择一个。	`switch {active|stand}`

目录

1.JindoFS基础使用（EMR-3.27.0之前版本）

1.1. JindoFS使用说明（EMR-3.20.0~3.22.0版本）

本文主要介绍JindoFS的配置使用方式，以及一些典型的应用场景。

概述

JindoFS是一种云原生的文件系统，结合OSS和本地存储，成为E-MapReduce产品的新一代存储系统，为上层计算提供了高效可靠的存储。

JindoFS 提供了块存储模式（Block）和缓存模式（Cache）的存储模式。

JindoFS 采用了本地存储和OSS的异构多备份机制，Storage Service提供了数据存储能力，首先使用OSS作为存储后端，保证数据的高可靠性，同时利用本地存储实现冗余备份，利用本地的备份，可以加速数据读取；另外，JindoFS 的元数据通过本地服务Namespace Service管理，从而保证了元数据操作的性能（和HDFS元数据操作性能相似）。

> 说明
> - E-MapReduce-3.20.0及以上版本支持Jindo FS，您可以在创建集群时勾选相关服务来使用JindoFS。
> - 本文主要是E-MapReduce-3.20.0及以上版本至E-MapReduce-3.22.0（但不包括）版本的介绍；E-MapReduce-3.22.0 及以上版本的JindoFS 使用说明，请参见JindoFS使用说明（EMR-3.22.0~3.26.3版本）。

应用场景

E-MapReduce目前提供了三种大数据存储系统，E-MapReduce OssFileSystem、E-MapReduce HDFS和E-MapReduce JindoFS，其中OssFileSystem和JindoFS都是云上存储的解决方案，下表为这三种存储系统和开源OSS各自的特点。

特点	开源OSS	E-MapReduce OssFileSystem	E-MapReduce HDFS	E-MapReduce JindoFS
存储空间	海量	海量	取决于集群规模	海量
可靠性	高	高	高	高
吞吐率因素	服务端	集群内磁盘缓存	集群内磁盘	集群内磁盘
元数据效率	慢	中	快	快
扩容操作	容易	容易	容易	容易
缩容操作	容易	容易	需Decommission	容易
数据本地化	无	弱	强	较强

JindoFS块存储模式具有以下几个特点：

- 海量弹性的存储空间，基于OSS作为存储后端，存储不受限于本地集群，而且本地集群能够自由弹性伸缩。
- 能够利用本地集群的存储资源加速数据读取，适合具有一定本地存储能力的集群，能够利用有限的本地存储提升吞吐率，特别对于一写多读的场景效果显著。
- 元数据操作效率高，能够与HDFS相当，能够有效规避OSS文件系统元数据操作耗时以及高频访问下可能引发不稳定的问题。
- 能够最大限度保证执行作业时的数据本地化，减少网络传输的压力，进一步提升读取性能。

环境准备

- 创建集群

  选择E-MapReduce-3.20.0及以上版本至E-MapReduce-3.22.0（但不包括）版本，勾选可选服务中的**SmartData**和**Bigboot**，创建集群详情请参见创建集群。Bigboot 服务提供了E-MapReduce平台上的基础的分布式数据管理交互服务以及一些组件管理监控和支持性服务，SmartData服务基于Bigboot之上对应用层提供了JindoFS文件系统。

- 配置集群

  SmartData提供的JindoFS文件系统使用OSS作为存储后端，因此在使用JindoFS之前需配置一些OSS相关参数。下面提供两种配置方式，第一种是先创建好集群，修改Bigboot相关参数，需重启SmartData服务生效；第二种是创建集群过程中添加自定义配置，这样集群创建好后相关服务就能按照自定义参数启动。

  - 集群创建好后参数初始化
    - `oss.access.bucket` 为OSS bucket的名称。
    - `oss.data-dir` 为JindoFS在OSS bucket中所使用的目录（注：该目录为 JindoFS后端存储目录，生成的数据不能人为破坏，并且保证该目录仅用于JindoFS后端存储，JindoFS在写入数据时会自动创建用户所配置的目录，无需在OSS上事先创建）。
    - `oss.access.endpoint` 为bucket所在的区域。
    - `oss.access.key` 为存储后端OSS的AccessKey ID。
    - `oss.access.secret` 为存储后端OSS的AccessKey Secret。

    考虑到性能和稳定性，推荐使用同region下的OSS bucket作为存储后端，此时，E-MapReduce集群能够免密访问OSS，无需配置AccessKey ID和AccessKey Secret。

    所有JindoFS相关配置都在Bigboot组件中，配置如下图所示，红框中为必填的配置项。

    

    > **说明** JindoFS支持多命名空间，本文命名空间以test为例。

    配置完成后保存并部署，然后在SmartData服务中重启所有组件，即开始使用JindoFS。

    

  - 创建集群时添加自定义配置

    E-MapReduce集群在创建集群时支持添加自定义配置，以同region下免密访问OSS为例，如下图勾选**软件自定义配置**，添加如下配置，配置 `oss.data-dir` 和 `oss.access.bucket` 。

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

    

使用JindoFS

JindoFS使用上与HDFS类似，提供jfs前缀，将jfs替代hdfs即可使用。简单示例：

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

目前，JindoFS能够支持 E-MapReduce 集群上的 Hadoop、Hive、Spark的作业进行访问，其余组件尚未完全支持。

磁盘空间水位控制

JindoFS后端基于OSS，可以提供海量的存储，但是本地盘的容量是有限的，因此JindoFS会自动淘汰本地较冷的数据备份。我们提供了node.data-dirs.watermark.high.ratio和node.data-dirs.watermark.low.ratio这两个参数用来调节本地存储的使用容量，值均为0～1的小数表示使用比例，JindoFS默认使用所有数据盘，每块盘的使用容量默认即为数据盘大小。前者表示使用量上水位比例，每块数据盘的JindoFS占用的空间到达上水位即会开始清理淘汰；后者表示使用量下水位比例，触发清理后会将JindoFS的占用空间清理到下水位。用户可以通过设置上水位比例调节期望分给JindoFS的磁盘空间，下水位必须小于上水位，设置合理的值即可。

存储策略

JindoFS提供了Storage Policy功能，提供更加灵活的存储策略适应不同的存储需求，可以对目录设置以下四种存储策略。

策略	策略说明
COLD	表示数据仅在OSS上有一个备份，没有本地备份，适用于冷数据存储。
WARM	默认策略。 表示数据在OSS和本地分别有一个备份， 本地备份能够有效的提供后续的读取加速。
HOT	表示数据在OSS上有一个备份，本地有多个备份，针对一些最热的数据提供更进一步的加速效果。
TEMP	表示数据仅有一个本地备份，针对一些临时性数据，提供高性能的读写，但降低了数据的高可靠性，适用于一些临时数据的存取。

JindoFS提供了Admin工具设置目录的Storage Policy（默认为 WARM），新增的文件将会以父目录所指定的Storage Policy进行存储，使用方式如下所示。

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

通过以下命令，获取某个目录的存储策略。

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

> **说明** 其中*[path]*为设置policy的路径名称， *-R*表示递归设置该路径下的所有路径。

Admin工具还提供archive命令，实现对冷数据的归档。

此命令提供了一种用户显式淘汰本地数据块的方式。Hive分区表按天分区，假如业务上对一周前的分区数据认为不会再经常访问，那么就可以定期将一周前的分区目录执行archive，淘汰本地备份，文件备份将仅仅保留在后端OSS上。

Archive命令的使用方式如下：

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

> **说明** *[path]*为需要归档文件的所在目录路径。

1.2. JindoFS使用说明（EMR-3.22.0~3.26.3版本）

JindoFS是一种云原生的文件系统，结合OSS和本地存储，成为E-MapReduce产品的新一代存储系统，为上层计算提供了高效可靠的存储。本文主要说明JindoFS的配置使用方式，以及介绍一些典型的应用场景。

概述

JindoFS提供了块存储模式（Block）和缓存模式（Cache）的存储模式。

JindoFS采用了本地存储和OSS的异构多备份机制，Storage Service提供了数据存储能力，首先使用OSS作为存储后端，保证数据的高可靠性，同时利用本地存储实现冗余备份，利用本地的备份，可以加速数据读取；另外，JindoFS的元数据通过本地服务Namespace Service管理，从而保证了元数据操作的性能（和HDFS元数据操作性能相似）。

> 说明
>
> - E-MapReduce-3.20.0及以上版本支持Jindo FS，您可以在创建集群时勾选相关服务来使用JindoFS。
> - 本文主要是E-MapReduce-3.22.0及以上版本的介绍；E-MapReduce-3.20.0及以上版本至E-MapReduce-3.22.0（但不包括）版本的JindoFS使用说明，请参见JindoFS使用说明（EMR-3.20.0~3.22.0版本）。

环境准备

- 创建集群

  选择E-MapReduce-3.22.0及以上版本，勾选可选服务中的**SmartData**，创建集群详情请参见创建集群。

- 配置集群

  SmartData提供的JindoFS文件系统使用OSS作为存储后端，因此在使用JindoFS之前需配置一些OSS相关参数。下面提供两种配置方式，第一种是先创建好集群，修改Bigboot相关参数，需重启SmartData服务生效；第二种是创建集群过程中添加自定义配置，这样集群创建好后相关服务就能按照自定义参数启动：

- 集群创建好后初始化参数

  所有JindoFS相关配置都在Bigboot组件中，配置如下所示：

  a. 在**服务配置**页面，单击**bigboot**页签。

  

  b. 单击**自定义配置**。

  

  > **说明**
  > - 红框中为必填的配置项。
  > - JindoFS支持多命名空间，本文命名空间以test为例。

参数	参数说明	示例
jfs.namespaces	表示当前JindoFS支持的命名空间，多个命名空间时以逗号隔开。	test
jfs.namespaces.test.uri	表示test命名空间的后端存储。	*oss://oss-bucket/oss-dir* **说明** 该配置也可以配置到OSS bucket下的具体目录，该命名空间即以该目录作为根目录来读写数据。
jfs.namespaces.test.mode	表示test命名空间为块存储模式。	block **说明** JindoFS支持block和cache两种存储模式。
jfs.namespaces.test.oss.access.key	表示存储后端OSS的AccessKey ID。	*xxxx* **说明** 考虑到性能和稳定性，推荐使用同账户、同region下的OSS bucket作为存储后端，此时，E-MapReduce集群能够免密访问OSS，无需配置AccessKey ID和AccessKey Secret。
jfs.namespaces.test.oss.access.secret	表示存储后端OSS的AccessKey Secret。	

配置完成后保存并部署，然后在SmartData服务中重启所有组件，即开始使用JindoFS。

- 创建集群时添加自定义配置

  E-MapReduce集群在创建集群时支持添加自定义配置，以同region下免密访问OSS为例，如下图勾选**软件自定义配置**，配置命名空间test的相关配置，详情如下。

```
[
{
"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"
}
]
```

使用JindoFS

JindoFS使用上与HDFS类似，提供jfs前缀，将jfs替代hdfs即可使用。

目前，JindoFS能够支持EMR集群上的大部分计算组件，包括Hadoop、Hive、Spark、Flink、Presto和Impala。

简单示例：

- Shell命令

```
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 ./
```

- MapReduce作业

```
hadoop jar /usr/lib/hadoop-current/share/hadoop/mapreduce/hadoop-mapreduce-examples-2.8.5.jar teragen -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 terasort -Dmapred.reduce.tasks=1000 jfs://your-namespace/terasort/input jfs://your-namespace/terasort/output
```

- Spark-SQL

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

磁盘空间水位控制

JindoFS后端基于OSS，可以提供海量的存储，但是本地盘的容量是有限的，因此JindoFS会自动淘汰本地较冷的数据备份。我们提供了node.data-dirs.watermark.high.ratio和node.data-dirs.watermark.low.ratio这两个参数用来调节本地存储的使用容量，值均为0～1的小数表示使用比例，JindoFS默认使用所有数据盘，每块盘的使用容量默认即为数据盘大小。前者表示使用量上水位比例，每块数据盘的JindoFS占用的空间到达上水位即会开始清理淘汰；后者表示使用量下水位比例，触发清理后会将JindoFS的占用空间清理到下水位。用户可以通过设置上水位比例调节期望分给JindoFS的磁盘空间，下水位必须小于上水位，设置合理的值即可。

存储策略

JindoFS提供了Storage Policy功能，提供更加灵活的存储策略适应不同的存储需求，可以对目录设置以下四种存储策略。

策略	策略说明
COLD	表示数据仅在OSS上有一个备份，没有本地备份，适用于冷数据存储。
WARM	默认策略。 表示数据在OSS和本地分别有一个备份， 本地备份能够有效的提供后续的读取加速。
HOT	表示数据在OSS上有一个备份，本地有多个备份，针对一些最热的数据提供更进一步的加速效果。
TEMP	表示数据仅有一个本地备份，针对一些临时性数据，提供高性能的读写，但降低了数据的高可靠性，适用于一些临时数据的存取。

JindoFS提供了Admin工具设置目录的Storage Policy（默认为 WARM），新增的文件将会以父目录所指定的Storage Policy进行存储，使用方式如下。

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

通过以下命令，获取某个目录的存储策略：

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

> **说明** 其中*[path]*为设置policy的路径名称， *-R*表示递归设置该路径下的所有路径。

Admin工具

JindoFS提供了Admin工具的archive和jindo命令。

- Admin工具提供archive命令，实现对冷数据的归档。

  此命令提供了一种用户显式淘汰本地数据块的方式。Hive分区表按天分区，假如业务上对一周前的分区数据认为不会再经常访问，那么就可以定期将一周前的分区目录执行archive，淘汰本地备份，文件备份将仅仅保留在后端OSS上。

  Archive命令的使用方式如下。

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

  > **说明** *[path]*为需要归档文件的所在目录路径。

- Admin工具提供jindo命令，为Namespace Service提供了一些管理员功能命令。

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

  > **说明** 可以通过 `jindo dfsadmin --help` 命令获取帮助信息。

Admin工具对Cache模式提供了diff和sync命令。

- diff命令主要用来显示本地数据与后端存储系统数据之间的差异。

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

> **说明** 默认情况下比较 [path] 目录的子目录中元数据之间的差异， -R 选项表示递归比较 [path] 目录下所有的路径。

- sync命令用于同步本地与后端存储之前的元数据。

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

> **说明** [path] 表示需要同步元数据的路径，默认只会同步 [path] 的下一级目录， -R 选项表示递归比较 [path] 目录下所有的路径。

1.3. 使用JindoFS SDK免密功能

本文介绍使用JindoFS SDK时，E-MapReduce（简称EMR）集群外如何以免密方式访问E-MapReduce JindoFS的文件系统。

前提条件

适用环境：ECS（EMR环境外）+Hadoop+JavaSDK。

背景信息

使用JindoFS SDK时，需要把环境中相关Jindo的包从环境中移除，如*jboot.jar*、*smartdata-aliyun-jfs-*.jar*。如果要使用Spark则需要把*/opt/apps/spark-current/jars/*里面的包也删除，从而可以正常使用。

步骤一：创建实例RAM角色

1. 使用云账号登录RAM的控制台。
2. 单击左侧导航栏的**RAM角色管理**。
3. 单击**创建 RAM 角色**，选择当前可信实体类型为**阿里云服务**。
4. 单击**下一步**。
5. 输入**角色名称**，从**选择授信服务**列表中，选择**云服务器**。
6. 单击**完成**。

步骤二：为RAM角色授予权限

1. 使用云账号登录RAM的控制台。
2. （可选）如果您不使用系统权限，可以参见账号访问控制创建自定义权限策略章节创建一个自定义策略。
3. 单击左侧导航栏的**RAM角色管理**。
4. 单击新创建RAM角色名称所在行的**精确授权**。
5. 选择权限类型为**系统策略**或**自定义策略**。
6. 输入策略名称。
7. 单击**确定**。

步骤三：为实例授予RAM角色

1. 登录ECS管理控制台。
2. 在左侧导航栏，单击**实例与镜像 > 实例**。
3. 在顶部状态栏左上角处，选择地域。
4. 找到要操作的ECS实例，选择**更多 > 实例设置 > 授予/收回RAM角色**。

5. 在弹窗中，选择创建好的实例RAM角色，单击**确定**完成授予。

步骤四：在ECS上设置环境变量

执行如下命令，在ECS上设置环境变量。

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

或者执行如下命令。

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

步骤五：测试免密方式访问的方法

1. 使用Shell访问OSS。

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

2. 使用Hadoop FileSystem访问OSS。JindoFS SDK支持使用Hadoop FileSystem访问OSS，示例代码如下。

   ```
   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. JindoFS块存储模式

本文主要介绍JindoFS的块存储模式（Block），以及一些典型的应用场景。

概念

块存储模式提供了最为高效的数据读写能力和元数据访问能力，并且能够支持更加全面的Hadoop文件系统语义。同时，JindoFS也提供了外部客户端，能够从集群外部访问建立在E-MapReduce集群内的JindoFS文件系统。

数据以Block形式存储在后端存储OSS上，本地Namespace服务维护元数据信息，该模式在性能上较优，无论是数据性能还是元数据性能。

应用场景

E-MapReduce目前提供了三种大数据存储系统，E-MapReduce OssFileSystem、E-MapReduce HDFS和E-MapReduce JindoFS，其中OssFileSystem和JindoFS都是云上存储的解决方案，下表为这三种存储系统和开源OSS各自的特点。

特点	开源OSS	E-MapReduce OssFileSystem	E-MapReduce HDFS	E-MapReduce JindoFS
存储空间	海量	海量	取决于集群规模	海量
可靠性	高	高	高	高
吞吐率因素	服务端	集群内磁盘缓存	集群内磁盘	集群内磁盘
元数据效率	慢	中	快	快
扩容操作	容易	容易	容易	容易
缩容操作	容易	容易	需Decommission	容易
数据本地化	无	弱	强	较强

JindoFS块存储模式具有以下几个特点：

- 海量弹性的存储空间，基于OSS作为存储后端，存储不受限于本地集群，而且本地集群能够自由弹性伸缩。
- 能够利用本地集群的存储资源加速数据读取，适合具有一定本地存储能力的集群，能够利用有限的本地存储提升吞吐率，特别对于一写多读的场景效果显著。
- 元数据操作效率高，能够与HDFS相当，能够有效规避OSS文件系统元数据操作耗时以及高频访问下可能引发不稳定的问题。
- 能够最大限度保证执行作业时的数据本地化，减少网络传输的压力，进一步提升读取性能。

配置集群

所有JindoFS相关配置都在Bigboot组件中，配置如下图所示。

修改配置项

新增配置项

> **说明**
> - 红框中为必填的配置项。
> - JindoFS支持多命名空间，本文命名空间以test为例。

参数	参数说明	示例
jfs.namespaces	表示当前JindoFS支持的命名空间，多个命名空间时以逗号隔开。	test
jfs.namespaces.test.uri	表示test命名空间的后端存储。	*oss://oss-bucket/oss-dir* **说明** 该配置也可以配置到OSS bucket下的具体目录，该命名空间即以该目录作为根目录来读写数据。
jfs.namespaces.test.mode	表示test命名空间为块存储模式。	block

参数	参数说明	示例
jfs.namespaces.test.oss.access.key	表示存储后端OSS的AccessKey ID。	*xxxx*
jfs.namespaces.test.oss.access.secret	表示存储后端OSS的AccessKey Secret。	**说明** 考虑到性能和稳定性，推荐使用同账户、同region下的OSS bucket作为存储后端，此时，E-MapReduce集群能够免密访问OSS，无需配置AccessKey ID和AccessKey Secret。

配置完成后保存并部署，然后在SmartData服务中重启Namespace Service，即可开始使用JindoFS。

存储策略

JindoFS提供了Storage Policy功能，提供更加灵活的存储策略适应不同的存储需求，可以对目录设置以下四种存储策略。

策略	策略说明
COLD	表示数据仅在OSS上有一个备份，没有本地备份，适用于冷数据存储。
WARM	默认策略。 表示数据在OSS和本地分别有一个备份， 本地备份能够有效的提供后续的读取加速。
HOT	表示数据在OSS上有一个备份，本地有多个备份，针对一些最热的数据提供更进一步的加速效果。
TEMP	表示数据仅有一个本地备份，针对一些临时性数据，提供高性能的读写，但降低了数据的高可靠性，适用于一些临时数据的存取。

JindoFS提供了Admin工具设置目录的Storage Policy（默认为 WARM），新增的文件将会以父目录所指定的Storage Policy进行存储，使用方式如下所示。

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

通过以下命令，获取某个目录的存储策略。

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

说明 其中*[path]*为设置policy的路径名称， *-R*表示递归设置该路径下的所有路径。

Admin工具还提供archive命令，实现对冷数据的归档。

此命令提供了一种用户显式淘汰本地数据块的方式。Hive分区表按天分区，假如业务上对一周前的分区数据认为不会再经常访问，那么就可以定期将一周前的分区目录执行archive，淘汰本地备份，文件备份将仅仅保留在后端OSS上。

Archive命令的使用方式如下：

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

说明 *[path]*为需要归档文件的所在目录路径。

1.5. JindoFS缓存模式

本文主要介绍JindoFS的缓存模式（Cache），以及一些典型的应用场景。

概述

缓存模式兼容现有OSS存储方式，文件以对象的形式存储在OSS上，每个文件根据实际访问情况会在本地进行数据和元数据的缓存，从而提高访问数据以及元数据的性能，Cache模式提供不同元数据同步策略以满足您在不同场景下的需求。

应用场景

缓存模式最大的特点就是兼容性，保持了OSS原有的对象语义，集群中仅做缓存，因此JindoFS和OSS客户端、OssFileSystem等，或者其他的各种OSS的交互程序是完全兼容的，对原有OSS上的存量数据也不需要做任何的迁移、转换工作即可使用。同时集群中的数据和元数据缓存也能一定程度上提升数据访问性能。

配置集群

所有JindoFS相关配置都在Bigboot组件中，配置如下图所示。

修改配置项

新增配置项

> **说明**
> - 红框中为必填的配置项。
> - JindoFS支持多命名空间，本文命名空间以test为例。

参数	参数说明	示例
jfs.namespaces	表示当前JindoFS支持的命名空间，多个命名空间时以逗号隔开。	test
jfs.namespaces.test.uri	表示test命名空间的后端存储。	*oss://oss-bucket/* **说明** 该配置也可以配置到OSS bucket下的具体目录，该命名空间即以该目录作为根目录来读写数据，但一般情况下配置bucket即可，这样路径就和原生OSS保持一致。
jfs.namespaces.test.mode	表示test命名空间为缓存模式。	cache
jfs.namespaces.test.oss.access.key	表示存储后端OSS的AccessKey ID。	*xxxx*
jfs.namespaces.test.oss.access.secret	表示存储后端OSS的AccessKey Secret。	**说明** 考虑到性能和稳定性，推荐使用同账户、同region下的OSS bucket作为存储后端，此时，E-MapReduce集群能够免密访问OSS，无需配置AccessKey ID和AccessKey Secret。

配置完成后保存并部署，然后在SmartData服务中重启Namespace Service，即可开始使用JindoFS。

元数据同步策略

缓存模式下可能存在JindoFS集群构建之前，您已经在OSS上保存了大量数据的场景，对于这种场景，后续的数据访问会同步数据和元数据到JindoFS集群，数据同步策略为了访问数据都会在本地保留一份；元数据同步策略分为两部分，包括元数据同步间隔策略和元数据load策略：

- 元数据同步间隔策略：

  配置参数为**namespace.sync.interval**，该参数默认值为**-1**，表示不会同步OSS上的元数据。

  - 当**namespace.sync.interval=0**时，表示每次操作都会同步OSS上的元数据。
  - 当**namespace.sync.interval>0**时，表示会以固定的时间间隔来同步OSS上的元数据。

    > **说明** 例如当namespace.sync.interval=5时，表示每隔5秒会去同步OSS上的元数据。

- 元数据Load策略：

  配置参数为**namespace.sync.loadtype**，该配置参数为枚举类型**{never, once, always}**，**never**表示从不同步OSS上的元数据；**once**为默认配置，表示只从OSS同步一次元数据；**always**表示每次操作都会同步OSS上的元数据。

  > **说明** 当不配置namespace.sync.interval参数时，才会去使用Load策略；如果已配置namespace.sync.interval参数，则Load策略配置不生效。

1.6. JindoFS外部客户端

本文主要介绍JindoFS的外部客户端，以及一些典型的应用场景。

概述

JindoFS外部客户端，主要是为E-MapReduce集群外部访问JindoFS集群提供一种可行的方法。现在JindoFS外部客户端只能访问块存储模式下的JindoFS，不支持访问缓存模式下的JindoFS。实际上，缓存模式兼容OSS原始语义，因此外部访问仅需用普通OSS客户端即可。

应用场景

JindoFS外部客户端实现了Hadoop文件系统的接口，在用户程序跟E-MapReduce JindoFS Namespace服务网络相通的情况下，用户可以通过JindoFS外部客户端去访问JindoFS上存储的数据，但外部客户端不能利用E-MapReduce JindoFS的数据缓存能力，相比E-MapReduce集群内部访问JindoFS集群，性能有所损失。

配置外部客户端

已配置JindoFS块存储模式的Namespace，详情请参见JindoFS块存储模式。

1. 获取Bigboot程序包。

   在E-MapReduce集群内部*/usr/lib/bigboot-current*路径下，获取Bigboot程序包。

   > **说明** 一般情况下，程序使用Native开发，若实际系统类型与E-MapReduce集群差别较大，相关的程序需重新编译，可以通过联系我们处理。

2. 配置环境。

   设置环境变量**BIGBOOT_HOME**为程序安装根目录，将程序根目录下*ext*和*lib*的路径，添加到用户使用的大数据组件（Hadoop或Spark等）的**Classpath**中。

3. 从E-MapReduce集群内部拷贝配置文件*/usr/lib/bigboot-current/conf/bigboot.cfg.external*，到用户客户机上对应的安装目录*conf/bigboot.cfg*。

4. 配置Namespace Service。

   - `client.namespace.rpc.port`：配置JindoFS Namespace Service的监听端口。
   - `client.namespace.rpc.address`：配置JindoFS Namespace Service的监听地址。

     > **说明** 默认E-MapReduce集群中的配置文件已经配置好这两项。

5. 配置数据访问相关的配置项。

   - `client.namespaces.{YourNamespace}.oss.access.bucket`：配置OSS bucket选项。
   - `client.namespaces.{YourNamespace}.oss.access.endpoint`：配置OSS endpoint选项。

- `client.namespaces.{YourNamespace}.oss.access.key`：配置OSS的AccessKey ID。
- `client.namespaces.{YourNamespace}.oss.access.secret`：配置OSS的AccessKey Secret。

> **说明** 其中`{YourNamespace}`为外部客户端要访问的Namespace的名称，本文Namespace的名称以test为例。

配置示例如下。

