Alibaba Cloud E-MapReduce

Developer Guide

Issue: 20190806

Legal disclaimer

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

1. **You shall download and obtain this document from the Alibaba Cloud website or other Alibaba Cloud-authorized channels, and use this document for your own legal business activities only. The content of this document is considered confidential information of Alibaba Cloud. You shall strictly abide by the confidentiality obligations. No part of this document shall be disclosed or provided to any third party for use without the prior written consent of Alibaba Cloud.**
2. **No part of this document shall be excerpted, translated, reproduced, transmitted, or disseminated by any organization, company, or individual in any form or by any means without the prior written consent of Alibaba Cloud.**
3. **The content of this document may be changed due to product version upgrades , adjustments, or other reasons. Alibaba Cloud reserves the right to modify the content of this document without notice and the updated versions of this document will be occasionally released through Alibaba Cloud-authorized channels. You shall pay attention to the version changes of this document as they occur and download and obtain the most up-to-date version of this document from Alibaba Cloud-authorized channels.**
4. **This document serves only as a reference guide for your use of Alibaba Cloud products and services. Alibaba Cloud provides the document in the context that Alibaba Cloud products and services are provided on an "as is", "with all faults " and "as available" basis. Alibaba Cloud makes every effort to provide relevant operational guidance based on existing technologies. However, Alibaba Cloud hereby makes a clear statement that it in no way guarantees the accuracy, integrity , applicability, and reliability of the content of this document, either explicitly or implicitly. Alibaba Cloud shall not bear any liability for any errors or financial losses incurred by any organizations, companies, or individuals arising from their download, use, or trust in this document. Alibaba Cloud shall not, under any circumstances, bear responsibility for any indirect, consequential, exemplary, incidental, special, or punitive damages, including lost profits arising from the use**

or trust in this document, even if Alibaba Cloud has been notified of the possibility of such a loss.

5. By law, all the content of the Alibaba Cloud website, including but not limited to works, products, images, archives, information, materials, website architecture, website graphic layout, and webpage design, are intellectual property of Alibaba Cloud and/or its affiliates. This intellectual property includes, but is not limited to, trademark rights, patent rights, copyrights, and trade secrets. No part of the Alibaba Cloud website, product programs, or content shall be used, modified , reproduced, publicly transmitted, changed, disseminated, distributed, or published without the prior written consent of Alibaba Cloud and/or its affiliates . The names owned by Alibaba Cloud shall not be used, published, or reproduced for marketing, advertising, promotion, or other purposes without the prior written consent of Alibaba Cloud. The names owned by Alibaba Cloud include, but are not limited to, "Alibaba Cloud", "Aliyun", "HiChina", and other brands of Alibaba Cloud and/or its affiliates, which appear separately or in combination, as well as the auxiliary signs and patterns of the preceding brands, or anything similar to the company names, trade names, trademarks, product or service names, domain names, patterns, logos, marks, signs, or special descriptions that third parties identify as Alibaba Cloud and/or its affiliates).
6. Please contact Alibaba Cloud directly if you discover any errors in this document.

Generic conventions

Table -1: Style conventions

Style	Description	Example
	This warning information indicates a situation that will cause major system changes, faults, physical injuries, and other adverse results.	 Danger: Resetting will result in the loss of user configuration data.
	This warning information indicates a situation that may cause major system changes, faults, physical injuries, and other adverse results.	 Warning: Restarting will cause business interruption. About 10 minutes are required to restore business.
	This indicates warning information, supplementary instructions, and other content that the user must understand.	 Notice: Take the necessary precautions to save exported data containing sensitive information.
	This indicates supplemental instructions, best practices, tips, and other content that is good to know for the user.	 Note: You can use Ctrl + A to select all files.
>	Multi-level menu cascade.	Settings > Network > Set network type
Bold	It is used for buttons, menus , page names, and other UI elements.	Click OK.
`Courier font`	It is used for commands.	Run the `cd /d C:/windows` command to enter the Windows system folder.
Italics	It is used for parameters and variables.	`bae log list --instanceid` *Instance_ID*
[] or [a|b]	It indicates that it is a optional value, and only one item can be selected.	`ipconfig` *[-all|-t]*

Style	Description	Example
{} or {a|b}	It indicates that it is a required value, and only one item can be selected.	`swich {stand | slave}`

Contents

Legal disclaimer..... I
Generic conventions..... I
1 Use Python on E-MapReduce..... 1
2 Preparations.....3
2.1 Development preparations.....3
2.2 Configure the OSS URI to use E-MapReduce.....3
2.3 Use sample projects.....4
3 Spark.....16
3.1 Preparations..... 16
3.2 Parameter description.....18
3.3 Operate OSS files.....22
3.4 Use Spark to access OSS..... 23
3.5 Use MaxCompute in Spark..... 23
3.6 Use Spark Streaming to consume MQ data..... 25
3.7 Consume Table Store data in Spark..... 26
3.8 Use Spark Streaming to consume MNS data..... 28
3.9 Use Spark to write data to HBase.....29
3.10 Use Spark Streaming to process Kafka data..... 30
3.11 Use Spark and write data to MySQL..... 31
3.12 Configure Spark-Submit parameters..... 32
4 Hadoop.....40
4.1 Parameter description.....40
4.2 Create and run MapReduce jobs..... 41
4.3 Create and run a Hive job..... 50
4.4 Create and run a Pig job..... 54
4.5 Create a Hadoop streaming job..... 58
4.6 Process Table Store data in Hive.....59
4.7 Process Table Store data in MR..... 61
5 Create an HBase cluster and use the HBase storage service...65
6 Back up HBase.....70
7 Guide for E-MapReduce cluster operations..... 72

1 Use Python on E-MapReduce

You can use Python on E-MapReduce (EMR) 2.0.0 or later. This topic describes how to install Python.

Python 2.7

You can use Python 2.7 on EMR 2.0.0 or later.

By default, Python is installed in the `/usr/local/Python-2.7.11/` directory and the NumPy package is included.

Python 3.6

You can use Python 3.6.4 on EMR 2.10.0 or later and 3.10.0 or later. By default, Python 3.6.4 is installed in the `/usr/bin/python3.6` directory. You can use the following command to check whether Python 3 is installed on your instance.

```
[root@emr-header-1 ~]# python36
```

The pip3 tool is not preinstalled by default. You can install the tool as required.

By default, you cannot use Python 3 on versions earlier than EMR 2.10.0 or EMR 3.10.0. You must download and install Python 3 as follows:

1. Download the installation package of Python 3 from this link.
2. Extract the downloaded file and install Python 3

```
tar zxvf Python-3.6.4.tgz
cd Python-3.6.4 ./configure --prefix=/usr/local/Python-3.6.4
make && make install
ln -s /usr/local/Python-3.6.4/bin/python3.6 /bin/python3
ln -s /usr/local/Python-3.6.4/bin/pip3 /bin/pip3
```

3. Verify the installation result of Python 3

```
[root@emr-header-1 bin]# python3
```

```
Python 3.6.4 (default, Mar 12 2018, 14:03:26)
[GCC 4.8.5 20150623 (Red Hat 4.8.5-16)] on linuxType
"help", "copyright", "credits" or "license" for more informatio n.
```

```
[ root @ emr - header - 1   bin ]#  pip3  - V
```

```
pip   9 . 0 . 1   from  / usr / local / Python - 3 . 6 . 4 / lib /
python3 . 6 / site - packages  ( python   3 . 6 )
```

The preceding result indicates a successful installation.

2 Preparations

2.1 Development preparations

This topic describes the preparations required for E-MapReduce development.

- You have activated the Alibaba Cloud service and created an AccessKey ID and AccessKey Secret. ID represents the AccessKey ID and KEY represents the AccessKey Secret. If you have not activated or do not know about the OSS service, log on to the OSS product page for more help.
- You have a basic understanding of Spark, Hadoop, Hive, and Pig. This topic does not describe the development practices of Spark, Hadoop, Hive, and Pig. For more information, see the Apache website.
- You have a basic understanding of Alibaba Cloud Elastic MapReduce (E-MapReduce) development components. The open source community is welcome to contribute to the project. For example, we appreciate your problem feedback, bug fixes, and new components.

2.2 Configure the OSS URI to use E-MapReduce

This topic describes how to configure the OSS URI to use E-MapReduce.

OSS URI

When using E-MapReduce, you can use two types of OSS URIs:

- native URI: oss://[accessKeyId:accessKeySecret@]bucket[.endpoint]/object/path

  This URI is used for specifying input and output data sources for a job, which is similar to hdfs://. When you operate OSS data, you can configure the accessKey ID, accessKey Secret, and the endpoint. Alternatively, you can specify the accessKey ID , accessKey Secret, and the endpoint in the URI.
- ref URI: ossref://bucket/object/path

  It is only valid in the configuration of an E-MapReduce job and is used to specify the resources needed for running the job.

We call prefixes, such as oss and ossref, as schemes. Pay special attention to the scheme difference in URI.

Notice:

Currently, operations only support OSS for standard storage types.

- E-MapReduce uses the multipart mode to upload large files to OSS. Note that if your job is interrupted, some of the result data remains in OSS. You must remove it manually. The procedure here is the same with the use of HDFS. One difference , however, is that E-MapReduce uses the multipart mode to upload large files. The file fragments are uploaded to the OSS fragment management. Therefore, you must delete the remaining job files in OSS file management and clean the file fragments in OSS fragment management. Otherwise, you are charged for data storage.
- Except for the preceding manual cleanup, you can also configure the lifecycle of the fragments, so that the expired fragments are automatically cleaned. For more information, see Lifecycle management of OSS files.

2.3 Use sample projects

This sample project is a complete, compilable, and runnable project, including the sample code of MapReduce, Pig, Hive, and Spark.

Sample project

See the open source project. The details are as follows:

- MapReduce

  WordCount: Counts the number of words.
- Hive

  sample.hive: A simple search for the tables.
- Pig

  sample.pig: OSS data instances processed by Pig.

- Spark
  - SparkPi: Calculates Pi.
  - SparkWordCount: Counts the number of words.
  - LinearRegression: Linear regression.
  - OSSSample: OSS sample.
  - ODPSSample: ODPS sample.
  - MNSSample: MNS sample.
  - LoghubSample: Loghub sample.

Dependencies

- Test data (under the data directory)
  - The_Sorrows_of_Young_Werther.txt: Used as the input data of WordCount (MapReduce/Spark).
  - patterns.txt: Filter characters of the WordCount (MapReduce) job.
  - u.data: The data of the test table of the sample.hive script.
  - abalone: Test data for linear regression algorithms.
- The JAR dependencies (under the lib directory)

  tutorial.jar: The JAR dependencies required for the sample.pig job.

Preparations

This project provides some test data. You can simply upload it to OSS to use it. For other samples, such as MaxCompute, MNS, ONS, and Log Service, you need to prepare the data as follows:

- (Optional) Enable Log Service. See the User Guide for Log Service.
- (Optional) Create MaxCompute projects and tables. See Create a MaxCompute project and Quick Start for MaxCompute.
- (Optional) Use ONS. See Quick start for primary accounts.
- (Optional) Use MNS. See topics for using the Message Service console.

Concepts

- OSSURI: oss://accessKeyId:accessKeySecret@bucket.endpoint/a/b/c.txt. Specify the input and output data sources. OSSURI is similar to URLs like hdfs://.
- The combination of AccessKey ID and AccessKey Secret is the key for you to access an Alibaba Cloud API. Click here to obtain it.

Run jobs in clusters

- Spark
  - SparkWordCount:

```
spark - submit  -- class   SparkWordC  ount   examples - 1 . 0 -
SNAPSHOT - shaded . jar  < inputPath >
               < outputPath > < numPartiti  on >
```

Parameters are described as follows:

■ inputPath: Data input path.

■ outputPath: Data output path.

■ numPartition: The number of RDD partitions of the input data.

  - SparkPi: `spark - submit  -- class   SparkPi   examples - 1 . 0 - SNAPSHOT - shaded . jar`
  - OSSSample:

```
spark - submit  -- class   OSSSample   examples - 1 . 0 -
SNAPSHOT - shaded . jar  < inputPath >
               < numPartiti  on >
```

Parameters are described as follows:

■ inputPath: Data input path.

■ numPartition: Input the number of data RDD partitions.

  - ONSSample:

```
spark - submit  -- class   ONSSample   examples - 1 . 0 -
SNAPSHOT - shaded . jar  < accessKeyI  d >
               < accessKeyS  ecret > < consumerId > < topic > <
subExpress  ion > < parallelis  m >
```

Parameters are described as follows:

■ accessKeyId: Alibaba Cloud AccessKey ID.

■ accessKeySecret: Alibaba Cloud AccessKey Secret.

■ topic: Each message queue has a topic.

■ subExpression: See Message filtering.

■ parallelism: Specifies the number of receivers to consume messages in the queue.

  - MaxComputeSample:

```
spark - submit  -- class   MaxCompute  Sample   examples - 1 . 0
- SNAPSHOT - shaded . jar  < accessKeyI  d >
```

```
                < accessKeyS  ecret > < envType > < project > <
table > < numPartiti  ons >
```

Parameters are described as follows:

- accessKeyId: Alibaba Cloud AccessKey ID.
- accessKeySecret: Alibaba Cloud AccessKey Secret.
- envType: 0 indicates the public network, and 1 indicates the private network. Select 0 for local debugging, and select 1 for execution on E-MapReduce.
- project: see MaxCompute Quick Start.
- numPartition: The number of RDD partitions of the input data.

- MNSSample:

```
spark - submit  -- class   MNSSample   examples - 1 . 0 -
SNAPSHOT - shaded . jar  < queueName >
               < accessKeyI  d > < accessKeyS  ecret > < endpoint
>
```

Parameters are described as follows:

- queueName: Queue name. See MNS glossary.
- accessKeyId: Alibaba Cloud AccessKey ID.
- accessKeySecret: Alibaba Cloud AccessKey Secret.
- endpoint: Address for accessing the queue data.

- LoghubSample:

```
spark - submit  -- class   LoghubSamp  le   examples - 1 . 0 -
SNAPSHOT - shaded . jar  < sls   project > < sls
               logstore > < loghub   group   name > < sls
endpoint > < access   key   id > < access   key   secret > <
batch
               interval   seconds >
```

Parameters are described as follows:

- sls project: The name of the Log Service project.
- sls logstore: Logstore name.
- loghub group name: The name of the group that consumes Logstore data in jobs. You can specify a name as needed. When the values of sls project and sls store are the same, jobs with the same group name consume data in sls store

collaboratively. Jobs with different group names consume data in sls store separately.

- sls endpoint: See Service endpoint.
- accessKeyId: Alibaba Cloud AccessKey ID.
- accessKeySecret: Alibaba Cloud AccessKey Secret.
- batch interval seconds: The interval between Spark Streaming jobs (unit: seconds).

- LinearRegression:

```
spark - submit  -- class   LinearRegr  ession   examples - 1 . 0
- SNAPSHOT - shaded . jar  < inputPath >
               < numPartiti  ons >
```

Parameters are described as follows:

- inputPath: Data input path.
- numPartition: The number of RDD partitions of the input data.

· MapReduce

- WordCount:

```
hadoop   jar   examples - 1 . 0 - SNAPSHOT - shaded . jar
WordCount
               - Dwordcount . case . sensitive = true  <
inputPath > < outputPath > - skip  < patternPat  h >
```

Parameters are described as follows:

- inputPath: Data input path.
- outputPath: Data output path.
- patternPath: The file that contains characters to be filtered. You can use data/patterns.txt.

· Hive

- `hive  - f   sample . hive  - hiveconf   inputPath =< inputPath >`

Parameters are described as follows:

- inputPath: Data input path.

· Pig

- ```
  pig  - x   mapreduce  - f   sample . pig  - param   tutorial =<
  tutorialJa  rPath > - param
  ```
  ```