```
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 = {YourOssAccessKeyID}
client.namespaces.test.oss.access.secret = {YourOssAccessKeySecret}
```

配置验证

验证如下信息：

- 通过以下命令，验证Namespace是否正确。

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

- 通过以下命令，验证数据是否可以上传或者下载。

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

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

2.JindoFS基础使用（EMR-3.27.x版本）

2.1. SmartData 2.6.x版本简介

SmartData的2.6.x版本，包含多个重大特性的发布以及大幅的性能优化。例如，Namespace服务后端存储支持Tablestore（OTS）以及Raft、Namespace服务支持HA、读写性能优化、块存储模式和缓存模式使用方式优化等。

元数据服务后端存储方案升级

在原有RocksDB方案的基础上，新版本推出了Tablestore和Raft的后端存储方案，实现元数据上云。

针对使用Cache模式且对于元数据存储以及HA没有高要求的场景，默认的RocksDB是一种简单、实用而且高效的方案。Tablestore和Raft的方案，实现了元数据服务的高可用，可以通过多个Namespace服务提供HA方案。

各方案详情请参见：

- 使用Tablestore作为存储后端
- 使用Raft-RocksDB-Tablestore作为存储后端
- 使用RocksDB作为元数据后端

使用模式优化

支持块存储模式和缓存模式两种使用模式：

- 块存储模式（Block）：使用方式与EMR 3.26.3之前版本基本一致，详情请参见JindoFS块存储模式使用说明。
- 缓存模式（Cache）：支持多种使用方式。例如，既支持与Block模式一致的使用方式，也支持原有OSS文件系统的使用方式，以满足用户不同的需要，详情请参见JindoFS缓存模式使用说明。

支持权限

Block模式支持Unix权限和Ranger权限两种文件系统权限功能：

- Unix权限：可以使用文件的777权限。
- Ranger权限：可以使用Ranger路径通配符等高级配置。

权限功能详细请参见JindoFS权限功能。

2.2. JindoFS块存储模式使用说明

块存储模式（Block）提供了最为高效的数据读写能力和元数据访问能力。数据以Block形式存储在后端存储OSS上，本地提供缓存加速，元数据则由本地Namespace服务维护，提供高效的元数据访问性能。本文主要介绍JindoFS的块存储模式及其使用方式。

背景信息

JindoFS块存储模式具有以下几个特点：

- 海量弹性的存储空间，基于OSS作为存储后端，存储不受限于本地集群，而且本地集群能够自由弹性伸缩。
- 能够利用本地集群的存储资源加速数据读取，适合具有一定本地存储能力的集群，能够利用有限的本地存储提升吞吐率，特别对于一写多读的场景效果显著。
- 元数据操作效率高，能够与HDFS相当，能够有效规避OSS文件系统元数据操作耗时以及高频访问下可能引发不稳定的问题。
- 能够最大限度保证执行作业时的数据本地化，减少网络传输的压力，进一步提升读取性能。

配置使用方式

1. 进入SmartData服务。
   i. 登录阿里云E-MapReduce控制台。
   ii. 在顶部菜单栏处，根据实际情况选择地域（Region）和资源组。
   iii. 单击上方的**集群管理**页签。
   iv. 在**集群管理**页面，单击相应集群所在行的**详情**。
   v. 在左侧导航栏单击**集群服务** > SmartData。
2. 进入bigboot服务配置。

i. 单击**配置**页签。

ii. 单击**bigboot**。

3. 配置以下参数。JindoFS支持多命名空间，本文命名空间以test为例。

i. 修改**jfs.namespaces**为**test**。**test**表示当前JindoFS支持的命名空间，多个命名空间时以逗号隔开。

ii. 单击**自定义配置**，在**新增配置项**对话框中增加以下参数，单击**确定**。

参数	参数说明	示例
jfs.namespaces.test.oss.uri	表示test命名空间的后端存储。	*oss://<oss_bucket>/<oss_dir>/* **说明** 推荐配置到OSS bucket下的某一个具体目录，该命名空间即会将Block模式的数据块存放在该目录下。
jfs.namespaces.test.mode	表示test命名空间为块存储模式。	block
jfs.namespaces.test.oss.access.key	表示存储后端OSS的AccessKey ID。	*xxxx* **说明** 考虑到性能和稳定性，推荐使用同账户、同Region下的OSS bucket作为存储后端，此时，E-MapReduce集群能够免密访问OSS，无需配置AccessKey ID和AccessKey Secret。
jfs.namespaces.test.oss.access.secret	表示存储后端OSS的AccessKey Secret。	

iii. 单击**确定**。

4. 单击右上角的**保存**。

5. 单击右上角的**操作 > 重启 Jindo Namespace Service**。重启后即可通过 `jfs://test/<path_of_file>` 的形式访问JindoFS上的文件。

磁盘空间水位控制

JindoFS后端基于OSS，可以提供海量的存储，但是本地盘的容量是有限的，因此JindoFS会自动淘汰本地较冷的数据备份。我们提供了 `storage.watermark.high.ratio` 和 `storage.watermark.low.ratio` 两个参数来调节本地存储的使用容量，值均为0~1的小数，表示使用磁盘空间的比例。

1. 修改磁盘水位配置。

可在**服务配置**区域的**storage**页签，修改以下参数。

参数	描述
storage.watermark.high.ratio	表示磁盘使用量的上水位比例，每块数据盘的JindoFS数据目录占用的磁盘空间到达上水位即会触发清理。默认值：0.4。
storage.watermark.low.ratio	表示使用量的下水位比例，触发清理后会自动清理冷数据，将JindoFS数据目录占用空间清理到下水位。默认值：0.2。

说明 您可以通过设置上水位比例调节期望分给JindoFS的磁盘空间，下水位必须小于上水位，设置合理的值即可。

2. 保存配置。

i. 单击右上角的**保存**。

ii. 在**确认修改**对话框中，输入执行原因，开启**自动更新配置**。

iii. 单击**确定**。

3. 重启Jindo Storage Service使配置生效。

i. 单击右上角的**操作 > 重启Jindo Storage Service**。

ii. 在**执行集群操作**对话框中，设置相关参数。

iii. 单击**确定**。

iv. 在**确认**对话框中，单击**确定**。

2.3. JindoFS缓存模式使用说明

缓存模式（Cache）主要兼容原生OSS存储方式，文件以对象的形式存储在OSS上，每个文件根据实际访问情况会在本地进行缓存，提升EMR集群内访问OSS的效率，同时兼容了原有OSS原有文件形式，数据访问上能够与其他OSS客户端完全兼容。本文主要介绍JindoFS的缓存模式及其使用方式。

背景信息

缓存模式最大的特点就是兼容性，保持了OSS原有的对象语义，集群中仅做缓存，因此和其他的各种OSS客户端是完全兼容的，对原有OSS上的存量数据也不需要做任何的迁移、转换工作即可使用。同时集群中的缓存也能一定程度上提升数据访问性能，缓解读写OSS的带宽压力。

配置使用方式

JindoFS缓存模式提供了以下两种基本使用方式，以满足不同的使用需求。

- OSS Scheme

  详情请参见配置OSS Scheme（推荐）。

- JFS Scheme

  详情请参见配置JFS Scheme。

配置OSS Scheme（推荐）

OSS Scheme保留了原有OSS文件系统的使用习惯，即直接通过 `oss://<bucket_name>/<path_of_your_file>` 的形式访问OSS上的文件。使用该方式访问OSS，无需进行额外的配置，创建EMR集群后即可使用，对于原有读写OSS的作业也无需做任何修改即可运行。

配置JFS Scheme

1. 进入SmartData服务。

i. 登录阿里云E-MapReduce控制台。

ii. 在顶部菜单栏处，根据实际情况选择地域（Region）和资源组。

iii. 单击上方的**集群管理**页签。

iv. 在**集群管理**页面，单击相应集群所在行的**详情**。

v. 在左侧导航栏单击**集群服务 > SmartData**。

2. 进入bigboot服务配置。

i. 单击**配置**页签。

ii. 单击**bigboot**。

3. 配置以下参数。JindoFS支持多命名空间，本文命名空间以test为例。

i. 修改**jfs.namespaces**为**test**。**test**表示当前JindoFS支持的命名空间，多个命名空间时以逗号隔开。

ii. 单击**自定义配置**，在**新增配置项**对话框中增加以下参数。

参数	参数说明	示例
jfs.namespaces.test.oss.uri	表示test命名空间的后端存储。	*oss://<oss_bucket>/<oss_dir>/* **说明** 该配置必须配置到OSS Bucket下的具体目录，也可以直接使用根目录。
jfs.namespaces.test.mode	表示test命名空间为缓存模式。	cache

4. 单击右上角的**保存**。
5. 单击右上角的**操作 > 重启 Jindo Namespace Service**。重启后即可通过 `jfs://test/<path_to_your_file>` 的形式访问，该命名空间下的文件会以**jfs.namespaces.test.oss.uri**所配置的目录作为根目录进行组织，例如 `jfs://test/hello.txt` 对应实际OSS上的文件为*oss://<oss_bucket>/<oss_dir>/hello.txt*。

启用缓存

启用缓存会利用本地磁盘对访问的热数据块进行缓存，默认状态为禁用，即所有OSS读取都直接访问OSS上的数据。

1. 在**集群服务 > SmartData**的**配置**页面，单击**client**页签。
2. 修改**jfs.cache.data-cache.enable**为**1**，表示启用缓存。此配置为客户端配置，不需要重启SmartData服务。

缓存启用后，Jindo服务会自动管理本地缓存备份，通过水位清理本地缓存，请您根据需求配置一定的比例用于缓存，详情请参见磁盘空间水位控制。

磁盘空间水位控制

JindoFS后端基于OSS，可以提供海量的存储，但是本地盘的容量是有限的，因此JindoFS会自动淘汰本地较冷的数据备份。我们提供了 `storage.watermark.high.ratio` 和 `storage.watermark.low.ratio` 两个参数来调节本地存储的使用容量，值均为0～1的小数，表示使用磁盘空间的比例。

1. 修改磁盘水位配置。

   可在**服务配置**区域的**storage**页签，修改以下参数。

参数	描述
storage.watermark.high.ratio	表示磁盘使用量的上水位比例，每块数据盘的JindoFS数据目录占用的磁盘空间到达上水位即会触发清理。默认值：0.4。
storage.watermark.low.ratio	表示使用量的下水位比例，触发清理后会自动清理冷数据，将JindoFS数据目录占用空间清理到下水位。默认值：0.2。

   说明 您可以通过设置上水位比例调节期望分给JindoFS的磁盘空间，下水位必须小于上水位，设置合理的值即可。

2. 保存配置。
   i. 单击右上角的**保存**。
   ii. 在**确认修改**对话框中，输入执行原因，开启**自动更新配置**。
   iii. 单击**确定**。
3. 重启Jindo Storage Service使配置生效。
   i. 单击右上角的**操作 > 重启Jindo Storage Service**。
   ii. 在**执行集群操作**对话框中，设置相关参数。
   iii. 单击**确定**。
   iv. 在**确认**对话框中，单击**确定**。

访问OSS Bucket

在EMR集群中访问同账号、同区域的OSS Bucket时，默认支持免密访问，即无需配置任何AccessKey即可访问。如果访问非以上情况的OSS Bucket需要配置相应的AccessKey ID、AccessKey Secret以及Endpoint，针对两种使用方式相应的配置分别如下：

- OSS Scheme
  i. 在**集群服务 > SmartData的配置**页面，单击**smartdata-site**页签。
  ii. 单击**自定义配置**，在**新增配置项**对话框中增加以下参数，单击**确定**。

参数	参数说明
fs.jfs.cache.oss-accessKeyId	表示存储后端OSS的AccessKey ID。
fs.jfs.cache.oss-accessKeySecret	表示存储后端OSS的AccessKey Secret。
fs.jfs.cache.oss-endpoint	表示存储后端OSS的endpoint。

- JFS Scheme
  i. 在**集群服务 > SmartData的配置**页面，单击**bigboot**页签。
  ii. 修改**jfs.namespaces**为**test**。
  iii. 单击**自定义配置**，在**新增配置项**对话框中增加以下参数，单击**确定**。

参数	参数说明
jfs.namespaces.test.oss.uri	表示test命名空间的后端存储。示例：*oss://<oss_bucket.endpoint>/<oss_dir>*。 endpoint信息直接配置在oss.uri中。
jfs.namespaces.test.oss.access.key	表示存储后端OSS的AccessKey ID。
jfs.namespaces.test.oss.access.secret	表示存储后端OSS的AccessKey Secret。

高级配置

Cache模式还包含一些高级配置，用于性能调优，以下配置均为客户端配置，修改后无需重启SmartData服务。

- 在**服务配置**区域的**client**页签，配置以下参数。

参数	参数说明
client.oss.upload.threads	每个文件写入流的OSS上传线程数。默认值：4。
client.oss.upload.max.parallelism	进程级别OSS上传总并发度上限，防止过多上传线程造成过大的带宽压力以及过大的内存消耗。默认值：16。

- 在**服务配置**区域的**smartdata-site**页签，配置以下参数。

参数	参数说明
fs.jfs.cache.copy.simple.max.byte	rename过程使用普通copy接口的文件大小上限（小于阈值的使用普通 copy接口，大于阈值的使用multipart copy接口以提高copy效率）。 **说明** 如果确认已开通OSS fast copy功能，参数值设为-1，表示所有大小均使用普通copy接口，从而有效利用fast copy获得最优的rename性能。
fs.jfs.cache.write.buffer.size	文件写入流的buffer大小，参数值必须为2的幂次，最大为8MB，如果作业同时打开的写入流较多导致内存使用过大，可以适当调小此参数。默认值：1048576。

参数	参数说明
fs.oss.committer.magic.enabled	启用Jindo Job Committer，避免Job Committer的rename操作，来提升性能。默认值：true。 **说明** 针对Cache模式下，由于OSS这类对象存储rename操作性能较差的问题，推出了Jindo Job Committer。

2.4. JindoFS元数据服务

2.4.1. 使用Tablestore作为存储后端

JindoFS元数据服务支持不同的存储后端，本文介绍使用Tablestore（OTS）作为元数据后端时需要进行的配置。

前提条件

- 已创建EMR集群。

  详情请参见创建集群。
- 已创建Tablestore实例，推荐使用高性能实例。

  详情请参见创建实例。

  说明 需要开启事务功能。

背景信息

JindoFS在新版本中，支持使用Tablestore作为JindoFS元数据服务（Namespace Service）的存储。一个EMR JindoFS集群可以绑定一个Tablestore实例（Instance）作为JindoFS元数据服务的存储介质，元数据服务会自动为每个Namespace创建独立的Tablestore表进行管理和存储元数据信息。

元数据服务（双机Tablestore和HA）架构图如下所示。

配置Tablestore

使用Tablestore功能，需要把创建的Tablestore实例和JindoFS的Namespace服务进行绑定，详细步骤如下：

1. 进入SmartData服务。
   i. 登录阿里云E-MapReduce控制台。
   ii. 在顶部菜单栏处，根据实际情况选择地域（Region）和资源组。
   iii. 单击上方的**集群管理**页签。
   iv. 在**集群管理**页面，单击相应集群所在行的**详情**。
   v. 在左侧导航栏单击**集群服务 > SmartData**。
2. 进入bigboot服务配置。
   i. 单击**配置**页签。
   ii. 单击**bigboot**。

   
3. 配置以下参数。例如，在华东1（杭州）地域下，创建了emr-jfs的Tablestore实例，EMR集群使用VPC网络，访问Tablestore的AccessKey ID为kkkkkk，Access Secret为XXXXXX。

参数	参数说明	是否必选	示例

参数	参数说明	是否必选	示例
namespace.backend.type	设置namespace后端存储类型，支持： ○ rocksdb ○ ots ○ raft 默认为rocksdb。	是	ots
namespace.ots.instance	Tablestore实例名称。	是	emr-jfs
namespace.ots.accessKey	Tablestore实例的AccessKey ID。	否	kkkkkk
namespace.ots.accessSecret	Tablestore实例的AccessKey Secret。	否	XXXXXX
namespace.ots.endpoint	Tablestore实例的Endpoint地址，普通EMR集群，推荐使用VPC地址。	是	*http://emr-jfs.cn-hangzhou.vpc.tablestore.aliyuncs.com*

4. 保存配置。
   i. 单击右上角的**保存**。
   ii. 在**确认修改**对话框中，输入执行原因，开启**自动更新配置**。
   iii. 单击**确定**。
5. 单击右上角的**操作 > 重启 Jindo Namespace Service**。

配置Tablestore（高可用方案）

针对EMR的高可用集群，可以通过配置开启Namespace高可用模式。

Namespace高可用模式采用Active和Standby互备方式，支持自动故障转移，当Active Namespace出现异常或者异常中止时，客户端可以请求自动切换到新的Active节点。

1. 进入SmartData的bigboot服务配置，配置以下参数。
   i. 修改**jfs.namespace.server.rpc-address**值为**emr-header-1:8101,emr-header-2:8101**。
   ii. 单击右上角的**自定义配置**，添加**namespace.backend.ots.ha**为**true**。
   iii. 单击**确定**。
   iv. 保存配置。
      a. 单击右上角的**保存**。
      b. 在**确认修改**对话框中，输入执行原因，开启**自动更新配置**。
      c. 单击**确定**。
2. 单击右上角的**操作 > 重启Jindo Namespace Service**。
3. 单击右上角的**操作 > 重启Jindo Storage Service**。

2.4.2. 使用RocksDB作为元数据后端

JindoFS元数据服务支持不同的存储后端，默认配置RocksDB为元数据存储后端。本文介绍使用RocksDB作为元数据后端时需要进行的相关配置。

背景信息

RocksDB作为元数据后端时不支持高可用。如果需要高可用，推荐配置Tablestore（OTS）或者Raft作为元数据后端，详情请参

见使用Tablestore作为存储后端和使用Raft-RocksDB-Tablestore作为存储后端。

单机RocksDB作为元数据服务的架构图如下所示。

配置RocksDB

1. 进入SmartData服务。
   i. 登录阿里云E-MapReduce控制台。
   ii. 在顶部菜单栏处，根据实际情况选择地域（Region）和资源组。
   iii. 单击上方的**集群管理**页签。
   iv. 在**集群管理**页面，单击相应集群所在行的**详情**。
   v. 在左侧导航栏单击**集群服务 > SmartData**。
2. 进入bigboot服务配置。
   i. 单击**配置**页签。
   ii. 单击**bigboot**。

   
3. 设置**namespace.backend.type**为**rocksdb**。
4. 保存配置。
   i. 单击右上角的**保存**。
   ii. 在**确认修改**对话框中，输入执行原因，开启**自动更新配置**。
   iii. 单击**确定**。
5. 单击右上角的**操作 > 重启 Jindo Namespace Service**。

2.4.3. 使用Raft-RocksDB-Tablestore作为存储后端

JindoFS在EMR-3.27.0及之后版本中支持使用Raft-RocksDB-OTS作为Jindo元数据服务（Namespace Service）的存储。1个EMR JindoFS集群创建3个Master节点组成1个Raft实例，实例的每个Peer节点使用本地RocksDB存储元数据信息。

前提条件

- 创建Tablestore实例，推荐使用高性能实例，详情请参见创建实例。

  > **说明** 需要开启事务功能。

- 创建3 Master的EMR集群，详情请参见创建集群。

  

  > **说明** 如果没有部署方式，请提交工单处理。

背景信息

RocksDB通过Raft协议实现3个节点之间的复制。集群可以绑定1个Tablestore（OTS）实例，作为Jindo的元数据服务的额外存储介质，本地的元数据信息会实时异步地同步到用户的Tablestore实例上。

元数据服务-多机Raft-RocksDB-Tablestore+HA如下图所示。

配置本地raft后端

1. 新建EMR集群后，暂停SmartData所有服务。
   i. 登录阿里云E-MapReduce控制台。
   ii. 在顶部菜单栏处，根据实际情况选择地域（Region）和资源组。
   iii. 单击上方的**集群管理**页签。

iv. 在**集群管理**页面，单击相应集群所在行的**详情**。

v. 在左侧导航栏，单击**集群服务 > SmartData**。

vi. 单击右上角的**操作 > 停止 All Components**。

2. 根据使用需求，添加需要的namespace。

3. 进入SmartData服务的**bigboot**页签。

i. 在左侧导航栏单击**集群服务 > SmartData**。

ii. 单击**配置**页签。

iii. 在**服务配置**区域，单击**bigboot**页签。

4. 在SmartData服务的**bigboot**页签，设置以下参数。

参数	描述	示例
namespace.backend.type	设置namespace后端存储类型，支持： ○ rocksdb ○ ots ○ raft 默认为rocksdb。	raft
namespace.backend.raft.initial-conf	部署raft实例的3个Master地址（固定值）。	emr-header-1:8103:0,emr-header-2:8103:0,emr-header-3:8103:0
jfs.namespace.server.rpc-address	Client端访问raft实例的3个Master地址（固定值）	emr-header-1:8101,emr-header-2:8101,emr-header-3:8101

> **说明** 如果不需要使用OTS远端存储，直接执行步骤6和步骤7；如果需要使用OTS远端存储，请执行步骤5~步骤7。

5. （可选）配置远端OTS异步存储。在SmartData服务的**bigboot**页签，设置以下参数。

参数	参数说明	示例
namespace.ots.instance	Tablestore实例名称。	emr-jfs
namespace.ots.accessKey	Tablestore实例的AccessKey ID。	*kkkkkk*
namespace.ots.accessSecret	Tablestore实例的AccessKey Secret。	*XXXXXX*
namespace.ots.endpoint	Tablestore实例的endpoint地址，通常EMR集群，推荐使用VPC地址。	*http://emr-jfs.cn-hangzhou.vpc.tablestore.aliyuncs.com*
namespace.backend.raft.async.ots.enabled	是否开启OTS异步上传，包括： ○ true ○ false 当设置为true时，需要在SmartData服务完成初始化前，开启OTS异步上传功能。 **说明** 如果SmartData服务已完成初始化，则不能再开启该功能。因为OTS的数据已经落后于本地RocksDB的数据。	true

6. 保存配置。

i. 单击右上角的**保存**。

ii. 在**确认修改**对话框中，输入执行原因，开启**自动更新配置**。

iii. 单击**确定**。

7. 单击右上角的**操作 > 启动 All Components**。

从Tablestore恢复元数据信息

如果您在原始集群开启了远端Tablestore异步存储，则Tablestore上会有1份完整的JindoFS元数据的副本。您可以在停止或释放原始集群后，在新创建的集群上恢复原先的元数据，从而继续访问之前保存的文件。

1. （可选）准备工作。
   i. （可选）统计原始集群的元数据信息（文件和文件夹数量）。

```
[hadoop@emr-header-1 ~]$ hadoop fs -count jfs://test/
      1596     1482809              25 jfs://test/
  (文件夹个数）（文件个数）
```

   ii. 停止原始集群的作业，等待30~120秒左右，等待原始集群的数据已经完全同步到Tablestore。执行以下命令查看状态。如果LEADER节点显示 `_synced=1` ，则表示Tablestore为最新数据，同步完成。

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

   iii. 停止或释放原始集群，确保没有其它集群正在访问当前的Tablestore实例。
2. 创建新集群。新建与Tablestore实例相同Region的EMR集群，暂停SmartData所有服务。详情请参见配置本地raft后端中的步骤1。
3. 初始化配置。在SmartData服务的**bigboot**页签，设置以下参数。

参数	描述	示例
namespace.backend.raft.async.ots.enabled	是否开启OTS异步上传，包括： ○ true ○ false	false
namespace.backend.raft.recovery.mode	是否开启从OTS恢复元数据，包括： ○ true ○ false	true

4. 保存配置。
   i. 单击右上角的**保存**。
   ii. 在**确认修改**对话框中，输入执行原因，开启**自动更新配置**。
   iii. 单击**确定**。
5. 单击右上角的**操作 > 启动 All Components**。
6. 新集群的SmartData服务启动后，自动从OTS恢复元数据到本地Raft-RocksDB上，可以通过以下命令查看恢复进度。

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

   如图所示，LEADER节点的state为FINISH表示恢复完成。
7. （可选）执行以下操作，可以比较一下文件数量与原始集群是否一致。此时的集群为恢复模式，也是只读模式。

```
# 对比文件数量一致
[hadoop@emr-header-1 ~]$ hadoop fs -count jfs://test/
      1596     1482809              25 jfs://test/

# 文件可正常读取(cat、get命令)
[hadoop@emr-header-1 ~]$ hadoop fs -cat jfs://test/testfile
this is a test file
# 查看目录
[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

# 只读状态，不可修改文件
[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. 修改配置，将集群设置为正常模式，开启OTS异步上传功能。在SmartData服务的**bigboot**页签，设置以下参数。

参数	描述	示例
namespace.backend.raft.async.ots.enabled	是否开启OTS异步上传，包括： ○ true ○ false	true
namespace.backend.raft.recovery.mode	是否开启从OTS恢复元数据，包括： ○ true ○ false	false

9. 重启集群。
   i. 单击上方的**集群管理**页签。
   ii. 在**集群管理**页面，单击相应集群所在行的**更多 > 重启**。

2.5. JindoFS权限功能

本文介绍JindoFS的Block模式支持的文件系统权限功能，包括Unix权限和Ranger权限两种。

背景信息

您可以在Apache Ranger组件上配置用户权限，在JindoFS上开启Ranger插件后，就可以在Ranger上对JindoFS权限（和其它组件权限）进行一站式管理。

Block模式支持Unix权限和Ranger权限两种文件系统权限功能：

- Unix权限：可以使用文件的777权限。
- Ranger权限：可以使用Ranger路径通配符等高级配置。

启用JindoFS Unix权限

1. 进入SmartData服务。
   i. 登录阿里云E-MapReduce控制台。
   ii. 在顶部菜单栏处，根据实际情况选择地域（Region）和资源组。

iii. 单击上方的**集群管理**页签。

iv. 在**集群管理**页面，单击相应集群所在行的**详情**。

v. 在左侧导航栏单击**集群服务 > SmartData**。

2. 进入bigboot服务配置。

i. 单击**配置**页签。

ii. 单击**bigboot**。

3. 单击**自定义配置**，在**新增配置项**对话框中，设置**Key**为jfs.namespaces.<namespace>.permission.method，**Value**为unix，单击**确定**。

4. 保存配置。

i. 单击右上角的**保存**。

ii. 在**确认修改**对话框中，输入执行原因，开启**自动更新配置**。

iii. 单击**确定**。

5. 重启配置。

i. 单击右上角的**操作 > 重启 Jindo Namespace Service**。

ii. 输入执行原因，单击**确定**。

开启文件系统权限后，使用方式跟HDFS一样。支持以下命令。

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

如果用户对某一个文件没有权限，将返回如下错误信息。

启用JindoFS Ranger权限

1. 添加Ranger。

i. 在**bigboot**页签，单击**自定义配置**。

ii. 在**新增配置项**对话框中，设置**Key**为jfs.namespaces.<namespace>.permission.method，**Value**为ranger。

iii. 保存配置。

a. 单击右上角的**保存**。

b. 在**确认修改**对话框中，输入执行原因，开启**自动更新配置**。

c. 单击**确定**。

iv. 重启配置。

a. 单击右上角的**操作 > 重启 Jindo Namespace Service**。

b. 输入执行原因，单击**确定**。

2. 配置Ranger。

i. 进入Ranger UI页面。详情请参见概述。

ii. Ranger UI添加HDFS service。

iii. 配置相关参数。

参数	说明
Service Name	jfs-{namespace_name}。
Username	自定义。
Password	自定义。
Namenode URL	输入*jfs://{namespace_name}*。
Authorization Enabled	使用默认值No。
Authentication Type	使用默认值Simple。
dfs.datanode.kerberos.principal	不填写。
dfs.namenode.kerberos.principal	
dfs.secondary.namenode.kerberos.principal	
Add New Configurations	

iv. 单击**Add**。

启用JindoFS Ranger权限+LDAP用户组

如果您在Ranger UserSync上开启了从LDAP同步用户组信息的功能，则JindoFS也需要修改相应的配置，才能获取LDAP的用户组信息，从而对当前用户组进行Ranger权限的校验。

1. 在**bigboot**页签，单击**自定义配置**。
2. 在**新增配置项**对话框中，设置以下参数配置LDAP，单击**确定**。

参数	示例
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
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

> **说明** 配置项请遵循开源HDFS内容。

3. 保存配置。
   i. 单击右上角的**保存**。
   ii. 在**确认修改**对话框中，输入执行原因，开启**自动更新配置**。
   iii. 单击**确定**。
4. 重启配置。

i. 单击右上角的**操作 > 重启 All Components**。

ii. 输入执行原因，单击**确定**。

5. 通过SSH登录emr-header-1节点，配置Ranger UserSync并启用LDAP选项。详情请参见Ranger Usersync集成LDAP。

2.6. Jindo Job Committer使用说明

本文主要介绍JindoOssCommitter的使用说明。

背景信息

Job Committer是MapReduce和Spark等分布式计算框架的一个基础组件，用来处理分布式任务写数据的一致性问题。

Jindo Job Committer是阿里云E-MapReduce针对OSS场景开发的高效Job Committer的实现，基于OSS的Multipart Upload接口，结合OSS Filesystem层的定制化支持。使用Jindo Job Committer时，Task数据直接写到最终目录中，在完成Job Commit前，中间数据对外不可见，彻底避免了Rename操作，同时保证数据的一致性。

> 注意
>
> - OSS拷贝数据的性能，针对不同的用户或Bucket会有差异，可能与OSS带宽以及是否开启某些高级特性等因素有关，具体问题可以咨询OSS的技术支持。
> - 在所有任务都完成后，MapReduce Application Master或Spark Driver执行最终的Job Commit操作时，会有一个短暂的时间窗口。时间窗口的大小和文件数量线性相关，可以通过增大 `fs.oss.committer.threads` 可以提高并发处理的速度。
> - Hive和Presto等没有使用Hadoop的Job Committer。
> - E-MapReduce集群中默认打开Jindo Oss Committer的参数。

在MapReduce中使用Jindo Job Committer

1. 进入YARN服务的**mapred-site**页签。
   i. 登录阿里云E-MapReduce控制台。
   ii. 在顶部菜单栏处，根据实际情况选择地域（Region）和资源组。
   iii. 单击上方的**集群管理**页签。
   iv. 在**集群管理**页面，单击相应集群所在行的**详情**。
   v. 在左侧导航栏单击**集群服务 > YARN**。
   vi. 单击**配置**页签。
   vii. 在**服务配置**区域，单击**mapred-site**页签。
2. 针对Hadoop不同版本，在YARN服务中配置以下参数。
   - Hadoop 2.x版本

     在YARN服务的**mapred-site**页签，设置**mapreduce.outputcommitter.class**为com.aliyun.emr.fs.oss.commit.JindoOssCommitter。
   - Hadoop 3.x版本

     在YARN服务的**mapred-site**页签，设置**mapreduce.outputcommitter.factory.scheme.oss**为com.aliyun.emr.fs.oss.commit.JindoOssCommitterFactory。
3. 保存配置。
   i. 单击右上角的**保存**。
   ii. 在**确认修改**对话框中，输入执行原因，开启**自动更新配置**。
   iii. 单击**确定**。
4. 进入SmartData服务的**smartdata-site**页签。
   i. 在左侧导航栏单击**集群服务 > SmartData**。
   ii. 单击**配置**页签。
   iii. 在**服务配置**区域，单击**smartdata-site**页签。
5. 在SmartData服务的**smartdata-site**页签，设置**fs.oss.committer.magic.enabled**为true。

6. 保存配置。
   i. 单击右上角的**保存**。
   ii. 在**确认修改**对话框中，输入执行原因，开启**自动更新配置**。
   iii. 单击**确定**。

> **说明** 在设置**mapreduce.outputcommitter.class**为com.aliyun.emr.fs.oss.commit.JindoOssCommitter后，可以通过开关**fs.oss.committer.magic.enabled**便捷地控制所使用的Job Committer。当打开时，MapReduce任务会使用无需Rename操作的Jindo Oss Magic Committer，当关闭时，JindoOssCommitter和FileOutputCommitter行为一样。

在Spark中使用Jindo Job Committer

1. 进入Spark服务的**spark-defaults**页签。
   i. 在左侧导航栏单击**集群服务 > Spark**。
   ii. 单击**配置**页签。
   iii. 在**服务配置**区域，单击**spark-defaults**页签。
2. 在Spark服务的**spark-defaults**页签，设置以下参数。

参数	参数值
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

   这三个参数分别用来设置写入数据到Spark DataSource表、Spark Parquet格式的DataSource表和Hive表时使用的Job Committer。
3. 保存配置。
   i. 单击右上角的**保存**。
   ii. 在**确认修改**对话框中，输入执行原因，开启**自动更新配置**。
   iii. 单击**确定**。
4. 进入SmartData服务的**smartdata-site**页签。
   i. 在左侧导航栏单击**集群服务 > SmartData**。
   ii. 单击**配置**页签。
   iii. 在**服务配置**区域，单击**smartdata-site**页签。
5. 在SmartData服务的**smartdata-site**页签，设置**fs.oss.committer.magic.enabled**为true。

> **说明** 您可以通过开关 `fs.oss.committer.magic.enabled` 便捷地控制所使用的Job Committer。当打开时，Spark任务会使用无需Rename操作的Jindo Oss Magic Committer，当关闭时，JindoOssCommitter和FileOutputCommitter行为一样。

6. 保存配置。
   i. 单击右上角的**保存**。
   ii. 在**确认修改**对话框中，输入执行原因，开启**自动更新配置**。
   iii. 单击**确定**。

优化Jindo Job Committer性能

当MapReduce或Spark任务写大量文件的时候，您可以调整MapReduce Application Master或Spark Driver中并发执行Commit相关任务的线程数量，提升Job Commit性能。

1. 进入SmartData服务的**smartdata-site**页签。
   i. 在左侧导航栏单击**集群服务 > SmartData**。

ii. 单击**配置**页签。

iii. 在**服务配置**区域，单击**smartdata-site**页签。

2. 在SmartData服务的**smartdata-site**页签，设置**fs.oss.committer.threads**为8。 默认值为8。

3. 保存配置。

i. 单击右上角的**保存**。

ii. 在**确认修改**对话框中，输入执行原因，开启**自动更新配置**。

iii. 单击**确定**。

3.JindoFS基础使用（EMR-3.28.x版本）

3.1. Jindo DistCp使用说明

本文介绍JindoFS的数据迁移工具Jindo DistCp的使用方法。

前提条件

已创建EMR-3.28.0或后续版本的集群，详情请参见创建集群。

使用Jindo Distcp

执行以下命令，获取帮助信息。

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

返回信息如下。

```
--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
```

Jindo DistCp参数详细信息如下：

- --src和--dest
- --parallelism
- --srcPattern
- --deleteOnSuccess
- --outputCodec
- --outputManifest和--requirePreviousManifest
- --outputManifest和--previousManifest
- --copyFromManifest
- --srcPrefixesFile
- --groupBy和-targetSize
- --enableBalancePlan
- --enableDynamicPlan
- --enableTransaction

--src和--dest

`--src` 表示指定源文件的路径， `--dest` 表示目标文件的路径。

Jindo DistCp默认将 `--src` 目录下的所有文件拷贝到指定的 `--dest` 路径下。您可以通过指定 `--dest` 路径来确定拷贝后的文件目录，如果不指定根目录，Jindo DistCp会自动创建根目录。

例如，如果您需要将*/opt/tmp*下的文件拷贝到OSS bucket，可以执行以下命令。

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

--parallelism

`--parallelism` 用于指定MapReduce作业里的mapreduce.job.reduces参数。该参数默认为7，您可以根据集群的资源情况，通过自定义 `--parallelism` 大小来控制DistCp任务的并发度。

例如，将HDFS上*/opt/tmp*目录拷贝到OSS bucket，可以执行以下命令。

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

--srcPattern

`--srcPattern` 使用正则表达式，用于选择或者过滤需要复制的文件。您可以编写自定义的正则表达式来完成选择或者过滤操作，正则表达式必须为全路径正则匹配。

例如，如果您需要复制*/data/incoming/hourly_table/2017-02-01/03*下所有log文件，您可以通过指定 `--srcPattern` 的正则表达式来过滤需要复制的文件。

执行以下命令，查看*/data/incoming/hourly_table/2017-02-01/03*下的文件。

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

返回信息如下。

```
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
```

执行以下命令，复制以log结尾的文件。

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

执行以下命令，查看目标bucket的内容。

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

返回信息如下，显示只复制了以log结尾的文件。

```
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` 可以移动数据并从源位置删除文件。

例如，执行以下命令，您可以将*/data/incoming/*下的*hourly_table*文件移动到OSS bucket中，并删除源位置文件。

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

--outputCodec

`--outputCodec` 可以在线高效地存储数据和压缩文件。

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

目标文件夹中的文件已经使用gz编解码器压缩了。

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

返回信息如下：

```
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
```

Jindo DistCp当前版本支持编解码器gzip、gz、lzo、lzop、snappy以及关键字none和keep（默认值）。关键字含义如下：

- none表示保存为未压缩的文件。如果文件已压缩，则Jindo DistCp会将其解压缩。
- keep表示不更改文件压缩形态，按原样复制。

> **说明** 如果您想在开源Hadoop集群环境中使用编解码器lzo，则需要安装gplcompression的native库和hadoop-lzo包。

--outputManifest和--requirePreviousManifest

`--outputManifest` 可以指定生成DistCp的清单文件，用来记录copy过程中的目标文件、源文件和数据量大小等信息。

如果您需要生成清单文件，则指定 `--requirePreviousManifest` 为 `false` 。当前outputManifest文件默认且必须为gz类型压缩文件。

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

查看outputManifest文件内容。

```
[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
```

返回信息如下。

```
{"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和--previousManifest

`--outputManifest` 表示包含所有已复制文件（旧文件和新文件）的列表，`--previousManifest` 表示只包含之前复制文件的列表。您可以使用 `--outputManifest` 和 `--previousManifest` 重新创建完整的操作历史记录，查看运行期间复制的文件。

例如，在源文件夹中新增加了两个文件，命令如下所示。

```
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
```

执行以下命令，查看运行期间复制的文件。

```
[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
```

返回信息如下。

```
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

使用 `--outputManifest` 生成清单文件后，您可以使用 `--copyFromManifest` 指定 `--outputManifest` 生成的清单文件，并将dest目录生成的清单文件中包含的文件信息拷贝到新的目录下。

```
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 可以一次性完成多个文件夹的复制。

示例如下，查看*hourly_table*下文件。

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

返回信息如下。

```
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
```

执行以下命令，复制*hourly_table*下文件到*folders.txt*。

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

查看*folders.txt*文件的内容。

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

返回信息如下。

```
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和-targetSize

因为Hadoop可以从HDFS中读取少量的大文件，而不再读取大量的小文件，所以在大量小文件的场景下，您可以使用Jindo DistCp将小文件聚合为指定大小的大文件，以便于优化分析性能和降低成本。

例如，执行以下命令，查看如下文件夹中的数据。

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

返回信息如下。

```
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
```

执行以下命令，将如下文件夹中的TXT文件合并为不超过10M的文件。

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

经过合并后，可以看到两个TXT文件被合并成了一个文件。

```
[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

在您要拷贝的数据大小均衡、小文件和大文件混合的场景下，因为DistCp默认的执行计划是随机进行文件分配的，所以您可以指定 `--enableBalancePlan` 来更改Jindo DistCp的作业分配计划，以达到更好的DistCp性能。

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

> **说明** 该参数不支持和 `--groupby` 或 `--targetSize` 同时使用。

--enableDynamicPlan

当您要拷贝的数据大小分化严重、小文件数据较多的场景下，您可以指定 `--enableDynamicPlan` 来更改Jindo DistCp的作业分配计划，以达到更好的DistCp性能。

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

> **说明** 该参数不支持和 `--groupby` 或 `--targetSize` 参数一起使用。

--enableTransaction

`--enableTransaction` 可以保证Job级别的完整性以及保证Job之间的事务支持。示例如下。

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

--diff

DistCp任务完成后，您可以使用 `--diff` 查看当前DistCp的文件差异。

例如，执行以下命令，查看*/data/incoming/*。

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

如果全部任务完成则会提示如下信息。

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

如果src的文件未能同步到dest上，则会在当前目录下生成*manifest*文件，您可以使用 `--copyFromManifest` 和 `--previousManifest` 拷贝剩余文件，从而完成数据大小和文件个数的校验。如果您的DistCp任务包含压缩或者解压缩，则 `--diff` 不能显示正确的文件差异，因为压缩或者解压缩会改变文件的大小。

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

> **说明** 如果您的 `--dest` 为HDFS路径，目前仅支持*/path*、*hdfs://hostname:ip/path*和*hdfs://headerIp:ip/path*的写法，暂不支持*hdfs:///path*、*hdfs:/path*和其他自定义写法。

查看Distcp Counters

执行以下命令，在MapReduce的Counter信息中查找Distcp Counters的信息。

```
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
```

说明 如果您的DistCp操作中包含压缩或者解压缩文件，则 Bytes Destination Copied 和 Bytes Source Read 的大小可能不相等。

使用OSS Accesskey

在E-MapReduce外或者免密服务出现问题的情况下，您可以通过指定Accesskey来获得访问OSS的权限。您可以在命令中使用--key、--secret、--endPoint选项来指定Accesskey。

命令示例如下。

```
jindo distcp --src /data/incoming/hourly_table --dest oss://<your_bucket>/hourly_table --key yourkey --secret yoursecret --endPoint oss-cn-hangzhou.aliyuncs.com --parallelism 20
```

使用归档或低频写入OSS

在您的Distcp任务写入OSS时，您可以通过如下模式写入OSS，数据存储：

- 使用归档（--archive）示例命令如下。

  ```
  jindo distcp --src /data/incoming/hourly_table --dest oss://<your_bucket>/hourly_table --policy archive --parallelism 20
  ```

- 使用低频（--ia）示例命令如下。

  ```
  jindo distcp --src /data/incoming/hourly_table --dest oss://<your_bucket>/hourly_table --policy ia --parallelism 20
  ```

清理残留文件

在您的DistCp任务过程中，由于某种原因在您的目标目录下，产生未正确上传的文件，这部分文件通过uploadId的方式由OSS管理，并且对用户不可见时，您可以通过指定--cleanUpPending选项，指定任务结束时清理残留文件，或者您也可以通过OSS控制台进行清理。

命令示例如下。

```
jindo distcp --src /data/incoming/hourly_table --dest oss://<your_bucket>/hourly_table --cleanUpPending --parallelism 20
```

4.JindoFS基础使用（EMR-3.29.x版本）

4.1. JindoFS块存储模式使用说明

Block模式提供了最为高效的数据读写能力和元数据访问能力。数据以Block形式存储在后端存储OSS上，本地提供缓存加速，元数据则由本地Namespace服务维护，提供高效的元数据访问性能。本文主要介绍JindoFS的Block模式及其使用方式。

背景信息

JindoFS Block模式具有以下几个特点：

- 海量弹性的存储空间，基于OSS作为存储后端，存储不受限于本地集群，而且本地集群能够自由弹性伸缩。
- 能够利用本地集群的存储资源加速数据读取，适合具有一定本地存储能力的集群，能够利用有限的本地存储提升吞吐率，特别对于一写多读的场景效果显著。
- 元数据操作效率高，能够与HDFS相当，能够有效规避OSS文件系统元数据操作耗时以及高频访问下可能引发不稳定的问题。
- 能够最大限度保证执行作业时的数据本地化，减少网络传输的压力，进一步提升读取性能。

配置使用方式

1. 进入SmartData服务。
   i. 登录阿里云E-MapReduce控制台。
   ii. 在顶部菜单栏处，根据实际情况选择地域（Region）和资源组。
   iii. 单击上方的**集群管理**页签。
   iv. 在**集群管理**页面，单击相应集群所在行的**详情**。
   v. 在左侧导航栏单击**集群服务 > SmartData**。
2. 进入**namespace**服务配置。
   i. 单击**配置**页签。
   ii. 单击**namespace**。

   
3. 配置以下参数。JindoFS支持多命名空间，本文命名空间以test为例。
   i. 修改**jfs.namespaces**为**test**。**test**表示当前JindoFS支持的命名空间，多个命名空间时以逗号隔开。
   ii. 单击**自定义配置**，在**新增配置项**对话框中增加以下参数，单击**确定**。

参数	参数说明	示例
jfs.namespaces.test.oss.uri	表示test命名空间的后端存储。	*oss://<oss_bucket>/<oss_dir>/* **说明** 推荐配置到OSS bucket下的某一个具体目录，该命名空间即会将Block模式的数据块存放在该目录下。
jfs.namespaces.test.mode	表示test命名空间为块存储模式。	block
jfs.namespaces.test.oss.access.key	表示存储后端OSS的AccessKey ID。	*xxxx* **说明** 考虑到性能和稳定性，推荐使用同账户、同Region下的OSS bucket作为存储后端，此时，E-MapReduce集群能够免密访问OSS，无需配置AccessKey ID和AccessKey Secret。
jfs.namespaces.test.oss.access.secret	表示存储后端OSS的AccessKey Secret。	

   iii. 单击**确定**。

4. 单击右上角的**保存**。
5. 单击右上角的**操作 > 重启 Jindo Namespace Service**。重启后即可通过 `jfs://test/<path_of_file>` 的形式访问JindoFS上的文件。

磁盘空间水位控制

JindoFS后端基于OSS，可以提供海量的存储，但是本地盘的容量是有限的，因此JindoFS会自动淘汰本地较冷的数据备份。我们提供了 `storage.watermark.high.ratio` 和 `storage.watermark.low.ratio` 两个参数来调节本地存储的使用容量，值均为0～1的小数，表示使用磁盘空间的比例。

1. 修改磁盘水位配置。 在**服务配置**区域的**storage**页签，修改如下参数。

参数	描述
storage.watermark.high.ratio	表示磁盘使用量的上水位比例，每块数据盘的JindoFS数据目录占用的磁盘空间到达上水位即会触发清理。默认值：0.4。
storage.watermark.low.ratio	表示使用量的下水位比例，触发清理后会自动清理冷数据，将JindoFS数据目录占用空间清理到下水位。默认值：0.2。

> **说明** 您可以通过设置上水位比例调节期望分给JindoFS的磁盘空间，下水位必须小于上水位，设置合理的值即可。

2. 保存配置。
   i. 单击右上角的**保存**。
   ii. 在**确认修改**对话框中，输入执行原因，开启**自动更新配置**。
   iii. 单击**确定**。
3. 重启Jindo Storage Service使配置生效。
   i. 单击右上角的**操作 > 重启Jindo Storage Service**。
   ii. 在**执行集群操作**对话框中，设置相关参数。
   iii. 单击**确定**。
   iv. 在**确认**对话框中，单击**确定**。

4.2. JindoFS缓存模式使用说明

缓存模式（Cache）主要兼容原生OSS存储方式，文件以对象的形式存储在OSS上，每个文件根据实际访问情况会在本地进行缓存，提升EMR集群内访问OSS的效率，同时兼容了原有OSS原有文件形式，数据访问上能够与其他OSS客户端完全兼容。本文主要介绍JindoFS的缓存模式及其使用方式。

背景信息

缓存模式最大的特点就是兼容性，保持了OSS原有的对象语义，集群中仅做缓存，因此和其他的各种OSS客户端是完全兼容的，对原有OSS上的存量数据也不需要做任何的迁移、转换工作即可使用。同时集群中的缓存也能一定程度上提升数据访问性能，缓解读写OSS的带宽压力。

配置使用方式

JindoFS缓存模式提供了以下两种基本使用方式，以满足不同的使用需求。

- OSS Scheme

  详情请参见配置OSS Scheme（推荐）。

- JFS Scheme

  详情请参见配置JFS Scheme。

配置OSS Scheme（推荐）

OSS Scheme保留了原有OSS文件系统的使用习惯，即直接通过 `oss://<bucket_name>/<path_of_your_file>` 的形式访问OSS上的文件。使用该方式访问OSS，无需进行额外的配置，创建EMR集群后即可使用，对于原有读写OSS的作业也无需做任何修改即可运行。

配置JFS Scheme

1. 进入SmartData服务。
   i. 登录阿里云E-MapReduce控制台。
   ii. 在顶部菜单栏处，根据实际情况选择地域（Region）和资源组。
   iii. 单击上方的**集群管理**页签。
   iv. 在**集群管理**页面，单击相应集群所在行的**详情**。
   v. 在左侧导航栏单击**集群服务 > SmartData**。
2. 进入**namespace**服务配置。
   i. 单击**配置**页签。
   ii. 单击**namespace**。
3. 配置以下参数。JindoFS支持多命名空间，本文命名空间以test为例。
   i. 修改**jfs.namespaces**为**test**。**test**表示当前JindoFS支持的命名空间，多个命名空间时以逗号隔开。
   ii. 单击**自定义配置**，在**新增配置项**对话框中增加以下参数。

参数	参数说明	示例
jfs.namespaces.test.oss.uri	表示test命名空间的后端存储。	*oss://<oss_bucket>/<oss_dir>/* **说明** 该配置必须配置到OSS Bucket下的具体目录，也可以直接使用根目录。
jfs.namespaces.test.mode	表示test命名空间为缓存模式。	cache

4. 单击右上角的**保存**。
5. 单击右上角的**操作 > 重启 Jindo Namespace Service**。重启后即可通过 `jfs://test/<path_of_file>` 的形式访问JindoFS上的文件。

启用缓存

启用缓存会利用本地磁盘对访问的热数据块进行缓存，默认状态为禁用，即所有OSS读取都直接访问OSS上的数据。

1. 在**集群服务 > SmartData**的**配置**页面，单击**client**页签。
2. 修改**jfs.cache.data-cache.enable**为**1**，表示启用缓存。此配置为客户端配置，不需要重启SmartData服务。

缓存启用后，Jindo服务会自动管理本地缓存备份，通过水位清理本地缓存，请您根据需求配置一定的比例用于缓存，详情请参见磁盘空间水位控制。

磁盘空间水位控制

JindoFS后端基于OSS，可以提供海量的存储，但是本地盘的容量是有限的，因此JindoFS会自动淘汰本地较冷的数据备份。我们提供了 `storage.watermark.high.ratio` 和 `storage.watermark.low.ratio` 两个参数来调节本地存储的使用容量，值均为0～1的小数，表示使用磁盘空间的比例。

1. 修改磁盘水位配置。 在**服务配置**区域的**storage**页签，修改如下参数。

参数	描述

参数	描述
storage.watermark.high.ratio	表示磁盘使用量的上水位比例，每块数据盘的JindoFS数据目录占用的磁盘空间到达上水位即会触发清理。默认值：0.4。
storage.watermark.low.ratio	表示使用量的下水位比例，触发清理后会自动清理冷数据，将JindoFS数据目录占用空间清理到下水位。默认值：0.2。

> **说明** 您可以通过设置上水位比例调节期望分给JindoFS的磁盘空间，下水位必须小于上水位，设置合理的值即可。

2. 保存配置。
   i. 单击右上角的**保存**。
   ii. 在**确认修改**对话框中，输入执行原因，开启**自动更新配置**。
   iii. 单击**确定**。
3. 重启Jindo Storage Service使配置生效。
   i. 单击右上角的**操作 > 重启Jindo Storage Service**。
   ii. 在**执行集群操作**对话框中，设置相关参数。
   iii. 单击**确定**。
   iv. 在**确认**对话框中，单击**确定**。

访问OSS bucket

在EMR集群中访问同账号、同区域的OSS Bucket时，默认支持免密访问，即无需配置任何AccessKey即可访问。如果访问非以上情况的OSS Bucket需要配置相应的AccessKey ID、AccessKey Secret以及Endpoint，针对两种使用方式相应的配置分别如下：

- OSS Scheme
  i. 在**集群服务 > SmartData**的**配置**页面，单击**smartdata-site**页签。
  ii. 单击**自定义配置**，在**新增配置项**对话框中增加以下参数，单击**确定**。

参数	参数说明
fs.jfs.cache.oss-accessKeyId	表示存储后端OSS的AccessKey ID。
fs.jfs.cache.oss-accessKeySecret	表示存储后端OSS的AccessKey Secret。
fs.jfs.cache.oss-endpoint	表示存储后端OSS的endpoint。

- JFS Scheme
  i. 在**集群服务 > SmartData**的**配置**页面，单击**bigboot**页签。
  ii. 修改**jfs.namespaces**为**test**。
  iii. 单击**自定义配置**，在**新增配置项**对话框中增加以下参数，单击**确定**。

参数	参数说明
jfs.namespaces.test.oss.uri	表示test命名空间的后端存储。示例：*oss://<oss_bucket.endpoint>/<oss_dir>*。 endpoint信息直接配置在oss.uri中。
jfs.namespaces.test.oss.access.key	表示存储后端OSS的AccessKey ID。
jfs.namespaces.test.oss.access.secret	表示存储后端OSS的AccessKey Secret。

- OSS Scheme
  i.

- JFS Scheme
  i. 在**集群服务 > SmartData的配置**页面，单击**namespace**页签。
  ii.
  iii.

高级配置

Cache模式还包含一些高级配置，用于性能调优，以下配置均为客户端配置，修改后无需重启SmartData服务。

- 在**服务配置**区域的**client**页签，配置以下参数。

参数	参数说明
client.oss.upload.threads	每个文件写入流的OSS上传线程数。默认值：4。
client.oss.upload.max.parallelism	进程级别OSS上传总并发度上限，防止过多上传线程造成过大的带宽压力以及过大的内存消耗。默认值：16。

- 在**服务配置**区域的**smartdata-site**页签，配置以下参数。

参数	参数说明
fs.jfs.cache.copy.simple.max.byte	rename过程使用普通copy接口的文件大小上限（小于阈值的使用普通 copy接口，大于阈值的使用multipart copy接口以提高copy效率）。 **说明** 如果确认已开通OSS fast copy功能，参数值设为-1，表示所有大小均使用普通copy接口，从而有效利用fast copy获得最优的rename性能。
fs.jfs.cache.write.buffer.size	文件写入流的buffer大小，参数值必须为2的幂次，最大为8MB，如果作业同时打开的写入流较多导致内存使用过大，可以适当调小此参数。默认值：1048576。
fs.oss.committer.magic.enabled	启用Jindo Job Committer，避免Job Committer的rename操作，来提升性能。默认值：true。 **说明** 针对Cache模式下，由于OSS这类对象存储rename操作性能较差的问题，推出了Jindo Job Committer。

4.3. 使用JindoFS SDK免密功能

本文介绍使用JindoFS SDK时，E-MapReduce（简称EMR）集群外如何以免密方式访问E-MapReduce JindoFS的文件系统。

前提条件

适用环境：ECS（EMR环境外）+Hadoop+JavaSDK。

背景信息

使用JindoFS SDK时，需要把环境中相关Jindo的包从环境中移除，如*jboot.jar*、*smartdata-aliyun-jfs-*.jar*。如果要使用Spark则需要把*/opt/apps/spark-current/jars/*里面的包也删除，从而可以正常使用。

步骤一：创建实例RAM角色

1. 使用云账号登录RAM的控制台。
2. 单击左侧导航栏的**RAM角色管理**。
3. 单击**创建 RAM 角色**，选择当前可信实体类型为**阿里云服务**。
4. 单击**下一步**。

5. 输入**角色名称**，从**选择授信服务**列表中，选择**云服务器**。
6. 单击**完成**。

步骤二：为RAM角色授予权限

1. 使用云账号登录RAM的控制台。
2. （可选）如果您不使用系统权限，可以参见账号访问控制创建自定义权限策略章节创建一个自定义策略。
3. 单击左侧导航栏的**RAM角色管理**。
4. 单击新创建RAM角色名称所在行的**精确授权**。
5. 选择权限类型为**系统策略**或**自定义策略**。
6. 输入策略名称。
7. 单击**确定**。

步骤三：为实例授予RAM角色

1. 登录ECS管理控制台。
2. 在左侧导航栏，单击**实例与镜像 > 实例**。
3. 在顶部状态栏左上角处，选择地域。
4. 找到要操作的ECS实例，选择**更多 > 实例设置 > 授予/收回RAM角色**。

5. 在弹窗中，选择创建好的实例RAM角色，单击**确定**完成授予。

步骤四：在ECS上设置环境变量

执行如下命令，在ECS上设置环境变量。

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

或者执行如下命令。

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

步骤五：测试免密方式访问的方法

1. 使用Shell访问OSS。

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

2. 使用Hadoop FileSystem访问OSS。JindoFS SDK支持使用Hadoop FileSystem访问OSS，示例代码如下。

```
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);
      }
   }
}
```

4.4. JindoFS元数据服务

4.4.1. 使用Tablestore作为存储后端

JindoFS元数据服务支持不同的存储后端，本文介绍使用Tablestore（OTS）作为元数据后端时需要进行的配置。

前提条件

- 已创建EMR集群。

  详情请参见创建集群。
- 已创建Tablestore实例，推荐使用高性能实例。

  详情请参见创建实例。

  > **说明** 需要开启事务功能。

背景信息

JindoFS在新版本中，支持使用Tablestore作为JindoFS元数据服务（Namespace Service）的存储。一个EMR JindoFS集群可以绑定一个Tablestore实例（Instance）作为JindoFS元数据服务的存储介质，元数据服务会自动为每个Namespace创建独立的Tablestore表进行管理和存储元数据信息。

元数据服务（双机Tablestore和HA）架构图如下所示。

配置Tablestore

使用Tablestore功能，需要把创建的Tablestore实例和JindoFS的Namespace服务进行绑定，详细步骤如下：

1. 进入SmartData服务。
   i. 登录阿里云E-MapReduce控制台。
   ii. 在顶部菜单栏处，根据实际情况选择地域（Region）和资源组。
   iii. 单击上方的**集群管理**页签。

iv. 在**集群管理**页面，单击相应集群所在行的**详情**。

v. 在左侧导航栏单击**集群服务 > SmartData**。

2. 进入**namespace**服务配置。

i. 单击**配置**页签。

ii. 单击**namespace**。

3. 配置以下参数。例如，在华东1（杭州）地域下，创建了emr-jfs的Tablestore实例，EMR集群使用VPC网络，访问Tablestore的AccessKey ID为kkkkkk，Access Secret为XXXXXX。

参数	参数说明	是否必选	示例
namespace.backend.type	设置namespace后端存储类型，支持： ○ rocksdb ○ ots ○ raft 默认为rocksdb。	是	ots
namespace.ots.instance	Tablestore实例名称。	是	emr-jfs
namespace.ots.accessKey	Tablestore实例的AccessKey ID。	否	kkkkkk
namespace.ots.accessSecret	Tablestore实例的AccessKey Secret。	否	XXXXXX
namespace.ots.endpoint	Tablestore实例的Endpoint地址，普通EMR集群，推荐使用VPC地址。	是	*http://emr-jfs.cn-hangzhou.vpc.tablestore.aliyuncs.com*

4. 保存配置。

i. 单击右上角的**保存**。

ii. 在**确认修改**对话框中，输入执行原因，开启**自动更新配置**。

iii. 单击**确定**。

5. 单击右上角的**操作 > 重启 Jindo Namespace Service**。

配置Tablestore（高可用方案）

针对EMR的高可用集群，可以通过配置开启Namespace高可用模式。

Namespace高可用模式采用Active和Standby互备方式，支持自动故障转移，当Active Namespace出现异常或者异常中止时，客户端可以请求自动切换到新的Active节点。

1. 进入SmartData的**namespace**服务配置，配置以下参数。

i. 修改**jfs.namespace.server.rpc-address**值为**emr-header-1:8101,emr-header-2:8101**。

ii. 单击右上角的**自定义配置**，添加**namespace.backend.ots.ha**为**true**。

iii. 单击**确定**。

iv. 保存配置。

a. 单击右上角的**保存**。

b. 在**确认修改**对话框中，输入执行原因，开启**自动更新配置**。

c. 单击**确定**。

2. 单击右上角的**操作 > 重启Jindo Namespace Service**。

3. 单击右上角的**操作** > **重启Jindo Storage Service**。

4.4.2. 使用RocksDB作为元数据后端

JindoFS元数据服务支持不同的存储后端，默认配置RocksDB为元数据存储后端。本文介绍使用RocksDB作为元数据后端时需要进行的相关配置。

背景信息

RocksDB作为元数据后端时不支持高可用。如果需要高可用，推荐配置Tablestore（OTS）或者Raft作为元数据后端，详情请参见使用Tablestore作为存储后端和使用Raft-RocksDB-Tablestore作为存储后端。

单机RocksDB作为元数据服务的架构图如下所示。

配置RocksDB

1. 进入SmartData服务。
   i. 登录阿里云E-MapReduce控制台。
   ii. 在顶部菜单栏处，根据实际情况选择地域（Region）和资源组。
   iii. 单击上方的**集群管理**页签。
   iv. 在**集群管理**页面，单击相应集群所在行的**详情**。
   v. 在左侧导航栏单击**集群服务** > **SmartData**。
2. 进入**namespace**服务配置。
   i. 单击**配置**页签。
   ii. 单击**namespace**。

   
3. 设置**namespace.backend.type**为**rocksdb**。
4. 保存配置。
   i. 单击右上角的**保存**。
   ii. 在**确认修改**对话框中，输入执行原因，开启**自动更新配置**。
   iii. 单击**确定**。
5. 单击右上角的**操作** > **重启 Jindo Namespace Service**。

4.4.3. 使用Raft-RocksDB-Tablestore作为存储后端

JindoFS在EMR-3.27.0及之后版本中支持使用Raft-RocksDB-OTS作为Jindo元数据服务（Namespace Service）的存储。1个EMR JindoFS集群创建3个Master节点组成1个Raft实例，实例的每个Peer节点使用本地RocksDB存储元数据信息。

前提条件

- 创建Tablestore实例，推荐使用高性能实例，详情请参见创建实例。

  > **说明** 需要开启事务功能。

- 创建3 Master的EMR集群，详情请参见创建集群。

  

  > **说明** 如果没有部署方式，请提交工单处理。

背景信息

RocksDB通过Raft协议实现3个节点之间的复制。集群可以绑定1个Tablestore（OTS）实例，作为Jindo的元数据服务的额外存储介质，本地的元数据信息会实时异步地同步到用户的Tablestore实例上。

元数据服务-多机Raft-RocksDB-Tablestore+HA如下图所示。

配置本地raft后端

1. 新建EMR集群后，暂停SmartData所有服务。
   i. 登录阿里云E-MapReduce控制台。
   ii. 在顶部菜单栏处，根据实际情况选择地域（Region）和资源组。
   iii. 单击上方的**集群管理**页签。
   iv. 在**集群管理**页面，单击相应集群所在行的**详情**。
   v. 在左侧导航栏，单击**集群服务 > SmartData**。
   vi. 单击右上角的**操作 > 停止 All Components**。
2. 根据使用需求，添加需要的namespace。
3. 进入SmartData服务的**namespace**页签。
   i. 在左侧导航栏，单击**集群服务 > SmartData**。
   ii. 单击**配置**页签。
   iii. 在**服务配置**区域，单击**namespace**页签。
4. 在SmartData服务的**namespace**页签，设置如下参数。

参数	描述	示例
namespace.backend.type	设置namespace后端存储类型，支持： ○ rocksdb ○ ots ○ raft 默认为rocksdb。	raft
namespace.backend.raft.initial-conf	部署raft实例的3个Master地址（固定值）。	emr-header-1:8103:0,emr-header-2:8103:0,emr-header-3:8103:0
jfs.namespace.server.rpc-address	Client端访问raft实例的3个Master地址（固定值）	emr-header-1:8101,emr-header-2:8101,emr-header-3:8101

> **说明** 如果不需要使用OTS远端存储，直接执行步骤6和步骤7；如果需要使用OTS远端存储，请执行步骤5~步骤7。

5. （可选）配置远端OTS异步存储。在SmartData服务的**namespace**页签，设置如下参数。

参数	参数说明	示例
namespace.ots.instance	Tablestore实例名称。	emr-jfs
namespace.ots.accessKey	Tablestore实例的AccessKey ID。	*kkkkkk*
namespace.ots.accessSecret	Tablestore实例的AccessKey Secret。	*XXXXXX*
namespace.ots.endpoint	Tablestore实例的endpoint地址，通常EMR集群，推荐使用VPC地址。	*http://emr-jfs.cn-hangzhou.vpc.tablestore.aliyuncs.com*

参数	参数说明	示例
namespace.backend.raft.async.ots.enabled	是否开启OTS异步上传，包括： ○ true ○ false 当设置为true时，需要在SmartData服务完成初始化前，开启OTS异步上传功能。 **说明** 如果SmartData服务已完成初始化，则不能再开启该功能。因为OTS的数据已经落后于本地RocksDB的数据。	true

6. 保存配置。
   i. 单击右上角的**保存**。
   ii. 在**确认修改**对话框中，输入执行原因，开启**自动更新配置**。
   iii. 单击**确定**。
7. 单击右上角的**操作 > 启动 All Components**。

从Tablestore恢复元数据信息

如果您在原始集群开启了远端Tablestore异步存储，则Tablestore上会有1份完整的JindoFS元数据的副本。您可以在停止或释放原始集群后，在新创建的集群上恢复原先的元数据，从而继续访问之前保存的文件。

1. （可选）准备工作。
   i. （可选）统计原始集群的元数据信息（文件和文件夹数量）。

```
[hadoop@emr-header-1 ~]$ hadoop fs -count jfs://test/
      1596     1482809              25 jfs://test/
  (文件夹个数）（文件个数）
```

   ii. 停止原始集群的作业，等待30~120秒左右，等待原始集群的数据已经完全同步到Tablestore。执行以下命令查看状态。如果LEADER节点显示 `_synced=1` ，则表示Tablestore为最新数据，同步完成。

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

   iii. 停止或释放原始集群，确保没有其它集群正在访问当前的Tablestore实例。
2. 创建新集群。新建与Tablestore实例相同Region的EMR集群，暂停SmartData所有服务。详情请参见配置本地raft后端。
3. 初始化配置。在SmartData服务的**namespace**页签，设置以下参数。

参数	描述	示例
namespace.backend.raft.async.ots.enabled	是否开启OTS异步上传，包括： ○ true ○ false	false
namespace.backend.raft.recovery.mode	是否开启从OTS恢复元数据，包括： ○ true ○ false	true

4. 保存配置。
   i. 单击右上角的**保存**。
   ii. 在**确认修改**对话框中，输入执行原因，开启**自动更新配置**。

iii. 单击**确定**。

5. 单击右上角的**操作 > 启动 All Components**。
6. 新集群的SmartData服务启动后，自动从OTS恢复元数据到本地Raft-RocksDB上，可以通过以下命令查看恢复进度。