```
                    input =< inputPath > - param   result =<
resultPath >
```

Parameters are described as follows:

- tutorialJarPath: The JAR dependencies. You can use lib/tutorial.jar.
- inputPath: Data input path.
- resultPath: Data output path.

Notice:

- If you are working on E-MapReduce, upload the testing data and the JAR dependencies to OSS. The path rule should follow the definition of OSSURI, as described above.
- If used in a cluster, it can be stored locally.

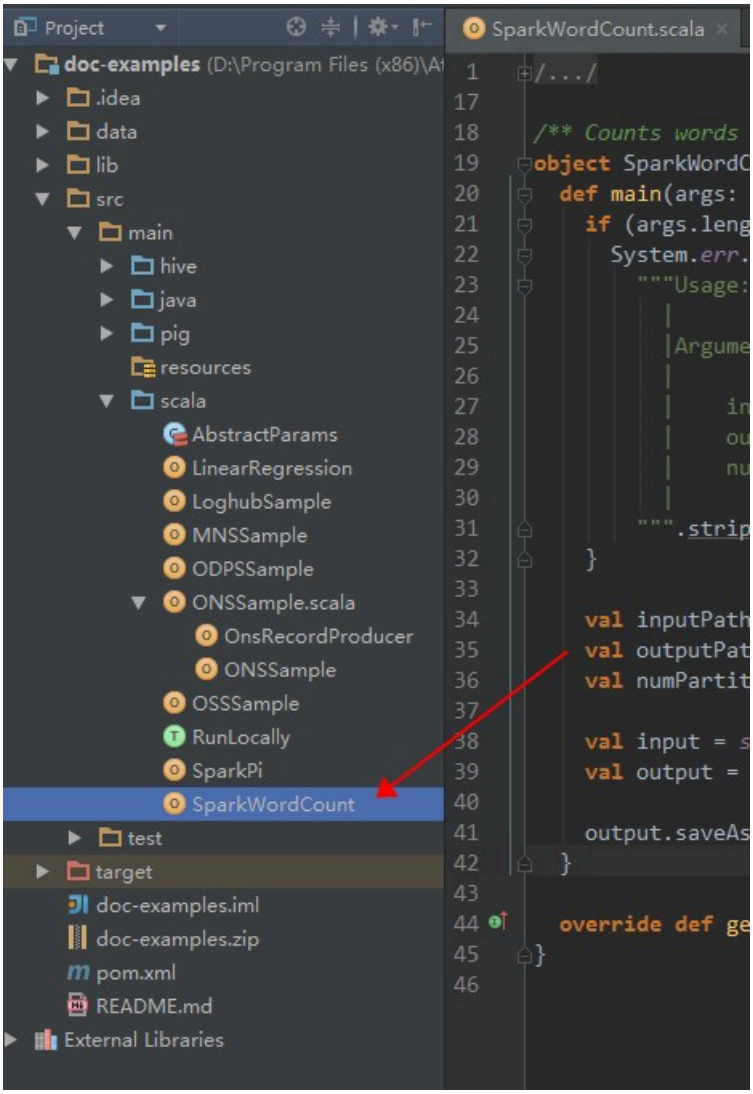
Run Locally

Here we describe how to run a Spark program locally to visit Alibaba Cloud's data sources, such as OSS. If you want to debug and run the program locally, we recommend that you use some development tools, such as IntelliJ IDEA or Eclipse, especially when using Windows. Otherwise, you need to configure Hadoop and Spark runtime environments on Windows machines.

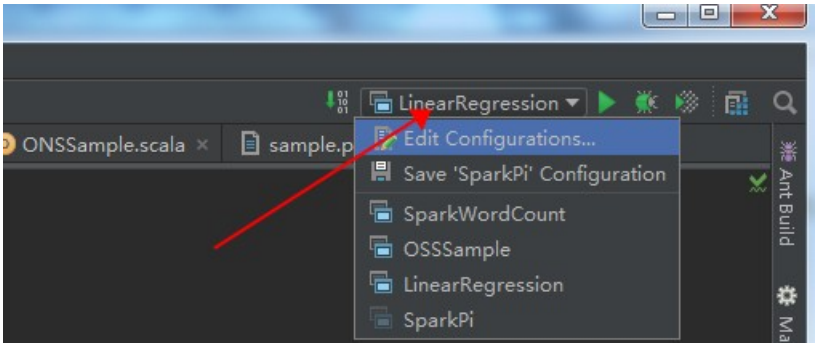
- IntelliJ IDEA
  - Preparations

    Install IntelliJ IDEA, Maven, Maven plugin for IntelliJ IDEA, Scala and Scala plugin for IntelliJ IDEA.
  - Development process

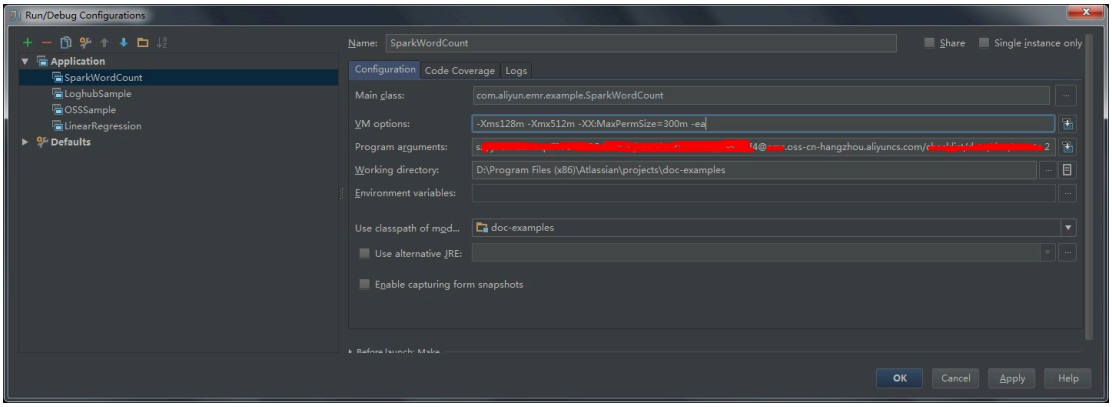
    1. Double-click to enter SparkWordCount.scala.



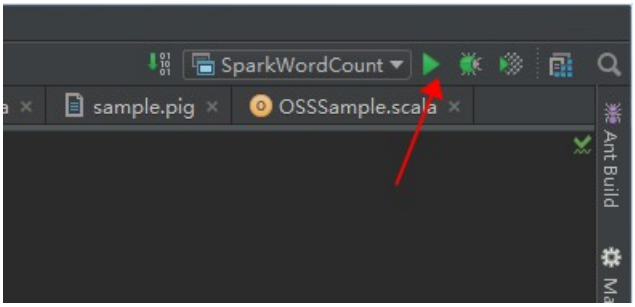
    2. Enter the job configuration page from the arrow as shown in the following figure.



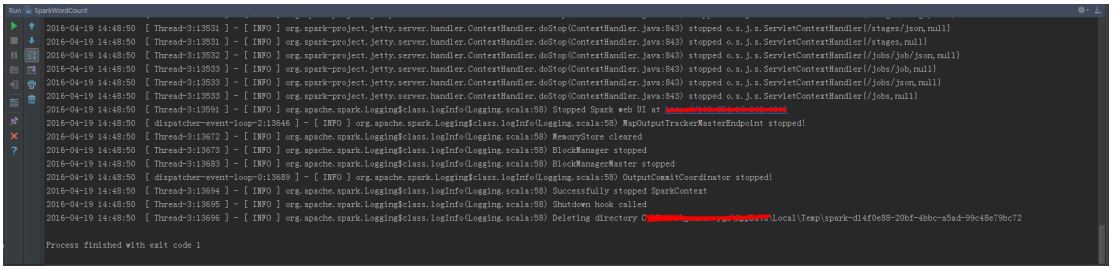
3. Select SparkWordCount and enter the required job parameters in the job parameter box.



4. Click OK.
5. Click Run to run the job.



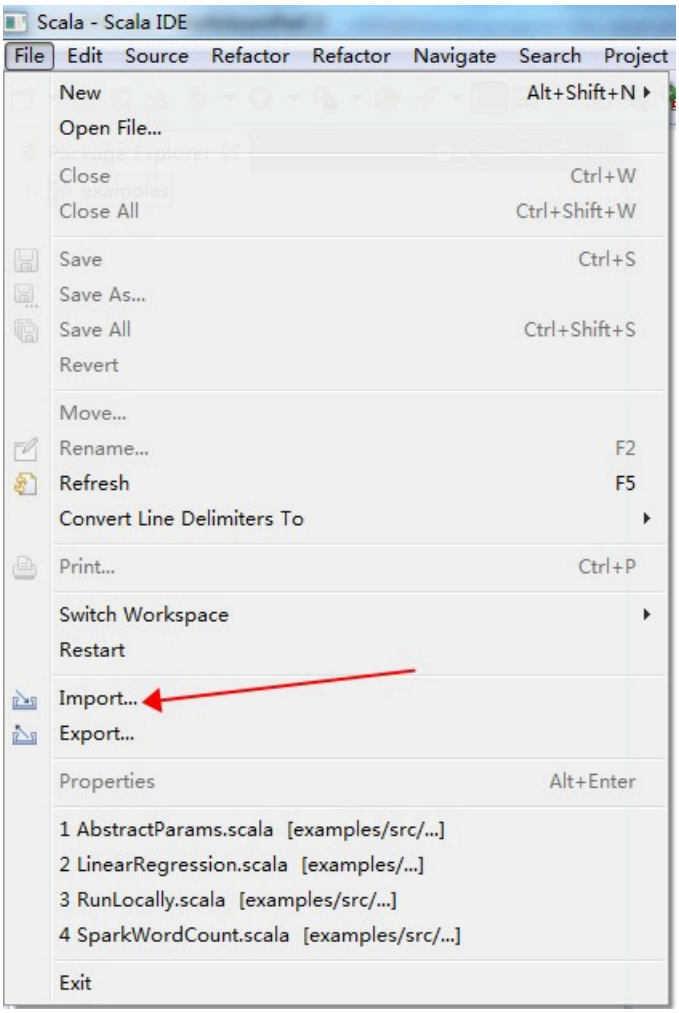
6. View job logs

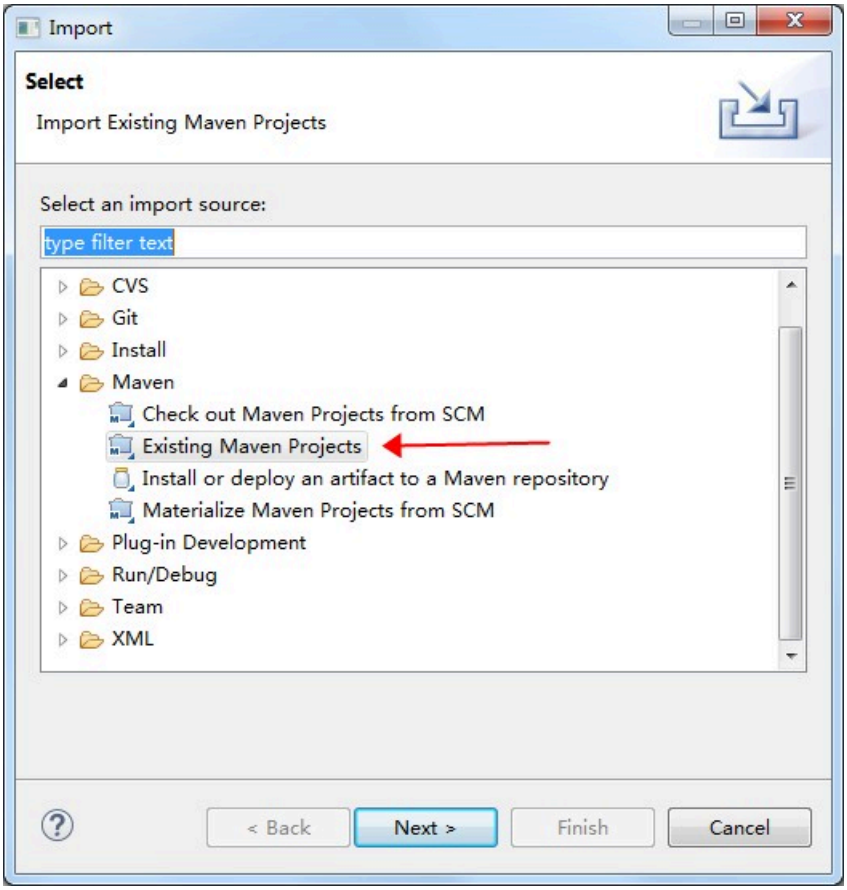


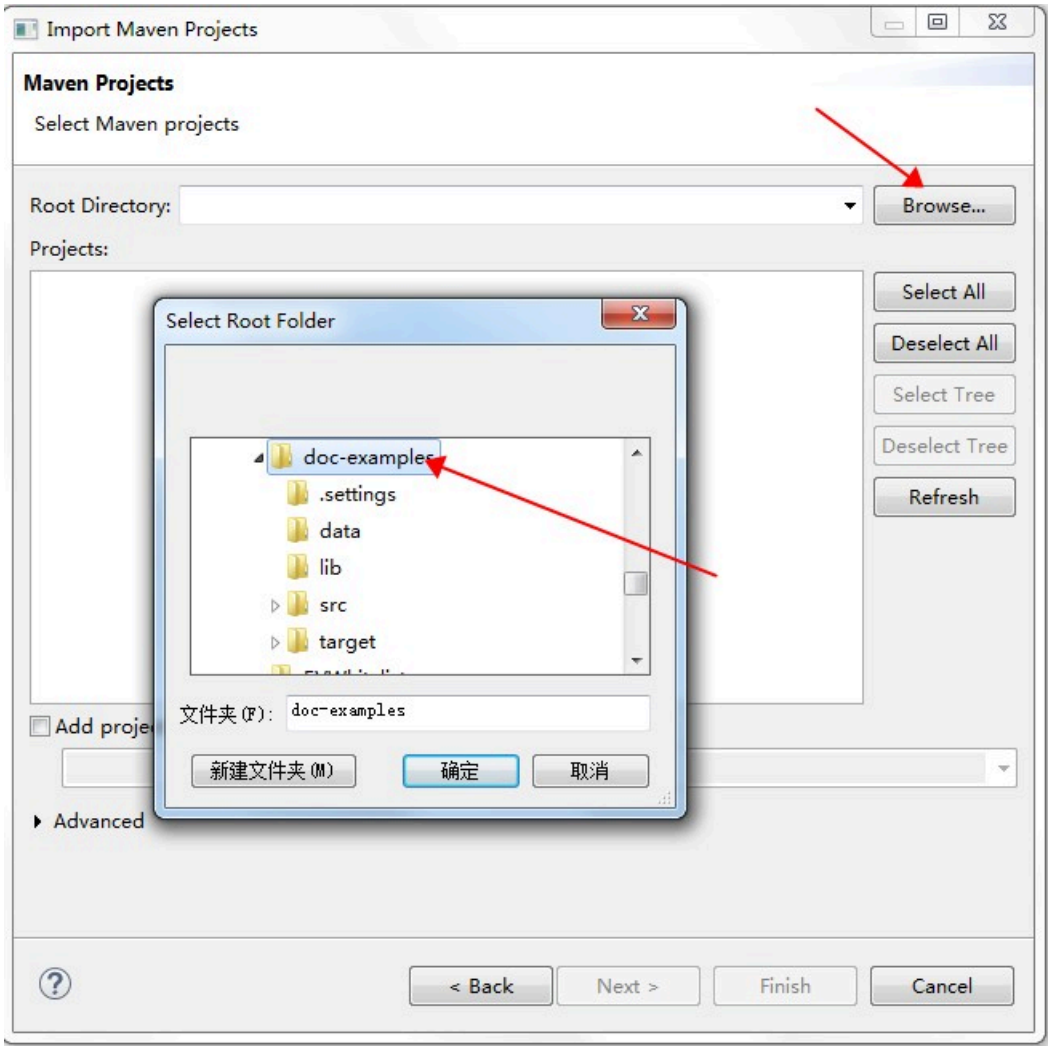
- Scala IDE for Eclipse
  - Preparations

    Install the Scala IDE for Eclipse, Maven and the Maven plugin for Eclipse.
  - Development process

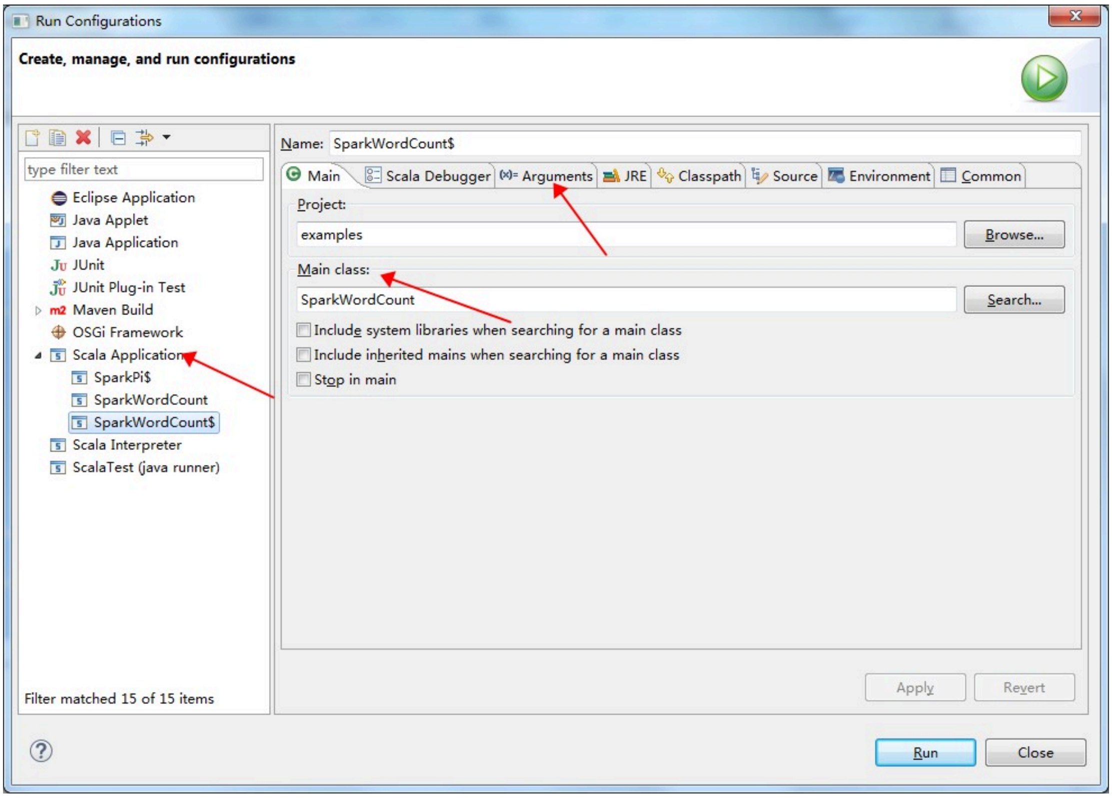
    1. Import a project as described in the figure below:



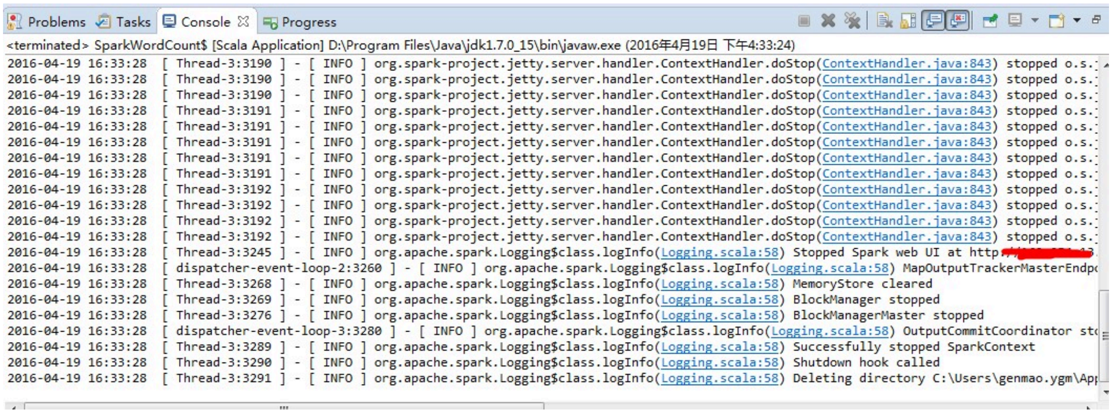

Import
Select
Import Existing Maven Projects
Select an import source:
type filter text
CVS
Git
Install
Maven
Check out Maven Projects from SCM
Existing Maven Projects
Install or deploy an artifact to a Maven repository
Materialize Maven Projects from SCM
Plug-in Development
Run/Debug
Team
XML
< Back
Next >
Finish
Cancel



2. The shortcut for Run as Maven build is Alt + Shift + X, M. You can also right-click the project name and choose Run As > Maven build.
3. Right-click on the job to run after it has been compiled, select Run Configuration to enter the configuration page.
4. In the configuration page, select Scala Application and configure the Main Class and parameters of the job, As shown in the following figure:



5. Click Run.
6. View the output log of the console, as shown in the following figure:



3 Spark

3.1 Preparations

This topic describes how to prepare for Spark development.

Install E-MapReduce SDK

You can install E-MapReduce SDK using either of the following methods:

- Method 1: Directly use the JAR package in Eclipse. Follow these steps:
  1. Download the dependencies required by E-MapReduce from the Maven repository.
  2. Copy the required JAR files to your project directory. (Choose SDK version 2.10 or 2.11 based on the Spark version you use in the cluster in which your jobs run. Version 2.10 is recommended for Spark1.x, and 2.11 for Spark2.x.)
  3. Right-click the project name in Eclipse, and then choose Properties > Java Build Path > Add JARs.
  4. Select the downloaded SDK.
  5. After you have completed the preceding steps, you can read and write data in OSS, Log Service, MNS, ONS, Table Store, and MaxCompute.
- Method 2: Edit the Maven project to add the dependencies as follows:

```
<! -- Support for OSS data sources -->
    < dependency >
        < groupId > com . aliyun . emr </ groupId >
        < artifactId > emr - core </ artifactId >
        < version > 1 . 4 . 1 </ version >
    </ dependency >
    <! -- Support for Table Store data sources -->
    < dependency >
        < groupId > com . aliyun . emr </ groupId >
        < artifactId > emr - tablestore </ artifactId >
        < version > 1 . 4 . 1 </ version >
    </ dependency >
    <! -- Support for Message Service ( MNS ), Open
 Notificati on Service ( ONS ), Log Service , MaxCompute
 data sources ( Spark 1 . x environmen t ) -->
    < dependency >
        < groupId > com . aliyun . emr </ groupId >
        < artifactId > emr - mns_2 . 10 </ artifactId >
        < version > 1 . 4 . 1 </ version >
    </ dependency >
    < dependency >
        < groupId > com . aliyun . emr </ groupId >
        < artifactId > emr - logservice _2 . 10 </ artifactId >
```

```
        < version > 1 . 4 . 1 </ version >
    </ dependency >
    < dependency >
        < groupId > com . aliyun . emr </ groupId >
        < artifactId > emr - maxcompute  _2 . 10 </ artifactId >
        < version > 1 . 4 . 1 </ version >
    </ dependency >
    < dependency >
        < groupId > com . aliyun . emr </ groupId >
        < artifactId > emr - ons_2 . 10 </ artifactId >
        < version > 1 . 4 . 1 </ version >
    </ dependency >
    <! --  Support   for   MNS ,  ONS ,  Log   Service ,  MaxCompute
   data   sources  ( spark   2 . x   environmen  t ) -->
    < dependency >
        < groupId > com . aliyun . emr </ groupId >
        < artifactId > emr - mns_2 . 11 </ artifactId >
        < version > 1 . 4 . 1 </ version >
    </ dependency >
    < dependency >
        < groupId > com . aliyun . emr </ groupId >
        < artifactId > emr - logservice  _2 . 11 </ artifactId >
        < version > 1 . 4 . 1 </ version >
    </ dependency >
    < dependency >
        < groupId > com . aliyun . emr </ groupId >
        < artifactId > emr - maxcompute  _2 . 11 </ artifactId >
        < version > 1 . 4 . 1 </ version >
    </ dependency >
    < dependency >
        < groupId > com . aliyun . emr </ groupId >
        < artifactId > emr - ons_2 . 11 </ artifactId >
        < version > 1 . 4 . 1 </ version >
    </ dependency >
```

Debug Spark code locally

The *spark.hadoop.mapreduce.job.run-local* property is for scenarios where you need to debug for Spark reading and writing OSS. Use the default settings in other scenarios.

If you need to debug for Spark reading and writing OSS, configure the SparkConf object by setting *spark.hadoop.mapreduce.job.run-local* to `true` . See the following example code:

```
val   conf  =  new   SparkConf (). setAppName ( getAppName ).
setMaster (" local [ 4 ]")
   conf . set (" spark . hadoop . fs . oss . impl ", " com . aliyun .
fs . oss . nat . NativeOssF  ileSystem ")
   conf . set (" spark . hadoop . mapreduce . job . run - local ", "
true ")
   val   sc  =  new   SparkConte  xt ( conf )
   val   data  =  sc . textFile (" oss ://...")
```

```
println ( s " count : ${ data . count ()}")
```

Third-party dependency description

To allow E-MapReduce to access Alibaba Cloud data sources (such as OSS and MaxCompute), you need to install the required third-party dependencies.

See the pom file to add or delete required third-party dependencies.

OSS output directory settings

Configure the parameter `fs . oss . buffer . dirs` to the local hadoop configuration file. The parameter value is a local directory path. If the parameter value is not set, a null pointer exception will occur when OSS data is written during the process of running Spark.

Clean up data

When a Spark job fails, the data generated is not deleted automatically. If a job fails , check the OSS output directory to see if any file exists. You must also check OSS fragment management for any fragments that have not been submitted. Remove fragments that are found.

Use PySpark

If you use Python to create a Spark job, see the instructions on configuring the environment.

3.2 Parameter description

The description of the parameters in Spark code

The following parameters can be configured in Spark code:

Property	Default	Description
spark.hadoop.fs.oss.accessKeyId	None.	(Optional) The AccessKey ID required for accessing OSS.
spark.hadoop.fs.oss.accessKeySecret	None.	(Optional) The AccessKey Secret required for accessing OSS.
spark.hadoop.fs.oss.securityToken	None.	(Optional) The STS token required for accessing OSS .

Property	Default	Description
spark.hadoop.fs.oss.endpoint	None.	(Optional) The endpoint used for accessing OSS.
spark.hadoop.fs.oss.multipart.thread.number	5	The number of threads (concurrency) used by OSS for the upload part - copy operation.
spark.hadoop.fs.oss.copy.simple.max.byte	134217728	The file size limit for copying files between buckets in OSS using common APIs.
spark.hadoop.fs.oss.multipart.split.max.byte	67108864	The file multipart limit for copying files between buckets in OSS using common APIs.
spark.hadoop.fs.oss.multipart.split.number	5	The number of multipart files for copying files between buckets in OSS using common APIs. By default, the number is the same as the number of threads (concurrency) used.
spark.hadoop.fs.oss.impl	com.aliyun.fs.oss.nat.NativeOssFileSystem	The implementation class for the native file system of OSS.
spark.hadoop.fs.oss.buffer.dirs	/mnt/disk1,/mnt/disk2,…	The OSS local temporary directory. It uses a data disk in the cluster by default.
spark.hadoop.fs.oss.buffer.dirs.exists	false	Whether the OSS temporary directory exists.
spark.hadoop.fs.oss.client.connection.timeout	50000	Timeout period for OSS client connections (unit: milliseconds).
spark.hadoop.fs.oss.client.socket.timeout	50000	Timeout period for OSS Client sockets (unit: milliseconds).

Property	Default	Description
spark.hadoop.fs.oss.client.connection.ttl	-1	The value of time-to-live for clients
spark.hadoop.fs.oss.connection.max	1024	The maximum connections allowed.
spark.hadoop.job.runlocal	false	If the data source is OSS and you need to run and debug Spark code locally, set this parameter to true. Otherwise, set this parameter to false.
spark.logservice.fetch.interval.millis	200	The interval for receivers to retrieve data from LogHub.
spark.logservice.fetch.inOrder	true	Whether to consume the Shard data in order after the data has been parted.
spark.logservice.heartbeat.interval.millis	30000	The heartbeat interval for the data consumption processes (unit: milliseconds).
spark.mns.batchMsg.size	16	The number of MNS messages to fetch in bulk, with a maximum of 16.
spark.mns.pollingWait.seconds	30	The polling wait time if the MNS queue is empty.
spark.hadoop.io.compression.codec.snappy.native	false	Whether the Snappy files are in the standard Snappy format. By default, Hadoop recognizes Snappy files edited in Hadoop.

Smart Shuffle optimized configurations

Smart Shuffle is a shuffle implementation provided by Spark SQL in EMR version 3.16.0. It processes queries that have a large amount of shuffle data to improve the execution efficiency of Spark SQL. After Smart Shuffle is started, shuffle output data is not stored on ephemeral disks. Instead, data is transmitted to a remote node through

a network in advance. Data in the same partition is transmitted to the same node and stored in the same file. Currently, Smart Shuffle has the following limitations:

- Smart Shuffle cannot guarantee the atomictiy of task execution. When the execution of a task fails, you need to restart all the Spark jobs.
- Smart Shuffle currently does not provide the External Smart Shuffle implementation, nor support Dynamic Resource Allocation.
- Speculative tasks are not supported.
- Smart Shuffle is incompatible with Adaptive Execution.

Smart Shuffle related configurations are as follows. You can enable Smart Shuffle by using Spark configuration files or the spark-submit parameters.

Parameter	Default	Description
spark.shuffle.manager	sort	A shuffle method that allows you to enable the Smart Shuffle feature with Spark Session by configurating spark.shuffle.manager=org.apache.spark.shuffle.sort.SmartShuffleManager or spark.shuffle.manager=smart.
spark.shuffle.smart.spill.memorySizeForceSpillThreshold	128m	When Smart Shuffle is enabled, the memory used for each shuffle task has a threshold. When the threshold is reached, shuffle data is sent to the corresponding remote node through the network according to partitions.
spark.shuffle.smart.transfer.blockSize	1m	The size of the cache used for network transfers when Smart Shuffle is enabled.

3.3 Operate OSS files

This topic describes the possible issues of using OSS SDK and the recommended practices.

Issues in the use of OSS SDK

If you cannot directly use OSS SDK to operate OSS files in Spark or Hadoop jobs, it is because conflicts exist between the http-client-4.4.x version on which the OSS SDK is dependent and the http-client version in the Spark or Hadoop running environments . If such operations are required, you must resolve the dependency conflicts first. In fact, Spark and Hadoop are already seamlessly compatible with OSS in E-MapReduce, and you can operate OSS files as you are using HDFS.

- The current E-MapReduce environment supports MetaService, which allows you to access OSS data in the E-MapReduce environment without an AccessKey. The previous way of using an AccessKey is still supported. Internal network endpoints are preferred when using OSS.
- When you test locally, you need to use a public network endpoint of OSS so that you can access OSS data from your local machine.

For more information about a complete endpoint list, see OSS endpoints.

Recommended practices (without using an AccessKey)

We recommend that you use the following methods to query files under OSS directories:

```
[ Scala ]
    import   org . apache . hadoop . conf . Configurat  ion
    import   org . apache . hadoop . fs .{  Path ,  FileSystem }
    val   dir  = " oss :// bucket / dir "
    val   path  =  new   Path ( dir )
    val   conf  =  new   Configurat  ion ()
    conf . set (" fs . oss . impl ", " com . aliyun . fs . oss . nat .
 NativeOssF  ileSystem ")
    val   fs  =  FileSystem . get ( path . toUri ,  conf )
    val   fileList  =  fs . listStatus ( path )
   ...
   [ Java ]
    import   org . apache . hadoop . conf . Configurat  ion ;
    import   org . apache . hadoop . fs . Path ;
    import   org . apache . hadoop . fs . FileStatus ;
    import   org . apache . hadoop . fs . FileSystem ;
    String   dir  = " oss :// bucket / dir ";
    Path   path  =  new   Path ( dir );
    Configurat  ion   conf  =  new   Configurat  ion ();
    conf . set (" fs . oss . impl ", " com . aliyun . fs . oss . nat .
 NativeOssF  ileSystem ");
    FileSystem   fs  =  FileSystem . get ( path . toUri (),  conf );
```

```
    FileStatus []  fileList  =  fs . listStatus ( path );
   ...
```

3.4 Use Spark to access OSS

E-MapReduce supports MetaService, which allows you to access OSS data in the E-MapReduce environment without an AccessKey. The previous way of using an AccessKey and endpoint is also supported. Make sure that you use an internal IP address for the OSS endpoint. For more information about a complete list of endpoints, see OSS endpoints.

Allow Spark to access OSS

The following example shows how Spark reads data from OSS without an AccessKey and writes the processed data back to OSS.

```
val   conf  =  new   SparkConf (). setAppName (" Test   OSS ")
    val   sc  =  new   SparkConte  xt ( conf )
    val   pathIn  = " oss :// bucket / path / to / read "
    val   inputData  =  sc . textFile ( pathIn )
    val   cnt  =  inputData . count
    println ( s " count : $ cnt ")
    val   outputPath  = " oss :// bucket / path / to / write "
    val   outpuData  =  inputData . map ( e  =>  s "$ e   has   been
  processed .")
    outpuData . saveAsText  File ( outputPath )
```

Appendix

For the complete sample code, see:

- Allow Spark to access OSS

3.5 Use MaxCompute in Spark

This topic describes how to use the E-MapReduce SDK to read and write MaxCompute data in Spark.

Allow Spark to access MaxCompute

1. Initialize an OdpsOps object. In Spark, data operations of MaxCompute are performed using the OdpsOps class. Follow these steps to create an OdpsOps object:

```
import   com . aliyun . odps . TableSchem  a
     import   com . aliyun . odps . data . Record
     import   org . apache . spark . aliyun . odps . OdpsOps
     import   org . apache . spark .{  SparkConte  xt ,  SparkConf
}
```

```
     object   Sample  {
       def   main ( args :  Array [ String ]):  Unit  = {
        // ==  Step - 1  ==
         val   accessKeyI  d  = "< accessKeyI  d >"
         val   accessKeyS  ecret  = "< accessKeyS  ecret >"
        //  Take   the   internal   network   address   as   an
example
         val   urls  =  Seq (" http :// odps - ext . aliyun - inc .
com / api ", " http :// dt - ext . odps . aliyun - inc . com ")
         val   conf  =  new   SparkConf (). setAppName (" Test
Odps ")
         val   sc  =  new   SparkConte  xt ( conf )
         val   odpsOps  =  OdpsOps ( sc ,  accessKeyI  d ,
accessKeyS  ecret ,  urls ( 0 ),  urls ( 1 ))
        //  A   part   of   the   calling   code   is   shown   as
  follows
        // ==  Step - 2  ==
        ...
        // ==  Step - 3  ==
        ...
      }
      // ==  Step - 2  ==
      //  Method   definition   1
      // ==  Step - 3  ==
      //  Method   definition   2
    }
```

2. Load the table data from MaxCompute to Spark. By using the readTable method of the OdpsOps object, you can load a MaxCompute table to Spark and create an RDD as follows:

```
// ==  Step - 2  ==
         val   project  = < odps - project >
         val   table  = < odps - table >
         val   numPartiti  ons  =  2
         val   inputData  =  odpsOps . readTable ( project ,  table
 ,  read ,  numPartiti  ons )
         inputData . top ( 10 ). foreach ( println )
        // ==  Step - 3  ==
        ...
```

In the preceding code, you must define a read function to parse and pre-process the MaxCompute table data, as follows:

```
def   read ( record :  Record ,  schema :  TableSchem  a ):  String
 = {
          record . getString ( 0 )
       }
```

This function loads the first column of the MaxCompute table into the Spark runtime environment.

3. Save the result data in Spark to the MaxCompute table. Using the saveToTable method of the OdpsOps object, you can save Spark RDD to MaxCompute.

```
val   resultData  =  inputData . map ( e  =>  s "$ e   has   been
processed .")
```

```
          odpsOps . saveToTabl  e ( project ,  table ,  dataRDD ,
write )
```

In the preceding code, you must define a write function for data pre-processing before writing to the MaxCompute table as follows:

```
def   write ( s :  String ,  emptyReord :  Record ,  schema :
TableSchem  a ):  Unit  = {
           val   r  =  emptyReord
           r . set ( 0 ,  s )
        }
```

This function writes RDD data in each row to the first column of the corresponding MaxCompute table.

4. Notation of the partition table parameters.

   The SDK supports reading and writing MaxCompute partition tables. The standard naming of the tables is partition_column_name=partition_name (with multiple partitions separated by commas). Assume that we have partition columns pt and ps :

   - Read the table data of the partition where pt is 1.
   - Read the table data of the partition where pt is 1 and ps is 2.

Appendix

For the complete sample code, see:

- Allow Spark to access MaxCompute

3.6 Use Spark Streaming to consume MQ data

This topic describes how to use Spark Streaming to consume MQ data and count the words in each batch.

Use Spark to access MQ

The following example shows how to use Spark Streaming to consume MQ data and count the words in each batch.

```
val   Array ( cId ,  topic ,  subExpress  ion ,  parallelis  m ,
interval ) =  args
    val   accessKeyI  d  = "< accessKeyI  d >"
    val   accessKeyS  ecret  = "< accessKeyS  ecret >"
    val   numStreams  =  parallelis  m . toInt
    val   batchInter  val  =  Millisecon  ds ( interval . toInt )
    val   conf  =  new   SparkConf (). setAppName (" Test   ONS
Streaming ")
    val   ssc  =  new   StreamingC  ontext ( conf ,  batchInter  val
)
```

```
    def   func :  Message  =>  Array [ Byte ] =  msg  =>  msg .
getBody
    val   onsStreams  = ( 0   until   numStreams ). map  {  i  =>
      println ( s " starting   stream  $ i ")
      OnsUtils . createStre  am (
        ssc ,
        cId ,
        topic ,
        subExpress  ion ,
        accessKeyI  d ,
        accessKeyS  ecret ,
        StorageLev  el . MEMORY_AND  _DISK_2 ,
        func )
   }
    val   unionStrea  ms  =  ssc . union ( onsStreams )
    unionStrea  ms . foreachRDD ( rdd  => {
      rdd . map ( bytes  =>  new   String ( bytes )). flatMap ( line
=>  line . split (" "))
       . map ( word  => ( word ,  1 ))
       . reduceByKe  y ( _  +  _ ). collect (). foreach ( e  =>
println ( s " word : ${ e . _1 },  cnt : ${ e . _2 }"))
   })
    ssc . start ()
    ssc . awaitTermi  nation ()
```

Appendix

For the complete sample code, see:

- Allow Spark to access ONS

3.7 Consume Table Store data in Spark

This topic describes how to consume Table Store data in Spark.

Allow Spark to access Table Store

- Prepare a table

Create a table named pet. Set the name field as the primary key field.

name	owner	species	sex	birth	death
Fluffy	Harold	cat	f	1993-02-04	
Claws	Gwen	cat	m	1994-03-17	
Buffy	Harold	dog	f	1989-05-13	
Fang	Benny	dog	m	1990-08-27	
Bowser	Diane	dog	m	1979-08-31	1995-07-29
Chirpy	Gwen	bird	f	1998-09-11	
Whistler	Gwen	bird		1997-12-09	
Slim	Benny	snake	m	1996-04-29	

name	owner	species	sex	birth	death
Puffball	Diane	hamster	f	1999-03-30	

- The following example shows how Spark consumes Table Store data.

```
 private   static   RangeRowQu  eryCriteri  a   fetchCrite  ria () {
     RangeRowQu  eryCriteri  a   res  =  new   RangeRowQu
 eryCriteri  a (" pet ");
     res . setMaxVers  ions ( 1 );
     List < PrimaryKey  Column >  lower  =  new   ArrayList <
 PrimaryKey  Column >();
     List < PrimaryKey  Column >  upper  =  new   ArrayList <
 PrimaryKey  Column >();
     lower . add ( new   PrimaryKey  Column (" name ",  PrimaryKey
 Value . INF_MIN ));
     upper . add ( new   PrimaryKey  Column (" name ",  PrimaryKey
 Value . INF_MAX ));
     res . setInclusi  veStartPri  maryKey ( new   PrimaryKey (
 lower ));
     res . setExclusi  veEndPrima  ryKey ( new   PrimaryKey ( upper
 ));
     return   res ;
}
 public   static   void   main ( String []  args ) {
     SparkConf   sparkConf  =  new   SparkConf (). setAppName ("
 RowCounter ");
     JavaSparkC  ontext   sc  =  new   JavaSparkC  ontext (
 sparkConf );
     Configurat  ion   hadoopConf  =  new   Configurat  ion ();
     JavaSparkC  ontext   sc  =  null ;
     try  {
         sc  =  new   JavaSparkC  ontext ( sparkConf );
         Configurat  ion   hadoopConf  =  new   Configurat  ion ();
         TableStore . setCredent  ial (
                 hadoopConf ,
                 new   Credential ( accessKeyI  d ,  accessKeyS
 ecret ,  securityTo  ken ));
         Endpoint   ep  =  new   Endpoint ( endpoint ,  instance );
         TableStore . setEndpoin  t ( hadoopConf ,  ep );
         TableStore  InputForma  t . addCriteri  a ( hadoopConf ,
 fetchCrite  ria ());
         JavaPairRD  D < PrimaryKey  Writable ,  RowWritabl  e >
 rdd  =  sc . newAPIHado  opRDD (
                 hadoopConf ,  TableStore  InputForma  t . class ,
                 PrimaryKey  Writable . class ,  RowWritabl  e .
 class );
         System . out . println (
             new   Formatter (). format (" TOTAL : % d ",  rdd .
 count ()). toString ());
    }  finally  {
         if  ( sc  ! =  null ) {
             sc . close ();
        }
    }
}
```

Appendix

For the complete sample code, see:

- Allow Spark to access Table Store

3.8 Use Spark Streaming to consume MNS data

This topic describes how to use Spark Streaming to consume data in MNS and calculate the word count of every batch.

Allow Spark to access MNS

The sample code is as follows:

```
val   conf  =  new   SparkConf (). setAppName (" Test   MNS
Streaming ")
    val   batchInter  val  =  Seconds ( 10 )
    val   ssc  =  new   StreamingC  ontext ( conf ,  batchInter  val
)
    val   queuename  = " queuename "
    val   accessKeyI  d  = "< accessKeyI  d >"
    val   accessKeyS  ecret  = "< accessKeyS  ecret >"
    val   endpoint  = " http :// xxx . yyy . zzzz / abc "
    val   mnsStream  =  MnsUtils . createPull  ingStreamA  sRawBytes
( ssc ,  queuename ,  accessKeyI  d ,  accessKeyS  ecret ,  endpoint
,
      StorageLev  el . MEMORY_ONL  Y )
    mnsStream . foreachRDD (  rdd  => {
      rdd . map ( bytes  =>  new   String ( bytes )). flatMap ( line
=>  line . split (" "))
       . map ( word  => ( word ,  1 ))
       . reduceByKe  y ( _  +  _ ). collect (). foreach ( e  =>
println ( s " word : ${ e . _1 },  cnt : ${ e . _2 }"))
   })
    ssc . start ()
    ssc . awaitTermi  nation ()
```

Use Spark Streaming with MetaService

In the preceding example, we explicitly pass the AccessKey to the API. However, since E-MapReduce SDK 1.3.2, Spark Streaming can process MNS data using MetaService without an AccessKey. For more information, see the description of the MnsUtils class in E-MapReduce SDK:

```
MnsUtils . createPull  ingStreamA  sBytes ( ssc ,  queueName ,
endpoint ,  storageLev  el )
MnsUtils . createPull  ingStreamA  sRawBytes ( ssc ,  queueName ,
endpoint ,  storageLev  el )
```

Appendix

For the complete sample code, see:

- Allow Spark to access MNS

3.9 Use Spark to write data to HBase

This topic describes how Spark writes data to HBase. Note that

 **Notice:**

computing clusters must be in the same security group as HBase clusters. Otherwise, the network cannot be connected. When you create a cluster in E-MapReduce, make sure that you select the security group where the HBase cluster is located.

Allow Spark to access HBase

Use the following code:

```
object   Connection  Util   extends   Serializab  le  {
      private   val   conf  =  HBaseConfi  guration . create ()
      conf . set ( HConstants . ZOOKEEPER_  QUORUM ," ecs1 , ecs1 ,
ecs3 ")
      conf . set ( HConstants . ZOOKEEPER_  ZNODE_PARE  NT , "/ hbase
")
      private   val   connection  =  Connection  Factory .
createConn  ection ( conf )
      def   getDefault  Conn :  Connection  =  connection
   }
   // Create   data   streaming   unionStrea  ms
    unionStrea  ms . foreachRDD ( rdd  => {
      rdd . map ( bytes  =>  new   String ( bytes ))
       . flatMap ( line  =>  line . split (" "))
       . map ( word  => ( word ,  1 ))
       . reduceByKe  y ( _  +  _ )
       . mapPartiti  ons  { words  => {
          val   conn  =  Connection  Util . getDefault  Conn
          val   tableName  =  TableName . valueOf ( tname )
          val   t  =  conn . getTable ( tableName )
          try  {
            words . sliding ( 100 ,  100 ). foreach ( slice  => {
              val   puts  =  slice . map ( word  => {
                println ( s " word : $ word ")
                val   put  =  new   Put ( Bytes . toBytes ( word .
_1  +  System . currentTim  eMillis ()))
                put . addColumn ( COLUMN_FAM  ILY_BYTES ,
COLUMN_QUA  LIFIER_BYT  ES ,
                  System . currentTim  eMillis (),  Bytes . toBytes (
word . _2 ))
                put
             }). toList
              t . put ( puts )
           })
         }  finally  {
          t . close ()
         }
          Iterator . empty
       }}. count ()
   })
    ssc . start ()
```

```
    ssc . awaitTermi  nation ()
```

Appendix

For the complete sample code, see:

- Allow Spark to access HBase

3.10 Use Spark Streaming to process Kafka data

This topic describes how to run Spark Streaming jobs in Hadoop clusters of E-MapReduce to process data in Kafka clusters.

Programming reference

The Hadoop and Kafka clusters in E-MapReduce are based on completely open source software. Therefore, you can refer to the corresponding official documentation during your development.

- Spark official documentation: streaming-kafka-integration
- Spark official documentation: structured-streaming-kafka-integration
- E-MapReduce-demo: Github

Allow Spark Streaming jobs to access Kerberos Kafka clusters

E-MapReduce allows you to create Kafka clusters based on Kerberos authentication. Jobs in Hadoop clusters can access Kerberos Kafka clusters in two ways:

- Non-Kerberos Hadoop cluster: Provide the kafka_client_jaas.conf file for Kerberos authentication of Kafka clusters.
- Kerberos Hadoop cluster: Cross-domain communication based on Kerberos clusters. Provide the kafka_client_jaas.conf file for Kerberos authentication of Hadoop clusters.

Both methods require that you provide the kafka_client_jaas.conf file for Kerberos authentication when you run a job.

The file format of kafka_client_jaas.conf is as follows:

```
KafkaClien  t  {
    com . sun . security . auth . module . Krb5LoginM  odule
required
    useKeyTab = true
    storeKey = true
    serviceNam  e =" kafka "
    keyTab ="/ path / to / kafka . keytab "
    principal =" kafka / emr - header - 1 . cluster - 12345 @ EMR .
12345 .  COM ";
```

```
};
```

For more information about how to obtain the keytab file, see Kerberos authentica tion in the *User Guide*.

Allow Spark Streaming jobs to access Kerberos Kafka clusters

When you run Spark Streaming jobs to access Kerberos Kafka clusters, you can provide the kafka_client_jaas.conf file and the kafka.keytab file as needed in the spark-submit command line parameters.

```
spark - submit  -- conf   spark . driver . extraJavaO  ptions =-
Djava . security . auth . login . config ={{ PWD }}/ kafka_clie
nt_jaas . conf  -- conf   spark . executor . extraJavaO  ptions =-
Djava . security . auth . login . config ={{ PWD }}/ kafka_clie
nt_jaas . conf  -- files  / local / path / to / kafka_clie  nt_jaas
. conf ,/ local / path / to / kafka . keytab  -- class   xx . xx . xx
. KafkaSampl  e  -- num - executors   2  -- executor - cores   2  --
executor - memory   1g  -- master   yarn - cluster   xxx . jar   arg1
  arg2   arg3
```

Note that in the kafka_client_jaas.conf file, the keytab file path must be a relative path. Make sure that you configure the path in the following format:

```
KafkaClien  t  {
    com . sun . security . auth . module . Krb5LoginM  odule
required
    useKeyTab = true
    storeKey = true
    serviceNam  e =" kafka "
    keyTab =" kafka . keytab "
    principal =" kafka / emr - header - 1 . cluster - 12345 @ EMR .
12345 .  COM ";
};
```

3.11 Use Spark and write data to MySQL

This topic describes how to run Spark jobs in Hadoop clusters of E-MapReduce to calculate the word count and write the result to MySQL. A Spark Streaming job writes data to MySQL in a similar way as a Spark job does.

Allow Spark to access MySQL

Use the following code:

```
val   input  =  getSparkCo  ntext . textFile ( inputPath ,
numPartiti  ons )
    input . flatMap ( _ . split (" ")). map ( x  => ( x ,  1 )).
reduceByKe  y ( _  +  _ )
     . mapPartiti  ons ( e  => {
        var   conn :  Connection  =  null
        var   ps :  PreparedSt  atement  =  null
```

```
        val   sql  =  s " insert   into  $ tbName ( word ,  count )
values  (?, ?)"
        try  {
          conn  =  DriverMana  ger . getConnect  ion ( s " jdbc :
mysql ://$ dbUrl :$ dbPort /$ dbName ",  dbUser ,  dbPwd )
          ps  =  conn . prepareSta  tement ( sql )
          e . foreach ( pair  => {
            ps . setString ( 1 ,  pair . _1 )
            ps . setLong ( 2 ,  pair . _2 )
            ps . executeUpd  ate ()
         })

          ps . close ()
          conn . close ()
       }  catch  {
          case   e :  Exception  =>  e . printStack  Trace ()
       }  finally  {
          if  ( ps  ! =  null ) {
            ps . close ()
         }
          if  ( conn  ! =  null ) {
            conn . close ()
         }
       }
      Iterator . empty
   }). count ()
```

Appendix

For the complete sample code, see:

- Allow Spark to write data to RDS

3.12 Configure Spark-Submit parameters

This topic describes how to configure spark-submit parameters in E-MapReduce.

Cluster configuration

- Software configuration

  E-MapReduce version 1.1.0

  - Hadoop 2.6.0
  - Spark 1.6.0

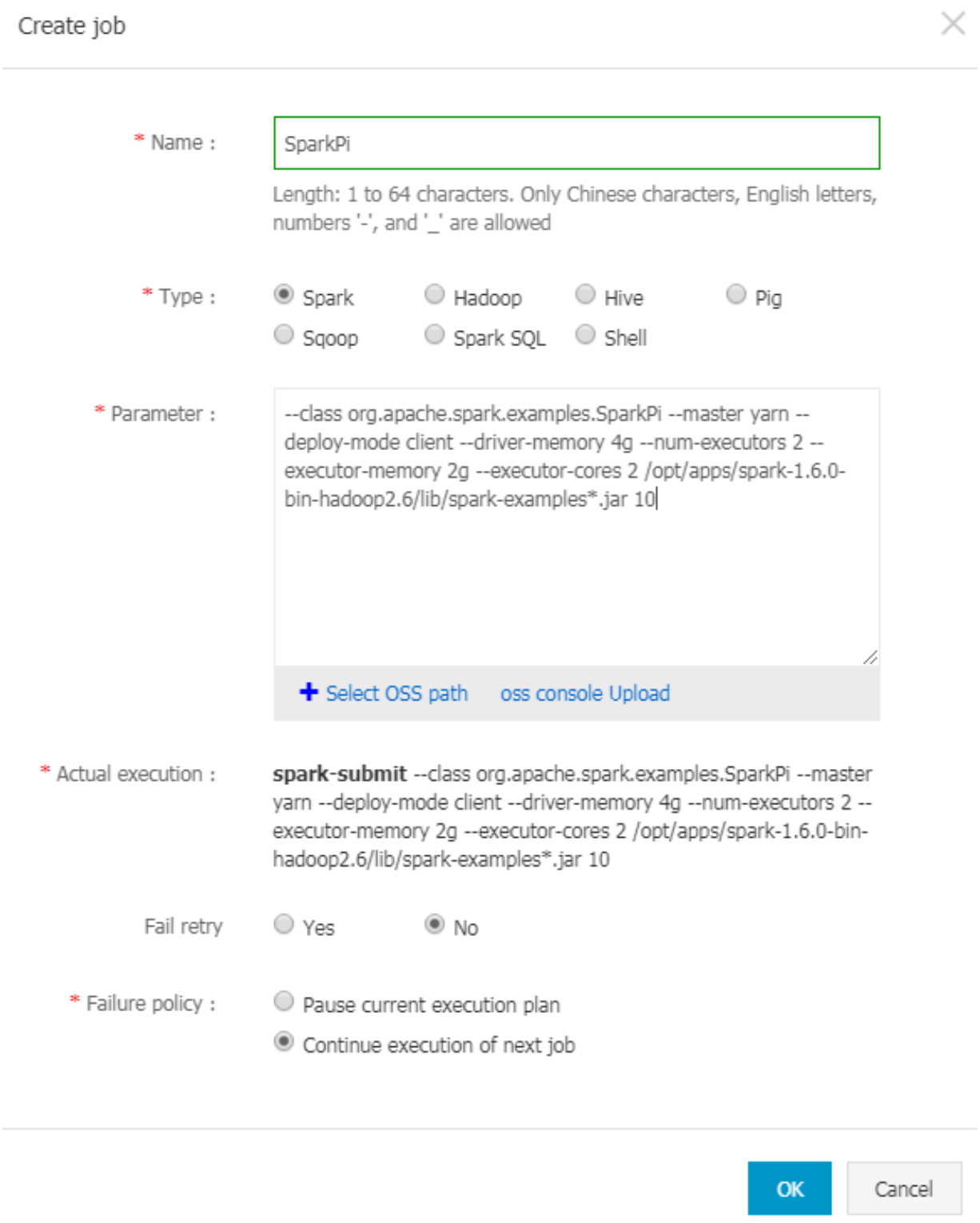
- Hardware configuration
  - Master node
    - 8-core, 16 GB RAM, 500 GB Ultra Disk
    - 1 unit
  - Worker node (x 10)
    - 8-core, 16 GB RAM, 500 GB Ultra Disk
    - 10 units
  - Total: 8-core 16GB (Worker) x 10 + 8-core 16GB (Master)

    Notice:
    Note: Only CPU and memory resources are calculated when a job is submitted. Therefore, the disk size is not included in the total resources.
  - Yarn total available resources: 12-core, 12.8 GB (worker) x 10

    Notice:
    By default, cores available for Yarn = number of cores x 1.5, memory available for Yarn = machine memory x 0.8.

Submit a job

After a cluster has been created, you can submit a job. First, you need to create a job in E-MapReduce with the following configuration:



The job in the preceding figure directly uses the official Spark example package, so you do not need to upload your own JAR package.

The parameters are listed as follows:

```
-- class   org . apache . spark . examples . SparkPi  -- master
 yarn  -- deploy - mode   client  -- driver - memory   4g  -- num -
 executors   2  -- executor - memory   2g  -- executor - cores   2
```

```
/ opt / apps / spark - 1 . 6 . 0 - bin - hadoop2 . 6 / lib / spark -
examples *. jar   10
```

The details of the parameters are described as follows:

Parameter	Reference value	Description
class	org.apache.spark.examples.SparkPi	The main class of the job.
master	yarn	E-MapReduce uses the Yarn mode. Therefore, the mode must be in the Yarn mode.
	yarn-client	Equivalent to setting the master parameter to yarn and the deploy-mode parameter to client. You do not need to set the deploy-mode parameter separately here.
	yarn-cluster	Equivalent to setting the master parameter to yarn and the deploy-mode parameter to cluster. You do not need to set the deploy-mode parameter separately here.
deploy-mode	client	The client mode indicates that the AM runs on the master node. You must set the preceding master parameter to yarn at the same time when setting this parameter.
	cluster	The cluster mode indicates that the AM runs randomly on one of the worker nodes. However, if you set this parameter, you must also set the preceding master parameter to yarn.

Parameter	Reference value	Description
driver-memory	4g	The memory used by the driver cannot exceed the total number of cores of a single machine.
num-executors	2	The number of executors to be created.
executor-memory	2g	The maximum memory used by each executor cannot exceed the maximum available memory of a single machine.
executor-cores	2	The number of threads used by each executor , which is also the maximum number of tasks that can be executed concurrently by each executor.

Resource calculations

The resources used by jobs running in different modes and settings are shown in the following table:

- Resource calculation for the yarn-client mode

Node	Resource type	Resource amount (the result is calculated based on the preceding examples)
master	core	1
	mem	driver-memory = 4 GB
worker	core	num-executors * executor -cores = 4

Node	Resource type	Resource amount (the result is calculated based on the preceding examples)
	mem	num-executors * executor -memory = 4 GB

- The primary program of the job (the driver program) runs on the master node. 4 GB memory (specified by --driver-memory) is allocated to the primary program based on the job settings (the memory may not be fully used).
- Two executors (specified by --num-executors) are initiated on the worker node , with each executor allocated with 2 GB (specified by --executor-memory) memory, and each executor supports a maximum of 2 (specified by - -executor-cores) concurrent tasks.

· Resource calculation for the yarn-cluster mode

Node	Resource type	Resource amount (the result is calculated using the preceding examples)
master		A tiny client program , which is responsibl e for synchronizing job information and consumes significantly less space.
worker	core	num-executors * executor -cores + spark.driver. cores = 5
	mem	num-executors * executor -memory + driver-memory = 8g

 Note:
By default, the value of spark.driver.cores is 1. You can set it to a value greater than 1.

Resource optimization

- yarn-client mode

  If you have a large job in the yarn-client mode and want to use more resources of the cluster, see the following configurations:

```
-- master   yarn - client  -- driver - memory   5g  -- num -
 executors   20  -- executor - memory   4g  -- executor - cores   4
```

> **Notice:**
>
> - When allocating memory, Spark allocates 375 MB or 7% (whichever is higher) extra memory in addition to the memory value you have set.
> - When allocating memory to containers, Yarn rounds up to the nearest integer gigabyte. Here the memory must be the integral multiple of 1 GB.

  Based on the preceding resource formula:

  - The resource amount for the master is:
    - cores: 1
    - mem: 6 GB (5 GB + 375 MB, rounded up to 6 GB)
  - The resource amount for the workers is:
    - core: 20*4 = 80
    - mem: 20*5 GB (4 GB + 375 MB, rounded up to 5 GB) = 100 GB

  We can see that the total resources allocated to the job have not exceeded the total resources for the cluster. Following this rule, you have multiple available configurations, such as:

```
-- master   yarn - client  -- driver - memory   5g  -- num -
 executors   40  -- executor - memory   1g  -- executor - cores   2
```

```
-- master   yarn - client  -- driver - memory   5g  -- num -
 executors   15  -- executor - memory   4g  -- executor - cores   4
```

```
-- master   yarn - client  -- driver - memory   5g  -- num -
 executors   10  -- executor - memory   9g  -- executor - cores   6
```

  In principle, you only have to make sure that the total resources calculated using the preceding formula do not exceed the total resources of the cluster. However, in production scenarios, the services of the system, HDFS, and E-MapReduce occupy certain core resources and memory resources. Therefore, if you use up the core

resources and memory resources, the job performance declines or the job fails to run.

Usually, the number of executor-cores is set to the same value as the core number of the cluster. If you set the value too large, the CPU switches frequently without benefiting the performance as expected.

- yarn-cluster mode

  In the yarn-cluster mode, the driver program runs on worker nodes. Resources in the resource pool of worker nodes are used. If you want to use more resources of this cluster, use the following configurations:

```
-- master   yarn - cluster  -- driver - memory   5g  -- num -
 executors   15  -- executor - memory   4g  -- executor - cores   4
```

Recommendations on configuration

- If you set the memory to a very large value, you should pay attention to the overhead caused by garbage collection. In general, we recommend that you set an executor with memory less than or equal to 64GB.
- If you are executing an HDFS read/write job, we recommend that you set the number of concurrent jobs for each executor to be less than or equal to 5 for reading and writing.
- If you are executing an OSS read/write job, we recommend that you distribute executors to different ECS instances so that the bandwidth of every ECS instance can be used. For example, if you have 10 ECS instances, you can set num-executors to 10, and set the appropriate memory and number of concurrent jobs.
- If the code you use in the job is not thread-safe, you need to monitor whether the concurrency causes job errors when setting the executor-cores. If yes, we recommend that you set executor-cores to 1.

4 Hadoop

4.1 Parameter description

Description of parameters in Hadoop code

The following parameters can be used in Hadoop code:

Property	Default	Description
fs.oss.accessKeyId	None	(Optional) The required AccessKey ID used to access OSS.
fs.oss.accessKeySecret	None	(Optional) The required AccessKey Secret used to access OSS.
fs.oss.securityToken	None	(Optional) The required STS token used to access OSS.
fs.oss.endpoint	None	(Optional) The endpoint used to access OSS.
fs.oss.multipart.thread.number	5	The number of threads (concurrency) used by OSS for upload part copy.
fs.oss.copy.simple.max.byte	134217728	The file limit of the internal copy in OSS using common APIs.
fs.oss.multipart.split.max.byte	67108864	The file multipart limit of copying files between buckets in OSS using common APIs.
fs.oss.multipart.split.number	5	The file multipart limit of copying files between buckets in OSS that is using common APIs. By default, the limit is equal to the number of threads used in the copy.

Property	Default	Description
fs.oss.impl	com.aliyun.fs.oss.nat.NativeOssFileSystem	The implementation class for the native OSS file system.
fs.oss.buffer.dirs	/mnt/disk1,/mnt/disk2,…	OSS local temporary directory. It uses a cluster data disk by default.
fs.oss.buffer.dirs.exists	false	Indicates whether the OSS temporary directory exists.
fs.oss.client.connection.timeout	50000	The connection timeout for the OSS client (ms).
fs.oss.client.socket.timeout	50,000	The socket timeout for the OSS client (ms).
fs.oss.client.connection.ttl	-1	The connection alive time
fs.oss.connection.max	1024	The maximum connections allowed.
io.compression.codec.snappy.native	false	Indicates whether to mark Snappy files as standard. By default, Hadoop recognizes Snappy files modified by Hadoop.

4.2 Create and run MapReduce jobs

This topic describes how to create and run a MapReduce job on E-MapReduce cluster.

Use OSS in MapReduce

To read and write data to OSS in MapReduce, configure the following parameters:

```
conf . set (" fs . oss . accessKeyI  d ", "${ accessKeyI  d }");
    conf . set (" fs . oss . accessKeyS  ecret ", "${ accessKeyS
ecret }");
    conf . set (" fs . oss . endpoint ","${ endpoint }");
```

Parameter descriptions:

- ${accessKeyId}: The AccessKey ID of your account.
- ${accessKeySecret}: The AccessKey Secret corresponding to the AccessKey ID.

- ${endpoint}: The network used for accessing OSS. It depends on the region in which your cluster exists, and the corresponding OSS must also be in the region in which your cluster exists.

  For more information about specific values, see OSS endpoints.

Word count

The following example describes how to read text from OSS and calculate the word count. The procedure is as follows:

1. Write the program

   Take the Java code as an example. Modify the WordCount sample on the official website of Hadoop as follows: Modify the instance by adding the configuration of the AccessKey ID and AccessKey Secret in the code so that the job has the permission to access OSS files.