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

如图所示，LEADER节点的state为FINISH表示恢复完成。

7. （可选）执行以下操作，可以比较一下文件数量与原始集群是否一致。此时的集群为恢复模式，也是只读模式。

```
# 对比文件数量一致
[hadoop@emr-header-1 ~]$ hadoop fs -count jfs://test/
      1596     1482809              25 jfs://test/

# 文件可正常读取(cat、get命令)
[hadoop@emr-header-1 ~]$ hadoop fs -cat jfs://test/testfile
this is a test file
# 查看目录
[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

# 只读状态，不可修改文件
[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. 修改配置，将集群设置为正常模式，开启OTS异步上传功能。在SmartData服务的**namespace**页签，设置以下参数。

参数	描述	示例
namespace.backend.raft.async.ots.enabled	是否开启OTS异步上传，包括： ○ true ○ false	true
namespace.backend.raft.recovery.mode	是否开启从OTS恢复元数据，包括： ○ true ○ false	false

9. 重启集群。
   i. 单击上方的**集群管理**页签。
   ii. 在**集群管理**页面，单击相应集群所在行的**更多 > 重启**。

4.5. Jindo Job Committer使用说明

本文主要介绍JindoOssCommitter的使用说明。

背景信息

Job Committer是MapReduce和Spark等分布式计算框架的一个基础组件，用来处理分布式任务写数据的一致性问题。

Jindo Job Committer是阿里云E-MapReduce针对OSS场景开发的高效Job Committer的实现，基于OSS的Multipart Upload接口，结合OSS Filesystem层的定制化支持。使用Jindo Job Committer时，Task数据直接写到最终目录中，在完成Job Commit前，中间数据对外不可见，彻底避免了Rename操作，同时保证数据的一致性。

> 注意
>
> - OSS拷贝数据的性能，针对不同的用户或Bucket会有差异，可能与OSS带宽以及是否开启某些高级特性等因素有关，具体问题可以咨询OSS的技术支持。
> - 在所有任务都完成后，MapReduce Application Master或Spark Driver执行最终的Job Commit操作时，会有一个短暂的时间窗口。时间窗口的大小和文件数量线性相关，可以通过增大 `fs.oss.committer.threads` 可以提高并发处理的速度。
> - Hive和Presto等没有使用Hadoop的Job Committer。
> - E-MapReduce集群中默认打开Jindo Oss Committer的参数。

在MapReduce中使用Jindo Job Committer

1. 进入YARN服务的**mapred-site**页签。
   i. 登录阿里云E-MapReduce控制台。
   ii. 在顶部菜单栏处，根据实际情况选择地域（Region）和资源组。
   iii. 单击上方的**集群管理**页签。
   iv. 在**集群管理**页面，单击相应集群所在行的**详情**。
   v. 在左侧导航栏单击**集群服务 > YARN**。
   vi. 单击**配置**页签。
   vii. 在**服务配置**区域，单击**mapred-site**页签。
2. 针对Hadoop不同版本，在YARN服务中配置以下参数。
   - Hadoop 2.x版本

     在YARN服务的**mapred-site**页签，设置**mapreduce.outputcommitter.class**为com.aliyun.emr.fs.oss.commit.JindoOssCommitter。
   - Hadoop 3.x版本

     在YARN服务的**mapred-site**页签，设置**mapreduce.outputcommitter.factory.scheme.oss**为com.aliyun.emr.fs.oss.commit.JindoOssCommitterFactory。
3. 保存配置。
   i. 单击右上角的**保存**。
   ii. 在**确认修改**对话框中，输入执行原因，开启**自动更新配置**。
   iii. 单击**确定**。
4. 进入SmartData服务的**smartdata-site**页签。
   i. 在左侧导航栏单击**集群服务 > SmartData**。
   ii. 单击**配置**页签。
   iii. 在**服务配置**区域，单击**smartdata-site**页签。
5. 在SmartData服务的**smartdata-site**页签，设置**fs.oss.committer.magic.enabled**为true。
6. 保存配置。
   i. 单击右上角的**保存**。
   ii. 在**确认修改**对话框中，输入执行原因，开启**自动更新配置**。
   iii. 单击**确定**。

> **说明** 在设置**mapreduce.outputcommitter.class**为com.aliyun.emr.fs.oss.commit.JindoOssCommitter后，可以通过开关**fs.oss.committer.magic.enabled**便捷地控制所使用的Job Committer。当打开时，MapReduce任务会使用无需Rename操作的Jindo Oss Magic Committer，当关闭时，JindoOssCommitter和FileOutputCommitter行为一样。

在Spark中使用Jindo Job Committer

1. 进入Spark服务的**spark-defaults**页签。
   i. 在左侧导航栏单击**集群服务 > Spark**。
   ii. 单击**配置**页签。
   iii. 在**服务配置**区域，单击**spark-defaults**页签。
2. 在Spark服务的**spark-defaults**页签，设置以下参数。

参数	参数值
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

   这三个参数分别用来设置写入数据到Spark DataSource表、Spark Parquet格式的DataSource表和Hive表时使用的Job Committer。
3. 保存配置。
   i. 单击右上角的**保存**。
   ii. 在**确认修改**对话框中，输入执行原因，开启**自动更新配置**。
   iii. 单击**确定**。
4. 进入SmartData服务的**smartdata-site**页签。
   i. 在左侧导航栏单击**集群服务 > SmartData**。
   ii. 单击**配置**页签。
   iii. 在**服务配置**区域，单击**smartdata-site**页签。
5. 在SmartData服务的**smartdata-site**页签，设置**fs.oss.committer.magic.enabled**为true。

   > **说明** 您可以通过开关 fs.oss.committer.magic.enabled 便捷地控制所使用的Job Committer。当打开时，Spark任务会使用无需Rename操作的Jindo Oss Magic Committer，当关闭时，JindoOssCommitter和FileOutputCommitter行为一样。

6. 保存配置。
   i. 单击右上角的**保存**。
   ii. 在**确认修改**对话框中，输入执行原因，开启**自动更新配置**。
   iii. 单击**确定**。

优化Jindo Job Committer性能

当MapReduce或Spark任务写大量文件的时候，您可以调整MapReduce Application Master或Spark Driver中并发执行Commit相关任务的线程数量，提升Job Commit性能。

1. 进入SmartData服务的**smartdata-site**页签。
   i. 在左侧导航栏单击**集群服务 > SmartData**。
   ii. 单击**配置**页签。
   iii. 在**服务配置**区域，单击**smartdata-site**页签。
2. 在SmartData服务的**smartdata-site**页签，设置**fs.oss.committer.threads**为8。 默认值为8。
3. 保存配置。
   i. 单击右上角的**保存**。
   ii. 在**确认修改**对话框中，输入执行原因，开启**自动更新配置**。
   iii. 单击**确定**。

4.6. Jindo DistCp使用说明

本文介绍JindoFS的数据迁移工具Jindo DistCp的使用方法。

前提条件

- 本地安装了Java JDK 8。
- 已创建EMR-3.28.0或后续版本的集群，详情请参见创建集群。

使用Jindo Distcp

执行以下命令，获取帮助信息。

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

返回信息如下。

```
--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
--ossKey=VALUE  -  Specify your oss key if needed
--ossSecret=VALUE  -  Specify your oss secret if needed
--ossEndPoint=VALUE  -  Specify your oss endPoint if needed
--policy=VALUE  -  Specify your oss storage policy
--cleanUpPending  -  clean up the incomplete upload when distcp job finish
--queue=VALUE  -  Specify yarn queuename if needed
--bandwidth=VALUE  -  Specify bandwidth per map/reduce in MB if needed
--s3Key=VALUE  -  Specify your s3 key
--s3Secret=VALUE  -  Specify your s3 Sercet
--s3EndPoint=VALUE  -  Specify your s3 EndPoint
```

--src和--dest

`--src`表示指定源文件的路径，`--dest`表示目标文件的路径。

Jindo DistCp默认将`--src`目录下的所有文件拷贝到指定的`--dest`路径下。您可以通过指定`--dest`路径来确定拷贝后的文件目录，如果不指定根目录，Jindo DistCp会自动创建根目录。

例如，如果您需要将*/opt/tmp*下的文件拷贝到OSS bucket，可以执行以下命令。

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

--parallelism

`--parallelism` 用于指定MapReduce作业里的mapreduce.job.reduces参数。该参数默认为7，您可以根据集群的资源情况，通过自定义 `--parallelism` 大小来控制DistCp任务的并发度。

例如，将HDFS上*/opt/tmp*目录拷贝到OSS bucket，可以执行以下命令。

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

--srcPattern

`--srcPattern` 使用正则表达式，用于选择或者过滤需要复制的文件。您可以编写自定义的正则表达式来完成选择或者过滤操作，正则表达式必须为全路径正则匹配。

例如，如果您需要复制*/data/incoming/hourly_table/2017-02-01/03*下所有log文件，您可以通过指定 `--srcPattern` 的正则表达式来过滤需要复制的文件。

执行以下命令，查看*/data/incoming/hourly_table/2017-02-01/03*下的文件。

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

返回信息如下。

```
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
```

执行以下命令，复制以log结尾的文件。

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

执行以下命令，查看目标bucket的内容。

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

返回信息如下，显示只复制了以log结尾的文件。

```
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` 可以移动数据并从源位置删除文件。

例如，执行以下命令，您可以将*/data/incoming/*下的*hourly_table*文件移动到OSS bucket中，并删除源位置文件。

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

--outputCodec

`--outputCodec` 可以在线高效地存储数据和压缩文件。

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

目标文件夹中的文件已经使用gz编解码器压缩了。

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

返回信息如下：

```
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
```

Jindo DistCp当前版本支持编解码器gzip、gz、lzo、lzop、snappy以及关键字none和keep（默认值）。关键字含义如下：

- none表示保存为未压缩的文件。如果文件已压缩，则Jindo DistCp会将其解压缩。
- keep表示不更改文件压缩形态，按原样复制。

> **说明** 如果您想在开源Hadoop集群环境中使用编解码器lzo，则需要安装gplcompression的native库和hadoop-lzo包。

--outputManifest和--requirePreviousManifest

`--outputManifest` 可以指定生成DistCp的清单文件，用来记录copy过程中的目标文件、源文件和数据量大小等信息。

如果您需要生成清单文件，则指定 `--requirePreviousManifest` 为 `false` 。当前outputManifest文件默认且必须为gz类型压缩文件。

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

查看outputManifest文件内容。

```
[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
```

返回信息如下。

{"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和--previousManifest

`--outputManifest` 表示包含所有已复制文件（旧文件和新文件）的列表， `--previousManifest` 表示只包含之前复制文件的列表。您可以使用 `--outputManifest` 和 `--previousManifest` 重新创建完整的操作历史记录，查看运行期间复制的文件。

例如，在源文件夹中新增加了两个文件，命令如下所示。

```
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
```

执行以下命令，查看运行期间复制的文件。

```
[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
```

返回信息如下。

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

使用 `--outputManifest` 生成清单文件后，您可以使用 `--copyFromManifest` 指定 `--outputManifest` 生成的清单文件，并将dest目录生成的清单文件中包含的文件信息拷贝到新的目录下。

```
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` 可以一次性完成多个文件夹的复制。

示例如下，查看*hourly_table*下文件。

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

返回信息如下。

```
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
```

执行以下命令，复制*hourly_table*下文件到*folders.txt*。

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

查看*folders.txt*文件的内容。

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

返回信息如下。

```
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和-targetSize

因为Hadoop可以从HDFS中读取少量的大文件，而不再读取大量的小文件，所以在大量小文件的场景下，您可以使用Jindo DistCp将小文件聚合为指定大小的大文件，以便于优化分析性能和降低成本。

例如，执行以下命令，查看如下文件夹中的数据。

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

返回信息如下。

```
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
```

执行以下命令，将如下文件夹中的TXT文件合并为不超过10M的文件。

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

经过合并后，可以看到两个TXT文件被合并成了一个文件。

```
[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

在您要拷贝的数据大小均衡、小文件和大文件混合的场景下，因为DistCp默认的执行计划是随机进行文件分配的，所以您可以指定 --enableBalancePlan 来更改Jindo DistCp的作业分配计划，以达到更好的DistCp性能。

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

> **说明** 该参数不支持和 --groupby 或 --targetSize 同时使用。

--enableDynamicPlan

当您要拷贝的数据大小分化严重、小文件数据较多的场景下，您可以指定 --enableDynamicPlan 来更改Jindo DistCp的作业分配计划，以达到更好的DistCp性能。

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

> **说明** 该参数不支持和 --groupby 或 --targetSize 参数一起使用。

--enableTransaction

--enableTransaction 可以保证Job级别的完整性以及保证Job之间的事务支持。示例如下。

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

--diff

DistCp任务完成后，您可以使用 --diff 查看当前DistCp的文件差异。

例如，执行以下命令，查看*/data/incoming/*。

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

如果全部任务完成则会提示如下信息。

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

如果src的文件未能同步到dest上，则会在当前目录下生成*manifest*文件，您可以使用 --copyFromManifest 和 --previousManifest 拷贝剩余文件，从而完成数据大小和文件个数的校验。如果您的DistCp任务包含压缩或者解压缩，则 --diff 不能显示正确的文件差异，因为压缩或者解压缩会改变文件的大小。

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

> **说明** 如果您的 --dest 为HDFS路径，目前仅支持*/path*、*hdfs://hostname:ip/path*和*hdfs://headerIp:ip/path*的写法，暂不支持*hdfs:///path*、*hdfs:/path*和其他自定义写法。

--queue

您可以使用--queue来指定本次DistCp任务所在Yarn队列的名称。

命令示例如下。

```
jindo distcp --src /data/incoming/hourly_table --dest oss://<your_bucket>/hourly_table --queue yarnqueue
```

--bandwidth

您可以使用--bandwidth来指定本次DistCp任务所用的带宽（以MB为单位），避免占用过大带宽。

使用OSS Accesskey

在E-MapReduce外或者免密服务出现问题的情况下，您可以通过指定Accesskey来获得访问OSS的权限。您可以在命令中使用--key、--secret、--endPoint选项来指定Accesskey。

命令示例如下。

```
jindo distcp --src /data/incoming/hourly_table --dest oss://<your_bucket>/hourly_table --key yourkey --secret yoursecret --endP
oint oss-cn-hangzhou.aliyuncs.com --parallelism 20
```

使用归档或低频写入OSS

在您的Distcp任务写入OSS时，您可以通过如下模式写入OSS，数据存储：

- 使用归档（--archive）示例命令如下。

  ```
  jindo distcp --src /data/incoming/hourly_table --dest oss://<your_bucket>/hourly_table --policy archive --parallelism 20
  ```

- 使用低频（--ia）示例命令如下。

  ```
  jindo distcp --src /data/incoming/hourly_table --dest oss://<your_bucket>/hourly_table --policy ia --parallelism 20
  ```

清理残留文件

在您的DistCp任务过程中，由于某种原因在您的目标目录下，产生未正确上传的文件，这部分文件通过uploadId的方式由OSS管理，并且对用户不可见时，您可以通过指定--cleanUpPending选项，指定任务结束时清理残留文件，或者您也可以通过OSS控制台进行清理。

命令示例如下。

```
jindo distcp --src /data/incoming/hourly_table --dest oss://<your_bucket>/hourly_table --cleanUpPending --parallelism 20
```

使用s3作为数据源

您可以在命令中使用--s3Key、--s3Secret、--s3EndPoint选项来指定连接s3的相关信息。

代码示例如下。

```
jindo distcp jindo-distcp-2.7.3.jar --src s3a://yourbucket/ --dest oss://<your_bucket>/hourly_table --s3Key yourkey --s3Secret you
rsecret --s3EndPoint s3-us-west-1.amazonaws.com
```

您可以配置s3Key、s3Secret、s3EndPoint在Hadoop的*core-site.xml*文件里，避免每次使用时填写Accesskey。

```
<configuration>
  <property>
    <name>fs.s3a.access.key</name>
    <value>xxx</value>
  </property>

  <property>
    <name>fs.s3a.secret.key</name>
    <value>xxx</value>
  </property>

  <property>
    <name>fs.s3.endpoint</name>
    <value>s3-us-west-1.amazonaws.com</value>
  </property>
</configuration>
```

此时代码示例如下。

```
jindo distcp /tmp/jindo-distcp-2.7.3.jar --src s3://smartdata1/ --dest s3://smartdata1/tmp --s3EndPoint  s3-us-west-1.amazonaws.com
```

查看Distcp Counters

执行以下命令，在MapReduce的Counter信息中查找Distcp Counters的信息。

```
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
```

> **说明** 如果您的DistCp操作中包含压缩或者解压缩文件，则 `Bytes Destination Copied` 和 `Bytes Source Read` 的大小可能不相等。

4.7. Jindo AuditLog使用说明

Jindo AuditLog提供块存储模式的审计功能，记录Namespace端的增加、删除和重命名操作信息。

前提条件

- 已创建EMR-3.29.x版本的集群，详情请参见创建集群。
- 已创建存储空间，详情请参见创建存储空间。

背景信息

AuditLog可以分析Namespace端访问信息、发现异常请求和追踪错误等。JindoFS AuditLog存储日志文件至OSS，单个Log文件不超过5 GB。基于OSS的生命周期策略，您可以自定义日志文件的保留天数，清理策略等。因为JindoFS AuditLog提供分析功能，所以您可以通过Shell命令分析指定的日志文件。

审计信息

审计信息示例。

```
2020-07-09 18:29:24.689 allowed=true ugi=hadoop (auth:SIMPLE) ip=127.0.0.1 ns=test-block cmd=CreateFileletRequest src=jfs://test-block/test/test.snappy.parquet dst=null perm=::rwxrwxr-x
```

块存储模式记录的审计信息参数如下所示。

参数	描述
时间	时间格式yyyy-MM-dd hh:mm:ss.SSS。
allowed	本次操作是否被允许： • true • false
ugi	操作用户（包含认证方式信息）。
ip	Client IP。
ns	块存储模式Namespace的名称。
cmd	操作命令。
src	源路径。
dest	目标路径，可以为空。
perm	操作文件Permission信息。

使用AuditLog

1. 进入SmartData服务。
   i. 登录阿里云E-MapReduce控制台。
   ii. 在顶部菜单栏处，根据实际情况选择地域（Region）和资源组。
   iii. 单击上方的**集群管理**页签。
   iv. 在**集群管理**页面，单击相应集群所在行的**详情**。
   v. 在左侧导航栏单击**集群服务 > SmartData**。
2. 进入**namespace**服务配置。
   i. 单击**配置**页签。
   ii. 单击**namespace**。
   
3. 配置如下参数。
   i. 在**namespace**页签，单击右上角的**自定义配置**。

ii. 在**新增配置项**对话框中，新增如下参数。

参数	描述	是否必填
namespace.auditlog.enable	■ true：打开AuditLog功能。 ■ false：关闭AuditLog功能。	是
namespace.auditlog.oss.uri	存储AuditLog的OSS Bucket。 请参见oss://<yourbucket>/auditLog格式配置。 <yourbucket>请替换为待存储的Bucket的名称。	是
namespace.auditlog.oss.accessKey	存储OSS的AccessKey ID。	否
namespace.auditlog.oss.accessSecret	存储OSS的AccessKey Secret。	否
namespace.auditlog.oss.endpoint	存储OSS的Endpoint。	否

iii. 单击**部署客户端配置**。

iv. 在**执行集群操作**对话框中，输入**执行原因**，单击**确定**。

v. 在**确认**对话框中，单击**确定**。

4. 重启服务。

i. 单击右上角的**操作 > 重启Jindo Namespace Service**。

ii. 在**执行集群操作**对话框中，输入**执行原因**，单击**确定**。

iii. 在**确认**对话框中，单击**确定**。

5. 配置清理策略。OSS提供了lifeCycle功能来管理OSS上文件的生命周期，您可以利用该功能来自定义Log文件的清理或者保存时间。

i. 登录 OSS管理控制台。

ii. 单击创建的存储空间。

iii. 在左侧导航栏，单击**基础设置 > 生命周期**，在**生命周期**单击**设置**。

iv. 单击**创建规则**，在**创建生命周期规则**配置各项参数。详情请参见设置生命周期规则。

v. 单击**确定**。

使用Jindo auditLog分析功能

JindoFS为存储在OSS上的AuditLog文件提供Shell命令的分析功能，通过MR任务分析Log文件，提供Top-N活跃操作命令分析、Top-N活跃IP分析。您可以使用 `jindo auditlog` 命令，使用该功能。

Jindo Auditlog的参数说明如下表。

参数	描述	是否必填
--src	存储AuditLog的OSS Bucket。默认为步骤3中namespace.auditlog.oss.uri的值，您也可以自定义该参数。	否
--ns	指定待分析的Namespace。默认为block模式下所有ns。	否
--type	指定分析： • ip：IP地址活跃度。 • cmd： 操作命令活跃度。	是

参数	描述	是否必填
--min	指定时间范围，分钟级别。	否 说明 --min和--day，需要二选一。
--day	指定时间范围，天级别。 day 1，表示当天。	

在E-MapReduce控制台，创建MR类型作业，作业内容示例如下。

```
jindo auditlog --src oss://<yourbucket>/auditlog/ --ns test --type ip --day 1 --top 2
```

返回信息如下。

```
16 openFileStatusRequest
6  deleteFileletRequest
```

4.8. JindoFS FUSE使用说明

本文介绍如何通过FUSE客户端访问JindoFS。FUSE支持Block和JFS Scheme的Cache两种模式。

前提条件

已创建集群，详情请参见创建集群。

背景信息

FUSE是Linux系统内核提供的一种挂载文件系统的方式。通过JindoFS的FUSE客户端，将JindoFS集群上的文件映射到本地磁盘，您可以像访问本地磁盘一样访问JindoFS集群上的数据，无需再使用 `hadoop fs -ls jfs://<namespace>/` 方式访问数据。

挂载

说明 依次在每个节点上执行挂载操作。

1. 使用SSH方式登录到集群主节点，详情请参见使用SSH连接主节点。
2. 执行如下命令，新建目录。

   ```
   mkdir /mnt/jfs
   ```

3. 执行如下命令，挂载目录。

   ```
   jindofs-fuse /mnt/jfs
   ```

   */mnt/jfs*作为FUSE的挂载目录。

读写文件

1. 列出/mnt/jfs/下的所有目录。

   ```
   ls /mnt/jfs/
   ```

   返回用户在服务端配置的所有命名空间列表。

   ```
   test  testcache
   ```

2. 列出命名空间test下面的文件列表。

   ```
   ls /mnt/jfs/test/
   ```

3. 创建目录。

```
mkdir /mnt/jfs/test/dir1
ls /mnt/jfs/test/
```

4. 写入文件。

```
echo "hello world" > /tmp/hello.txt
cp /tmp/hello.txt /mnt/jfs/test/dir1/
```

5. 读取文件。

```
cat /mnt/jfs/test/dir1/hello.txt
```

返回如下信息。

```
hello world
```

如果您想使用Python方式写入和读取文件，请参见如下示例：

1. 使用Python写*write.py*文件，包含如下内容。

```
#!/usr/bin/env python36
with open("/mnt/jfs/test/test.txt",'w',encoding = 'utf-8') as f:
  f.write("my first file\n")
  f.write("This file\n\n")
  f.write("contains three lines\n")
```

2. 使用Python读文件。创建脚本*read.py*文件，包含如下内容。

```
#!/usr/bin/env python36
with open("/mnt/jfs/test/test.txt",'r',encoding = 'utf-8') as f:
   lines = f.readlines()
   [print(x, end = '') for x in lines]
```

读取写入*test.txt*文件的内容。

```
[hadoop@emr-header-1 ~]$ ./read.py
```

返回如下信息。

```
my first file
This file
```

卸载

> **说明** 依次在每个节点上执行卸载操作。

1. 使用SSH方式登录到集群主节点，详情请参见使用SSH连接主节点。
2. 执行如下命令，卸载FUSE。

```
umount jindofs-fuse
```

如果出现 `target is busy` 错误，请切换到其它目录，停止所有正在读写FUSE文件的程序，再执行卸载操作。

5.JindoFS基础使用（EMR-3.30.x版本）

5.1. SmartData版本说明

SmartData组件是EMR JindoFS引擎的存储部分，为EMR各个计算引擎提供统一的存储、缓存和计算优化以及功能扩展。SmartData组件主要包括JindoFS，JindoTable和相关工具集。本文介绍SmartData（3.0.0）版本的更新内容。

JindoFS存储优化

- 改进JindoFS Namespace服务单机配置，单机情况下也可以更新并异步写入元数据至Tablestore。
- 移除JindoFS Namespace服务的Tablestore作为元数据后端的配置，不再支持基于Tablestore的HA方案。
- 支持归档存储，允许文件数据按照OSS归档类型进行存储，以节省成本。
- 提供JindoFS分层存储的Archive、Unarchive和Status命令，允许归档至指定目录，查看归档操作进度和相关状态。
- 提供JindoFS ls2命令，允许查看文件信息。
- 支持JindoFS存储系统fsimage的离线导出和分析查询。
- 支持跨集群访问JindoFS存储系统。

JindoFS分层存储命令详情请参见JindoFS分层存储命令使用说明

JindoFS缓存优化

- 改进缓存数据磁盘组织，解除对系统盘的依赖，实现数据盘之间完全独立，增强磁盘下线操作。
- 改进缓存服务，增强节点容错处理和节点下线操作。
- 改进缓存块写入磁盘的选择策略，默认支持轮询（Round Robin）。
- 改进读写流程，增强容错处理。
- 提供JindoFS分层存储的Cache、Uncache和Status命令，允许缓存至指定目录，支持数据预加载，查看缓存进度和相关状态。
- 优化小文件占用缓存空间的问题，准确地统计相关指标。

JindoTable计算优化

- 提供JindoTable Optimize命令，支持优化Hive表操作，例如分区小文件合并。
- 提供JindoTable Archive、Unarchive和Status命令，允许归档至指定表和分区，查看归档操作进度和相关状态。
- 支持JindoTable Cache、Uncache和Status命令，允许缓存至指定表和分区，支持数据预加载，查看缓存进度和相关状态。
- 支持导出MaxCompute表至JindoFS缓存系统上，以实现机器学习训练前结构化数据的预加载机制。

JindoTable详情请参见JindoTable使用说明

JindoFS OSS扩展和支持

- 支持在客户端进行Ranger权限集成，获取OSS各种操作，通过JindoFS服务记录进行Ranger权限检查。
- 支持在客户端进行操作审计，获取OSS各种操作，通过JindoFS服务记录操作记录，作为审计用途。
- 支持Hadoop Credentials Provider框架，允许按照Hadoop常用方式指定OSS的Accesskey配置。
- 支持Flink Connector，允许Flink引擎将OSS作为source、sink和checkpoint存储。
- 提供JindoFS OSS SDK（Hadoop Connector）轻量版本（lite），主要适用于非标准环境，例如用户的IDC（Internet Data Center）集群环境。

JindoManager系统管理

支持通过UI来查看JindoFS存储系统上的系统状态、文件统计和缓存系统上的缓存指标统计。

JindoTools工具箱

改进JindoDistCp工具的分发机制，针对EMR集群内使用场景和非EMR集群环境使用场景，分别使用不同的发行包。

JindoDistCp提供轻量版本（lite），主要适用于非标准环境，例如用户的IDC集群环境。

5.2. JindoFS Block模式使用说明

Block模式提供了最为高效的数据读写能力和元数据访问能力。数据以Block形式存储在后端存储OSS上，本地提供缓存加速，元数据则由本地Namespace服务维护，提供高效的元数据访问性能。本文主要介绍JindoFS的Block模式及其使用方式。

背景信息

JindoFS Block模式具有以下几个特点：

- 海量弹性的存储空间，基于OSS作为存储后端，存储不受限于本地集群，而且本地集群能够自由弹性伸缩。
- 能够利用本地集群的存储资源加速数据读取，适合具有一定本地存储能力的集群，能够利用有限的本地存储提升吞吐率，特别对于一写多读的场景效果显著。
- 元数据操作效率高，能够与HDFS相当，能够有效规避OSS文件系统元数据操作耗时以及高频访问下可能引发不稳定的问题。
- 能够最大限度保证执行作业时的数据本地化，减少网络传输的压力，进一步提升读取性能。

配置使用方式

1. 进入SmartData服务。
   i. 登录阿里云E-MapReduce控制台。
   ii. 在顶部菜单栏处，根据实际情况选择地域（Region）和资源组。
   iii. 单击上方的**集群管理**页签。
   iv. 在**集群管理**页面，单击相应集群所在行的**详情**。
   v. 在左侧导航栏单击**集群服务 > SmartData**。
2. 进入**namespace**服务配置。
   i. 单击**配置**页签。
   ii. 单击**namespace**。
3. 配置以下参数。JindoFS支持多命名空间，本文命名空间以test为例。
   i. 修改**jfs.namespaces**为**test**。**test**表示当前JindoFS支持的命名空间，多个命名空间时以逗号隔开。
   ii. 单击**自定义配置**，在**新增配置项**对话框中增加以下参数，单击**确定**。

参数	参数说明	示例
jfs.namespaces.test.oss.uri	表示test命名空间的后端存储。	*oss://<oss_bucket>/<oss_dir>/* **说明** 推荐配置到OSS bucket下的某一个具体目录，该命名空间即会将Block模式的数据块存放在该目录下。
jfs.namespaces.test.mode	表示test命名空间为块存储模式。	block
jfs.namespaces.test.oss.access.key	表示存储后端OSS的AccessKey ID。	*xxxx* **说明** 考虑到性能和稳定性，推荐使用同账户、同Region下的OSS bucket作为存储后端，此时，E-MapReduce集群能够免密访问OSS，无需配置AccessKey ID和AccessKey Secret。
jfs.namespaces.test.oss.access.secret	表示存储后端OSS的AccessKey Secret。	

   iii. 单击**确定**。
4. 单击右上角的**保存**。
5. 单击右上角的**操作 > 重启 Jindo Namespace Service**。重启后即可通过 `jfs://test/<path_of_file>` 的形式访问JindoFS上的文件。

磁盘空间水位控制

JindoFS后端基于OSS，可以提供海量的存储，但是本地盘的容量是有限的，因此JindoFS会自动淘汰本地较冷的数据备份。我们提供了 storage.watermark.high.ratio 和 storage.watermark.low.ratio 两个参数来调节本地存储的使用容量，值均为0～1的小数，表示使用磁盘空间的比例。

1. 修改磁盘水位配置。 在**服务配置**区域的**storage**页签，修改如下参数。

参数	描述
storage.watermark.high.ratio	表示磁盘使用量的上水位比例，每块数据盘的JindoFS数据目录占用的磁盘空间到达上水位即会触发清理。默认值：0.4。
storage.watermark.low.ratio	表示使用量的下水位比例，触发清理后会自动清理冷数据，将JindoFS数据目录占用空间清理到下水位。默认值：0.2。

> **说明** 您可以通过设置上水位比例调节期望分给JindoFS的磁盘空间，下水位必须小于上水位，设置合理的值即可。

2. 保存配置。
   i. 单击右上角的**保存**。
   ii. 在**确认修改**对话框中，输入执行原因，开启**自动更新配置**。
   iii. 单击**确定**。
3. 重启Jindo Storage Service使配置生效。
   i. 单击右上角的**操作 > 重启Jindo Storage Service**。
   ii. 在**执行集群操作**对话框中，设置相关参数。
   iii. 单击**确定**。
   iv. 在**确认**对话框中，单击**确定**。

5.3. JindoFS缓存模式使用说明

缓存模式（Cache）主要兼容原生OSS存储方式，文件以对象的形式存储在OSS上，每个文件根据实际访问情况会在本地进行缓存，提升EMR集群内访问OSS的效率，同时兼容了原有OSS原有文件形式，数据访问上能够与其他OSS客户端完全兼容。本文主要介绍JindoFS的缓存模式及其使用方式。

背景信息

缓存模式最大的特点就是兼容性，保持了OSS原有的对象语义，集群中仅做缓存，因此和其他的各种OSS客户端是完全兼容的，对原有OSS上的存量数据也不需要做任何的迁移、转换工作即可使用。同时集群中的缓存也能一定程度上提升数据访问性能，缓解读写OSS的带宽压力。

配置使用方式

JindoFS缓存模式提供了以下两种基本使用方式，以满足不同的使用需求。

- OSS Scheme

  详情请参见配置OSS Scheme（推荐）。

- JFS Scheme

  详情请参见配置JFS Scheme。

配置OSS Scheme（推荐）

OSS Scheme保留了原有OSS文件系统的使用习惯，即直接通过 oss://<bucket_name>/<path_of_your_file> 的形式访问OSS上的文件。使用该方式访问OSS，无需进行额外的配置，创建EMR集群后即可使用，对于原有读写OSS的作业也无需做任何修改即可运行。

配置JFS Scheme

1. 进入SmartData服务。

i. 登录阿里云E-MapReduce控制台。
ii. 在顶部菜单栏处，根据实际情况选择地域（Region）和资源组。
iii. 单击上方的**集群管理**页签。
iv. 在**集群管理**页面，单击相应集群所在行的**详情**。
v. 在左侧导航栏单击**集群服务 > SmartData**。

2. 进入**namespace**服务配置。
   i. 单击**配置**页签。
   ii. 单击**namespace**。

3. 配置以下参数。JindoFS支持多命名空间，本文命名空间以test为例。
   i. 修改**jfs.namespaces**为**test**。**test**表示当前JindoFS支持的命名空间，多个命名空间时以逗号隔开。
   ii. 单击**自定义配置**，在**新增配置项**对话框中增加以下参数。

参数	参数说明	示例
jfs.namespaces.test.oss.uri	表示test命名空间的后端存储。	*oss://<oss_bucket>/<oss_dir>/* **说明** 该配置必须配置到OSS Bucket下的具体目录，也可以直接使用根目录。
jfs.namespaces.test.mode	表示test命名空间为缓存模式。	cache

4. 单击**确定**。
5. 保存配置。
   i. 单击右上角的**保存**。
   ii. 在**确认修改**对话框中，输入执行原因，开启**自动更新配置**。
   iii. 单击**确定**。
6. 单击右上角的**操作 > 重启 Jindo Namespace Service**。重启后即可通过 `jfs://test/<path_of_file>` 的形式访问JindoFS上的文件。

启用缓存

启用缓存会利用本地磁盘对访问的热数据块进行缓存，默认状态为禁用，即所有OSS读取都直接访问OSS上的数据。

1. 在**集群服务 > SmartData**的**配置**页面，单击**client**页签。
2. 修改**jfs.cache.data-cache.enable**为**true**，表示启用缓存模式。此配置无需重启SmartData服务。
3. 保存配置。
   i. 单击右上角的**保存**。
   ii. 在**确认修改**对话框中，输入执行原因，开启**自动更新配置**。
   iii. 单击**确定**。

缓存模式启用后，Jindo服务会自动管理本地缓存备份，通过水位清理本地缓存，请您根据需求配置一定的比例用于缓存，详情请参见磁盘空间水位控制。

磁盘空间水位控制

JindoFS后端基于OSS，可以提供海量的存储，但是本地盘的容量是有限的，因此JindoFS会自动淘汰本地较冷的数据备份。我们提供了 `storage.watermark.high.ratio` 和 `storage.watermark.low.ratio` 两个参数来调节本地存储的使用容量，值均为0～1的小数，表示使用磁盘空间的比例。

1. 修改磁盘水位配置。 在**服务配置**区域的**storage**页签，修改如下参数。

参数	描述
storage.watermark.high.ratio	表示磁盘使用量的上水位比例，每块数据盘的JindoFS数据目录占用的磁盘空间到达上水位即会触发清理。默认值：0.4。
storage.watermark.low.ratio	表示使用量的下水位比例，触发清理后会自动清理冷数据，将JindoFS数据目录占用空间清理到下水位。默认值：0.2。

> **说明** 您可以通过设置上水位比例调节期望分给JindoFS的磁盘空间，下水位必须小于上水位，设置合理的值即可。

2. 保存配置。
   i. 单击右上角的**保存**。
   ii. 在**确认修改**对话框中，输入执行原因，开启**自动更新配置**。
   iii. 单击**确定**。
3. 重启Jindo Storage Service使配置生效。
   i. 单击右上角的**操作 > 重启Jindo Storage Service**。
   ii. 在**执行集群操作**对话框中，设置相关参数。
   iii. 单击**确定**。
   iv. 在**确认**对话框中，单击**确定**。

访问OSS Bucket

在EMR集群中访问同账号、同区域的OSS Bucket时，默认支持免密访问，即无需配置任何AccessKey即可访问。如果访问非以上情况的OSS Bucket需要配置相应的AccessKey ID、AccessKey Secret以及Endpoint，针对两种使用方式相应的配置分别如下：

- OSS Scheme
  i. 在**集群服务 > SmartData**的**配置**页面，单击**smartdata-site**页签。
  ii. 单击**自定义配置**，在**新增配置项**对话框中增加以下参数，单击**确定**。

参数	参数说明
fs.jfs.cache.oss.accessKeyId	表示存储后端OSS的AccessKey ID。
fs.jfs.cache.oss.accessKeySecret	表示存储后端OSS的AccessKey Secret。
fs.jfs.cache.oss.endpoint	表示存储后端OSS的endpoint。

> **说明** 兼容EMR-3.30.0之前版本的配置项。

- JFS Scheme
  i. 在**集群服务 > SmartData**的**配置**页面，单击**namespace**页签。
  ii. 修改**jfs.namespaces**为**test**。
  iii. 单击**自定义配置**，在**新增配置项**对话框中增加以下参数，单击**确定**。

参数	参数说明
jfs.namespaces.test.oss.uri	表示test命名空间的后端存储。示例：*oss://<oss_bucket.endpoint>/<oss_dir>*。 endpoint信息直接配置在oss.uri中。
jfs.namespaces.test.oss.access.key	表示存储后端OSS的AccessKey ID。
jfs.namespaces.test.oss.access.secret	表示存储后端OSS的AccessKey Secret。

高级配置

Cache模式还包含一些高级配置，用于性能调优，以下配置均为客户端配置，修改后无需重启SmartData服务。

- 在**服务配置**区域的**client**页签，配置以下参数。

参数	参数说明
client.oss.upload.threads	每个文件写入流的OSS上传线程数。默认值：4。
client.oss.upload.max.parallelism	进程级别OSS上传总并发度上限，防止过多上传线程造成过大的带宽压力以及过大的内存消耗。默认值：16。

- 在**服务配置**区域的**smartdata-site**页签，配置以下参数。

参数	参数说明
fs.jfs.cache.write.buffer.size	文件写入流的buffer大小，参数值必须为2的幂次，最大为 8MB，如果作业同时打开的写入流较多导致内存使用过大，可以适当调小此参数。默认值：1048576。
fs.oss.committer.magic.enabled	启用Jindo Job Committer，避免Job Committer的rename操作，来提升性能。默认值：true。 **说明** 针对Cache模式下，这类OSS对象存储rename操作性能较差的问题，推出了Jindo Job Committer。

5.4. 使用JindoFS SDK免密功能

本文介绍使用JindoFS SDK时，E-MapReduce（简称EMR）集群外如何以免密方式访问E-MapReduce JindoFS的文件系统。

前提条件

适用环境：ECS（EMR环境外）+Hadoop+JavaSDK。

背景信息

使用JindoFS SDK时，需要把环境中相关Jindo的包从环境中移除，如*jboot.jar*、*smartdata-aliyun-jfs-*.jar*。如果要使用Spark则需要把*/opt/apps/spark-current/jars/*里面的包也删除，从而可以正常使用。

步骤一：创建实例RAM角色

1. 使用云账号登录RAM的控制台。
2. 单击左侧导航栏的**RAM角色管理**。
3. 单击**创建 RAM 角色**，选择当前可信实体类型为**阿里云服务**。
4. 单击**下一步**。
5. 输入**角色名称**，从**选择授信服务**列表中，选择**云服务器**。
6. 单击**完成**。

步骤二：为RAM角色授予权限

1. 使用云账号登录RAM的控制台。
2. （可选）如果您不使用系统权限，可以参见账号访问控制创建自定义权限策略章节创建一个自定义策略。
3. 单击左侧导航栏的**RAM角色管理**。
4. 单击新创建RAM角色名称所在行的**精确授权**。
5. 选择权限类型为**系统策略**或**自定义策略**。
6. 输入策略名称。
7. 单击**确定**。

步骤三：为实例授予RAM角色

1. 登录ECS管理控制台。
2. 在左侧导航栏，单击**实例与镜像 > 实例**。
3. 在顶部状态栏左上角处，选择地域。
4. 找到要操作的ECS实例，选择**更多 > 实例设置 > 授予/收回RAM角色**。

5. 在弹窗中，选择创建好的实例RAM角色，单击**确定**完成授予。

步骤四：在ECS上设置环境变量

执行如下命令，在ECS上设置环境变量。

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

或者执行如下命令。

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

步骤五：测试免密方式访问的方法

1. 使用Shell访问OSS。

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

2. 使用Hadoop FileSystem访问OSS。JindoFS SDK支持使用Hadoop FileSystem访问OSS，示例代码如下。

   ```
   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);
         }
      }
   }
   ```

5.5. 跨集群访问JindoFS

通常E-MapReduce集群之间相互独立，每个集群的客户端只能连接并访问本集群内配置的namespace。在多集群的情况下，您可以通过配置JindoFS实现跨集群互访。本文以集群A访问集群B为例，介绍如何跨集群访问JindoFS。

前提条件

- 已创建EMR-3.30.0及后续版本的同一VPC下的集群A和B，详情请参见创建集群。
- 配置*/etc/hosts*文件，同步B集群所有节点的hosts至A集群。

修改配置

1. 进入SmartData服务。
   i. 登录阿里云E-MapReduce控制台。
   ii. 在顶部菜单栏处，根据实际情况选择地域（Region）和资源组。
   iii. 单击上方的**集群管理**页签。
   iv. 在**集群管理**页面，单击相应集群所在行的**详情**。
   v. 在左侧导航栏单击**集群服务 > SmartData**。
2. 进入**client**服务配置。
   i. 单击**配置**页签。
   ii. 在**服务配置**区域，单击**client**页签。
3. 修改配置信息，实现跨集群访问。根据B集群的**namespace.backend.type**参数配置A集群：
   - 当B集群的**namespace.backend.type**为**rocksdb**时，执行如下操作：
     a. 单击右上角的**自定义配置**。
     b. 在**新增配置项**对话框中，添加**client.external.namespace.rpc.addresses**为 `emr-header-1.<cluster-B>:8101` ，单击**确定**。

     > **说明** <cluster-B>为集群B的集群ID。

   - 当B集群的**namespace.backend.type**为**raft**时，执行如下操作：
     a. 单击右上角的**自定义配置**。
     b. 在**新增配置项**对话框中，添加**client.external.namespace.rpc.addresses**为 `emr-header-1.<cluster-B>:8101,emr-header-2.<cluster-B>:8101,emr-header-3.<cluster-B>:8101` ，单击**确定**。
4. 保存配置。
   i. 单击右上角的**保存**。
   ii. 在**确认修改**对话框中，输入执行原因，开启**自动更新配置**。
   iii. 单击**确定**。

关联多个集群

client.external.namespace.rpc.addresses配置多个远端地址时，即可实现关联多个集群，不同的集群地址通过英文分号（;）隔开。

例如，集群A需要关联集群B和集群C，B集群（rocksdb实现）地址为 `emr-header-1.<cluster-B>:8101` ，C集群（raft实现）地址为 `emr-header-1.<cluster-C>:8101,emr-header-2.<cluster-C>:8101,emr-header-3.<cluster-C>:8101` 那A集群需要添加的配置信息为 `client.external.namespace.rpc.addresses=emr-header-1.<cluster-B>:8101;emr-header-1.<cluster-C>:8101,emr-header-2.<cluster-C>:8101,emr-header-3.<cluster-C>:8101` 。

5.6. 访问JindoFS Web UI

JindoFS提供了Web UI服务，您可以快速查看集群当前的状态。例如，当前的运行模式、命名空间、集群StorageService信息和启动状态等。

前提条件

通过SSH隧道方式才能访问Web UI，详情请参见通过SSH隧道方式访问开源组件Web UI。

访问JindoFS Web UI

打通SSH隧道后，您可以通过*http://emr-header-1:8101/*访问JindoFS Web UI功能。JindoFS 3.0版本提供总览信息（Overview）、Namespace信息、存储节点信息以及专家功能（Advanced）。

- 总览信息（Overview）

  包含Namespace启动时间、当前状态、元数据后端、当前Storage服务数量和版本信息等。

  

- Namespace信息

  包含当前节点可用的Namespace以及对应的模式和后端。Block模式的Namespace支持查看当前Namespace的统计信息，包括目录数、文件数以及文件总大小等。

  

- StorageService信息

  包含当前集群的 StorageService 列表，以及对应 StorageService的地址、状态、使用量、最近连接时间、启动时间、StorageService编号和内部版本信息等。

  

  单击Node对应链接，可以查看每个磁盘的空间使用情况。

  

- 专家功能（Advanced）

  专家功能目前仅用于JindoFS开发人员排查问题。

5.7. JindoFS元数据服务

5.7.1. 使用RocksDB作为元数据后端

JindoFS元数据服务支持不同的存储后端，默认配置RocksDB为元数据存储后端。本文介绍使用RocksDB作为元数据后端时需要进行的相关配置。

背景信息

RocksDB作为元数据后端时不支持高可用。如果需要高可用，推荐配置Raft作为元数据后端，详情请参见使用Raft-RocksDB-Tablestore作为存储后端。

单机RocksDB作为元数据服务的架构图如下所示。

配置RocksDB

1. 进入SmartData服务。
   i. 登录阿里云E-MapReduce控制台。
   ii. 在顶部菜单栏处，根据实际情况选择地域（Region）和资源组。
   iii. 单击上方的**集群管理**页签。
   iv. 在**集群管理**页面，单击相应集群所在行的**详情**。
   v. 在左侧导航栏单击**集群服务** > **SmartData**。
2. 进入**namespace**服务配置。
   i. 单击**配置**页签。
   ii. 单击**namespace**。

   
3. 设置**namespace.backend.type**为**rocksdb**。
4. 保存配置。
   i. 单击右上角的**保存**。
   ii. 在**确认修改**对话框中，输入执行原因，开启**自动更新配置**。
   iii. 单击**确定**。

5. 单击右上角的**操作 > 重启 Jindo Namespace Service**。
6. （可选）配置远端Tablestore（OTS）异步存储。

   您可以给集群绑定一个Tablestore（OTS）实例，作为Jindo的元数据服务的额外存储介质，本地的元数据信息会异步地同步至您的Tablestore实例上。

   在SmartData服务的**namespace**页签，设置如下参数。

参数	参数说明	示例
namespace.ots.instance	Tablestore实例名称。	emr-jfs
namespace.ots.accessKey	Tablestore实例的AccessKey ID。	*kkkkkk*
namespace.ots.accessSecret	Tablestore实例的AccessKey Secret。	*XXXXXX*
namespace.ots.endpoint	Tablestore实例的endpoint地址，推荐使用VPC地址。	*http://emr-jfs.cn-hangzhou.vpc.tablestore.aliyuncs.com*
namespace.backend.rocksdb.async.ots.enabled	是否开启OTS异步上传，包括： ○ true ○ false 当设置为true时，需要在SmartData服务完成初始化前，开启OTS异步上传功能。 **说明** 如果SmartData服务已完成初始化，则不能再开启该功能。因为OTS的数据已经落后于本地RocksDB的数据。	true

7. 保存配置。
   i. 单击右上角的**保存**。
   ii. 在**确认修改**对话框中，输入执行原因，开启**自动更新配置**。
   iii. 单击**确定**。
8. 单击右上角的**操作 > 启动 All Components**。

从Tablestore恢复元数据信息

如果您在原始集群开启了远端Tablestore异步存储，则Tablestore上会有一份完整的JindoFS元数据的副本。您可以在停止或释放原始集群后，在新创建的集群上恢复原先的元数据，从而继续访问之前保存的文件。

1. 准备工作。
   i. （可选）统计原始集群的元数据信息（文件和文件夹数量）。

   ```
   hadoop fs -count jfs://test/
   ```

   返回信息类似如下。

   ```
    1596    1482809            25 jfs://test/
   ```

   返回文件夹个数是1596，文件个数1481809。

   ii. 停止原始集群的作业，等待30~120秒左右，等待原始集群的数据已经完全同步到Tablestore。执行以下命令查看状态。如果显示`_synced=1`，则表示Tablestore为最新数据，同步完成。

   ```
   jindo jfs -metaStatus
   ```

   返回信息类似如下所示。

   iii. 停止或释放原始集群，确保没有其它集群正在访问当前的Tablestore实例。

2. 创建新集群。新建与Tablestore实例相同Region的EMR集群，暂停SmartData所有服务。
3. 初始化配置。在SmartData服务的**namespace**页签，添加如下参数。

参数	描述	示例
namespace.backend.rocksdb.async.ots.enabled	是否开启OTS异步上传，包括： ○ true ○ false	false
namespace.backend.rocksdb.recovery.mode	是否开启从OTS恢复元数据，包括： ○ true ○ false	true

4. 保存配置。
   i. 单击右上角的**保存**。
   ii. 在**确认修改**对话框中，输入执行原因，开启**自动更新配置**。
   iii. 单击**确定**。
5. 单击右上角的**操作 > 启动 All Components**。
6. 新集群的SmartData服务启动后，自动从OTS恢复元数据到本地Raft-RocksDB上，可以通过以下命令查看恢复进度。

```
jindo jfs -metaStatus
```

如图所示，state为FINISH时表示恢复完成。

7. （可选）执行以下操作，可以比较一下文件数量与原始集群是否一致。此时的集群为恢复模式，也是只读模式。

```
# 对比文件数量一致
[hadoop@emr-header-1 ~]$ hadoop fs -count jfs://test/
      1596     1482809              25 jfs://test/

# 文件可正常读取(cat、get命令)
[hadoop@emr-header-1 ~]$ hadoop fs -cat jfs://test/testfile
this is a test file
# 查看目录
[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

# 只读状态，不可修改文件
[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. 修改配置，将集群设置为正常模式，开启OTS异步上传功能。在SmartData服务的**namespace**页签，设置以下参数。

参数	描述	示例
namespace.backend.rocksdb.async.ots.enabled	是否开启OTS异步上传，包括： ○ true ○ false	true

参数	描述	示例
namespace.backend.rocksdb.recovery.mode	是否开启从OTS恢复元数据，包括： ○ true ○ false	false

9. 重启集群。
   i. 单击上方的**集群管理**页签。
   ii. 在**集群管理**页面，单击相应集群所在行的**更多 > 重启**。

5.7.2. 使用Raft-RocksDB-Tablestore作为存储后端

JindoFS在EMR-3.27.0及之后版本中支持使用Raft-RocksDB-OTS作为Jindo元数据服务（Namespace Service）的存储。1个EMR JindoFS集群创建3个Master节点组成1个Raft实例，实例的每个Peer节点使用本地RocksDB存储元数据信息。

前提条件

- 创建Tablestore实例，推荐使用高性能实例，详情请参见创建实例。

  > **说明** 需要开启事务功能。

- 创建3 Master的EMR集群，详情请参见创建集群。

  > **说明** 如果没有部署方式，请提交工单处理。

背景信息

RocksDB通过Raft协议实现3个节点之间的复制。集群可以绑定1个Tablestore（OTS）实例，作为Jindo的元数据服务的额外存储介质，本地的元数据信息会实时异步地同步到用户的Tablestore实例上。

元数据服务-多机Raft-RocksDB-Tablestore+HA如下图所示。

配置本地raft后端

1. 新建EMR集群后，暂停SmartData所有服务。
   i. 登录阿里云E-MapReduce控制台。
   ii. 在顶部菜单栏处，根据实际情况选择地域（Region）和资源组。
   iii. 单击上方的**集群管理**页签。
   iv. 在**集群管理**页面，单击相应集群所在行的**详情**。
   v. 在左侧导航栏，单击**集群服务 > SmartData**。
   vi. 单击右上角的**操作 > 停止 All Components**。
2. 根据使用需求，添加需要的namespace。
3. 进入SmartData服务的**namespace**页签。
   i. 在左侧导航栏，单击**集群服务 > SmartData**。
   ii. 单击**配置**页签。
   iii. 在**服务配置**区域，单击**namespace**页签。
4. 在SmartData服务的**namespace**页签，设置如下参数。

参数	描述	示例

参数	描述	示例
namespace.backend.type	设置namespace后端存储类型，支持： ○ rocksdb ○ ots ○ raft 默认为rocksdb。	raft
namespace.backend.raft.initial-conf	部署raft实例的3个Master地址（固定值）。	emr-header-1:8103:0,emr-header-2:8103:0,emr-header-3:8103:0
jfs.namespace.server.rpc-address	Client端访问raft实例的3个Master地址（固定值）	emr-header-1:8101,emr-header-2:8101,emr-header-3:8101

> **说明** 如果不需要使用OTS远端存储，直接执行步骤6和步骤7；如果需要使用OTS远端存储，请执行步骤5~步骤7。

5. （可选）配置远端OTS异步存储。在SmartData服务的**namespace**页签，设置如下参数。

参数	参数说明	示例
namespace.ots.instance	Tablestore实例名称。	emr-jfs
namespace.ots.accessKey	Tablestore实例的AccessKey ID。	*kkkkkk*
namespace.ots.accessSecret	Tablestore实例的AccessKey Secret。	*XXXXXX*
namespace.ots.endpoint	Tablestore实例的endpoint地址，通常EMR集群，推荐使用VPC地址。	*http://emr-jfs.cn-hangzhou.vpc.tablestore.aliyuncs.com*
namespace.backend.raft.async.ots.enabled	是否开启OTS异步上传，包括： ○ true ○ false 当设置为true时，需要在SmartData服务完成初始化前，开启OTS异步上传功能。 **说明** 如果SmartData服务已完成初始化，则不能再开启该功能。因为OTS的数据已经落后于本地RocksDB的数据。	true

6. 保存配置。
   i. 单击右上角的**保存**。
   ii. 在**确认修改**对话框中，输入执行原因，开启**自动更新配置**。
   iii. 单击**确定**。
7. 单击右上角的**操作 > 启动 All Components**。

从Tablestore恢复元数据信息

如果您在原始集群开启了远端Tablestore异步存储，则Tablestore上会有1份完整的JindoFS元数据的副本。您可以在停止或释放原始集群后，在新创建的集群上恢复原先的元数据，从而继续访问之前保存的文件。

1. （可选）准备工作。

i. （可选）统计原始集群的元数据信息（文件和文件夹数量）。

```
[hadoop@emr-header-1 ~]$ hadoop fs -count jfs://test/
      1596     1482809              25 jfs://test/
  (文件夹个数）（文件个数）
```

ii. 停止原始集群的作业，等待30~120秒左右，等待原始集群的数据已经完全同步到Tablestore。执行以下命令查看状态。如果LEADER节点显示 `_synced=1` ，则表示Tablestore为最新数据，同步完成。

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

iii. 停止或释放原始集群，确保没有其它集群正在访问当前的Tablestore实例。

2. 创建新集群。新建与Tablestore实例相同Region的EMR集群，暂停SmartData所有服务。详情请参见配置本地raft后端。
3. 初始化配置。在SmartData服务的**namespace**页签，设置以下参数。

参数	描述	示例
namespace.backend.raft.async.ots.enabled	是否开启OTS异步上传，包括： ○ true ○ false	false
namespace.backend.raft.recovery.mode	是否开启从OTS恢复元数据，包括： ○ true ○ false	true

4. 保存配置。
   i. 单击右上角的**保存**。
   ii. 在**确认修改**对话框中，输入执行原因，开启**自动更新配置**。
   iii. 单击**确定**。
5. 单击右上角的**操作 > 启动 All Components**。
6. 新集群的SmartData服务启动后，自动从OTS恢复元数据到本地Raft-RocksDB上，可以通过以下命令查看恢复进度。

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

如图所示，LEADER节点的state为FINISH表示恢复完成。

7. （可选）执行以下操作，可以比较一下文件数量与原始集群是否一致。此时的集群为恢复模式，也是只读模式。

```
# 对比文件数量一致
[hadoop@emr-header-1 ~]$ hadoop fs -count jfs://test/
      1596     1482809              25 jfs://test/

# 文件可正常读取(cat、get命令)
[hadoop@emr-header-1 ~]$ hadoop fs -cat jfs://test/testfile
this is a test file
# 查看目录
[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

# 只读状态，不可修改文件
[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. 修改配置，将集群设置为正常模式，开启OTS异步上传功能。在SmartData服务的**namespace**页签，设置以下参数。

参数	描述	示例
namespace.backend.raft.async.ots.enabled	是否开启OTS异步上传，包括： ○ true ○ false	true
namespace.backend.raft.recovery.mode	是否开启从OTS恢复元数据，包括： ○ true ○ false	false

9. 重启集群。
   i. 单击上方的**集群管理**页签。
   ii. 在**集群管理**页面，单击相应集群所在行的**更多 > 重启**。

5.8. Jindo Job Committer使用说明

本文主要介绍JindoOssCommitter的使用说明。

背景信息

Job Committer是MapReduce和Spark等分布式计算框架的一个基础组件，用来处理分布式任务写数据的一致性问题。

Jindo Job Committer是阿里云E-MapReduce针对OSS场景开发的高效Job Committer的实现，基于OSS的Multipart Upload接口，结合OSS Filesystem层的定制化支持。使用Jindo Job Committer时，Task数据直接写到最终目录中，在完成Job Commit前，中间数据对外不可见，彻底避免了Rename操作，同时保证数据的一致性。

注意

- OSS拷贝数据的性能，针对不同的用户或Bucket会有差异，可能与OSS带宽以及是否开启某些高级特性等因素有关，具体问题可以咨询OSS的技术支持。
- 在所有任务都完成后，MapReduce Application Master或Spark Driver执行最终的Job Commit操作时，会有一个短暂的时间窗口。时间窗口的大小和文件数量线性相关，可以通过增大 `fs.oss.committer.threads` 可以提高并发处理的速度。
- Hive和Presto等没有使用Hadoop的Job Committer。
- E-MapReduce集群中默认打开Jindo Oss Committer的参数。

在MapReduce中使用Jindo Job Committer

1. 进入YARN服务的**mapred-site**页签。
   i. 登录阿里云E-MapReduce控制台。
   ii. 在顶部菜单栏处，根据实际情况选择地域（Region）和资源组。
   iii. 单击上方的**集群管理**页签。
   iv. 在**集群管理**页面，单击相应集群所在行的**详情**。
   v. 在左侧导航栏单击**集群服务 > YARN**。
   vi. 单击**配置**页签。
   vii. 在**服务配置**区域，单击**mapred-site**页签。
2. 针对Hadoop不同版本，在YARN服务中配置以下参数。
   - Hadoop 2.x版本

     在YARN服务的**mapred-site**页签，设置**mapreduce.outputcommitter.class**为com.aliyun.emr.fs.oss.commit.JindoOssCommitter。
   - Hadoop 3.x版本

     在YARN服务的**mapred-site**页签，设置**mapreduce.outputcommitter.factory.scheme.oss**为com.aliyun.emr.fs.oss.commit.JindoOssCommitterFactory。
3. 保存配置。
   i. 单击右上角的**保存**。
   ii. 在**确认修改**对话框中，输入执行原因，开启**自动更新配置**。
   iii. 单击**确定**。
4. 进入SmartData服务的**smartdata-site**页签。
   i. 在左侧导航栏单击**集群服务 > SmartData**。
   ii. 单击**配置**页签。
   iii. 在**服务配置**区域，单击**smartdata-site**页签。
5. 在SmartData服务的**smartdata-site**页签，设置**fs.oss.committer.magic.enabled**为true。
6. 保存配置。
   i. 单击右上角的**保存**。
   ii. 在**确认修改**对话框中，输入执行原因，开启**自动更新配置**。
   iii. 单击**确定**。

说明 在设置**mapreduce.outputcommitter.class**为com.aliyun.emr.fs.oss.commit.JindoOssCommitter后，可以通过开关**fs.oss.committer.magic.enabled**便捷地控制所使用的Job Committer。当打开时，MapReduce任务会使用无需Rename操作的Jindo Oss Magic Committer，当关闭时，JindoOssCommitter和FileOutputCommitter行为一样。

在Spark中使用Jindo Job Committer

1. 进入Spark服务的**spark-defaults**页签。
   i. 在左侧导航栏单击**集群服务 > Spark**。

ii. 单击**配置**页签。

iii. 在**服务配置**区域，单击**spark-defaults**页签。

2. 在Spark服务的**spark-defaults**页签，设置以下参数。

参数	参数值
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

这三个参数分别用来设置写入数据到Spark DataSource表、Spark Parquet格式的DataSource表和Hive表时使用的Job Committer。

3. 保存配置。

i. 单击右上角的**保存**。

ii. 在**确认修改**对话框中，输入执行原因，开启**自动更新配置**。

iii. 单击**确定**。

4. 进入SmartData服务的**smartdata-site**页签。

i. 在左侧导航栏单击**集群服务 > SmartData**。

ii. 单击**配置**页签。

iii. 在**服务配置**区域，单击**smartdata-site**页签。

5. 在SmartData服务的**smartdata-site**页签，设置**fs.oss.committer.magic.enabled**为true。

> **说明** 您可以通过开关 `fs.oss.committer.magic.enabled` 便捷地控制所使用的Job Committer。当打开时，Spark任务会使用无需Rename操作的Jindo Oss Magic Committer，当关闭时，JindoOssCommitter和FileOutputCommitter行为一样。

6. 保存配置。

i. 单击右上角的**保存**。

ii. 在**确认修改**对话框中，输入执行原因，开启**自动更新配置**。

iii. 单击**确定**。

优化Jindo Job Committer性能

当MapReduce或Spark任务写大量文件的时候，您可以调整MapReduce Application Master或Spark Driver中并发执行Commit相关任务的线程数量，提升Job Commit性能。

1. 进入SmartData服务的**smartdata-site**页签。

i. 在左侧导航栏单击**集群服务 > SmartData**。

ii. 单击**配置**页签。

iii. 在**服务配置**区域，单击**smartdata-site**页签。

2. 在SmartData服务的**smartdata-site**页签，设置**fs.oss.committer.threads**为8。 默认值为8。

3. 保存配置。

i. 单击右上角的**保存**。

ii. 在**确认修改**对话框中，输入执行原因，开启**自动更新配置**。

iii. 单击**确定**。

5.9. JindoFS权限功能

本文介绍JindoFS的namespace的存储模式（Block或Cache）支持的文件系统权限功能。Block模式和Cache模式不支持切换。

背景信息

根据您namespace的存储模式，JindoFS支持的系统权限如下：

- 当您namespace的存储模式是Block模式时，支持Unix和Ranger权限。
  - Unix权限：你可以设置文件的777权限，以及Owner和Group。
  - Ranger权限：您可以执行复杂或高级操作。例如使用路径通配符。
- 当您namespace的存储模式是Cache模式时，仅支持Ranger权限。

  您可以执行复杂或高级操作。例如使用路径通配符。

启用JindoFS Unix权限

1. 进入SmartData服务。
   i. 登录阿里云E-MapReduce控制台。
   ii. 在顶部菜单栏处，根据实际情况选择地域（Region）和资源组。
   iii. 单击上方的**集群管理**页签。
   iv. 在**集群管理**页面，单击相应集群所在行的**详情**。
   v. 在左侧导航栏单击**集群服务 > SmartData**。
2. 进入namespace服务配置。
   i. 单击**配置**页签。
   ii. 单击**namespace**。

   
3. 单击**自定义配置**，在**新增配置项**对话框中，设置**Key**为jfs.namespaces.<namespace>.permission.method，**Value**为unix，单击**确定**。
4. 保存配置。
   i. 单击右上角的**保存**。
   ii. 在**确认修改**对话框中，输入执行原因，开启**自动更新配置**。
   iii. 单击**确定**。
5. 重启配置。
   i. 单击右上角的**操作 > 重启 Jindo Namespace Service**。
   ii. 输入执行原因，单击**确定**。

   开启文件系统权限后，使用方式跟HDFS一样。支持以下命令。

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

   如果用户对某一个文件没有权限，将返回如下错误信息。

   

启用JindoFS Ranger权限

您可以在Apache Ranger组件上配置用户权限，在JindoFS上开启Ranger插件后，就可以在Ranger上对JindoFS权限（和其它组件权限）进行一站式管理。

1. 添加Ranger。
   i. 在**namespace**页签，单击**自定义配置**。
   ii. 在**新增配置项**对话框中，设置**Key**为jfs.namespaces.<namespace>.permission.method，**Value**为ranger。
   iii. 保存配置。
      a. 单击右上角的**保存**。
      b. 在**确认修改**对话框中，输入执行原因，开启**自动更新配置**。
      c. 单击**确定**。

iv. 重启配置。

a. 单击右上角的**操作 > 重启 Jindo Namespace Service**。

b. 输入执行原因，单击**确定**。

2. 配置Ranger。

i. 进入Ranger UI页面。详情请参见概述。

ii. Ranger UI添加HDFS service。

iii. 配置相关参数。

参数	描述
Service Name	固定格式：jfs-{namespace_name}。 例如：jfs-test。
Username	自定义。
Password	自定义。
Namenode URL	输入*jfs://{namespace_name}/*。
Authorization Enabled	使用默认值No。
Authentication Type	使用默认值Simple。
dfs.datanode.kerberos.principal	不填写。
dfs.namenode.kerberos.principal	
dfs.secondary.namenode.kerberos.principal	
Add New Configurations	

iv. 单击**Add**。

启用JindoFS Ranger权限+LDAP用户组

如果您在Ranger UserSync上开启了从LDAP同步用户组信息的功能，则JindoFS也需要修改相应的配置，以获取LDAP的用户组信息，从而对当前用户组进行Ranger权限的校验。

1. 在**namespace**页签，单击**自定义配置**。
2. 在**新增配置项**对话框中，参见以下示例设置参数来配置LDAP，单击**确定**。

> **说明** 配置项请遵循开源HDFS内容，详情请参见core-default.xml。

参数	示例
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
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}))

参数	示例
hadoop.security.group.mapping.ldap.search.filter.group	(objectClass=groupOfNames)
hadoop.security.group.mapping.ldap.base	o=emr

3. 保存配置。
   i. 单击右上角的**保存**。
   ii. 在**确认修改**对话框中，输入执行原因，开启**自动更新配置**。
   iii. 单击**确定**。
4. 重启配置。
   i. 单击右上角的**操作 > 重启 All Components**。
   ii. 输入执行原因，单击**确定**。
5. 通过SSH登录emr-header-1节点，配置Ranger UserSync并启用LDAP选项。详情请参见Ranger Usersync集成LDAP。

5.10. JindoFS AuditLog使用说明

Jindo AuditLog提供缓存和Block模式的审计功能，记录Namespace端的增加、删除和重命名操作信息。

前提条件

- 已创建EMR-3.30.0版本的集群，详情请参见创建集群。
- 已创建存储空间，详情请参见创建存储空间。

背景信息

AuditLog可以分析Namespace端访问信息、发现异常请求和追踪错误等。JindoFS AuditLog存储日志文件至OSS，单个Log文件不超过5 GB。基于OSS的生命周期策略，您可以自定义日志文件的保留天数和清理策略等。因为JindoFS AuditLog提供分析功能，所以您可以通过Shell命令分析指定的日志文件。

审计信息

Block模式记录的审计信息参数如下所示。

参数	描述
时间	时间格式yyyy-MM-dd hh:mm:ss.SSS。
allowed	本次操作是否被允许，取值如下： • true • false
ugi	操作用户（包含认证方式信息）。
ip	Client IP。
ns	Block模式namespace的名称。
cmd	操作命令。
src	源路径。
dest	目标路径，可以为空。
perm	操作文件Permission信息。

审计信息示例。