```
package org . apache . hadoop . examples ;
 import java . io . IOExceptio n ;
 import java . util . StringToke nizer ;
 import org . apache . hadoop . conf . Configurat ion ;
 import org . apache . hadoop . fs . Path ;
 import org . apache . hadoop . io . IntWritabl e ;
 import org . apache . hadoop . io . Text ;
 import org . apache . hadoop . mapreduce . Job ;
 import org . apache . hadoop . mapreduce . Mapper ;
 import org . apache . hadoop . mapreduce . Reducer ;
 import org . apache . hadoop . mapreduce . lib . input .
FileInputF ormat ;
 import org . apache . hadoop . mapreduce . lib . output .
FileOutput Format ;
 import org . apache . hadoop . util . GenericOpt ionsParser ;
 public class EmrWordCou nt {
  public static class TokenizerM apper
       extends Mapper < Object , Text , Text , IntWritabl e
>{
    private final static IntWritabl e one = new
IntWritabl e ( 1 );
    private Text word = new Text ();
    public void map ( Object key , Text value , Context
  context
                  ) throws IOExceptio n , Interrupte
dException {
       StringToke nizer itr = new StringToke nizer ( value
. toString ());
       while ( itr . hasMoreTok ens ()) {
         word . set ( itr . nextToken ());
         context . write ( word , one );
      }
    }
  }
   public static class IntSumRedu cer
       extends Reducer < Text , IntWritabl e , Text ,
IntWritabl e > {
    private IntWritabl e result = new IntWritabl e ();
```

```
     public   void   reduce ( Text   key ,  Iterable < IntWritabl
e >  values ,
                       Context   context
                     )  throws   IOExceptio  n ,  Interrupte
dException  {
       int   sum  =  0 ;
       for  ( IntWritabl  e   val  :  values ) {
         sum  +=  val . get ();
      }
       result . set ( sum );
       context . write ( key ,  result );
    }
  }
   public   static   void   main ( String []  args )  throws
Exception  {
     Configurat  ion   conf  =  new   Configurat  ion ();
     String []  otherArgs  =  new   GenericOpt  ionsParser ( conf
,  args ). getRemaini  ngArgs ();
     if  ( otherArgs . length  <  2 ) {
       System . err . println (" Usage :  wordcount  < in > [< in
>...] < out >");
       System . exit ( 2 );
    }
     conf . set (" fs . oss . accessKeyI  d ", "${ accessKeyI  d
}");
     conf . set (" fs . oss . accessKeyS  ecret ", "${ accessKeyS
ecret }");
     conf . set (" fs . oss . endpoint ","${ endpoint }");
     Job   job  =  Job . getInstanc  e ( conf , " word   count ");
     job . setJarByCl  ass ( EmrWordCou  nt . class );
     job . setMapperC  lass ( TokenizerM  apper . class );
     job . setCombine  rClass ( IntSumRedu  cer . class );
     job . setReducer  Class ( IntSumRedu  cer . class );
     job . setOutputK  eyClass ( Text . class );
     job . setOutputV  alueClass ( IntWritabl  e . class );
     for  ( int   i  =  0 ;  i  <  otherArgs . length  -  1 ; ++ i
) {
       FileInputF  ormat . addInputPa  th ( job ,  new   Path (
otherArgs [ i ]));
    }
     FileOutput  Format . setOutputP  ath ( job ,
       new   Path ( otherArgs [ otherArgs . length  -  1 ]));
     System . exit ( job . waitForCom  pletion ( true ) ?  0  :  1
);
  }
}
```

2. **Compile the program**

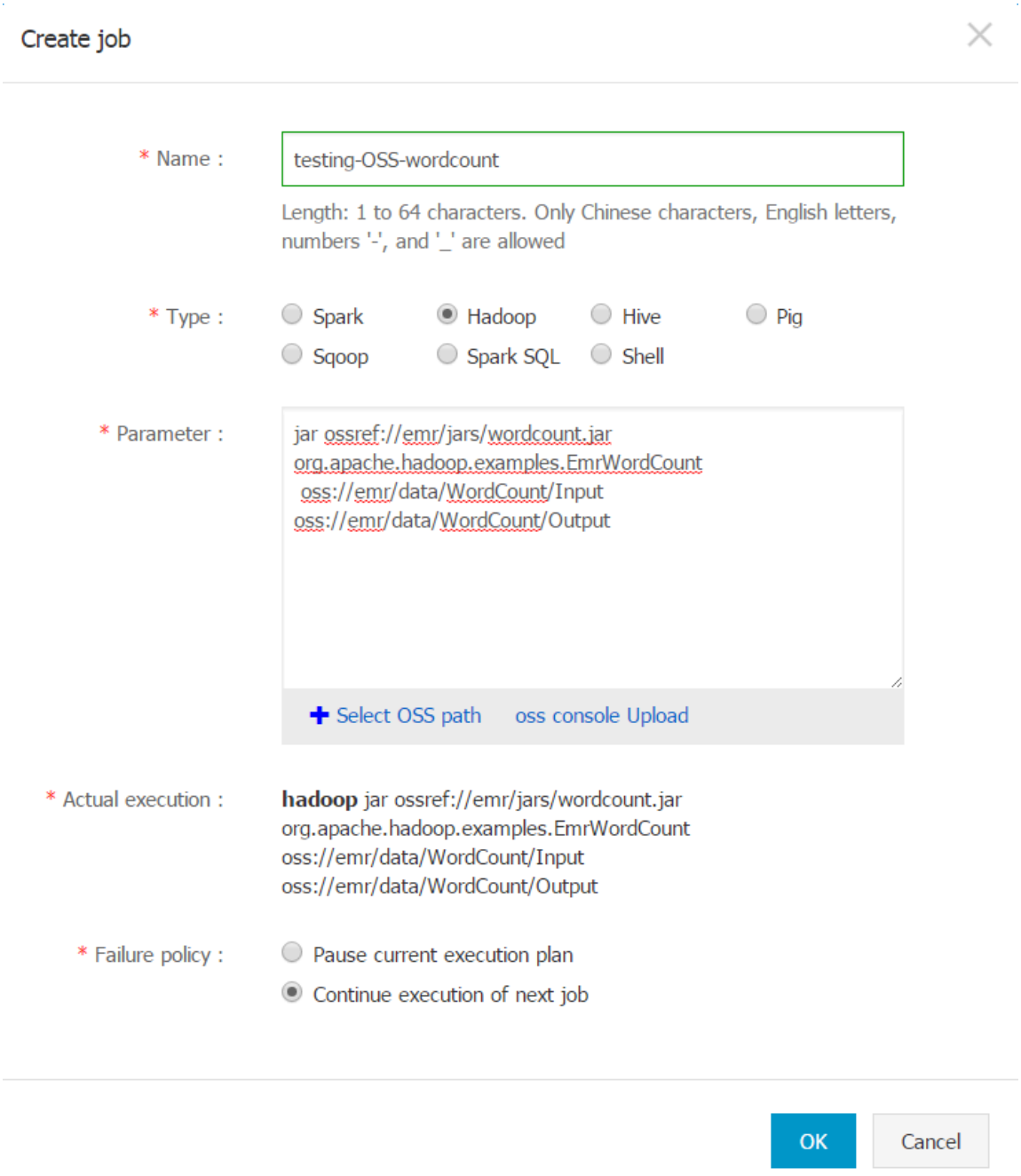
   First, you need to configure the JDK and Hadoop environments and then perform the following operations:

```
mkdir   wordcount_  classes
 javac  - classpath  ${ HADOOP_HOM  E }/ share / hadoop / common
/ hadoop - common - 2 . 6 . 0 . jar :${ HADOOP_HOM  E }/ share /
hadoop / mapreduce / hadoop - mapreduce - client - core - 2 . 6 . 0
. jar :${ HADOOP_HOM  E }/ share / hadoop / common / lib / commons
- cli - 1 . 2 . jar  - d   wordcount_  classes   EmrWordCou  nt .
java
```

```
jar cvf wordcount.jar -C wordcount_classes .
```

3. Create a job

   - Upload the JAR file prepared in the previous step to OSS. Log on to the OSS website for detailed operations. Assume that the path of the JAR file on OSS is `oss://emr/jars/wordcount.jar` and the input and output paths are `oss://emr/data/WordCount/Input` and `oss://emr/data/WordCount/Output`.
   - Create E-MapReduce jobs as follows:



4. Create an execution plan

   Create an execution plan in E-MapReduce and add the created job to the execution plan. Select Run Now as the policy so that the WordCount job runs in the selected cluster.

Manage MR jobs using Maven

When your project grows larger in size, it becomes considerably complicated to manage. We recommend that you use Maven or similar software management tools to manage projects. The procedure is as follows:

1. Install Maven

   First, make sure that you have installed Maven.

2. Generate the project framework.

   At the root directory of your project (suppose the root directory of your project is D :/workspace), execute the following commands:

   ```
   mvn   archetype : generate  - DgroupId = com . aliyun . emr .
   hadoop . examples  - DartifactI  d = wordcountv  2  - Darchetype
   ArtifactId = maven - archetype - quickstart  - Dinteracti  veMode =
   false
   ```

   Maven automatically generates an empty sample project at `D :/ workspace / wordcountv 2` (same with the artifactId you specified) containing a basic pom.xml file and an App class (the class package path is the same as the groupId you specified).

3. Add Hadoop dependencies

   Open the project with your favorite IDE and edit the pom.xml file. Add the following content to the dependencies:

   ```
   < dependency >
           < groupId > org . apache . hadoop </ groupId >
           < artifactId > hadoop - mapreduce - client - common </
    artifactId >
           < version > 2 . 6 . 0 </ version >
       </ dependency >
       < dependency >
           < groupId > org . apache . hadoop </ groupId >
           < artifactId > hadoop - common </ artifactId >
           < version > 2 . 6 . 0 </ version >
   ```

```
    </ dependency >
```

4. **Write the program**

   Add a new class named *WordCount2 . java* **at the same directory level as that of the App class under the** *com . aliyun . emr . hadoop . examples* **package. The content is as follows:**

```
package   com . aliyun . emr . hadoop . examples ;
 import   java . io . BufferedRe  ader ;
 import   java . io . FileReader ;
 import   java . io . IOExceptio  n ;
 import   java . net . URI ;
 import   java . util . ArrayList ;
 import   java . util . HashSet ;
 import   java . util . List ;
 import   java . util . Set ;
 import   java . util . StringToke  nizer ;
 import   org . apache . hadoop . conf . Configurat  ion ;
 import   org . apache . hadoop . fs . Path ;
 import   org . apache . hadoop . io . IntWritabl  e ;
 import   org . apache . hadoop . io . Text ;
 import   org . apache . hadoop . mapreduce . Job ;
 import   org . apache . hadoop . mapreduce . Mapper ;
 import   org . apache . hadoop . mapreduce . Reducer ;
 import   org . apache . hadoop . mapreduce . lib . input .
FileInputF  ormat ;
 import   org . apache . hadoop . mapreduce . lib . output .
FileOutput  Format ;
 import   org . apache . hadoop . mapreduce . Counter ;
 import   org . apache . hadoop . util . GenericOpt  ionsParser ;
 import   org . apache . hadoop . util . StringUtil  s ;
 public   class   WordCount2  {
     public   static   class   TokenizerM  apper
              extends   Mapper < Object ,  Text ,  Text ,
IntWritabl  e >{
         static   enum   CountersEn  um  {  INPUT_WORD  S  }
         private   final   static   IntWritabl  e   one  =  new
IntWritabl  e ( 1 );
         private   Text   word  =  new   Text ();
         private   boolean   caseSensit  ive ;
         private   Set < String >  patternsTo  Skip  =  new
HashSet < String >();
         private   Configurat  ion   conf ;
         private   BufferedRe  ader   fis ;
        @ Override
         public   void   setup ( Context   context )  throws
IOExceptio  n ,
                 Interrupte  dException  {
             conf  =  context . getConfigu  ration ();
             caseSensit  ive  =  conf . getBoolean (" wordcount .
case . sensitive ",  true );
             if  ( conf . getBoolean (" wordcount . skip . patterns
",  true )) {
                 URI []  patternsUR  Is  =  Job . getInstanc  e (
conf ). getCacheFi  les ();
                 for  ( URI   patternsUR  I  :  patternsUR  Is ) {
                     Path   patternsPa  th  =  new   Path (
patternsUR  I . getPath ());
                     String   patternsFi  leName  =  patternsPa  th
. getName (). toString ();
```

```
                    parseSkipF  ile ( patternsFi  leName );
                }
            }
        }
         private   void   parseSkipF  ile ( String   fileName ) {
             try  {
                 fis  =  new   BufferedRe  ader ( new   FileReader
( fileName ));
                 String   pattern  =  null ;
                 while  (( pattern  =  fis . readLine ()) ! =  null
) {
                     patternsTo  Skip . add ( pattern );
                 }
             }  catch  ( IOExceptio  n   ioe ) {
                 System . err . println (" Caught   exception
while   parsing   the   cached   file  '"
                         +  StringUtil  s . stringifyE  xception (
ioe ));
             }
         }
         @ Override
          public   void   map ( Object   key ,  Text   value ,
Context   context
         )  throws   IOExceptio  n ,  Interrupte  dException  {
             String   line  = ( caseSensit  ive ) ?
                     value . toString () :  value . toString ().
toLowerCas  e ();
             for  ( String   pattern  :  patternsTo  Skip ) {
                 line  =  line . replaceAll ( pattern , "");
             }
             StringToke  nizer   itr  =  new   StringToke  nizer (
line );
             while  ( itr . hasMoreTok  ens ()) {
                 word . set ( itr . nextToken ());
                 context . write ( word ,  one );
                 Counter   counter  =  context . getCounter (
CountersEn  um . class . getName (),
                         CountersEn  um . INPUT_WORD  S . toString
());
                 counter . increment ( 1 );
             }
         }
     }
      public   static   class   IntSumRedu  cer
             extends   Reducer < Text , IntWritabl  e , Text ,
IntWritabl  e > {
         private   IntWritabl  e   result  =  new   IntWritabl  e
();
         public   void   reduce ( Text   key ,  Iterable <
IntWritabl  e >  values ,
                           Context   context
         )  throws   IOExceptio  n ,  Interrupte  dException  {
             int   sum  =  0 ;
             for  ( IntWritabl  e   val  :  values ) {
                 sum  +=  val . get ();
             }
             result . set ( sum );
             context . write ( key ,  result );
         }
     }
      public   static   void   main ( String []  args )  throws
Exception  {
         Configurat  ion   conf  =  new   Configurat  ion ();
```

```
         conf . set (" fs . oss . accessKeyI  d ", "${ accessKeyI
d }");
        conf . set (" fs . oss . accessKeyS  ecret ", "${
accessKeyS  ecret }");
         conf . set (" fs . oss . endpoint ","${ endpoint }");
         GenericOpt  ionsParser   optionPars  er  =  new
GenericOpt  ionsParser ( conf ,  args );
         String []  remainingA  rgs  =  optionPars  er .
getRemaini  ngArgs ();
         if  (!(  remainingA  rgs . length  ! =  2  ||  remainingA
rgs . length  ! =  4 )) {
             System . err . println (" Usage :  wordcount  < in > <
out > [- skip   skipPatter  nFile ]");
             System . exit ( 2 );
        }
         Job   job  =  Job . getInstanc  e ( conf , " word   count
");
         job . setJarByCl  ass ( WordCount2 . class );
         job . setMapperC  lass ( TokenizerM  apper . class );
         job . setCombine  rClass ( IntSumRedu  cer . class );
         job . setReducer  Class ( IntSumRedu  cer . class );
         job . setOutputK  eyClass ( Text . class );
         job . setOutputV  alueClass ( IntWritabl  e . class );
         List < String >  otherArgs  =  new   ArrayList < String
>();
         for  ( int   i = 0 ;  i  <  remainingA  rgs . length ; ++
i ) {
             if  ("- skip ". equals ( remainingA  rgs [ i ])) {
                 job . addCacheFi  le ( new   Path ( EMapReduce
OSSUtil . buildOSSCo  mpleteUri ( remainingA  rgs [++ i ],  conf
)). toUri ());
                 job . getConfigu  ration (). setBoolean ("
wordcount . skip . patterns ",  true );
             }  else  {
                 otherArgs . add ( remainingA  rgs [ i ]);
             }
        }
         FileInputF  ormat . addInputPa  th ( job ,  new   Path (
EMapReduce  OSSUtil . buildOSSCo  mpleteUri ( otherArgs . get ( 0
),  conf )));
         FileOutput  Format . setOutputP  ath ( job ,  new   Path (
EMapReduce  OSSUtil . buildOSSCo  mpleteUri ( otherArgs . get ( 1
),  conf )));
         System . exit ( job . waitForCom  pletion ( true ) ?  0
:  1 );
     }
}
```

See the following sample code for the EMapReduceOSSUtil class, which is located in the same directory of WordCount2:

```
package   com . aliyun . emr . hadoop . examples ;
 import   org . apache . hadoop . conf . Configurat  ion ;
 public   class   EMapReduce  OSSUtil  {
     private   static   String   SCHEMA  = " oss ://";
     private   static   String   AKSEP  = ":";
     private   static   String   BKTSEP  = "@";
     private   static   String   EPSEP  = ".";
     private   static   String   HTTP_HEADE  R  = " http ://";
    /**
     *  complete   OSS   uri
```

```
     *  convert   uri   like :  oss :// bucket / path    to    oss
:// accessKeyI  d : accessKeyS  ecret @ bucket . endpoint / path
     *  ossref   do   not   need   this
     *
     * @ param   oriUri   original   OSS   uri
     */
     public   static   String   buildOSSCo  mpleteUri ( String
oriUri ,  String   akId ,  String   akSecret ,  String   endpoint )
{
         if  ( akId  ==  null ) {
             System . err . println (" miss   accessKeyI  d ");
             return   oriUri ;
         }
         if  ( akSecret  ==  null ) {
             System . err . println (" miss   accessKeyS  ecret ");
             return   oriUri ;
         }
         if  ( endpoint  ==  null ) {
             System . err . println (" miss   endpoint ");
             return   oriUri ;
         }
         int   index  =  oriUri . indexOf ( SCHEMA );
         if  ( index  == - 1  ||  index  ! =  0 ) {
             return   oriUri ;
         }
         int   bucketInde  x  =  index  +  SCHEMA . length ();
         int   pathIndex  =  oriUri . indexOf ("/",  bucketInde  x
);
         String   bucket  =  null ;
         if  ( pathIndex  == - 1 ) {
             bucket  =  oriUri . substring ( bucketInde  x );
         }  else  {
             bucket  =  oriUri . substring ( bucketInde  x ,
pathIndex );
         }
         StringBuil  der   retUri  =  new   StringBuil  der ();
         retUri . append ( SCHEMA )
                . append ( akId )
                . append ( AKSEP )
                . append ( akSecret )
                . append ( BKTSEP )
                . append ( bucket )
                . append ( EPSEP )
                . append ( stripHttp ( endpoint ));
         if  ( pathIndex  >  0 ) {
             retUri . append ( oriUri . substring ( pathIndex ));
         }
         return   retUri . toString ();
    }
     public   static   String   buildOSSCo  mpleteUri ( String
oriUri ,  Configurat  ion   conf ) {
         return   buildOSSCo  mpleteUri ( oriUri ,  conf . get ("
fs . oss . accessKeyI  d "),  conf . get (" fs . oss . accessKeyS
ecret "),  conf . get (" fs . oss . endpoint "));
    }
     private   static   String   stripHttp ( String   endpoint ) {
         if  ( endpoint . startsWith ( HTTP_HEADE  R )) {
             return   endpoint . substring ( HTTP_HEADE  R . length
());
         }
         return   endpoint ;
    }
```