```
2020-07-09 18:29:24.689 allowed=true ugi=hadoop (auth:SIMPLE) ip=127.0.0.1 ns=test-block cmd=CreateFileletRequest src=jfs://test-block/test/test.snappy.parquet dst=null perm=::rwxrwxr-x
```

使用AuditLog

1. 进入SmartData服务。
   i. 登录阿里云E-MapReduce控制台。
   ii. 在顶部菜单栏处，根据实际情况选择地域（Region）和资源组。
   iii. 单击上方的**集群管理**页签。
   iv. 在**集群管理**页面，单击相应集群所在行的**详情**。
   v. 在左侧导航栏单击**集群服务 > SmartData**。
2. 进入**namespace**服务配置。
   i. 单击**配置**页签。
   ii. 单击**namespace**。
3. 配置如下参数。
   i. 在**namespace**页签，单击右上角的**自定义配置**。
   ii. 在**新增配置项**对话框中，新增如下参数。

参数	描述	是否必填
jfs.namespaces.{ns}.auditlog.enable	打开指定namespaces的AuditLog开关，取值如下： ■ true：打开AuditLog功能。 ■ false：关闭AuditLog功能。	是
namespace.sysinfo.oss.uri	存储AuditLog的OSS Bucket。 请参见oss://<yourbucket>/auditLog格式配置。 <yourbucket>请替换为待存储的Bucket的名称。	是
namespace.sysinfo.oss.access.key	存储OSS的AccessKey ID。	否
namespace.sysinfo.oss.access.secret	存储OSS的AccessKey Secret。	否
namespace.sysinfo.oss.endpoint	存储OSS的Endpoint。	否

   iii. 单击**部署客户端配置**。
   iv. 在**执行集群操作**对话框中，输入**执行原因**，单击**确定**。
   v. 在**确认**对话框中，单击**确定**。
4. 重启服务。
   i. 单击右上角的**操作 > 重启Jindo Namespace Service**。
   ii. 在**执行集群操作**对话框中，输入**执行原因**，单击**确定**。
   iii. 在**确认**对话框中，单击**确定**。
5. 配置清理策略。OSS提供了lifeCycle功能来管理OSS上文件的生命周期，您可以利用该功能来自定义Log文件的清理或者保存时间。
   i. 登录 OSS管理控制台。
   ii. 单击创建的存储空间。

iii. 在左侧导航栏，单击**基础设置 > 生命周期**，在**生命周期**单击**设置**。

iv. 单击**创建规则**，在**创建生命周期规则**配置各项参数。详情请参见设置生命周期规则。

v. 单击**确定**。

使用Jindo AuditLog分析功能

JindoFS为存储在OSS上的AuditLog文件提供SQL的分析功能，通过SQL分析相关表，提供Top-N活跃操作命令分析和Top-N活跃IP分析。您可以使用 `jindo sql` 命令，使用该功能。

`jindo sql` 使用Spark-SQL语法，内部嵌入了audit_log_source（auditlog原始数据）、audit_log（auditlog清洗后数据）和fs_image（fsimage日志数据）三个表，audit_log_source和fs_image均为分区表。使用方法如下：

- `jindo sql --help` 查看支持参数的详细信息。常用参数如下。

参数	描述
-f	指定运行的SQL文件。
-i	启动jindo sql后自动运行初始化SQL脚本。

- `show partitions table_name` 获取所有分区。
- `desc formatted table_name` 查看表结构。

因为Jindo sql基于Spark的程序，所以初始资源可能较小，您可以通过环境变量JINDO_SPARK_OPTS来修改初始资源Jindo sql的启动参数，修改示例如下。

```
export JINDO_SPARK_OPTS="--conf spark.driver.memory=4G --conf spark.executor.instances=20 --conf spark.executor.cores=5 --conf spark.executor.memory=20G"
```

示例如下：

- 执行如下命令显示表。

```
show tables;
```

- 执行如下命令显示分区。

```
show partitions audit_log_source;
```

返回信息类似如下。

- 执行如下查询数据。

```
select * from audit_log_source limit 10;
```

返回信息类似如下。

```
select * from audit_log limit 10;
```

返回信息类似如下。

- 执行如下命令统计2020-10-20日不同命令的使用频次。

5.11. Jindo DistCp使用说明

本文介绍JindoFS的数据迁移工具Jindo DistCp的使用方法。

前提条件

- 本地安装了Java JDK 8。
- 已创建EMR-3.28.0或后续版本的集群，详情请参见创建集群。

使用Jindo Distcp

执行以下命令，获取帮助信息。

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

返回信息如下。

```
--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
--ossKey=VALUE  -  Specify your oss key if needed
--ossSecret=VALUE  -  Specify your oss secret if needed
--ossEndPoint=VALUE  -  Specify your oss endPoint if needed
--policy=VALUE  -  Specify your oss storage policy
--cleanUpPending  -  clean up the incomplete upload when distcp job finish
--queue=VALUE  -  Specify yarn queuename if needed
--bandwidth=VALUE  -  Specify bandwidth per map/reduce in MB if needed
--s3Key=VALUE  -  Specify your s3 key
--s3Secret=VALUE  -  Specify your s3 Sercet
--s3EndPoint=VALUE  -  Specify your s3 EndPoint
```

--src和--dest

`--src`表示指定源文件的路径，`--dest`表示目标文件的路径。

Jindo DistCp默认将`--src`目录下的所有文件拷贝到指定的`--dest`路径下。您可以通过指定`--dest`路径来确定拷贝后的文件目录，如果不指定根目录，Jindo DistCp会自动创建根目录。

例如，如果您需要将*/opt/tmp*下的文件拷贝到OSS bucket，可以执行以下命令。

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

--parallelism

`--parallelism` 用于指定MapReduce作业里的mapreduce.job.reduces参数。该参数默认为7，您可以根据集群的资源情况，通过自定义 `--parallelism` 大小来控制DistCp任务的并发度。

例如，将HDFS上*/opt/tmp*目录拷贝到OSS bucket，可以执行以下命令。

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

--srcPattern

`--srcPattern` 使用正则表达式，用于选择或者过滤需要复制的文件。您可以编写自定义的正则表达式来完成选择或者过滤操作，正则表达式必须为全路径正则匹配。

例如，如果您需要复制*/data/incoming/hourly_table/2017-02-01/03*下所有log文件，您可以通过指定 `--srcPattern` 的正则表达式来过滤需要复制的文件。

执行以下命令，查看*/data/incoming/hourly_table/2017-02-01/03*下的文件。

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

返回信息如下。

```
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
```

执行以下命令，复制以log结尾的文件。

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

执行以下命令，查看目标bucket的内容。

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

返回信息如下，显示只复制了以log结尾的文件。

```
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` 可以移动数据并从源位置删除文件。

例如，执行以下命令，您可以将*/data/incoming/*下的*hourly_table*文件移动到OSS bucket中，并删除源位置文件。

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

--outputCodec

`--outputCodec` 可以在线高效地存储数据和压缩文件。

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

目标文件夹中的文件已经使用gz编解码器压缩了。

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

返回信息如下：

```
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
```

Jindo DistCp当前版本支持编解码器gzip、gz、lzo、lzop、snappy以及关键字none和keep（默认值）。关键字含义如下：

- none表示保存为未压缩的文件。如果文件已压缩，则Jindo DistCp会将其解压缩。
- keep表示不更改文件压缩形态，按原样复制。

> **说明** 如果您想在开源Hadoop集群环境中使用编解码器lzo，则需要安装gplcompression的native库和hadoop-lzo包。

--outputManifest和--requirePreviousManifest

`--outputManifest` 可以指定生成DistCp的清单文件，用来记录copy过程中的目标文件、源文件和数据量大小等信息。

如果您需要生成清单文件，则指定 `--requirePreviousManifest` 为 `false` 。当前outputManifest文件默认且必须为gz类型压缩文件。

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

查看outputManifest文件内容。

```
[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
```

返回信息如下。

{"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和--previousManifest

`--outputManifest` 表示包含所有已复制文件（旧文件和新文件）的列表， `--previousManifest` 表示只包含之前复制文件的列表。您可以使用 `--outputManifest` 和 `--previousManifest` 重新创建完整的操作历史记录，查看运行期间复制的文件。

例如，在源文件夹中新增加了两个文件，命令如下所示。

```
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
```

执行以下命令，查看运行期间复制的文件。

```
[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
```

返回信息如下。

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

使用 `--outputManifest` 生成清单文件后，您可以使用 `--copyFromManifest` 指定 `--outputManifest` 生成的清单文件，并将dest目录生成的清单文件中包含的文件信息拷贝到新的目录下。

```
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` 可以一次性完成多个文件夹的复制。

示例如下，查看*hourly_table*下文件。

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

返回信息如下。

```
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
```

执行以下命令，复制*hourly_table*下文件到*folders.txt*。

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

查看*folders.txt*文件的内容。

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

返回信息如下。

```
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和-targetSize

因为Hadoop可以从HDFS中读取少量的大文件，而不再读取大量的小文件，所以在大量小文件的场景下，您可以使用Jindo DistCp将小文件聚合为指定大小的大文件，以便于优化分析性能和降低成本。

例如，执行以下命令，查看如下文件夹中的数据。

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

返回信息如下。

```
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
```

执行以下命令，将如下文件夹中的TXT文件合并为不超过10M的文件。

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

经过合并后，可以看到两个TXT文件被合并成了一个文件。

```
[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

在您要拷贝的数据大小均衡、小文件和大文件混合的场景下，因为DistCp默认的执行计划是随机进行文件分配的，所以您可以指定 --enableBalancePlan 来更改Jindo DistCp的作业分配计划，以达到更好的DistCp性能。

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

> **说明** 该参数不支持和 --groupby 或 --targetSize 同时使用。

--enableDynamicPlan

当您要拷贝的数据大小分化严重、小文件数据较多的场景下，您可以指定 --enableDynamicPlan 来更改Jindo DistCp的作业分配计划，以达到更好的DistCp性能。

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

> **说明** 该参数不支持和 --groupby 或 --targetSize 参数一起使用。

--enableTransaction

--enableTransaction 可以保证Job级别的完整性以及保证Job之间的事务支持。示例如下。

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

--diff

DistCp任务完成后，您可以使用 --diff 查看当前DistCp的文件差异。

例如，执行以下命令，查看*/data/incoming/*。

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

如果全部任务完成则会提示如下信息。

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

如果src的文件未能同步到dest上，则会在当前目录下生成*manifest*文件，您可以使用 --copyFromManifest 和 --previousManifest 拷贝剩余文件，从而完成数据大小和文件个数的校验。如果您的DistCp任务包含压缩或者解压缩，则 --diff 不能显示正确的文件差异，因为压缩或者解压缩会改变文件的大小。

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

> **说明** 如果您的 --dest 为HDFS路径，目前仅支持*/path*、*hdfs://hostname:ip/path*和*hdfs://headerIp:ip/path*的写法，暂不支持*hdfs:///path*、*hdfs:/path*和其他自定义写法。

--queue

您可以使用--queue来指定本次DistCp任务所在Yarn队列的名称。

命令示例如下。

```
jindo distcp --src /data/incoming/hourly_table --dest oss://<your_bucket>/hourly_table --queue yarnqueue
```

--bandwidth

您可以使用--bandwidth来指定本次DistCp任务所用的带宽（以MB为单位），避免占用过大带宽。

使用OSS Accesskey

在E-MapReduce外或者免密服务出现问题的情况下，您可以通过指定Accesskey来获得访问OSS的权限。您可以在命令中使用--key、--secret、--endPoint选项来指定Accesskey。

命令示例如下。

```
jindo distcp --src /data/incoming/hourly_table --dest oss://<your_bucket>/hourly_table --key yourkey --secret yoursecret --endPoint oss-cn-hangzhou.aliyuncs.com --parallelism 20
```

使用归档或低频写入OSS

在您的Distcp任务写入OSS时，您可以通过如下模式写入OSS，数据存储：

- 使用归档（--archive）示例命令如下。

  ```
  jindo distcp --src /data/incoming/hourly_table --dest oss://<your_bucket>/hourly_table --policy archive --parallelism 20
  ```

- 使用低频（--ia）示例命令如下。

  ```
  jindo distcp --src /data/incoming/hourly_table --dest oss://<your_bucket>/hourly_table --policy ia --parallelism 20
  ```

清理残留文件

在您的DistCp任务过程中，由于某种原因在您的目标目录下，产生未正确上传的文件，这部分文件通过uploadId的方式由OSS管理，并且对用户不可见时，您可以通过指定--cleanUpPending选项，指定任务结束时清理残留文件，或者您也可以通过OSS控制台进行清理。

命令示例如下。

```
jindo distcp --src /data/incoming/hourly_table --dest oss://<your_bucket>/hourly_table --cleanUpPending --parallelism 20
```

使用s3作为数据源

您可以在命令中使用--s3Key、--s3Secret、--s3EndPoint选项来指定连接s3的相关信息。

代码示例如下。

```
jindo distcp jindo-distcp-2.7.3.jar --src s3a://yourbucket/ --dest oss://<your_bucket>/hourly_table --s3Key yourkey --s3Secret yoursecret --s3EndPoint s3-us-west-1.amazonaws.com
```

您可以配置s3Key、s3Secret、s3EndPoint在Hadoop的*core-site.xml*文件里，避免每次使用时填写Accesskey。

```
<configuration>
  <property>
    <name>fs.s3a.access.key</name>
    <value>xxx</value>
  </property>

  <property>
    <name>fs.s3a.secret.key</name>
    <value>xxx</value>
  </property>

  <property>
    <name>fs.s3.endpoint</name>
    <value>s3-us-west-1.amazonaws.com</value>
  </property>
</configuration>
```

此时代码示例如下。

```
jindo distcp /tmp/jindo-distcp-2.7.3.jar --src s3://smartdata1/ --dest s3://smartdata1/tmp --s3EndPoint  s3-us-west-1.amazonaws.com
```

查看Distcp Counters

执行以下命令，在MapReduce的Counter信息中查找Distcp Counters的信息。

```
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
```

> **说明** 如果您的DistCp操作中包含压缩或者解压缩文件，则 `Bytes Destination Copied` 和 `Bytes Source Read` 的大小可能不相等。

5.12. Jindo DistCp场景化使用指导

本文通过场景化为您介绍如何使用Jindo DistCp。

前提条件

- 已创建EMR-3.30.0版本的集群，详情请参见创建集群。
- 已安装JDK 1.8。

- 根据您使用的Hadoop版本，下载*jindo-distcp-<version>.jar*。
  - Hadoop 2.7及后续版本，请下载jindo-distcp-3.0.0.jar。
  - Hadoop 3.x系列版本，请下载jindo-distcp-3.0.0.jar。

场景预览

Jindo DistCp常用使用场景如下所示：

- 场景一：导入HDFS数据至OSS，需要使用哪些参数？
- 场景二：使用JindoDistCp成功导完数据后，如何验证数据完整性？
- 场景三：导入HDFS数据至OSS时，DistCp任务存在随时失败的情况，该使用哪些参数支持断点续传？
- 场景四：成功导入HDFS数据至OSS，数据不断增量增加，在Distcp过程中可能已经产生了新文件，该使用哪些参数处理？
- 场景五：如果需要指定JindoDistCp作业在Yarn上的队列以及可用带宽，该使用哪些参数？
- 场景五：如果需要指定JindoDistCp作业在Yarn上的队列以及可用带宽，该使用哪些参数？
- 场景六：当通过低频或者归档形式写入OSS，该使用哪些参数？
- 场景七：针对小文件比例和文件大小情况，该使用哪些参数来优化传输速度？
- 场景八：如果需要使用S3作为数据源，该使用哪些参数？
- 场景九：如果需要写入文件至OSS上并压缩（LZO和GZ格式等）时，该使用哪些参数？
- 场景十：如果需要把本次Copy中符合特定规则或者同一个父目录下的部分子目录作为Copy对象，该使用哪些参数？
- 场景十一：如果想合并符合一定规则的文件，以减少文件个数，该使用哪些参数？
- 场景十二：如果Copy完文件，需要删除原文件，只保留目标文件时，该使用哪些参数？
- 场景十三：如果不想将OSS AccessKey这种参数写在命令行里，该如何处理？

场景一：导入HDFS数据至OSS，需要使用哪些参数？

如果您使用的不是EMR环境，当从HDFS上往OSS传输数据时，需要满足以下几点：

- HDFS可访问，有读数据权限。
- 需要提供OSS的AccessKey（AccessKey ID和AccessKey Secret），以及Endpoint信息，且该AccessKey具有写目标Bucket的权限。
- OSS Bucket不能为归档类型。
- 环境可以提交MapReduce任务。
- 已下载JindoDistCp JAR包

本场景示例如下。

```
hadoop jar jindo-distcp-<version>.jar --src /data/incoming/hourly_table --dest oss://yang-hhht/hourly_table --ossKey yourkey --ossSecret yoursecret --ossEndPoint oss-cn-hangzhou.aliyuncs.com --parallelism 10
```

> **说明** 各参数含义请参见Jindo DistCp使用说明。

场景二：使用JindoDistCp成功导完数据后，如何验证数据完整性？

您可以通过以下两种方式验证数据完整性：

- JindoDistCp Counters

  您可以在MapReduce任务结束的Counter信息中，获取Distcp Counters的信息。

```
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
```

参数含义如下：

- Bytes Destination Copied：表示目标端写文件的字节数大小。
- Bytes Source Read：表示源端读文件的字节数大小。
- Files Copied：表示成功Copy的文件数。

- JindoDistCp --diff

  您可以使用 `--diff` 命令，进行源端和目标端的文件比较，该命令会对文件名和文件大小进行比较，记录遗漏或者未成功传输的文件，存储在提交命令的当前目录下，生成manifest文件。

  在场景一的基础上增加 `--diff` 参数即可，示例如下。

  ```
  hadoop jar jindo-distcp-<version>.jar --src /data/incoming/hourly_table --dest oss://yang-hhht/hourly_table --ossKey yourkey
  --ossSecret yoursecret --ossEndPoint oss-cn-hangzhou.aliyuncs.com --diff
  ```

  当全部文件传输成功时，系统返回如下信息。

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

场景三：导入HDFS数据至OSS时，DistCp任务存在随时失败的情况，该使用哪些参数支持断点续传？

在场景一的基础上，如果您的Distcp任务因为各种原因中间失败了，而此时您想支持断点续传，只Copy剩下未Copy成功的文件，此时需要您在进行上一次Distcp任务完成后进行如下操作：

1. 增加一个 `--diff` 命令，查看所有文件是否都传输完成。

   ```
   hadoop jar jindo-distcp-<version>.jar --src /data/incoming/hourly_table --dest oss://yang-hhht/hourly_table --ossKey yourke
   y --ossSecret yoursecret --ossEndPoint oss-cn-hangzhou.aliyuncs.com --diff
   ```

   当所有文件都传输完成，则会提示如下信息。否则，执行。

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

2. 文件没有传输完成时会生成manifest文件，您可以使用 `--copyFromManifest` 和 `--previousManifest` 命令进行剩余文件的Copy。示例如下。

   ```
   hadoop jar jindo-distcp-<version>.jar --src /data/incoming/hourly_table --dest oss://yang-hhht/hourly_table --dest oss://ya
   ng-hhht/hourly_table --previousManifest=file:///opt/manifest-2020-04-17.gz --copyFromManifest --parallelism 20
   ```

   `file:///opt/manifest-2020-04-17.gz` 为您当前执行命令的本地路径。

场景四：成功导入HDFS数据至OSS，数据不断增量增加，在Distcp过程中可能已经产生了新文件，该使用哪些参数处理？

1. 未产生上一次Copy的文件信息，需要指定生成manifest文件，记录已完成的文件信息。

   在场景一的基础上增加 `--outputManifest=manifest-2020-04-17.gz` 和 `--requirePreviousManifest=false` 两个信息，示例如下。

   ```
   hadoop jar jindo-distcp-<version>.jar --src /data/incoming/hourly_table --dest oss://yang-hhht/hourly_table --ossKey yourkey --ossSecret yoursecret --ossEndPoint oss-cn-hangzhou.aliyuncs.com --outputManifest=manifest-2020-04-17.gz --requirePreviousManifest=false --parallelism 20
   ```

   参数含义如下：

   - `--outputManifest`：指定生成的manifest文件，文件名称自定义但必须以gz结尾，例如*manifest-2020-04-17.gz*，该文件会存放在 `--dest` 指定的目录下。
   - `--requirePreviousManifest`：无已生成的历史manifest文件信息。

2. 当前一次Distcp任务结束后，源目录可能已经产生了新文件，这时候需要增量同步新文件。

   在场景一的基础上增加 `--outputManifest=manifest-2020-04-17.gz` 和 `--previousManifest=oss://yang-hhht/hourly_table/manifest-2020-04-17.gz` 两个信息，示例如下。

   ```
   hadoop jar jindo-distcp-<version>.jar --src /data/incoming/hourly_table --dest oss://yang-hhht/hourly_table --ossKey yourkey --ossSecret yoursecret --ossEndPoint oss-cn-hangzhou.aliyuncs.com --outputManifest=manifest-2020-04-17.gz --requirePreviousManifest=false --parallelism 20
   ```

   ```
   hadoop jar jindo-distcp-2.7.3.jar --src /data/incoming/hourly_table --dest oss://yang-hhht/hourly_table --ossKey yourkey --ossSecret yoursecret --ossEndPoint oss-cn-hangzhou.aliyuncs.com --outputManifest=manifest-2020-04-18.gz --previousManifest=oss://yang-hhht/hourly_table/manifest-2020-04-17.gz --parallelism 10
   ```

3. 重复执行步骤2，不断同步增量文件。

场景五：如果需要指定JindoDistCp作业在Yarn上的队列以及可用带宽，该使用哪些参数？

在场景一的基础上需要增加两个参数。两个参数可以配合使用，也可以单独使用。

- `--queue`：指定Yarn队列的名称。
- `--bandwidth`：指定带宽的大小，单位为MB。

示例如下。

```
hadoop jar jindo-distcp-<version>.jar --src /data/incoming/hourly_table --dest oss://yang-hhht/hourly_table --ossKey yourkey --ossSecret yoursecret --ossEndPoint oss-cn-hangzhou.aliyuncs.com --queue yarnqueue --bandwidth 6 --parallelism 10
```

场景六：当通过低频或者归档形式写入OSS，该使用哪些参数？

- 当通过归档形式写入OSS时，需要在在场景一的基础上增加 `--archive` 参数，示例如下。

  ```
  hadoop jar jindo-distcp-<version>.jar --src /data/incoming/hourly_table --dest oss://yang-hhht/hourly_table --ossKey yourkey --ossSecret yoursecret --ossEndPoint oss-cn-hangzhou.aliyuncs.com --archive --parallelism 20
  ```

- 当通过低频形式写入OSS时，需要在在场景一的基础上增加 `--ia` 参数，示例如下。

  ```
  hadoop jar jindo-distcp-<version>.jar --src /data/incoming/hourly_table --dest oss://yang-hhht/hourly_table --ossKey yourkey --ossSecret yoursecret --ossEndPoint oss-cn-hangzhou.aliyuncs.com --ia --parallelism 20
  ```

场景七：针对小文件比例和文件大小情况，该使用哪些参数来优化传输速度？

- 小文件较多，大文件较大情况。

  如果要Copy的所有文件中小文件的占比较高，大文件较少，但是单个文件数据较大，在正常流程中是按照随机方式来进行Copy文件分配，此时如果不做优化很可能造成一个Copy进程分配到大文件的同时也分配到很多小文件，不能发挥最好的性能。

  在场景一的基础上增加 `--enableDynamicPlan` 开启优化选项，但不能和 `--enableBalancePlan` 一起使用。示例如下。

  ```
  hadoop jar jindo-distcp-<version>.jar --src /data/incoming/hourly_table --dest oss://yang-hhht/hourly_table --ossKey yourkey
  --ossSecret yoursecret --ossEndPoint oss-cn-hangzhou.aliyuncs.com --enableDynamicPlan --parallelism 10
  ```

  优化对比如下。

  

- 文件总体均衡，大小差不多情况。

  如果您要Copy的数据里文件大小总体差不多，比较均衡，您可以使用 `--enableBalancePlan` 优化。示例如下。

  ```
  hadoop jar jindo-distcp-<version>.jar --src /data/incoming/hourly_table --dest oss://yang-hhht/hourly_table --ossKey yourkey
  --ossSecret yoursecret --ossEndPoint oss-cn-hangzhou.aliyuncs.com --enableBalancePlan --parallelism 10
  ```

  优化对比如下。

  

场景八：如果需要使用S3作为数据源，该使用哪些参数？

需要在场景一的基础上替换OSS的AccessKey和endPoint信息转换成S3参数：

- `--s3Key` ：连接S3的AccessKey ID。
- `--s3Secret` ：连接S3的AccessKey Secret。
- `--s3EndPoint` ：连接S3的endPoint信息。

示例如下。

```
hadoop jar jindo-distcp-<version>.jar --src s3a://yourbucket/ --dest oss://yang-hhht/hourly_table --s3Key yourkey --s3Secret you
rsecret --s3EndPoint s3-us-west-1.amazonaws.com --parallelism 10
```

场景九：如果需要写入文件至OSS上并压缩（LZO和GZ格式等）时，该使用哪些参数？

如果您想压缩写入的目标文件，例如LZO和GZ等格式，以降低目标文件的存储空间，您可以使用 `--outputCodec` 参数来完成。

需要在场景一的基础上增加 `--outputCodec` 参数，示例如下。

```
hadoop jar jindo-distcp-<version>.jar --src /data/incoming/hourly_table --dest oss://yang-hhht/hourly_table --ossKey yourkey --o
ssSecret yoursecret --ossEndPoint oss-cn-hangzhou.aliyuncs.com --outputCodec=gz --parallelism 10
```

JindoDistCp支持编解码器GZIP、GZ、LZO、LZOP和SNAPPY以及关键字none和keep（默认值）。这些关键字含义如下：

- none表示保存为未压缩的文件。如果文件已压缩，则Jindo DistCp会将其解压缩。
- keep表示不更改文件压缩形态，按原样复制。

> **说明** 如您在开源Hadoop集群环境中使用LZO压缩功能，则您需要安装gplcompression的native库和hadoop-lzo包，

场景十：如果需要把本次Copy中符合特定规则或者同一个父目录下的部分子目录作为Copy对象，该使用哪些参数？

- 如果您需要将Copy列表中符合一定规则的文件进行Copy，需要在场景一的基础上增加 `--srcPattern` 参数，示例如下。

```
hadoop jar jindo-distcp-<version>.jar --src /data/incoming/hourly_table --dest oss://yang-hhht/hourly_table --ossKey yourkey
--ossSecret yoursecret --ossEndPoint oss-cn-hangzhou.aliyuncs.com --srcPattern .*\.log --parallelism 10
```

`--srcPattern`：进行过滤的正则表达式，符合规则进行Copy，否则抛弃。

- 如果您需要Copy同一个父目录下的部分子目录，需要在场景一的基础上增加 `--srcPrefixesFile` 参数。

```
hadoop jar jindo-distcp-<version>.jar --src /data/incoming/hourly_table --dest oss://yang-hhht/hourly_table --ossKey yourkey
--ossSecret yoursecret --ossEndPoint oss-cn-hangzhou.aliyuncs.com --srcPrefixesFile file:///opt/folders.txt --parallelism 20
```

`--srcPrefixesFile`：存储需要Copy的同父目录的文件夹列表的文件。

示例中的*folders.txt*内容如下。

```
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
```

场景十一：如果想合并符合一定规则的文件，以减少文件个数，该使用哪些参数？

需要在场景一的基础上增加如下参数：

- `--targetSize`：合并文件的最大大小，单位MB。
- `--groupBy`：合并规则，正则表达式。

示例如下。

```
hadoop jar jindo-distcp-<version>.jar --src /data/incoming/hourly_table --dest oss://yang-hhht/hourly_table --ossKey yourkey --o
ssSecret yoursecret --ossEndPoint oss-cn-hangzhou.aliyuncs.com --targetSize=10 --groupBy='.*/([a-z]+).*.txt' --parallelism 20
```

场景十二：如果Copy完文件，需要删除原文件，只保留目标文件时，该使用哪些参数？

需要在场景一的基础上，增加 `--deleteOnSuccess` 参数，示例如下。

```
hadoop jar jindo-distcp-<version>.jar --src /data/incoming/hourly_table --dest oss://yang-hhht/hourly_table --ossKey yourkey --o
ssSecret yoursecret --ossEndPoint oss-cn-hangzhou.aliyuncs.com --deleteOnSuccess --parallelism 10
```

场景十三：如果不想将OSS AccessKey这种参数写在命令行里，该如何处理？

通常您需要将OSS AccessKey和endPoint信息写在参数里，但是JindoDistcp可以将OSS的AccessKey ID、AccessKey Secret和endpoint预先写在Hadoop的*core-site.xml*文件里，以避免使用时多次填写的问题。

- 如果您需要保存OSS的Accesskey相关信息，您需要将以下信息保存在*core-site.xml*中。

```
<configuration>
  <property>
    <name>fs.jfs.cache.oss-accessKeyId</name>
    <value>xxx</value>
  </property>

  <property>
    <name>fs.jfs.cache.oss-accessKeySecret</name>
    <value>xxx</value>
  </property>

  <property>
    <name>fs.jfs.cache.oss-endpoint</name>
    <value>oss-cn-xxx.aliyuncs.com</value>
  </property>
</configuration>
```

- 如果您需要保存S3的AccessKey相关信息，您需要将以下信息保存在*core-site.xml*中。

```
<configuration>
  <property>
    <name>fs.s3a.access.key</name>
    <value>xxx</value>
  </property>
  <property>
    <name>fs.s3a.secret.key</name>
    <value>xxx</value>
  </property>
  <property>
    <name>fs.s3.endpoint</name>
    <value>s3-us-west-1.amazonaws.com</value>
  </property>
</configuration>
```

5.13. JindoFS支持Flink写入OSS

EMR-3.30.0版本及后续版本，JindoFS支持Flink写入OSS。当写入OSS的作业发生局部失败时，您可以通过Flink自有的检查点（checkpoint）机制，能够迅速恢复作业，并继续写入。

背景信息

开源Flink版本对写入OSS的支持尚不完整。在流式数据处理场景，如果数据是直接写入OSS的，则不能支持作业的可恢复性写入，即对于一个大规模分布式流处理系统，一旦发生局部失败（通常认为是很难避免的），就会丢失数据或出现重复数据（即不支持EXACTLY_ONCE语义）。

在Flink作业中的用法

- 通用配置

  为了支持EXACTLY_ONCE语义写入OSS，您需要执行如下配置：

  i. 打开Flink的检查点（checkpoint）。

  示例如下。

a. 通过如下方式建立的 StreamExecutionEnvironment。

```
StreamExecutionEnvironment env = StreamExecutionEnvironment.getExecutionEnvironment();
```

b. 执行如下命令，启动checkpoint。

```
env.enableCheckpointing(<userDefinedCheckpointInterval>, CheckpointingMode.EXACTLY_ONCE);
```

ii. 使用可重发的数据源，例如Kafka。

- 便捷使用

  您无需额外引入依赖，只需要选择合适的EMR版本，并使用带*oss://*前缀的路径，就可以使用该功能。EMR版本详情请参见版本概述。

  例如，您通过计算和转换，最终形成一个DataStream<String>的对象OutputStream，并期望将其写入OSS，你可以执行如下命令添加sink。

```
String outputPath = "oss://<user-defined-oss-bucket>/<user-defined-oss-dir>"
StreamingFileSink<String> sink = StreamingFileSink.forRowFormat(
      new Path(outputPath),
      new SimpleStringEncoder<String>("UTF-8")
).build();
outputStream.addSink(sink);
```

自定义配置

您在提交Flink作业时，可以自定义参数。

例如，以yarn-cluster模式提交Flink作业时，通过 `-yD` 配置的 `oss.upload.max.concurrent.uploads` 参数，以控制允许同时上传数据块（part）数量的最大值。示例如下。

```
<flink_home>/bin/flink run -m yarn-cluster -yD oss.upload.max.concurrent.uploads=2 ...
```

本文介绍的功能会自动调用OSS提供的高效的分片上传（Multipart Upload）机制，将待上传的文件分为多个part分别上传，最后组合。默认情况下，该值为当前可用的处理器数量。

5.14. JindoFS分层存储命令使用说明

EMR-3.30版本JindoFS引入分层存储功能。通过该功能您可以根据数据冷热程度选择不同的存储介质来存储数据，以减少数据存储成本，或者加速访问数据的速度。

使用Jindo jfs

执行以下命令，获取帮助信息。

```
[root@emr-header-1 ~]# jindo jfs -help archive
-archive -i/a <path> ... :
 Archive commands.
```

JindoFS分层存储命令均为异步执行，分层存储命令只是启动相关任务执行。

常用命令如下：

- Cache命令
- Uncache命令
- Archive命令
- Unarchive命令
- Status命令
- ls2命令

Cache命令

Cache命令可以备份对应路径的数据至本集群的磁盘，以便于后续可以读取本地数据，无需读取OSS上的数据。

```
jindo jfs -cache -p <path>
```

`-p` 参数可以保证本地数据不受磁盘水位清理。

Uncache命令

Uncache命令可以删除本地集群中的本地备份，只存储数据在OSS标准存储上，以便于后续读取OSS上的数据。

```
jindo jfs -uncache  <path>
```

Archive命令

Archive命令可以归档存储数据，删除本地磁盘上的数据备份，归档OSS上的数据至低频访问存储或者归档存储上。存储类型请参见对象存储OSS的存储类型介绍。

```
jindo jfs -archive -i|-a <path>
```

`-i` 参数可以归档数据至OSS低频存储类型。 `-a` 参数可以归档数据至OSS归档存储类型。

Unarchive命令

Unarchive命令可以将数据从归档（存储类型或者低频存储）类型恢复到低频存储或者标准存储，同时可以临时解冻归档存储类型，使数据临时可读，有效时间为1天。

```
jindo jfs -unarchive -i/-o <path>
```

`-i` 参数可以恢复数据至OSS低频存储类型。 `-o` 参数可以临时解冻归档存储类型，使数据临时可读。

Status命令

Status命令可以查看任务进度信息，默认会统计该路径需要执行分层存储的文件数目以及已经完成的数据。

```
jindo jfs -status -detail/-sync <path>
```

`-detail` 参数可以查看文件进度信息。 `-sync` 参数表示该命令需要同步等待分层存储任务结束才会退出。

ls2命令

JindoFS扩展hadoop ls相关操作，提供ls2命令可以查看文件存储状态。

```
hadoop fs -ls2 <path>
```

返回信息会包含文件的存储类型，示例如下。

```
drwxrwxrwx  - -       0    2020-06-05 04:27 oss://xxxx/warehouse
-rw-rw-rw-  1 Archive   1484 2020-09-23 16:40 oss://xxxx/wikipedia_data.csv
-rw-rw-rw-  1 Standard  1676 2020-06-07 20:04 oss://xxxx/wikipedia_data.json
```

5.15. Credential Provider使用说明

您可以使用Credential Provider配置加密后的AccessKey信息至文件中，避免泄露AccessKey信息。

背景信息

EMR-3.30.0版本支持JindoOSS Credential Provider功能。您可以通过使用Hadoop Credential Provider将加密后的AccessKey信息存入文件，从而避免配置明文AccessKey，根据不同情况选择合适的JindoOSS Credential Provider。

配置JindoOSS Credential Provider

1. 进入SmartData服务。
   i. 登录阿里云E-MapReduce控制台。
   ii. 在顶部菜单栏处，根据实际情况选择地域（Region）和资源组。
   iii. 单击上方的**集群管理**页签。
   iv. 在**集群管理**页面，单击相应集群所在行的**详情**。
   v. 在左侧导航栏单击**集群服务 > SmartData**。
2. 进入**smartdata-site**服务配置。
   i. 单击**配置**页签。
   ii. 在**服务配置**区域，单击**smartdata-site**页签。
3. 添加配置信息。
   i. 在**smartdata-site**页签，单击右上角的**自定义配置**
   ii. 在**新增配置项**对话框中，新增如下配置。

参数	描述
fs.jfs.cache.credentials.provider	配置com.aliyun.emr.fs.auth.AliyunCredentialsProvider的实现类，多个类时使用英文逗号（,）隔开，按照先后顺序读取Credential直至读到有效的Credential。例如，`com.aliyun.emr.fs.auth.TemporaryAliyunCredentialsProvider, com.aliyun.emr.fs.auth.SimpleAliyunCredentialsProvider,com.aliyun.emr.fs.auth.EnvironmentVariableCredentialsProvider`。

您可以根据情况，选择不同的Credential Provider，支持如下Provider:

- TemporaryAliyunCredentialsProvider
- SimpleAliyunCredentialsProvider
- EnvironmentVariableCredentialsProvider
- InstanceProfileCredentialsProvider

使用Hadoop Credential Providers存储AccessKey信息

> **说明** Hadoop Credential Provider详情的使用方法，请参见CredentialProvider API Guide。

fs.jfs.cache.oss.accessKeyId、**fs.jfs.cache.oss.accessKeySecret**和**fs.jfs.cache.oss.securityToken**可以存储至Hadoop Credential Providers。

使用Hadoop提供的命令，存储AccesKey和SecurityToken信息至Credential文件中。命令格式如下。

```
hadoop credential <subcommand> [options]
```

例如，存储AccessKey和Token信息至jceks文件中，jceks代表Java Keystore Provider，除了使用文件权限保护该文件外，您也可以指定密码加密存储信息，如果不指定密码则使用默认字符串加密。

```
hadoop credential create fs.jfs.cache.oss.accessKeyId -value AAA -provider jceks://file/root/oss.jceks
hadoop credential create fs.jfs.cache.oss.accessKeySecret -value BBB -provider jceks://file/root/oss.jceks
hadoop credential create fs.jfs.cache.oss.securityToken -value  CCC -provider jceks://file/root/oss.jceks
```

生成Credenttial文件后，您需要配置下面的参数来指定Provider的类型和位置。

参数	描述
fs.jfs.cache.oss.security.credential.provider.path	配置存储AccessKey的Credential文件。 例如，*jceks://file/${user.home}/oss.jceks*为HOME下的*oss.jceks*文件。

TemporaryAliyunCredentialsProvider

适合使用有时效性的AccessKey和SecurityToken访问OSS的情况。

参数	参数说明
fs.jfs.cache.credentials.provider	com.aliyun.emr.fs.auth.TemporaryAliyunCredentialsProvider
fs.jfs.cache.oss.accessKeyId	OSS的AccessKey Id。
fs.jfs.cache.oss.accessKeySecret	OSS的AccessKey Secret。
fs.jfs.cache.oss.securityToken	OSS的SecurityToken（临时安全令牌）。

SimpleAliyunCredentialsProvider

适合使用长期有效的AccessKey访问OSS的情况。

参数	参数说明
fs.jfs.cache.credentials.provider	com.aliyun.emr.fs.auth.SimpleAliyunCredentialsProvider
fs.jfs.cache.oss.accessKeyId	OSS的AccessKey Id。
fs.jfs.cache.oss.accessKeySecret	OSS的AccessKey Secret。

EnvironmentVariableCredentialsProvider

该方式需要在环境变量中配置以下参数。

参数	参数说明
fs.jfs.cache.credentials.provider	com.aliyun.emr.fs.auth.EnvironmentVariableCredentialsProvider
ALIYUN_ACCESS_KEY_ID	OSS的AccessKey Id。
ALIYUN_ACCESS_KEY_SECRET	OSS的AccessKey Secret。
ALIYUN_SECURITY_TOKEN	OSS的SecityToken（临时安全令牌）。 **说明** 仅配置有时效Token时需要。

InstanceProfileCredentialsProvider

该方式无需配置AccessKey，免密方式访问OSS。

参数	参数说明
fs.jfs.cache.credentials.provider	com.aliyun.emr.fs.auth.InstanceProfileCredentialsProvider

5.16. JindoTable使用说明

JindoTable提供表或分区级别的热度统计、存储分层和表文件优化的功能。本文为您介绍JindoTable的使用方法。

前提条件

- 本地安装了Java JDK 8。
- 已创建EMR-3.30.0或后续版本的集群，详情请参见创建集群。

使用JindoTable

常见命令如下：

- -accessStat
- -cache
- -archive
- -unarchive
- -uncache
- -status
- -optimize
- -showTable
- -showPartition
- -listTables
- -dumpmc

> 注意 指定表时使用 database.table 的格式，指定分区时使用 partitionCol1=1,partitionCol2=2,... 的格式。

-accessStat

- 语法

  jindo table -accessStat {-d} *<days>* **{-n}** <topNums>

- 功能

  查询在指定时间范围内访问最多表或分区的条数。

  <days>和<topNums>应为正整数。天数为1时，表示查询从本地时间当天0:00开始到现在的所有访问记录。

- 示例：查询近七天访问最多的表和分区的20条访问记录。

  ```
  jindo table -accessStat -d 7 -n 20
  ```

-cache

- 语法

  jindo table -cache {-t} <dbName.tableName> **[-p]** <partitionSpec> **[-pin]**

- 功能

  表示缓存指定表或分区的数据至集群本地磁盘上。

  表或分区的路径需要位于OSS或JindoFS。指定表时使用 database.table 的格式，指定分区时使用 partitionCol1=1,partitionCol2=2,... 的格式。指定 -pin 时，在缓存空间不足时尽量不删除相关数据。

- 示例：缓存2020-03-16日db1.t1表的数据至本地磁盘上。

  ```
  jindo table -cache -t db1.t1 -p date=2020-03-16
  ```

-uncache

- 语法

jindo table -uncache {-t} <dbName.tableName> **[-p]** <partitionSpec>

- 功能

  表示删除集群本地磁盘上指定表或分区的缓存数据。

  对应的路径需要位于OSS或JindoFS。指定表时使用 `database.table` 的格式，指定分区时使用 `partitionCol1=1,partitionCol2=2,...` 的格式。

- 示例：
  - 删除集群本地磁盘上表db1.t2的缓存数据。

    ```
    jindo table -cache -t db1.t2
    ```

  - 删除集群本地磁盘上表db1.t1的缓存数据。

    ```
    jindo table -uncache -t db1.t1 -p date=2020-03-16,category=1
    ```

-archive

- 语法

  jindo table -archive {-a|i } {-t} <dbName.tableName> **[-p]** <partitionSpec>

- 功能

  表示降低表或者分区的存储策略级别，默认改为归档存储。

  加上-i使用低频存储。指定表时使用database.table的格式，指定分区时使用`partitionCol1=1,partitionCol2=2,...`的格式。

- 示例：指定表db1.t1缓存至本地磁盘上。

  ```
  jindo table -archive -t db1.t1 -p date=2020-10-12
  ```

-unarchive

- 语法

  jindo table -archive [-o|-i] {-t} <dbName.tableName> **[-p]** <partitionSpec>

- 功能

  表示将归档数据转为标准存储。

  `-o` 将归档数据临时解冻， `-i` 将归档数据转为低频。

-status

- 语法

  jindo table -status {-t} <dbName.tableName> **[-p]** <partitionSpec>

- 功能

  表示查看指定表或者分区的存储状态。

- 示例：
  - 查看表db1.t2的状态。

    ```
    jindo table -status -t db1.t2
    ```

  - 查看表db1.t1在2020-03-16日的状态。

    ```
    jindo table -status -t db1.t1 -p date=2020-03-16
    ```

-optimize

- 语法

  jindo table -optimize {-t} <dbName.tableName>

- 功能

  优化表在存储层的数据组织。

- 示例：优化表db1.t1在存储层的数据组织。

```
jindo table -optimize -t db1.t1
```

-showTable

- 语法

  jindo table -showTable {-t} <dbName.tableName>

- 功能

  如果是分区表，则展示所有分区；如果是非分区表，则返回表的存储情况。

- 示例：展示db1.t1分区表的所有分区。

```
jindo table -showTable -t db1.t1
```

-showPartition

- 语法

  jindo table -showPartition {-t} <dbName.tableName> **[-p]** <partitionSpec>

- 功能

  表示返回分区的存储情况。

- 示例：返回分区表db1.t1在2020-10-12日的存储情况。

```
jindo table -showPartition -t db1.t1 -p date=2020-10-12
```

-listTables

- 语法

  jindo table -listTables [-db] <dbName.tableName>

- 功能

  展示指定数据库中的所有表。不指定 `[-db]` 时默认展示default库中的表。

- 示例：
  - 展示default库中的表。

    ```
    jindo table -listTables
    ```

  - 列出数据库db1中的表。

    ```
    jindo table -listTables -db db1
    ```

-dumpmc

- 语法

  jindo table -dumpmc {-i} <accessId> **{-k}** <accessKey> **{-m}** <numMaps> **{-t}** <tunnelUrl> **{-project}** <projectName> **{-table}** <tablename> **{-p}** <partitionSpec> **{-f}** <csv|tfrecord> **{-o}** <outputPath>

参数	描述	是否必选
-i	阿里云的AccessKey ID。	是
-k	阿里云的AccessKey Secret。	是
-m	map任务数。	是

参数	描述	是否必选
-t		是
-project	Maxcompute的项目空间名。	是
-table	Maxcompute的表名。	是
-p	分区信息 。例如 `pt=xxx` ，多个分区表时用英文逗号（,）分开 `pt=xxx,dt=xxx` 。	是
-f	文件格式。包括： ○ tfrecord ○ csv	是
-o	目的路径。	是

- 功能

  表示Dumpmc Maxcompute表至EMR集群或OSS存储。支持CSV格式和TFRECORD格式。

- 示例：
  - Dumpmc Maxcompute表（TFRECORD格式）至EMR集群。

    ```
    jindo table -dumpmc -m 10 -project mctest_project -table t1 -t http://dt.xxx.maxcompute.aliyun-inc.com -k xxxxxxxxx -i XXXXXX -o /tmp/outputtf1 -f tfrecord
    ```

  - Dumpmc Maxcompute表（CSV格式）至OSS存储。

    ```
    jindo table -dumpmc -m 10 -project mctest_project -table t1 -t http://dt.xxx.maxcompute.aliyun-inc.com -k xxxxxxxxx -i XXXXXX -o oss://bucket1/tmp/outputcsv -f csv
    ```

5.17. JindoFS文件元数据离线分析

EMR-3.30.0及后续版本的Block模式，支持dump整个namespace的元数据信息至OSS中，并通过Jindo Sql工具直接分析元数信息。

背景信息

在HDFS文件系统中，整个分布式文件的元数据存储在名为fsimage的快照文件中。文件中包含了整个文件系统的命名空间、文件、Block和文件系统配额等元数据信息。HDFS支持通过命令行下载整个fsimage文件（xml形式）到本地，以便离线分析元数据信息，而JindoFS无需下载元数据信息至本地。

上传文件系统元数据至OSS

使用Jindo命令行工具上传命名空间的元数据至OSS，命令格式如下。

```
jindo jfs -dumpMetadata <nsName>
```

`<nsName>` 为Block模式对应的namespace名称。

例如，上传并离线分析test-block的元数据。

```
jindo jfs -dumpMetadata test-block
```

当提示如下信息时，表示上传成功并以JSON格式的文件存放在OSS中。

```
Sucessfully upload namespace metadata to OSS.
```

元数据上传路径

元数据信息上传的路径为JindoFS中配置的*sysinfo*的子目录下的metadataDump子目录。

例如，配置的 namespace.sysinfo.oss.uri 为 oss://abc/ ，则上传的文件会在 oss://abc/metadataDump 子目录中。

参数	说明
namespace.sysinfo.oss.uri	存储Bucket和路径。
namespace.sysinfo.oss.endpoint	对应Endpoint信息，支持跨Region。
namespace.sysinfo.oss.access.key	阿里云的AccessKey ID。
namespace.sysinfo.oss.access.secret	阿里云的AccessKey Secret。

批次信息：因为分布式文件系统的元数据会跟随用户的使用发生变化，所以我们每次对元数据进行分析是基于命令执行当时的元数据信息的快照进行的。每次运行Jindo命令进行上传会在目录下，根据上传时间生成对应批次号作为本次上传文件的根目录，以保证每次上传的数据不会被覆盖，您可以根据需要删除历史数据。

- ①表示OSS系统信息配置路径。
- ②表示namespce。
- ③表示批次号。

元数据Schema

上传至OSS的文件系统元信息以JSON文件格式存放。其Schema信息如下。

```
{
 "type":"string",       /*INode类型，FILE文件 DIRECTORY目录*/
 "id": "string",         /*INode id*/
 "parentId" :"string",       /*父节点id*/
 "name":"string",       /*INode名称*/
 "size": "int",       /*INode大小， bigint*/
 "permission":"int",       /*permmsion 以int格式存放*/
 "owner":"string",       /*owner名称*/
 "ownerGroup":"string",    /*owner组名称*/
 "mtime":"int",          /*inode修改时间， bigint*/
 "atime":"int",          /*inode最近访问时间， bigint*/
 "attributes":"string",     /*文件相关属性*/
 "state":"string",         /*INode状态*/
 "storagePolicy":"string",  /*存储策略*/
 "etag":"string"         /*etag*/
}
```

使用Jindo Sql分析元数据

1. 执行如下命令，启动Jindo Sql。

   

2. 查询Jindo Sql可以分析的表格。
   - 使用 show tables 可以查看支持查询分析的表格。目前Jindo Sql内置了审计和元数据信息的分析功能，对应audit_log和fs_image。
   - 使用 show partitions fs_image 可以查看表的fs_image分区信息。每一个分区对应于一次上传 jindo jfs -dumpMetadata 生成的数据。

示例如下。

3. 查询分析元数据信息。Jindo Sql使用Spark-SQL语法。您可以使用sql进行分析和查询fs_image表。

   示例如下。

   namespace和datatime为jindo sql增加的两列，分别对应于namespace名称和上传元数据的时间戳。

   例如：根据某次dump的元数据信息统计该namespace下的目录个数。

使用Hive分析元数据

1. 在Hive中创建Table Schema。

   在Hive中创建对应的元信息以供查询，您可以参考下面的格式在Hive中创建文件系统元信息对应表的Schema。

```
CREATE EXTERNAL TABLE `table_name`
(`type` string,
 `id` string,
 `parentId` string,
 `name` string,
 `size` bigint,
 `permission` int,
 `owner` string,
 `ownerGroup` string,
 `mtime` bigint,
 `atime` bigint,
 `attr` string,
 `state` string,
 `storagePolicy` string,
 `etag` string)
 ROW FORMAT SERDE 'org.apache.hive.hcatalog.data.JsonSerDe'
 STORED AS TEXTFILE
 LOCATION '文件上传的OSS路径';
```

2. 使用Hive进行离线分析。创建完Hive表后，您可以使用Hive SQL分析元数据。

```
select * from table_name limit 200;
```

   示例如下。

5.18. JindoFS FUSE使用说明

本文介绍如何通过FUSE客户端访问JindoFS。FUSE支持Block和JFS Scheme的Cache两种模式。

前提条件

已创建集群，详情请参见创建集群。

背景信息

FUSE是Linux系统内核提供的一种挂载文件系统的方式。通过JindoFS的FUSE客户端，将JindoFS集群上的文件映射到本地磁盘，您可以像访问本地磁盘一样访问JindoFS集群上的数据，无需再使用 `hadoop fs -ls jfs://<namespace>/` 方式访问数据。

挂载

> **说明** 依次在每个节点上执行挂载操作。

1. 使用SSH方式登录到集群主节点，详情请参见使用SSH连接主节点。
2. 执行如下命令，新建目录。

   ```
   mkdir /mnt/jfs
   ```

3. 执行如下命令，挂载目录。

   ```
   jindofs-fuse /mnt/jfs
   ```

   */mnt/jfs*作为FUSE的挂载目录。

读写文件

1. 列出/mnt/jfs/下的所有目录。

   ```
   ls /mnt/jfs/
   ```

   返回用户在服务端配置的所有命名空间列表。

   ```
   test  testcache
   ```

2. 列出命名空间test下面的文件列表。

   ```
   ls /mnt/jfs/test/
   ```

3. 创建目录。

   ```
   mkdir /mnt/jfs/test/dir1
   ls /mnt/jfs/test/
   ```

4. 写入文件。

   ```
   echo "hello world" > /tmp/hello.txt
   cp /tmp/hello.txt /mnt/jfs/test/dir1/
   ```

5. 读取文件。

   ```
   cat /mnt/jfs/test/dir1/hello.txt
   ```

   返回如下信息。

   ```
   hello world
   ```

如果您想使用Python方式写入和读取文件，请参见如下示例：

1. 使用Python写*write.py*文件，包含如下内容。

   ```
   #!/usr/bin/env python36
   with open("/mnt/jfs/test/test.txt",'w',encoding = 'utf-8') as f:
     f.write("my first file\n")
     f.write("This file\n\n")
     f.write("contains three lines\n")
   ```

2. 使用Python读文件。创建脚本*read.py*文件，包含如下内容。

```
#!/usr/bin/env python36
with open("/mnt/jfs/test/test.txt",'r',encoding = 'utf-8') as f:
    lines = f.readlines()
    [print(x, end = '') for x in lines]
```

读取写入*test.txt*文件的内容。

```
[hadoop@emr-header-1 ~]$ ./read.py
```

返回如下信息。

```
my first file
This file
```

卸载

> **说明** 依次在每个节点上执行卸载操作。

1. 使用SSH方式登录到集群主节点，详情请参见使用SSH连接主节点。
2. 执行如下命令，卸载FUSE。

```
umount jindofs-fuse
```

如果出现`target is busy`错误，请切换到其它目录，停止所有正在读写FUSE文件的程序，再执行卸载操作。

6.JindoFS生态

6.1. 迁移Hadoop文件系统数据至JindoFS

本文以OSS为例，介绍如何将Hadoop文件系统上的数据迁移至JindoFS。

迁移数据

- Hadoop FsShell

  对于文件较少或者数据量较小的场景，可以直接使用Hadoop的FsShell进行同步：

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

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

- DistCp

  对于文件较多或者数据量较大的场景，推荐使用Hadoop内置的DistCp进行同步：

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

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

> **说明** 更多DistCp参数可参见DistCp Version2 Guide。

利用JindoFS缓存模式

缓存模式是兼容现有OSS的存储方式：文件会以原生对象的形式存储在OSS上，同时OSS文件通过JindoFS缓存模式访问时，也有机会在本地进行数据和元数据的缓存、加速访问，具体可参见JindoFS缓存模式。

6.2. 使用MapReduce处理JindoFS上的数据

本文介绍如何使用MapReduce读写JindoFS上的数据。

JindoFS配置

已创建名为emr-jfs的命名空间，示例如下：

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

MapReduce简介

Hadoop MapReduce作业一般通过HDFS进行读写，JindoFS目前已兼容大部分HDFS接口，通常只需要将MapReduce作业的输入、输出目录配置到JindoFS，即可实现读写JindoFS上的文件 。

Hadoop Map/Reduce是一个使用简易的软件框架，基于它写出来的应用程序能够运行在由上千个商用机器组成的大型集群上，并以一种可靠容错的方式并行处理上T级别的数据集。一个Map/Reduce作业（job） 通常会把输入的数据集切分为若干独立的数据块，由map任务（task）以完全并行的方式处理它们。框架会对map的输出先进行排序， 然后把结果输入给reduce任务。通常作业的输入和输出都会被存储在文件系统中。 整个框架负责任务的调度和监控，以及重新执行已经失败的任务。

作业的输入和输出

MapReduce作业一般会指明输入/输出的位置（路径），并通过实现合适的接口或抽象类提供map和reduce函数。Hadoop的job client 再加上其他作业的参数提交给ResourceManager，进行调度执行。这种情况下，我们直接修改作业的输入和输出目录即可实现JindoFS的读写。

MapReduce on JindoFS样例

以下是MapReduce作业通过修改输入输出实现JindoFS的读写的例子。

- Teragen数据生成样例

  Teragen是Example中生成随机数据演示程序，在指定目录上生成指定行数的数据，具体命令如下：

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

  替换输出路径，可以把数据输出到JindoFS 上：

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

- Terasort数据生成样例

  Terasort是Example中数据排序演示样例，有输入和输出目录，具体命令如下：

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

  替换输入和输出路径，即可处理JindoFS上的数据：

  ```
  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
  ```

6.3. 使用Hive查询JindoFS上的数据

Apache Hive是Hadoop生态中广泛使用的SQL引擎之一，让用户可以使用SQL实现分布式的查询，Hive中数据主要以undefinedDatabase、Table和Partition的形式进行管理，通过指定位置（Location）对应到后端的数据。

JindoFS配置

已创建名为emr-jfs的命名空间，示例如下：

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

Warehouse/Database/Table/Partition的Location

- Warehouse的Location

  Hive的hive-site中有hive.metastore.warehouse.dir，表示Hive数仓存放数据的默认路径，例如配置成：`jfs://emr-jfs/user/hive/warehouse`。

- Database的Location

  Hive的Database会有一个Location属性，database的Location作为下属Table的默认路径。默认情况下，创建Database不是必须指定Location，默认会使用hive-site中hive.metastore.warehouse.dir的值加上database的名字作为路径。通过下面的命令可以指定Database的Location到JindoFS：

  - 创建Database时指定Location到JindoFS。

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

    例如，创建名为database_on_jindofs，location为`jfs://emr-jfs/warehouse/database_on_jindofs`的Hive数据库。

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

- 修改Database的Location到JindoFS。
  a. 通过SHOW CREATE语句查看Database的Location。

```
SHOW CREATE DATABASE database_name;
```

  b. 一般情况下，默认为warehouse目录，查询结果如下。

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

  c. 通过修改Location，可以把默认路径指定到JindoFS上。此操作不会影响存量表，当新建表没有指定默认Location时，才会使用此目录。

  例如，查看表 jfs_table_name下的某个Partition。

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

- Table/Partition的Location

  Table/Partition的Location与Database类似，对于非Partition表，数据直接存放在Table Location下，Partition表的数据存放在Partition目录下，相关操作如下：

  - 创建Table时指定Location到JindoFS。

```
CREATE [EXTERNAL] TABLE table_name
 [(col_name data_type,...)]
LOCATION 'jfs://emr-jfs/database_dir/table_dir';
```

  - 修改Table/Partition指定Location到 JindoFS。
    a. 通过DESCRIBE语句查看Table/Partition的 location。

```
DESCRIBE FORMATTED [PARTITION partition_spec] table_name;
```

    b. 通过修改Location，可以把默认路径指定到JindoFS上。

```
ALTER TABLE table_name [PARTITION partition_spec] SET LOCATION "jfs_path";
```

    例如，查看表 jfs_table_name下的某个Partition。

```
DESCRIBE FORMATTED jfs_table_name PARTITION (partition_key1=123,partition_key2='xxxx');
```

Hive scratch目录

Hive会把一些临时输出文件和作业计划存储在scratch目录，可以通过设置hive-site的hive.exec.scratchdir把地址指向到JindoFS，也可以通过命令行传参。

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

或者

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

6.4. 使用Spark处理JindoFS上的数据

Spark处理JindoFS上的数据，主要有两种方式，一种是直接调用文件系统接口使用；一种是通过SparkSQL读取存在JindoFS的数据表。

JindoFS配置

已创建名为emr-jfs的命名空间，示例如下：

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

处理JindoFS上的数据

- 调用文件系统

  Spark中读写JindoFS上的数据，与处理其他文件系统的数据类似，以RDD操作为例，直接使用jfs的路径即可：

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

  写入数据：

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

- SparkSQL

  创建数据库、数据表以及分区时指定Location到JindoFS即可，详情请参见使用Hive查询JindoFS上的数据。对于已经创建好的存储在JindoFS上的数据表，直接查询即可。

6.5. 使用Flink处理JindoFS上的数据

本文介绍如何使用Flink处理JindoFS上的数据。

JindoFS配置

已创建名为emr-jfs的命名空间，示例如下：

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

使用JindoFS

Flink作业同样可以将作业的输入输出指定为JindoFS相应Namespace下的路径，即可实现Flink作业对JindoFS数据的交互。

例如，HDFS 上的作业命令如下：

```
flink run -m yarn-cluster -yD taskmanager.network.memory.fraction=0.4 -yD akka.ask.timeout=60s -yjm 2048 -ytm 2048 -ys 4 -yn 1
4 -c xxx.xxx.FlinkWordCount -p 56 XXX.jar --input hdfs:///test//large-input-flink --output hdfs:///runjob/test/large-output-flink"
```

相应的改成如下命令即可：

```
flink run -m yarn-cluster -yD taskmanager.network.memory.fraction=0.4 -yD akka.ask.timeout=60s -yjm 2048 -ytm 2048 -ys 4 -yn 1
4 -c xxx.xxx.FlinkWordCount -p 56 XXX.jar --input jfs://emr-jfs/test/large-input-flink --output jfs://emr-jfs/test/large-output-flink
"
```

6.6. 使用Impala/Presto查询JindoFS上的数据

本文介绍如何使用Impala/Presto查询JindoFS上的数据。

JindoFS配置

已创建名为emr-jfs的命名空间，示例如下：

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

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

使用JindoFS

目前， 在E-MapReduce 3.22.0及以上版本，Impala/Presto支持Hive元数据的读取，对于存储在JindoFS的Hive数据表，E-MapReduce Impala/Presto可以直接读取。

同样，也可以将建表语句的Location设置为JindoFS路径，即可实现表数据落在JindoFS上。

例如，原有HDFS上的建表语句如下：

```
Create external table lineitem (L_ORDERKEY INT, L_PARTKEY INT, L_SUPPKEY INT, L_LINENUMBER INT, L_QUANTITY DOUBLE, L_EXTENDEDPRICE DOUBLE, L_DISCOUNT DOUBLE, L_TAX DOUBLE, L_RETURNFLAG STRING, L_LINESTATUS STRING, L_SHIPDATE STRING, L_COMMITDATE STRING, L_RECEIPTDATE STRING, L_SHIPINSTRUCT STRING, L_SHIPMODE STRING, L_COMMENT STRING) ROW FORMAT DELIMITED FIELDS TERMINATED BY '|' STORED AS TEXTFILE LOCATION 'hdfs:///tpch_impala/lineitem';
```

相应的改成如下命令即可：