```
}
```

5. Compile, package, and upload the code

   In the project directory, run the following commands:

   ```
   mvn clean package -DskipTests
   ```

   You can see the *wordcountv2-1.0-SNAPSHOT.jar* file in the target directory of your project directory, which is the JAR package of the job. Upload the JAR package to OSS.

6. Create a job

   Create a new job in E-MapReduce with the following parameters:

   ```
   jar ossref://yourBucket/yourPath/wordcountv2-1.0-SNAPSHOT.jar com.aliyun.emr.hadoop.examples.WordCount2 -Dwordcount.case.sensitive=true oss://yourBucket/yourPath/The_Sorrow_s_of_Young_Werther.txt oss://yourBucket/yourPath/output -skip oss://yourBucket/yourPath/patterns.txt
   ```

   Here, yourBucket represents your OSS bucket, and yourPath represents a path in the bucket. Configure the settings as needed. You need to download the files for processing related resources, which are *oss://yourBucket/yourPath/The_Sorrow_s_of_Young_Werther.txt* and *oss://yourBucket/yourPath/patterns.txt*, and then store them in OSS. You can download the resources needed for the jobs and store these resources in the corresponding directories in OSS.

   Download: The_Sorrows_of_Young_Werther.txtpatterns.txt

7. Create and run the execution plan

   Create the execution plan in E-MapReduce, associate it with the job, and then run the execution plan.

4.3 Create and run a Hive job

This topic describes how to create and run a Hive job on E-MapReduce cluster.

Use OSS in Hive

To read and write OSS in Hive, use the following statement to create an external table:

```
CREATE EXTERNAL TABLE eusers (
  userid INT )
```

```
  LOCATION  ' oss :// emr / users ';
```

If your cluster version is obsolete and the preceding method is not supported, or if you want to access an OSS instance that is located in another region with the AccessKey of another account, follow these steps:

```
CREATE  EXTERNAL  TABLE  eusers  (
  userid  INT )
 LOCATION  ' oss ://${ AccessKeyI  d }:${ AccessKeyS  ecret }@${
bucket }.${ endpoint }/ users ';
```

Parameter descriptions:

- ${accessKeyId}: The AccessKeyId of your account.
- ${accessKeySecret}: The secret key corresponding to the AccessKeyId.
- ${endpoint}: The network used for accessing OSS. It depends on the region in which your cluster exists. The corresponding OSS instance should be in the region in which your cluster exists.

  For more information about specific values, see OSS endpoints.

Use Tez as the computing engine

Tez is implemented after E-MapReduce version 2.1.0+. Tez is a computing framework for optimizing complex DAG scheduling jobs. It dramatically improves the speed of Hive jobs in many scenarios.

You can optimize your jobs by setting Tez as the execution engine. Follow these steps:

```
set  hive . execution . engine = tez
```

- Example 1

  Follow these steps:

  1. 1. Write the following script, save it as hiveSample1.sql, and then upload it to OSS.

     ```
     USE  DEFAULT ;
      set  hive . input . format = org . apache . hadoop . hive . ql
     . io . HiveInputF  ormat ;
      set  hive . stats . autogather = false ;
      DROP  TABLE  emrusers ;
      CREATE  EXTERNAL  TABLE  emrusers  (
        userid  INT ,
        movieid  INT ,
        rating  INT ,
        unixtime  STRING  )
       ROW  FORMAT  DELIMITED
       FIELDS  TERMINATED  BY  '\ t '
       STORED  AS  TEXTFILE
     ```

```
  LOCATION  ' oss ://${ bucket }/ yourpath ';
 SELECT   COUNT (*)  FROM   emrusers ;
 SELECT  *  from   emrusers   limit   100 ;
 SELECT   movieid , count ( userid )  as   usercount   from
emrusers   group   by   movieid   order   by   usercount   desc
limit   50 ;
```

2. Data resources for testing.

   You can download required resources for Hive jobs from the following link and store the resources to the corresponding directory in your OSS instance.

   Download resources: Public testing data.

3. Create a job.

   Create a new job in E-MapReduce using the following configuration:

   ```
   - f   ossref ://${ bucket }/ yourpath / hiveSample  1 . sql
   ```

   In this example, ${bucket} represents your OSS buckets, and yourPath represents a path in the bucket. You need to replace yourPath with where the Hive script is saved.

4. Create and run an execution plan.

   Create an execution plan. You can associate it with an existing cluster, or create a cluster as needed and associate the plan with the cluster. Save the job as Run Manually. Return to the previous page and click Run Now to run the job.

- Example 2

  Take scan in HiBench as an example.

  Follow these steps:

  1. Write the following script:

     ```
     USE   DEFAULT ;
      set   hive . input . format = org . apache . hadoop . hive . ql
     . io . HiveInputF  ormat ;
      set   mapreduce . job . maps = 12 ;
      set   mapreduce . job . reduces = 6 ;
      set   hive . stats . autogather = false ;
      DROP   TABLE   uservisits ;
      CREATE   EXTERNAL   TABLE   uservisits  ( sourceIP   STRING
     , destURL   STRING , visitDate   STRING , adRevenue   DOUBLE
     , userAgent   STRING , countryCod  e   STRING , languageCo
     de   STRING , searchWord   STRING , duration   INT  )  ROW
     FORMAT   DELIMITED   FIELDS   TERMINATED   BY  ','  STORED   AS
     ```

```
SEQUENCEFI  LE   LOCATION  ' oss ://${ bucket }/ sample - data /
hive / Scan / Input / uservisits ';
```

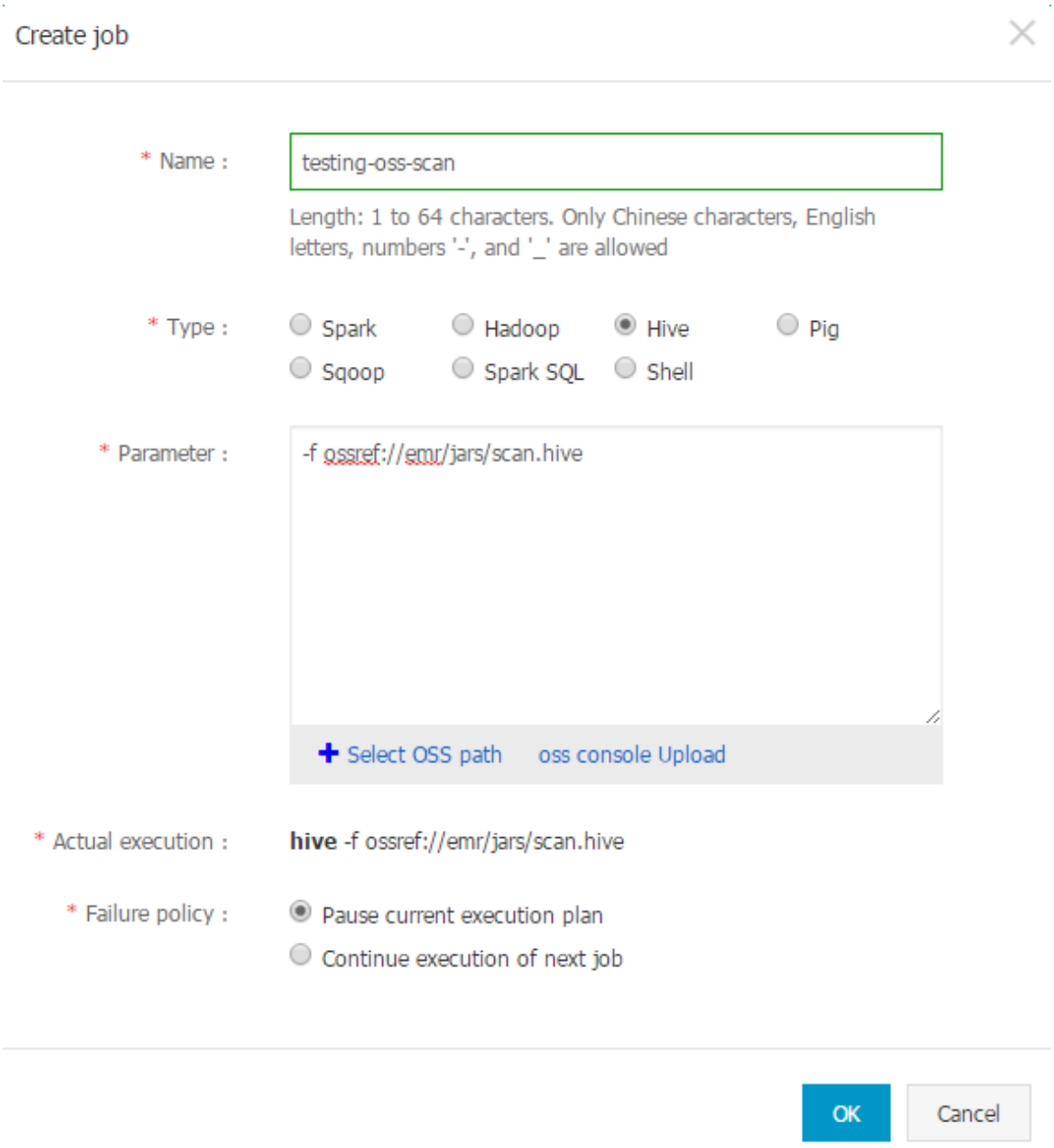
2. Prepare test data

   You can download required resources for the job from the following link and store the resources to the corresponding directory in your OSS instance.

   Download resources: uservisits

3. Create a job

   Store the script created in step 1 to OSS. For example, if the storage path is `oss :// emr / jars / scan . hive` , follow these steps to create a job in E-MapReduce:



4. Create and run an execution plan.

   Create an execution plan. You can associate it with an existing cluster, or create a cluster as needed and associate the plan with the cluster. Save the job as Run Manually. Return to the previous page and click Run Now to run the job.

4.4 Create and run a Pig job

This topic describes how to create and run a Pig job on E-MapReduce cluster.

Use OSS in Pig

Use the following format for OSS paths:

```
oss ://${ AccessKeyI  d }:${ AccessKeyS  ecret }@${ bucket }.${
endpoint }/${ path }
```

Parameter descriptions:

- ${accessKeyId}: The AccessKey ID of your account.
- ${accessKeySecret}: The AccessKey Secret corresponding to the AccessKey ID.
- ${bucket}: The bucket corresponding to the AccessKey ID.
- ${endpoint}: The network used for accessing OSS. It depends on the region in which your cluster exists, and the corresponding OSS should also be in the region in which your cluster exists.

  For more information about specific values, see OSS endpoint.
- ${path}: Path in the bucket.

Take script1-hadoop.pig in Pig as an example. Upload tutorial.jar and excite.log.bz2 in Pig to OSS. Suppose that the URLs for uploading files are *oss :// emr / jars / tutorial . jar* and *oss :// emr / data / excite . log . bz2* respectively.

Follow these steps:

1. Create scripts

   Edit the path of the JAR file and the input and output paths in the script, as shown below: Note that the format of the OSS path is *oss ://${ accesskeyI d }:${ accessKeyS ecret }@${ bucket }.${ endpoint }/ object / path .*

```
/*
 *  Licensed   to   the   Apache   Software   Foundation   ( ASF )
 under   one
*  or   more   contributo  r   license   agreements .   See   the
 NOTICE   file
```

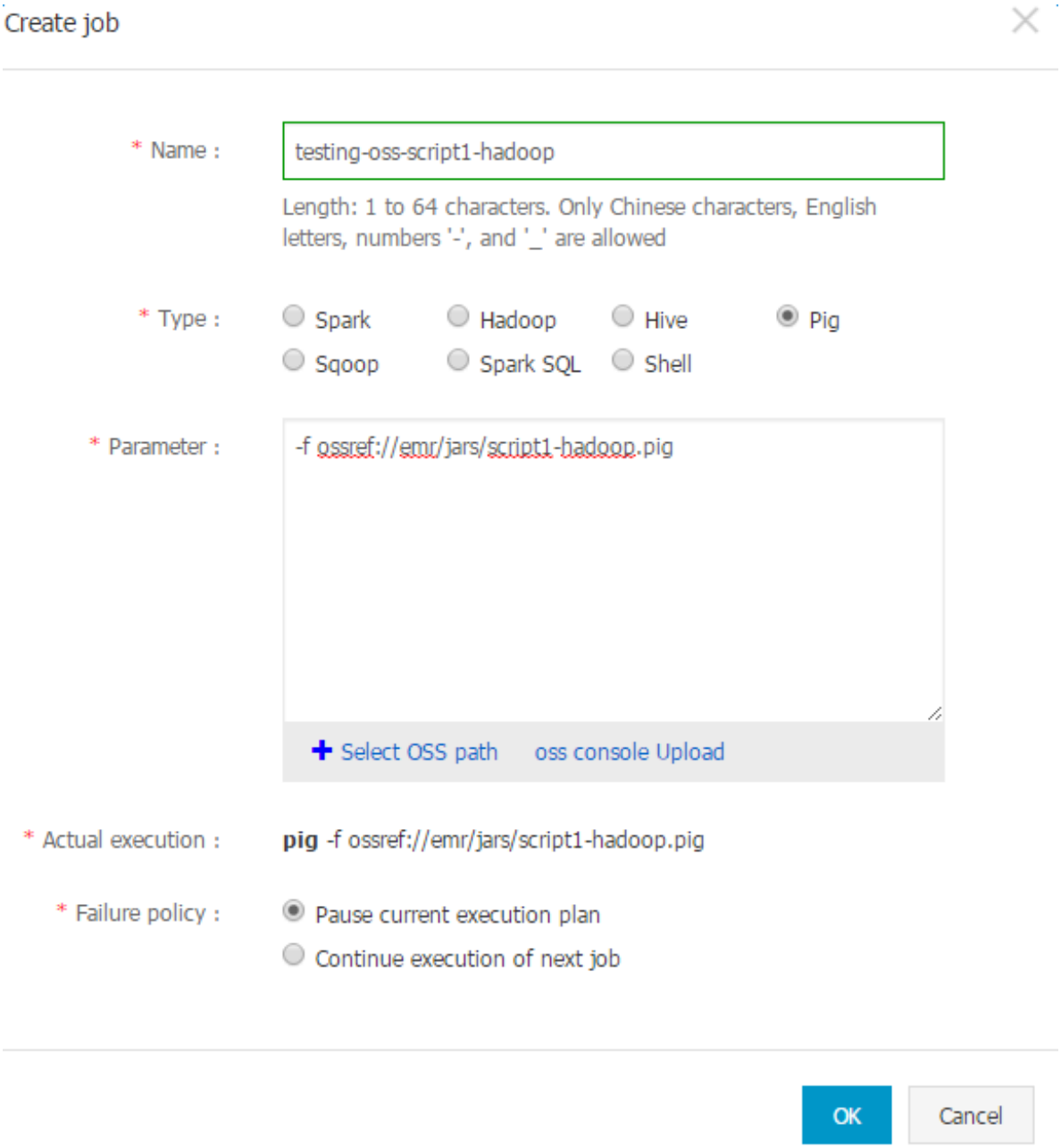
```
*  distribute  d   with   this   work   for   additional
 informatio  n
*  regarding   copyright   ownership .   The   ASF   licenses   this
   file
*  to   you   under   the   Apache   License ,  Version   2 . 0  (
 the
* " License ");  you   may   not   use   this   file   except   in
 compliance
*  with   the   License .   You   may   obtain   a   copy   of   the
   License   at
*
*      http :// www . apache . org / licenses / LICENSE - 2 . 0
*
*  Unless   required   by   applicable   law   or   agreed   to   in
   writing ,  software
*  distribute  d   under   the   License   is   distribute  d   on
 an  " AS   IS "  BASIS ,
*  WITHOUT   WARRANTIES   OR   CONDITIONS   OF   ANY   KIND ,
 either   express   or   implied .
*  See   the   License   for   the   specific   language   governing
   permission  s   and
*  limitation  s   under   the   License .
*/
--  Query   Phrase   Popularity  ( Hadoop   cluster )
--  This   script   processes   a   search   query   log   file
   from   the   Excite   search   engine   and   finds   search
 phrases   that   occur   with   particular   high   frequency
 during   certain   times   of   the   day .
--  Register   the   tutorial   JAR   file   so   that   the
 included   UDFs   can   be   called   in   the   script .
 REGISTER   oss ://${ AccessKeyI  d }:${ AccessKeyS  ecret }@${
 bucket }.${ endpoint }/ data / tutorial . jar ;
--  Use   the    PigStorage   function   to   load   the   excite
   log   file   into   the   raw   bag   as   an   array   of
 records .
--  Input : ( user , time , query )
 raw  =  LOAD  ' oss ://${ AccessKeyI  d }:${ AccessKeyS  ecret
 }@${ bucket }.${ endpoint }/ data / excite . log . bz2 '  USING
 PigStorage ('\ t ')  AS  ( user ,  time ,  query );
--  Call   the   NonURLDete  ctor   UDF   to   remove   records   if
   the   query   field   is   empty   or   a   URL .
 clean1  =  FILTER   raw   BY   org . apache . pig . tutorial .
 NonURLDete  ctor ( query );
--  Call   the   ToLower   UDF   to   change   the   query   field
 to   lowercase .
 clean2  =  FOREACH   clean1   GENERATE   user ,  time ,      org .
 apache . pig . tutorial . ToLower ( query )  as   query ;
--  Because   the   log   file   only   contains   queries   for   a
   single   day ,  we   are   only   interested   in   the   hour .
--  The   excite   query   log   timestamp   format   is
 YYMMDDHHMM  SS .
--  Call   the   ExtractHou  r   UDF   to   extract   the   hour   (
 HH )  from   the   time   field .
 houred  =  FOREACH   clean2   GENERATE   user ,  org . apache . pig
 . tutorial . ExtractHou  r ( time )  as   hour ,  query ;
--  Call   the   NGramGener  ator   UDF   to   compose   the   n -
 grams   of   the   query .
 ngramed1  =  FOREACH   houred   GENERATE   user ,  hour ,  flatten
 ( org . apache . pig . tutorial . NGramGener  ator ( query ))  as
 ngram ;
--  Use   the    DISTINCT   command   to   get   the   unique   n -
 grams   for   all   records .
 ngramed2  =  DISTINCT   ngramed1 ;
```

```
-- Use the GROUP command to group records by n -
 gram and hour .
 hour_frequ ency1 = GROUP ngramed2 BY ( ngram , hour );
-- Use the COUNT function to get the count (
 occurrence s ) of each n - gram .
 hour_frequ ency2 = FOREACH hour_frequ ency1 GENERATE
 flatten ($ 0 ), COUNT ($ 1 ) as count ;
-- Use the GROUP command to group records by n -
 gram only .
-- Each group now correspond s to a distinct n -
 gram and has the count for each hour .
 uniq_frequ ency1 = GROUP hour_frequ ency2 BY group ::
 ngram ;
-- For each group , identify the hour in which
 this n - gram is used with a particular ly high
 frequency .
-- Call the ScoreGener ator UDF to calculate a "
 popularity " score for the n - gram .
 uniq_frequ ency2 = FOREACH uniq_frequ ency1 GENERATE
   flatten ($ 0 ), flatten ( org . apache . pig . tutorial .
 ScoreGener ator ($ 1 ));
-- Use the FOREACH - GENERATE command to assign names
   to the fields .
 uniq_frequ ency3 = FOREACH uniq_frequ ency2 GENERATE $ 1
 as hour , $ 0 as ngram , $ 2 as score , $ 3 as count
 , $ 4 as mean ;
-- Use the FILTER command to move all records with
   a score less than or equal to 2 . 0 .
 filtered_u niq_freque ncy = FILTER uniq_frequ ency3 BY
 score > 2 . 0 ;
-- Use the ORDER command to sort the remaining
 records by hour and score .
 ordered_un iq_frequen cy = ORDER filtered_u niq_freque ncy
   BY hour , score ;
-- Use the PigStorage function to store the results
 .
-- Output : ( hour , n - gram , score , count , average_co
 unts_among _all_hours )
```

```
STORE   ordered_un  iq_frequen  cy   INTO  ' oss ://${ AccessKeyI
d }:${ AccessKeyS  ecret }@${ bucket }.${ endpoint }/ data /
script1 - hadoop - results '  USING   PigStorage ();
```

2. Create a job

   Store the script created in step 1 to OSS. For example, if the storage path is `oss ://emr / jars / script1 - hadoop . pig` , follow these steps to create a job in E-MapReduce:



3. Create and run an execution plan.

   Create the execution plan in E-MapReduce and add the created Pig job to the execution plan. Select Run Now as the policy so that the script1-hadoop job can run in the selected cluster.

4.5 Create a Hadoop streaming job

This topic describes how to use PythonH to create a Hadoop streaming job.

Use Python to create a Hadoop streaming job

The mapper code is as follows:

```
#! / usr / bin / env   python
 import   sys
 for   line   in   sys . stdin :
     line  =  line . strip ()
     words  =  line . split ()
     for   word   in   words :
         print  '% s \ t % s ' % ( word ,  1 )
```

The reducer code is as follows:

```
#! / usr / bin / env   python
 from   operator   import   itemgetter
 import   sys
 current_wo  rd  =  None
 current_co  unt  =  0
 word  =  None
 for   line   in   sys . stdin :
     line  =  line . strip ()
     word ,  count  =  line . split ('\ t ',  1 )
     try :
         count  =  int ( count )
     except   ValueError :
         continue
     if   current_wo  rd  ==  word :
         current_co  unt  +=  count
     else :
         if   current_wo  rd :
             print  '% s \ t % s ' % ( current_wo  rd ,  current_co
 unt )
         current_co  unt  =  count
         current_wo  rd  =  word
 if   current_wo  rd  ==  word :
     print  '% s \ t % s ' % ( current_wo  rd ,  current_co  unt )
```

Suppose that the mapper code is saved as */ home / hadoop / mapper . py* and the reducer code is saved as */ home / hadoop / reducer . py* , and that the paths for input and output are */ tmp / input* and */ tmp / output* in the HDFS file system respectively. Submit the following Hadoop command in the E-MapReduce cluster.

```
 hadoop   jar  / usr / lib / hadoop - current / share / hadoop / tools
 / lib / hadoop - streaming -*. jar  - file  / home / hadoop / mapper
 . py  - mapper   mapper . py  - file  / home / hadoop / reducer . py
```

```
 - reducer  reducer . py  - input  / tmp / hosts  - output  / tmp /
output
```

4.6 Process Table Store data in Hive

This topic describes how to process Table Store data in Hive.

Connect Hive to Table Store

- **Prepare a data table**

  Create a table named pet and set the name field as the primary key.

name	owner	species	sex	birth	death
Fluffy	Harold	cat	f	1993-02-04	
Claws	Gwen	cat	m	1994-03-17	
Buffy	Harold	dog	f	1989-05-13	
Fang	Benny	dog	m	1990-08-27	
Bowser	Diane	dog	m	1979-08-31	1995-07-29
Chirpy	Gwen	bird	f	1998-09-11	
Whistler	Gwen	bird		1997-12-09	
Slim	Benny	snake	m	1996-04-29	
Puffball	Diane	hamster	f	1999-03-30	

- **The following example describes how to process Table Store data in Hive.**

  1. **Command line**

```
$  HADOOP_HOM  E = YourHadoop  Dir   HADOOP_CLA  SSPATH = emr -
 tablestore -< version >. jar : tablestore - 4 . 1 . 0 - jar -
 with - dependenci  es . jar : joda - time - 2 . 9 . 4 . jar   bin
 / hive
```

  2. **Create an external table**

```
 CREATE   EXTERNAL   TABLE   pet
  ( name   STRING ,  owner   STRING ,  species   STRING ,  sex
STRING ,  birth   STRING ,  death   STRING )
  STORED   BY  ' com . aliyun . openservic  es . tablestore .
hive . TableStore  StorageHan  dler '
  WITH   SERDEPROPE  RTIES (
   " tablestore . columns . mapping "=" name , owner , species ,
sex , birth , death ")
  TBLPROPERT  IES  (
   " tablestore . endpoint "=" YourEndpoi  nt ",
   " tablestore . access_key  _id "=" YourAccess  KeyId ",
   " tablestore . access_key  _secret "=" YourAccess  KeySecret
",
```

```
    " tablestore . table . name "=" pet ");
```

3. **Insert data to the external table**

```
INSERT  INTO  pet  VALUES (" Fluffy ", " Harold ", " cat ", "
f ", " 1993 - 02 - 04 ",  null );
INSERT  INTO  pet  VALUES (" Claws ", " Gwen ", " cat ", " m
", " 1994 - 03 - 17 ",  null );
INSERT  INTO  pet  VALUES (" Buffy ", " Harold ", " dog ", " f
", " 1989 - 05 - 13 ",  null );
INSERT  INTO  pet  VALUES (" Fang ", " Benny ", " dog ", " m
", " 1990 - 08 - 27 ",  null );
INSERT  INTO  pet  VALUES (" Bowser ", " Diane ", " dog ", " m
", " 1979 - 08 - 31 ", " 1995 - 07 - 29 ");
INSERT  INTO  pet  VALUES (" Chirpy ", " Gwen ", " bird ", " f
", " 1998 - 09 - 11 ",  null );
INSERT  INTO  pet  VALUES (" Whistler ", " Gwen ", " bird ",
null , " 1997 - 12 - 09 ",  null );
INSERT  INTO  pet  VALUES (" Slim ", " Benny ", " snake ", " m
", " 1996 - 04 - 29 ",  null );
INSERT  INTO  pet  VALUES (" Puffball ", " Diane ", " hamster
", " f ", " 1999 - 03 - 30 ",  null );
```

4. **Query data**

```
>  SELECT  *  FROM  pet ;
 Bowser    Diane     dog       m         1979 - 08 - 31
 1995 - 07 - 29
 Buffy     Harold    dog       f         1989 - 05 - 13
 NULL
 Chirpy    Gwen      bird      f         1998 - 09 - 11
 NULL
 Claws     Gwen      cat       m         1994 - 03 - 17
 NULL
 Fang      Benny     dog       m         1990 - 08 - 27
 NULL
 Fluffy    Harold    cat       f         1993 - 02 - 04
 NULL
 Puffball          Diane     hamster   f         1999 - 03 - 30
     NULL
 Slim      Benny     snake     m         1996 - 04 - 29
 NULL
 Whistler          Gwen      bird      NULL      1997 - 12 - 09
     NULL
>  SELECT  *  FROM  pet  WHERE  birth  > " 1995 - 01 - 01 ";
 Chirpy    Gwen      bird      f         1998 - 09 - 11
 NULL
 Puffball          Diane     hamster   f         1999 - 03 - 30
     NULL
 Slim      Benny     snake     m         1996 - 04 - 29
 NULL
```

```
Whistler          Gwen      bird      NULL      1997 - 12 - 09
    NULL
```

Data type conversion

	TINYINT	SMALLINT	INT	BIGINT	FLOAT	DOUBLE	BOOLEAN	STRING	BINARY
INTEGER	Supported (with loss of precision)	Supported (with loss of precision)	Supported (with loss of precision)	Supported	Supported (with loss of precision)	Supported (with loss of precision)			
DOUBLE	Supported (with loss of precision)	Supported (with loss of precision)	Supported (with loss of precision)	Supported (with loss of precision)	Supported (with loss of precision)	Supported			
BOOLEAN							Supported		
STRING								Supported	
BINARY									Supported

Appendix

For the complete sample code, see

- Process Table Store data in Hive

4.7 Process Table Store data in MR

This topic describes how to process Table Store data in MapReduce.

Connect MR to Table Store

- Prepare a data table

  Create a table named pet and set the name field as the primary key.

name	owner	species	sex	birth	death
Fluffy	Harold	cat	f	1993-02-04	
Claws	Gwen	cat	m	1994-03-17	
Buffy	Harold	dog	f	1989-05-13	
Fang	Benny	dog	m	1990-08-27	
Bowser	Diane	dog	m	1979-08-31	1995-07-29

name	owner	species	sex	birth	death
Chirpy	Gwen	bird	f	1998-09-11	
Whistler	Gwen	bird		1997-12-09	
Slim	Benny	snake	m	1996-04-29	
Puffball	Diane	hamster	f	1999-03-30	

- The following example describes how to process Table Store data in MR.

```
public   class   RowCounter  {
    public   static   class   RowCounter  Mapper
      extends   Mapper < PrimaryKey  Writable ,  RowWritabl  e ,
Text ,  LongWritab  le > {
        private   final   static   Text   agg  =  new   Text ("
TOTAL ");
        private   final   static   LongWritab  le   one  =  new
LongWritab  le ( 1 );
       @ Override   public   void   map ( PrimaryKey  Writable
key ,  RowWritabl  e   value ,
            Context   context )  throws   IOExceptio  n ,
Interrupte  dException  {
            context . write ( agg ,  one );
       }
   }
    public   static   class   IntSumRedu  cer
      extends   Reducer < Text , LongWritab  le , Text ,
LongWritab  le > {
       @ Override   public   void   reduce ( Text   key ,  Iterable
< LongWritab  le >  values ,
            Context   context )  throws   IOExceptio  n ,
Interrupte  dException  {
            long   sum  =  0 ;
            for  ( LongWritab  le   val  :  values ) {
                sum  +=  val . get ();
           }
            context . write ( key ,  new   LongWritab  le ( sum ));
       }
   }
    private   static   RangeRowQu  eryCriteri  a   fetchCrite  ria
() {
        RangeRowQu  eryCriteri  a   res  =  new   RangeRowQu
eryCriteri  a (" pet ");
        res . setMaxVers  ions ( 1 );
        List < PrimaryKey  Column >  lower  =  new   ArrayList <
PrimaryKey  Column >();
        List < PrimaryKey  Column >  upper  =  new   ArrayList <
PrimaryKey  Column >();
        lower . add ( new   PrimaryKey  Column (" name ",
PrimaryKey  Value . INF_MIN ));
        upper . add ( new   PrimaryKey  Column (" name ",
PrimaryKey  Value . INF_MAX ));
        res . setInclusi  veStartPri  maryKey ( new   PrimaryKey (
lower ));
        res . setExclusi  veEndPrima  ryKey ( new   PrimaryKey (
upper ));
        return   res ;
   }
    public   static   void   main ( String []  args )  throws
Exception  {
```

```
        Configurat  ion   conf  =  new   Configurat  ion ();
        job . setJarByCl  ass ( RowCounter . class );
        job . setMapperC  lass ( RowCounter  Mapper . class );
        job . setCombine  rClass ( IntSumRedu  cer . class );
        job . setReducer  Class ( IntSumRedu  cer . class );
        job . setOutputK  eyClass ( Text . class );
        job . setOutputV  alueClass ( LongWritab  le . class );
        job . setInputFo  rmatClass ( TableStore  InputForma  t .
 class );
        TableStore . setCredent  ial ( job ,  accessKeyI  d ,
 accessKeyS  ecret ,  securityTo  ken );
        TableStore . setEndpoin  t ( job ,  endpoint ,  instance );
        TableStore  InputForma  t . addCriteri  a ( job ,
 fetchCrite  ria ());
        FileOutput  Format . setOutputP  ath ( job ,  new   Path (
 outputPath ));
        System . exit ( job . waitForCom  pletion ( true ) ?  0  :
 1 );
    }
}
```

- **The following example describes how to use MR to write data to Table Store.**

```
 public   static   class   OwnerMappe  r
   extends   Mapper < PrimaryKey  Writable ,  RowWritabl  e ,  Text
 ,  MapWritabl  e > {
    @ Override   public   void   map ( PrimaryKey  Writable   key ,
 RowWritabl  e   row ,
        Context   context )  throws   IOExceptio  n ,  Interrupte
 dException  {
        PrimaryKey  Column   pet  =  key . getPrimary  Key ().
 getPrimary  KeyColumn (" name ");
        Column   owner  =  row . getRow (). getLatestC  olumn ("
 owner ");
        Column   species  =  row . getRow (). getLatestC  olumn ("
 species ");
        MapWritabl  e   m  =  new   MapWritabl  e ();
        m . put ( new   Text ( pet . getValue (). asString ()),
            new   Text ( species . getValue (). asString ()));
        context . write ( new   Text ( owner . getValue ().
 asString ()),  m );
    }
}
 public   static   class   IntoTableR  educer
   extends   Reducer < Text , MapWritabl  e , Text , BatchWrite
 Writable > {
    @ Override   public   void   reduce ( Text   owner ,  Iterable <
 MapWritabl  e >  pets ,
        Context   context )  throws   IOExceptio  n ,  Interrupte
 dException  {
        List < PrimaryKey  Column >  pkeyCols  =  new   ArrayList <
 PrimaryKey  Column >();
        pkeyCols . add ( new   PrimaryKey  Column (" owner ",
                PrimaryKey  Value . fromString ( owner . toString
 ())));
        PrimaryKey   pkey  =  new   PrimaryKey ( pkeyCols );
        List < Column >  attrs  =  new   ArrayList < Column >();
        for ( MapWritabl  e   petMap :  pets ) {
            for ( Map . Entry < Writable ,  Writable >  pet :
 petMap . entrySet ()) {
                Text   name  = ( Text )  pet . getKey ();
                Text   species  = ( Text )  pet . getValue ();
                attrs . add ( new   Column ( name . toString (),
```

```
                        ColumnValu  e . fromString ( species .
 toString ()));
             }
         }
          RowPutChan  ge   putRow  =  new   RowPutChan  ge (
 outputTabl  e ,  pkey )
             . addColumns ( attrs );
          BatchWrite  Writable   batch  =  new   BatchWrite  Writable
 ();
          batch . addRowChan  ge ( putRow );
          context . write ( owner ,  batch );
    }
}
 public   static   void   main ( String []  args )  throws
 Exception  {
     Configurat  ion   conf  =  new   Configurat  ion ();
     Job   job  =  Job . getInstanc  e ( conf ,  TableStore
 OutputForm  atExample . class . getName ());
     job . setMapperC  lass ( OwnerMappe  r . class );
     job . setReducer  Class ( IntoTableR  educer . class );
     job . setMapOutp  utKeyClass ( Text . class );
     job . setMapOutp  utValueCla  ss ( MapWritabl  e . class );
     job . setInputFo  rmatClass ( TableStore  InputForma  t . class
 );
     job . setOutputF  ormatClass ( TableStore  OutputForm  at .
 class );
     TableStore . setCredent  ial ( job ,  accessKeyI  d ,
 accessKeyS  ecret ,  securityTo  ken );
     TableStore . setEndpoin  t ( job ,  endpoint ,  instance );
     TableStore  InputForma  t . addCriteri  a ( job , ...) ;
     TableStore  OutputForm  at . setOutputT  able ( job ,
 outputTabl  e );
     System . exit ( job . waitForCom  pletion ( true ) ?  0  :  1
 );
}
```

Appendix

For the complete sample code, see:

- MR reads data from Table Store
- MR writes data to Table Store

5 Create an HBase cluster and use the HBase storage service

This topic describes how to create and configure an HBase cluster and use the HBase storage service.

To make better use of HBase, we recommend that you use the following configurat ions when creating a cluster.

- Enable access from the public network.
- Select the zone where the server of the application that accesses HBase is located. Do not randomly assign a zone.
- Select four or more hardware nodes, including the master and slave nodes. E-MapReduce creates NameNode, DataNode, JournalNode, HMaster, RegionServer, and ZooKeeper roles on these nodes.
- Select 4-core 16 GB and 8-core 32 GB for servers. Lower configurations may impair the stability of HBase clusters.
- Select the SSD disk type for the data disk for cost-effectiveness. You can select Basic Disk as the disk type for services that are not frequently used but require a large amount of storage space.
- Configure the data capacity as needed.
- HBase clusters support memory expansion.

Configure HBase

You can use the Software Configuration feature on the Create Cluster page when creating an HBase cluster. You can optimize and modify the default parameters of HBase based in specific scenarios. See the following example:

```
{
  " configurat  ions ": [
    {
      " classifica  tion ": " hbase - site ",
      " properties ": {
        " hbase . hregion . memstore . flush . size ": " 268435456 ",
        " hbase . regionserv  er . global . memstore . size ": " 0 . 5
 ",
        " hbase . regionserv  er . global . memstore . lowerLimit ": "
 0 . 6 "
      }
    }
  ]
```

```
}
```

Some default configurations for HBase clusters are as follows:

key	value
zookeeper.session.timeout	180000
hbase.regionserver.global.memstore.size	0.35
hbase.regionserver.global.memstore.lowerLimit	0.3
hbase.hregion.memstore.flush.size	128MB

Access HBase

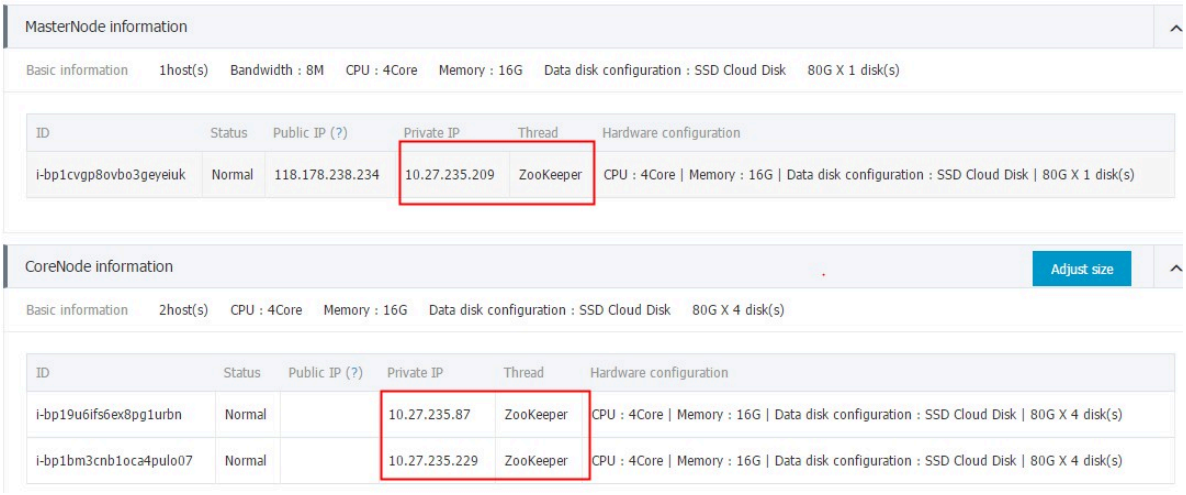
Notice:

- Considering network performance, we recommend that you initiate access requests to HBase clusters created by E-MapReduce from ECS instances in the same zone.
- The ECS instance accessing HBase clusters must be in the same security group as the HBase clusters. Otherwise, the access fails. Therefore, if you create a Hadoop, Spark, or Hive cluster in E-MapReduce, and the cluster you create needs to access HBase, then make sure that you select the same security group as the HBase cluster.

After you have created an HBase cluster in the E-MapReduce console, you can use the HBase storage service. The procedure is as follows:

1. Get the master IP address and the address of the ZooKeeper cluster. On the Cluster Overview page in the E-MapReduce console, you can view the IP addresses of the

master nodes and ZooKeeper access addresses (intranet IP addresses), as shown in the following figure.



For more information about master nodes with public IP addresses enabled, see How to log on to a master node and browse the HMaster WEB UI (*localhost : 16010*).

2. You can connect to the master node of a cluster over SSH using HBase Shell. You can use SSH to access the master node of a cluster. Switch to an HDFS account and use HBase Shell to access the clusters. (For more information about HBase Shell, see the Apache HBase website.)

```
[ root @ emr - header - 1  ~]#  su   hdfs
[ hadoop @ emr - header - 1   root ]$  hbase   shell
 HBase   Shell ;  enter  ' help < RETURN >'  for   a   list   of
 supported   commands .
 Type  " exit < RETURN >"  to   leave   the   HBase   Shell
 Version   1 . 1 . 1 ,  r374488 ,  Fri   Aug   21   09 : 18 : 22
 CST   2015
 hbase ( main ): 001 : 0 >
```

3. Access the cluster using HBase Shell from another ECS node (within the same security group). Download the HBase-1.x resource from the Apache HBase website (download link). Extract the resources, modify *conf / hbase - site . xml* , and then add the ZooKeeper address of the cluster as follows:

```
< configurat  ion >
     < property >
         < name > hbase . zookeeper . quorum </ name >
         < value >$ ZK_IP1 ,$ ZK_IP2 ,$ ZK_IP3 </ value >
     </ property >
```

```
</ configurat  ion >
```

Then you can access the cluster using the `bin / hbase  shell` command.

If the ECS instances are created using E-MapReduce, you only need to modify the *hbase - site . xml* file without downloading the HBase-1.x resource.

The configuration files of versions later than 3.2.0 are located in */ etc / ecm / hbase - conf / hbase - site . xml* .

The configuration files of versions earlier than 3.2.0 are located in */ etc / emr / hbase - conf / hbase - site . xml* .

4. To access the HBase cluster using APIs, add the following Maven dependency:

```
< groupId > org . apache . hbase </ groupId >
< artifactId > hbase - client </ artifactId >
< version > 1 . 1 . 1 </ version >
```

Configure the correct ZooKeeper address to connect to the cluster.