```
Create external table lineitem (L_ORDERKEY INT, L_PARTKEY INT, L_SUPPKEY INT, L_LINENUMBER INT, L_QUANTITY DOUBLE, L_EXTENDEDPRICE DOUBLE, L_DISCOUNT DOUBLE, L_TAX DOUBLE, L_RETURNFLAG STRING, L_LINESTATUS STRING, L_SHIPDATE STRING, L_COMMITDATE STRING, L_RECEIPTDATE STRING, L_SHIPINSTRUCT STRING, L_SHIPMODE STRING, L_COMMENT STRING) ROW FORMAT DELIMITED FIELDS TERMINATED BY '|' STORED AS TEXTFILE LOCATION 'jfs://emr-jfs/tpch_impala/lineitem';
```

6.7. 使用JindoFS作为HBase的底层存储

本文介绍如何使用JindoFS作为HBase的底层存储。

背景信息

HBase是Hadoop生态中的实时数据库，有很高的写入性能，E-MapReduce HBase（E-MapReduce 3.22.0及以上版本） 支持使用 JindoFS/OSS作为底层存储，相对于HDFS存储来说，使用更加灵活。

JindoFS配置

已创建名为emr-jfs的命名空间，示例如下：

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

指定HBase的存储路径

由于JindoFS和OSS在E-MapReduce 3.22.0版本暂不支持Sync操作，需要把hbase-site的hbase.rootdir指向JindoFS/OSS地址，hbase.wal.dir指向本地的undefinedHDFS地址，通过本地HDFS集群存储WAL文件。如果要释放集群，需要先Disable table，确保WAL文件已经完全更新到HFile。

配置文件	参数	参数说明	示例
hbase-site	hbase.rootdir	指定HBase的ROOT存储目录到JindoFS	*jfs://emr-jfs/hbase-root-dir*
	hbase.wal.dir	指定HBase的WAL存储目录到本地HDFS集群	*hdfs://emr-cluster/hbase*

创建集群

创建集群详情请参见创建集群。

添加软件自定义配置，如下图所示。

1

使用JindoFS

以JindoFS作为HBase后端为例，替换oss_bucket及对应路径，自定义配置如下：

```
[
 {
    "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"
 }
]
```

6.8. 基于JindoFS存储YARN MR/SPARK作业日志

本文介绍如何将MapReduce、Spark作业日志配置到JindoFS/OSS上。

概述

E-MapReduce集群支持按量计费以及包年包月的付费方式，满足不同用户的使用需求。对于按量计费的集群随时会被释放，而Hadoop默认会把日志存储在HDFS上，当集群释放以后，按量计费的用户就无法查询作业的日志了，这也给按量计费用户排查作业问题带来了困难。本文会介绍如何将MapReduce、Spark作业日志配置到JindoFS/OSS上，集群重新创建以后，也可以继续查询之前作业相关的日志。

JindoFS、YARN Container日志和Spark HistoryServer配置

- JindoFS配置

配置文件	参数	参数说明	示例
bigboot	jfs.namespaces	表示当前JindoFS支持的命名空间，多个命名空间时以逗号隔开。	emr-jfs
	jfs.namespaces.emr-jfs.uri	表示emr-jfs命名空间的后端存储。	*oss://oss-bucket/oss-dir*
	jfs.namespaces.test.mode	表示emr-jfs命名空间为块存储模式。	block **说明** JindoFS支持block和cache两种存储模式。

- YARN Container日志配置

配置文件	参数	参数说明	示例
yarn-site	yarn.nodemanager.remote-app-log-dir	当应用程序运行结束后，日志聚合的存储位置，YARN日志聚合功能默认已打开。	*jfs://emr-jfs/emr-cluster-log/yarn-apps-logs*或者*oss://${oss-bucket}/emr-cluster-log/yarn-apps-logs*
mapred-site	mapreduce.jobhistory.done-dir	JobHistory存放已经运行完的Hadoop作业记录的目录。	*jfs://emr-jfs/emr-cluster-log/jobhistory/done*或者*oss://${oss-bucket}/emr-cluster-log/jobhistory/done*
	mapreduce.jobhistory.intermediate-done-dir	JobHistory存放未归档的Hadoop作业记录的目录。	*jfs://emr-jfs/emr-cluster-log/jobhistory/done_intermediate*或者*oss://${oss-bucket}/emr-cluster-log/jobhistory/done_intermediate*

- Spark HistoryServer配置

配置文件	参数	参数说明	示例
spark-defaults	spark_eventlog_dir	存放Spark作业历史的目录。	*jfs://emr-jfs/emr-cluster-log/spark-history*或者*oss://${oss-bucket}/emr-cluster-log/spark-history*

创建集群

添加软件自定义配置，如下图所示。

JindoFS样例

以JindoFS存储日志为例，替换oss_bucket及对应路径，自定义配置如下：

```
[
 {
    "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"
 }
]
```

以OSS存储日志为例，替换oss_bucket及对应路径，自定义配置如下：

```
[
 {
    "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"
 }
]
```

6.9. 将Kafka数据导入JindoFS

Kafka广泛用于日志收集、监控数据聚合等场景，支持离线或流式数据处理、实时数据分析等。本文主要介绍Kafka数据导入到JindoFS的几种方式。

常见Kafka数据导入方式

- 通过Flume导入

  推荐使用Flume方式导入到JindoFS，利用Flume对HDFS的支持，替换路径到JindoFS即可完成。

  ```
  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
  ```

- 通过调用Kafka API导入

对于MapReduce、Spark以及其他调用Kafka API导入数据的方式，只需引用Hadoop FileSystem，然后使用JindoFS的路径写入即可。

- 通过Kafka Connector导入

  使用Kafka HDFS Connector也可以把Kafka数据导入到Hadoop生态，将sink的输出路径替换成JindoFS的路径即可。

7.JindoCube

7.1. E-MapReduce JindoCube使用说明

JindoCube在E-MapReduce 3.24.0及之后版本中可用。本文主要介绍E-MapReduce JindoCube的安装、部署和使用等。

前提条件

已创建表或者视图。

概述

JindoCube是E-MapReduce Spark支持的高级特性，通过预计算加速数据处理，实现十倍甚至百倍的性能提升。您可以将任意View表示的数据进行持久化，持久化的数据可以保存在HDFS或OSS等任意Spark支持的DataSource中。EMR Spark自动发现可用的已持久化数据，并优化执行计划，对用户完全透明。JindoCube主要用于查询模式相对比较固定的业务场景，通过提前设计JindoCube，对数据进行预计算和预组织，从而加速业务查询的速度，常见的使用场景包括MOLAP多维分析、报表生成、数据Dashboard和跨集群数据同步等。

JindoCube的安装与部署

JindoCube作为EMR Spark组件的高级特性，所有使用EMR Spark提交的Dataset、DataFrame API、SQL任务，均可以基于JindoCube进行加速，无须额外的组件部署与维护。

1. UI页面展示。

   JindoCube主要通过Spark的UI页面进行管理，包括JindoCube的创建、删除和更新等。通过UI创建JindoCube完成后，即可自动用于该集群所有Spark任务的查询加速。通过**spark.sql.cache.tab.display**参数可以控制是否在Spark UI页面展示JindoCube的Tab，可以通过EMR控制台在Spark服务中配置相关参数，或者在Spark提交命令中指定参数值，该参数默认值为**false**。

   JindoCube还提供了**spark.sql.cache.useDatabase**参数，可以针对业务方向，按不同的业务建立database，把需要建cache的view放在这个database中。对于分区表JindoCube还提供了**spark.sql.cache.cacheByPartition**参数，可指定cache使用分区字段进行存储。

参数	说明	示例值
spark.sql.cache.tab.display	显示Cube Mangament页面。	true
spark.sql.cache.useDatabase	cube存储数据库。	db1,db2,dbn
spark.sql.cache.cacheByPartition	按照分区字段存储cube。	true

2. 优化查询。

   spark.sql.cache.queryRewrite用于控制是否允许使用JindoCube中的Cache数据加速Spark查询任务，用户可以在集群、session、SQL等层面使用该配置，默认值为**true**。

JindoCube的使用

1. 创建JindoCube。
   i. 通过主账号登录阿里云 E-MapReduce 控制台。
   ii. 单击**集群管理**页签。
   iii. 单击待操作集群所在行的集群ID。
   iv. 单击左侧导航栏的**访问链接与端口**。
   v. 在**公网访问链接**页面，单击YARN UI所在行的链接，进入Knox代理的YARN UI页面。

      Knox相关使用说明请参见Knox。

   vi. 单击User为spark，Name为Thrift JDBC/ODBC Server所在行的ApplicationMaster。
   vii. 单击最上面的**Cube Management**页签。
   viii. 单击**New Cache**。

用户可以选择某一个表或视图，单击action中的链接继续创建Cache。可以选择的Cache类型分为两类：

- Raw Cache：某一个表或者视图的raw cache，表示将对应表或视图代表的表数据按照指定的方式持久化。

  在创建Raw Cache时，用户需要指定如下信息：

参数	描述	是否必选
Cache Name	指定Cache的名字，支持字母、数字、连接号（-）和下划线（_）的组合。	必选
Column Selector	选择需要Cache哪些列的数据。	必选
Rewrite	是否允许该Cache被用作后续查询的执行计划优化。	必选
Provider	Cache数据的存储格式，支持JSON、PARQUET、ORC等所有Spark支持的数据格式。	必选
Partition Columns	Cache数据的分区字段。	可选
ZOrder Columns	ZOrder是一种支持多列排序的方法，Cache数据按照ZOrder字段排序后，对于基于ZOrder字段过滤的查询会有更好的加速效果。	可选

- Cube Cache：基于某一个表或者视图的原始数据，按照用户指定的方式构建cube，并将cube数据持久化。

  在创建Cube Cache时，用户需要指定如下信息：

参数	描述	是否必选
Cache Name	指定Cache的名字，支持字母、数字、连接号（-）和下划线（_）的组合。	必选
Dimension Selector	选择构建Cube时的维度字段。	必选
Measure Selector	选择构建Cube时的measure字段和measure预计算函数。	必选
Rewrite	是否允许该Cache被用作后续查询的执行计划优化。	必选
Provider	Cache数据的存储格式，支持JSON、PARQUET、ORC等所有Spark支持的数据格式。	必选
Partition Columns	Cache数据的分区字段。	可选
ZOrder Columns	ZOrder是一种支持多列排序的方法，Cache数据按照ZOrder字段排序后，对于基于ZOrder字段过滤的查询会有更好的加速效果。	可选

  JindoCube通过用户指定的Dimension和Measure信息来构建Cube，对于上图的示例，创建的Cube Cache可以用SQL表示为：

```
SELECT c_city, c_nation, c_region, MAX(lo_quantity), SUM(lo_tax)
FROM lineorder_flatten
GROUP BY c_city, c_nation, c_region;
```

  JindoCube计算Cube的最细粒度维度组合，在优化使用更粗粒度的维度组合的查询时，基于Spark强大的现场计算能力，通过重聚合实现。在定义Cube Cache时，必须使用JindoCube支持的预计算函数。JindoCube支持的预计算函数和其对应的聚合函数类型如下：

聚合函数类型	预计算函数
COUNT	COUNT
SUM	SUM
MAX	MAX
MIN	MIN
AVG	COUNT，SUM
COUNT（DISTINCT）	PRE_COUNT_DISTINCT
APPROX_COUNT_DISTINCT	PRE_APPROX_COUNT_DISTINCT

  在**Cube Management**页面，展示所有的Cache列表。单击**Detail**进入Cache的详细页面，在Cache详细页会展示Cache的详细信息、包括基本信息、Cache数据分区信息、构建Cache信息以及构建历史信息等。

2. 构建JindoCube。

   创建JindoCube Cache只是进行元数据操作，Cache表示的数据并未持久化，需要继续构建Cache，从而持久化Cache数据到HDFS或OSS等存储中。此外Cache对应的源表数据可能会新增或者更新，需要更新Cache中的数据从而保持一致。JindoCube支持两类构建操作：

- Build Cache。

  通过Build Cache链接，用户可以主动触发一次构建操作，构建页面相关信息如下：

  

  在构建JindoCube的Cache时，相关用户选项如下：

参数	描述
Save Mode	支持Overwrite和Append两种模式。 ■ Overwrite：会覆盖之前曾经构建的Cache数据。 ■ Append：会新增数据到Cache中。

Optional Filter	用户可以选择额外的过滤条件，在构建时，将该Cache表示的数据过滤后再持久化。
	■ Column：过滤字段。
	■ Filter Type：过滤类型，支持固定值和范围值两种。
	  ■ Fixed Values：指定过滤值，可以多个，以“,”分隔。
	  ■ Range Values：指定范围值的最小和最大值，最大值可以为空，过滤条件包含最小值，不包含最大值。

  上图中构建任务想要构建lineorder_flatten视图的Raw Cache数据，要写入Cache中的数据可以使用如下SQL表示：

```
SELECT * FROM lineorder_flatten
WHERE s_region == 'ASIA' OR s_region == 'AMERICA';
```

  单击**Submit**，提交构建任务，返回到Cache详细页面，对应的构建任务会提交到Spark集群中执行，在Build Information中可以看到当前是否正在构建Cache的信息。在Cache构建完成后，可以在Build History中看到相关的信息。

  > **说明** Cache数据由Spark任务写到一个指定目录中，和普通的Spark写表或者写目录一样，对于Parquet、Json、ORC等数据格式，并发构建同一个Cache可能导致Cache数据不准确，不可用，应避免这种情况。如果无法避免并发构建、更新Cache，可以考虑使用delta等支持并发写的数据格式。

- Trigger Period Build。

  定期更新功能可以方便用户设置自动更新Cache的策略，保持Cache数据和源表数据的一致。相关页面如下：

  

  定期更新的相关用户选项如下：

参数	描述
Save Mode	支持Overwrite和Append两种模式。 ■ Overwrite：会覆盖之前曾经构建的Cache数据。 ■ Append：会新增数据到Cache中。
Trigger Strategy	触发策略，设置触发构建任务的开始时间和间隔时间。 ■ Start At：通过时间控件选择或者手工输入第一次触发构建任务的时间点，日期格式为yyyy-MM-dd hh:mm:ss。 ■ Period：设置触发构建任务的间隔时间。

参数	描述
Optional Step	设置每次触发构建任务的数据筛选条件，通过指定时间类型的字段，配合触发策略中的间隔时间，可以实现按照时间间隔增量的更新Cache。如果不选择，每次全量更新Cache。 ■ Step By：选择增量更新字段类型，只支持时间类型字段，包括Long类型的timestamp字段，以及指定dateformat信息的String类型字段。 ■ Column Name：增量更新字段名称。

在Cache详细页面中，可以看到当前设置的定期更新策略，用户可以随时通过Cancel Period Build取消定期更新。所有触发的构建任务信息在完成后也可以在Build History列表中看到。

> 说明
> ■ 定期更新任务是Spark集群级别的，相关设置保存在SparkContext中，并由Spark Driver定期触发，当Spark集群关闭后，定期更新任务也随之关闭。
> ■ 当前Spark集群所有的构建任务完成后，都会展示在Build History列表中，包含开始/结束时间、SaveMode、构建条件，任务最终状态等。Build History也是Spark集群级别的信息，当Spark集群关闭后，相关信息也随之释放。

3. 管理JindoCube。

   创建和构建JindoCube的Cache数据后，通过Cube Management的UI页面，可以对JindoCube的Cache数据进行进一步的管理。

   - 删除cache。

     在JindoCube Cache列表页面，可以通过action列的**Drop**删除对应Cache，删除成功后，Cache的相关元数据和存储数据都会被清理。

     

   - 开启或关闭Cache优化。

     JindoCube支持在Cache级别，设置是否允许用于Spark查询的优化，在Cache的详细页面，您可以通过基本信息中的**Enabled**或**Disabled**，启用或者停用该Cache，控制是否允许该Cache用于查询加速。

     

   - 删除分区数据。

     如果Cache的数据是按照分区存储的，当确认某些分区数据不再需要时，删除这些分区数据可以节省大量存储空间。在Cache的详细页面，分区Cache的相关分区会通过列表展示，用户可以通过**Delete**删除特定分区的数据。

     

     > 说明 在删除Cache分区数据之前，请谨慎确认，确保该分区数据不会被使用。如果用户的查询经过优化需要用到该Cache被删除的分区数据，会导致错误的查询结果。

4. 查询优化。

   目前JindoCube支持基于View的查询优化，当用户使用某个视图创建了Raw Cache或者Cube Cache后，后续的查询使用到了该视图，EMR Spark会在满足逻辑语义的前提下，尝试使用Cache重写查询的执行计划，新的执行计划直接访问Cache数据，从而加速查询速度。以如下场景为例，lineorder_flatten视图是将lineorder和其他维度表关联之后的大宽表视图，其相关定义如下：

   

   基于lineorder_flatten视图简单查询的执行计划如下：

   

在为line order_flatten视图创建Raw Cache并构建完成后，执行相同查询，EMR Spark会自动使用Cache数据优化执行计划，优化后的执行计划如下：

```
```

可以看到，优化后的执行计划省去了lineorder_flatten视图的所有计算逻辑，直接访问HDFS中Cache的数据。

注意事项

1. JindoCube并不保证Cache数据和源表数据的一致性，而是需要用户通过手工触发或者设置定期策略触发更新任务同步Cache中的数据，用户需要根据查询对于数据一致性的需求，触发Cache的更新任务。
2. 在对查询的执行计划进行优化的时候，JindoCube根据视图的元数据判断是否可以使用Cache优化查询的执行计划。优化后，如果Cache的数据不完整，可能会影响查询结果的完整性或正确性。可能导致Cache数据不完整的情况包括：用户在Cache详情页主动删除查询需要的Cache Partition数据，构建、更新Cache时指定的过滤条件过滤掉了查询需要的数据，查询需要的数据还未及时更新到Cache等。

8.JindoFS常见问题

基本概念

开源和生态

升级和迁移

OSS相关

安全相关

什么是JindoFS？

JindoFS是阿里云开源大数据E-MapReduce产品提供的一套Hadoop文件系统，主要对Hadoop和Spark大数据生态系统使用阿里云OSS提供多层次的封装支持和优化。

基础功能提供适配OSS和支持访问，您可以直接使用JindoFS SDK；标准功能针对OSS提供分布式缓存优化，以对应JindoFS缓存模式；高级功能上针对使用OSS一些特殊或重要场景进行了深度定制，例如，JindoFS Block模式。

已经有阿里云OSS，为什么还要使用JindoFS？

阿里云OSS是对象存储系统，提供基于对象语义的REST API和各种语言SDK封装。JindoFS主要是对阿里云OSS提供HCFS（Hadoop Compatible FileSystem）接口封装，并且在此基础上提供缓存加速能力和高级优化定制的功能。因为Hadoop和Spark生态组件依赖HCFS的抽象接口，所以需要使用JindoFS。

JindoFS有哪些使用方式？使用场景是什么？

JindoFS使用方式包括JindoFS SDK（*jindo-sdk_xxx.jar*）、缓存和Block模式。

针对三种方式，使用场景如下：

- JindoFS SDK模式：简单情况下，您可以使用此模式，上传JindoFS SDK的JAR包至组件的classpath目录。
- 缓存模式：当计算性能受限于IO和存储吞吐时，您可以使用此模式，在计算集群的Core节点上配备、增加或扩容磁盘，以开启数据缓存。
- Block模式：特殊场景，例如对元数据操作性能和一致性要求高时，使用此模式。

JindoFS SDK和缓存模式的区别是什么？

JindoFS SDK和缓存模式完全兼容阿里云OSS，通过这两种方式您可以通过OSS产品提供的API和SDK，直接读取写入OSS的文件。

缓存模式需要部署和配置Jindo分布式缓存服务，打开数据缓存开关，而JindoFS SDK则不需要。如果缓存服务出现故障，系统自动切换至JindoFS SDK方式，直接读写OSS文件。

JindoFS缓存模式和Block模式的区别是怎么？

Block模式可以管理文件的元数据，组织数据的块，把OSS作为磁盘来使用，类似HDFS。读写Block模式的数据需要通过JindoFS SDK 客户端。

缓存模式兼容OSS，可以直接读取数据。例如，您通过缓存模式写一个大文件，可以通过OSS Web页面在对应目录下找到这个大文件。如果是块缓存模式时，您只能找到很多文件块，这些块只能通过JindoFS SDK 客户端拼接成大文件。

如果更新了OSS上的数据，如何保证JindoFS缓存数据的一致性？

如果OSS对象被修改、覆盖或删除，JindoFS在读取OSS对象的时候，首先会检查OSS对象的meta信息和，然后对比本地缓存的信息，检查是否发生了变化。如果发生了变化，本次读取放弃本地缓存直接读取OSS，并更新缓存。

JindoFS Block模式的数据可以通过OSS API读取吗？

不可以。只能通过JindoFS SDK客户端读取数据。

对象存储OSS不支持rename操作，那JindoFS支持rename操作吗？

支持。因为JindoFS支持HCFS文件系统接口，所以支持文件和目录的rename操作。

对象存储OSS因为没有文件和目录的概念，所以不支持文件和目录的rename操作，需要通过模拟文件系统的方式来实现rename操作（先拷贝对象至新位置，再删除旧的对象）。

JindoFS的rename性能如何？

JindoFS的rename性能优于社区版本。如果是文件，OSS支持大对象 Fast Copy 优化，JindoFS 利用该优化做到比社区版本快很多；如果是目录，涉及到很多文件，JindoFS通过充分并发优化，也比社区版本快多倍。

JindoFS支持类似于Hadoop S3A的Magic Committer吗？

JindoFS支持无需rename操作的Magic Committer。

JindoFS对百万千万级文件数目录的支持情况如何？

针对百万千万级文件数的大目录，JindoFS支持并发访问和内存优化，不会出现OOM（Out Of Memory）或者挂起。

JindoFS是如何保证数据可靠性的？

因为JindoFS无论使用哪种方式，数据都存放在OSS上，本地磁盘只缓存数据，所以数据可靠性是由OSS来保证的。

JindoFS支持文件和目录操作的一致性吗？

支持。JindoFS Block模式实现HDFS文件系统语义，支持强一致性。

JindoFS支持文件和目录操作的原子性吗？

JindoFS兼容模式不支持原子性。JindoFS兼容模式因为要兼容OSS，受限于OSS对象存储限制，不支持跨对象操作的原子性。例如rename操作，至少涉及到源和目标两个对象，如果是目录的rename，涉及的对象更多。

JindoFS Block模式严格实现HDFS文件系统语义，支持原子性，包括rename操作。

JindoFS Block模式如何保证HA？

JindoFS Block模式基于Raft分布式一致性协议可以部署多个Jindo NamespaceService节点，并且元数据的更新支持异步备份至阿里云Tablestore数据库上。

JindoFS Block模式保存文件数据在集群上，重建集群时数据怎么处理？

JindoFS Block模式的元数据的更新支持异步备份至阿里云Tablestore数据库上，在确保生产集群停止更新，所有修改同步至Tablestore后，停掉JindoFS集群，此时，所有数据在OSS和Tablestore上。重建集群时，恢复OSS和Tablestore上数据至重建集群。

> **说明** 重建集群时，需要考虑版本的兼容性。例如，EMR-2.7.x版本之间都是兼容的，但EMR-2.7.x和EMR-2.6.x之间则不一定。如果是升级到不兼容的大版本时，建议通过Jindo DistCp同步Block模式数据至OSS。

EMR已经支持HDFS，为什么还要有JindoFS Block模式？

JindoFS Block模式从技术架构和功能上确实和HDFS相似，都是自定义管理文件元数据并组织数据，具有强一致性。

JindoFS Block模式的优势在于，数据备份至OSS上，支持弹性扩展、低成本且无需维护磁盘。

JindoFS和Alluxio相比有什么技术差异和优势？

对比项	JindoFS	Alluxio
相同点	JindoFS缓存模式在技术架构上与Alluxio类似，都提供对OSS的缓存加速能力，支持Master + Workers形式，Master维护缓存块的位置信息，Workers提供缓存块的管理和读写能力。	
	JindoFS不需要挂载，可以直接访问*oss://*路径，只需打开数据缓存开关即可。	Alluxio需要先挂载OSS Bucket位置至名字空间，再使用*alluxio://*访问

对比项 不同点	JindoFS	Alluxio
	JindoFS核心支持的是OSS，性能极致优化。	Alluxio支持数据源很多，可以同时挂载到统一的名字空间。
	JindoFS提供基础的SDK模式支持访问适配OSS，全面对接各种开源引擎。	无

跟HDFS相比，使用JindoFS和OSS能节省成本吗？

HDFS存储时，不能弹性伸缩，预算不足就会面临存储空间不足，或者相存在空间浪费的情况。

阿里云OSS是海量对象存储，支持弹性伸缩，具有归档存储功能，可以备份冷数据。JindoFS基于OSS，支持数据冷热分层和数据归档存储策略，使用得当，整体上可以降低成本。

Hadoop社区版本也提供OSS支持，JindoFS有什么优势？

Hadoop社区版本支持的OSS，受到社区整体约束，只具备OSS基本适配功能。

JindoFS对OSS的支持优势如下：

- 更全面：对接各种开源引擎。
- 更活跃：对OSS最新功能提供同步更新和升级。
- 更高级：具有高阶缓存加速能力和高级定制功能Block模式。
- 更快：性能更优。JindoFS核心代码采用C++ native代码开发，各种基本操作性能优于社区版本。

FindoFS提供Fuse支持吗？和OSS自带的Fuse有什么优势？

提供。FindoFS提供的Fuse优势在于能够利用JindoFS分布式缓存和Block模式功能。

JindoFS支持哪些开源组件？

支持按照HCFS接口读写数据的组件，例如，Hadoop、Hive、Spark、Flink、Presto、HBase、Impala、Druid、Kafka和Flume。

JindoFS吞吐如何？会不会影响Spark或Hive大规模分析计算？

JindoFS在适配上充分发挥OSS并发吞吐能力，实现异步并发读取，利用Concurrent Multipart Upload特性进行并发分块写入，在读写吞吐上面比社区版本具有较大优势。

JindoFS缓存模式和Block模式可以利用集群本地磁盘或内存来缓存数据，对于新写入的数据和重复读取的数据具有显著加速效果。在同样集群条件下，对于Spark或Hive分析计算，跟HDFS相比集群吞吐是相当的，甚至优于HDFS。

JindoFS写性能如何？Flume或Kafka在写入数据时碰到瓶颈如何处理？

因为HDFS需要写三备份才算写入成功，JindoFS只需写入OSS一备份就算写入成功，所以通常情况下，JindoFS写性能优于HDFS。

如果集群的Flume或Kafka在写入数据时碰到瓶颈，请提交工单处理。

JindoFS支持Flink实时计算场景吗？

支持。JindoFS支持Flink读OSS source，checkpoint和sink到OSS以及Exactly-Once语义。

JindoFS和OSS场景下，可以使用Presto做交互式分析吗？

可以。JindoFS缓存模式和Block模式都支持Presto交互式分析，且性能稳定。

使用Impala时，可以通过JindoFS查询OSS上的数据吗？

可以。Impala 3.4及后续版本支持JindoFS，可以读写OSS。

JindoFS支持使用Delta Lake，或者Hudi和IceBerg时，存放数据在OSS上吗？

支持。

数据存放在OSS上，JindoFS支持机器学习训练吗？

支持。您可以使用JindoFS缓存模式，通过预加载将OSS数据提前写入内存或者SSD做缓存，然后训练引擎可以通过JindoFuse支持直接读取。

基于MaxCompute数仓上的数据，JindoFS如何帮助机器学习训练？

有如下两种方式：

- MaxCompute数仓作业将数据通过MaxCompute外表方式写入至OSS，然后在训练集群通过JindoFS缓存模式和JindoFuse来加载训练。
- 通过JindoTable从MaxCompute拉取数据写入至JindoFS缓存模式，然后使用JindoFuse来加载训练。

基于Hive数仓上的数据，JindoFS如何帮助机器学习训练？

类似于MaxCompute数仓上的数据处理方式，方式详情请参见基于MaxCompute数仓上的数据，JindoFS如何帮助机器学习训练？。

如果使用JindoFS，如何迁移HDFS上的数据？

您可以使用Jindo DistCp同步HDFS数据至JindoFS或OSS。Jindo DistCp比Hadoop DistCp性能高，且支持OSS归档。

JindoFS在新版本才有，如果需要在EMR集群上使用JindoFS，该如何处理？

如果集群规模不大，建议重建集群来使用JindoFS和EMR新版本。如果规模较大，请提交工单处理。

JindoFS支持哪些Hadoop版本和发行厂商？

JindoFS SDK提供OSS适配功能，明确支持Hadoop 2.7后续版本和Hadoop 3.x版本。

Hortonworks版本（Hortonworks Data Platform，简称HDP）和Cloudera版本（Cloudera's Distribution Including Apache Hadoop，简称CDH）都可以使用，但可能会存在冲突，需要修改配置 `fs.oss.impl = JindoOssFileSystem` 。

JindoFS可以在ECS自建集群上使用吗？

可以。需要您下载JindoFS SDK手工部署即可。如果您需要使用JindoFS缓存模式和Block模式功能，建议您登录阿里云E-MapReduce控制台，直接使用E-MapReduce产品。

JindoFS可以在阿里云ACK环境上使用吗？

可以。

使用JindoFS会被阿里云E-MapReduce绑定吗？

不会。JindoFS遵循标准HCFS接口，兼容和支持全面开源生态，不会绑定。

JindoFS可以在IDC机房的Hadoop集群使用吗？

可以。您可以直接下载开源JindoFS SDK按照文档部署使用即可。如果集群出现兼容性问题，请提交工单处理。

如何查看JindoFS上的数据量？

您可以直接使用如下命令查看统计情况。

```
hadoop dfs -du/count
```

JindoFS查看的数据量和OSS产品控制台上看到的数据量不一致时如何处理？

请提交工单处理。

什么情况下建议打开OSS Bucket的多版本控制？

对于重要的数据建议打开OSS Bucket多版本，防止误删时数据不丢失。

打开OSS Bucket多版本控制对EMR和JindoFS的影响是什么？

对于Hive或Spark中间结果存放的数据以及频繁修改的数据，不建议使用多版本Bucket，会影响计算性能。

OSS归档存储可以大量节省存储成本，JindoFS提供相应的支持吗？

JindoFS支持相应的OSS归档存储。Block模式上，提供专门的存储策略与OSS归档匹配。

使用JindoFS，会泄露AccessKey吗？

JindoFS支持在集群上配置使用AccessKey，但存在泄漏Accesskey的风险。在EMR集群里或者在ECS环境，如果节点绑定了ECS role，您可以使用权限管理，不使用AccessKey。

什么是AccessKey免密？

EMR集群提供AccessKey免密，该功能通过EMR管控得到用户授权，方便用户拿到具有权限的阿里云STS (Security Token Service)，然后使用该 Token访问阿里云资源，例如OSS。

如果支持AccessKey免密，那如何区分不同的用户和权限限制？

AccessKey免密不是适用于所有的场景。

如果有多个用户需要区分权限，有如下两种方式：

- 您可以通过阿里云RAM子账号权限控制，每个用户使用RAM子账号来访问OSS。
- 您可以使用JindoFS权限控制，通过Ranger来授权。

> 注意 JindoFS仅能在Namespace上设定权限控制。

如何使用不同的AccessKey，通过FindoFS访问不同的OSS Bucket？

您可以使用JindoFS multi namespace，每个Namespace配置不同的OSS bucket和对应的AccessKey信息。

在无EMR管控支持情况下，想使用自建的IDC集群，又不想在集群节点上配置AccessKey，该如何处理？

您可以使用Hadoop Crendential Provider机制，详情请参见Credential Provider使用说明

JindoFS支持Auditlog吗？

支持。JindoFS支持Multi Namespaces，每个Namespace上可以设定Auditlog，默认不打开。

JindoFS支持Ranger集成吗？

支持。JindoFS支持Multi Namespaces，每个Namespace上可以设定支持Ranger，默认不打开。

EMR中的SmartData和JindoFS是什么关系？

SmartData是产品组件，该组件包括JindoFS服务。

EMR中的Bigboot和JindoFS是什么关系？

Bigboot是SmartData组件的基础设施，对该组件所包含服务提供毫秒级进程监测和日志清理等功能。