```
Configurat  ion  config  =  HBaseConfi  guration . create ();
config . set ( HConstants . ZOOKEEPER_  QUORUM ,"$ ZK_IP1 ,$ ZK_IP2
,$ ZK_IP3 ");
Connection  connection  =  Connection  Factory . createConn
ection ( config );
try  {
     Table  table  =  connection . getTable ( TableName . valueOf
(" myLittleHB  aseTable "));
     try  {
        // Do  table  operation
    } finally  {
         if  ( table  ! =  null )  table . close ();
    }
}  finally  {
     connection . close ();
}
```

For more information about the Apache HBase development, see the Apache HBase website.

Examples

- Prerequisites

  The ECS instance accessing the Hbase cluster must be in the same security group as the HBase cluster.

- Allow Spark to access HBase

  See spark-hbase-connector.

- Allow Hadoop to access HBase

  See HBase MapReduce Examples.

- Allow Hive to access HBase

  Only Hive running in clusters with a version of E-MapReduce 1.2.0 or later can access the HBase cluster. Follow these steps:

  1. Log on to the Hive cluster, and modify hosts by adding the following line:

     ```
     $ zk_ip   emr - cluster  //$ zk_ip   is   the   IP   address   of
        the   ZooKeeper   node   in   the   HBase   cluster .
     ```

  2. For more information about how to operate Hive, see Hive HBase Integration.

6 Back up HBase

The HBase cluster created in E-MapReduce allows you to use the snapshot feature integrated into HBase to back up HBase tables, and export the backup to OSS.

Example:

1. Create an HBase cluster

   For more information, see Create clusters.

2. Create a table

```
> create  ' test ',' cf '
```

3. Add data

```
>  put  ' test ',' a ',' cf : c1 ', 1
>  put  ' test ',' a ',' cf : c2 ', 2
>  put  ' test ',' b ',' cf : c1 ', 3
>  put  ' test ',' b ',' cf : c2 ', 4
>  put  ' test ',' c ',' cf : c1 ', 5
>  put  ' test ',' c ',' cf : c2 ', 6
```

4. Create a snapshot

```
hbase   snapshot   create  - n   test_snaps  hot  - t   test
```

   List snapshots

```
> list_snaps  hots
 SNAPSHOT                 TABLE  +  CREATION   TIME
  test_snaps  hot           test  ( Sun   Sep   04   20 : 31 : 00
+ 0800   2016 )
1   row ( s )  in   0 . 2080   seconds
```

5. Export the snapshot to OSS.

```
hbase   org . apache . hadoop . hbase . snapshot . ExportSnap  shot
 - snapshot   test_snaps  hot  - copy - to   oss ://$ accessKeyI
d :$ accessKeyS  ecret @$ bucket . oss - cn - hangzhou - internal .
aliyuncs . com / hbase / snapshot / test
```

   Note:

   Access OSS using internal endpoints.

6. Create another HBase cluster
7. Export snapshots from OSS

```
hbase   org . apache . hadoop . hbase . snapshot . ExportSnap  shot
 - snapshot   test_snaps  hot  - copy - from   oss ://$ accessKeyI
```

```
d :$ accessKeyS  ecret @$ bucket . oss - cn - hangzhou - internal .
aliyuncs . com / hbase / snapshot / test  - copy - to  / hbase /
```

8. Restore data from snapshots

```
> restore   _snapshot  ' test_snaps  hot '
```

```
> scan  ' test '
ROW                      COLUMN + CELL
 a                       column = cf : c1 ,  timestamp =
1472992081  375 ,  value = 1
 a                       column = cf : c2 ,  timestamp =
1472992090  434 ,  value = 2
 b                       column = cf : c1 ,  timestamp =
1472992104  339 ,  value = 3
 b                       column = cf : c2 ,  timestamp =
1472992099  611 ,  value = 4
 c                       column = cf : c1 ,  timestamp =
1472992112  657 ,  value = 5
 c                       column = cf : c2 ,  timestamp =
1472992118  964 ,  value = 6
3   row ( s )  in   0 . 0540   seconds
```

9. Create new tables from snapshots

```
> clone_snap  shot  ' test_snaps  hot ',' test_2 '
```

```
> scan  ' test_2 '
ROW                      COLUMN + CELL
 a                       column = cf : c1 ,  timestamp =
1472992081  375 ,  value = 1
 a                       column = cf : c2 ,  timestamp =
1472992090  434 ,  value = 2
 b                       column = cf : c1 ,  timestamp =
1472992104  339 ,  value = 3
 b                       column = cf : c2 ,  timestamp =
1472992099  611 ,  value = 4
 c                       column = cf : c1 ,  timestamp =
1472992112  657 ,  value = 5
 c                       column = cf : c2 ,  timestamp =
1472992118  964 ,  value = 6
3   row ( s )  in   0 . 0540   seconds
```

7 Guide for E-MapReduce cluster operations

This topic describes multiple operation methods for E-MapReduce clusters, which make it easier for you to independently perform operations on components.

Notice:

For later cluster versions (3.2 and later), all operations on components can be performed through the Cluster Management page in the console. We recommend that you manage components using web pages.

Some general environment variables

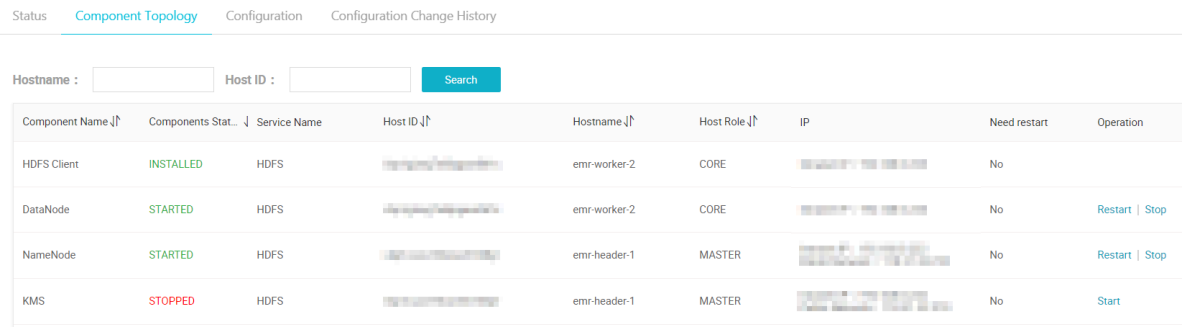
Enter the `env` command to see environment variable configurations that are similar to those in the following example. The actual configurations can be different. This example is for reference only.

```
JAVA_HOME =/ usr / lib / jvm / java
HADOOP_HOM  E =/ usr / lib / hadoop - current
HADOOP_CLA  SSPATH =/ usr / lib / hbase - current / lib /*:/ usr /
lib / tez - current /*:/ usr / lib / tez - current / lib /*:/ etc /
emr / tez - conf :/ usr / lib / hbase - current / lib /*:/ usr / lib
/ tez - current /*:/ usr / lib / tez - current / lib /*:/ etc / emr
/ tez - conf :/ opt / apps / extra - jars /*:/ opt / apps / extra -
jars /*
HADOOP_CON  F_DIR =/ etc / emr / hadoop - conf
SPARK_HOME =/ usr / lib / spark - current
SPARK_CONF  _DIR =/ etc / emr / spark - conf
HBASE_HOME =/ usr / lib / hbase - current
HBASE_CONF  _DIR =/ etc / emr / hbase - conf
HIVE_HOME =/ usr / lib / hive - current
HIVE_CONF_  DIR =/ etc / emr / hive - conf
PIG_HOME =/ usr / lib / pig - current
PIG_CONF_D  IR =/ etc / emr / pig - conf
TEZ_HOME =/ usr / lib / tez - current
TEZ_CONF_D  IR =/ etc / emr / tez - conf
ZEPPELIN_H  OME =/ usr / lib / zeppelin - current
ZEPPELIN_C  ONF_DIR =/ etc / emr / zeppelin - conf
HUE_HOME =/ usr / lib / hue - current
HUE_CONF_D  IR =/ etc / emr / hue - conf
PRESTO_HOM  E =/ usr / lib / presto - current
PRESTO_CON  F_DIR =/ etc / emr / presot - conf
```

Start and stop a component using the Web UI

You can use the Web UI of the E-MapReduce service to start, stop, and restart the component that runs on the specified ECS instance. Operations on different components are similar, for example the HDFS service. Follow these steps to start, stop, and restart the DataNode component for the emr-worker-1 instance.

1. On the Cluster list page, click Manage in the Actions column for the cluster to perform operations on.
2. On the Clusters and Services page, click the HDFS link to go to the HDFS service page.
3. Click the Component Topology tab to see the list of components that run on all instances in the cluster.
4. Click Start in the Actions column for the DataNode component that runs on the `emr-worker-1` instance. Enter the commit record in the dialog box and click OK. Refresh the page after 10 seconds. The value in the Components Status column for DataNode switches from STOPPED to STARTED.



5. After the component is started, click Restart in the Actions column. Enter the commit record in the dialog box and click OK.

   Refresh the page after 40 seconds. The value in the Components Status column for DataNode switches from STOPPED to STARTED.
6. Click Stop in the Actions column. Enter the commit record in the dialog box and click OK.

   Refresh the page after 10 seconds. The value of the Components Status column for DataNode switches from STARTED to STOPPED.

Perform bulk operations on components using the Web UI

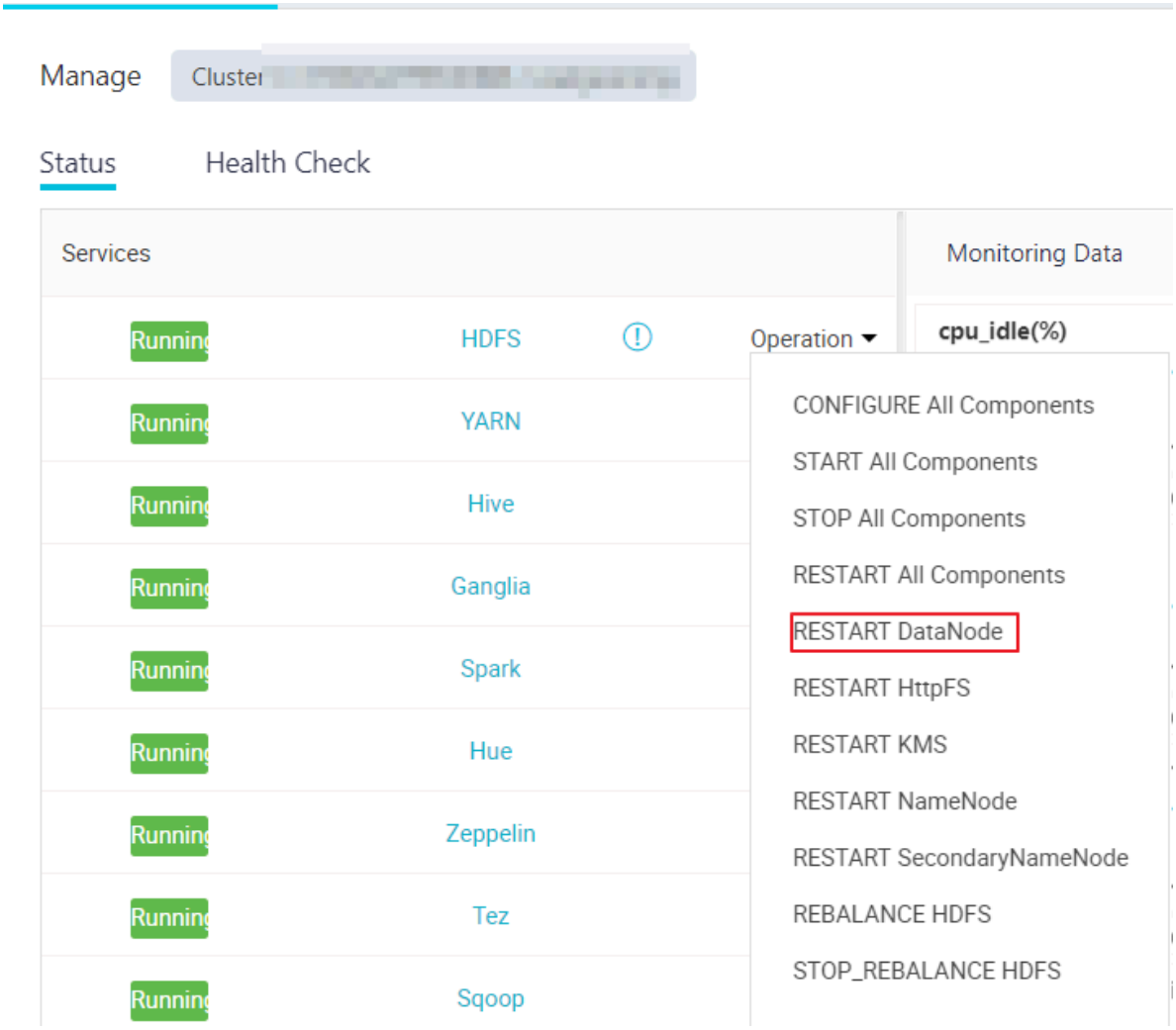
You can perform bulk operations on components for multiple ECS instances instead of a single specified ECS instance. The HDFS service as an example. Follow these steps to restart the DataNode components for all instances.

1. On the Cluster list page, click Manage in the Actions column for the cluster to perform operations on.
2. On the Clusters and Services page, click Actions for HDFS in the Services list.

3. Select RESTART DataNode from the Actions drop-down list. Enter the commit record in the dialog box and click OK.

   Click HDFS and click the Component Topology tab. View the status of each process on the Component Topology tab page.

   > **Notice:**
   > Console reports an error if you restart nodes manually after performing a rolling restart of the cluster.



Start and stop a component using CLI

- YARN

  Account: hadoop

  - ResourceManager (a master component)

    ```
    // Starts   a   component .
    ```

```
/ usr / lib / hadoop - current / sbin / yarn - daemon . sh   start
   resourcema  nager
// Stops   a   component .
/ usr / lib / hadoop - current / sbin / yarn - daemon . sh   stop
   resourcema  nager
```

- NodeManager (a core component)

```
// Starts   a   component .
/ usr / lib / hadoop - current / sbin / yarn - daemon . sh   start
   nodemanage  r
// Stops   a   component .
/ usr / lib / hadoop - current / sbin / yarn - daemon . sh   stop
   nodemanage  r
```

- JobHistoryServer (a master component)

```
// Starts   a   component .
/ usr / lib / hadoop - current / sbin / mr - jobhistory - daemon .
 sh   start   historyser  ver
// Stops   a   component .
/ usr / lib / hadoop - current / sbin / mr - jobhistory - daemon .
 sh   stop   historyser  ver
```

- JobHistoryServer (a master component)

```
// Starts   a   component .
/ usr / lib / hadoop - current / sbin / mr - jobhistory - daemon .
 sh   start   historyser  ver
// Stops   a   component .
/ usr / lib / hadoop - current / sbin / mr - jobhistory - daemon .
 sh   stop   historyser  ver
```

- WebProxyServer (a master component)

```
// Starts   a   component .
/ usr / lib / hadoop - current / sbin / yarn - daemon . sh   start
   proxyserve  r
// Stops   a   component .
/ usr / lib / hadoop - current / sbin / yarn - daemon . sh   stop
   proxyserve  r
```

· HDFS

Account: hdfs

- NameNode (a master component)

```
// Starts   a   component .
/ usr / lib / hadoop - current / sbin / hadoop - daemon . sh
 start   namenode
// Stops   a   component .
/ usr / lib / hadoop - current / sbin / hadoop - daemon . sh
 stop   namenode
```

- DataNode (a core component)

```
// Starts   a   component .
/ usr / lib / hadoop - current / sbin / hadoop - daemon . sh
 start   datanode
```

```
// Stops   a   component .
/ usr / lib / hadoop - current / sbin / hadoop - daemon . sh
 stop   datanode
```

- Hive

  Account: hadoop

  - MetaStore (a master component)

```
// Starts   a   component .  You   can   set   the   HADOOP_HEA
 PSIZE   environmen  t   variable   to   a   greater   value   for
   a   larger   maximum   memory .
 HADOOP_HEA  PSIZE = 512  / usr / lib / hive - current / bin /
 hive  -- service   metastore  >/ var / log / hive / metastore .
 log   2 >& 1  &
```

  - HiveServer2 (a master component)

```
// Starts   a   component .
 HADOOP_HEA  PSIZE = 512  / usr / lib / hive - current / bin
 / hive  -- service   hiveserver  2  >/ var / log / hive /
 hiveserver  2 . log   2 >& 1  &
```

- HBase

  Operational account: hdfs

  Note: The following example is based on the HBase service. An error occurs when you start a component without corresponding configurations.

  - HMaster (a master component)

```
// Starts   a   component .
/ usr / lib / hbase - current / bin / hbase - daemon . sh   start
   master
// Restarts   a   component .
/ usr / lib / hbase - current / bin / hbase - daemon . sh
 restart   master
// Stops   a   component .
/ usr / lib / hbase - current / bin / hbase - daemon . sh   stop
 master
```

  - HRegionServer (a core component)

```
// Starts   a   component .
/ usr / lib / hbase - current / bin / hbase - daemon . sh   start
   regionserv  er
// Restarts   a   component .
/ usr / lib / hbase - current / bin / hbase - daemon . sh
 restart   regionserv  er
// Stops   a   component .
/ usr / lib / hbase - current / bin / hbase - daemon . sh   stop
 regionserv  er
```

  - ThriftServer (a master component)

```
// Starts   a   component .
```

```
/ usr / lib / hbase - current / bin / hbase - daemon . sh   start
   thrift  - p   9099  >/ var / log / hive / thriftserv  er . log
   2 >& 1  &
// Stops   a   component .
/ usr / lib / hbase - current / bin / hbase - daemon . sh   stop
 thrift
```

- **Hue**

  Account: hadoop

```
// Starts   a   component .
 su  - l   root  - c  "${ HUE_HOME }/ build / env / bin / supervisor
  >/ dev / null   2 >& 1  &"
// Stops   a   component .
 ps   aux  |  grep   hue      // Lists   informatio  n   about   the
   currently   running   Hue   processes .
 kill  - 9   huepid         // Kills   the   Hue   process   by
 its   PID .
```

- **Zeppelin**

  Account: hadoop

```
// Starts   a   component .  You   can   set   the   HADOOP_HEA
 PSIZE   environmen  t   variable   to   a   greater   value   for
 a   larger   maximum   memory .
 su  - l   root  - c  " ZEPPELIN_M  EM =\"- Xmx512m  - Xms512m \" ${
 ZEPPELIN_H  OME }/ bin / zeppelin - daemon . sh   start "
// Stops   a   component .
 su  - l   root  - c  "${ ZEPPELIN_H  OME }/ bin / zeppelin - daemon
 . sh   stop "
```

- **Presto**

  Account: hadoop

  - **PrestoServer (a master component)**

```
// Starts   a   component .
/ usr / lib / presto - current / bin / launcher  -- config =/
 usr / lib / presto - current / etc / coordinato  r - config .
 properties   start
// Stops   a   component .
/ usr / lib / presto - current / bin / launcher  -- config =/
 usr / lib / presto - current / etc / coordinato  r - config .
 properties   stop
```

  - **(a core component)**

```
// Starts   a   component .
/ usr / lib / presto - current / bin / launcher  -- config =/ usr
 / lib / presto - current / etc / worker - config . properties
 start
// Stops   a   component .
```

```
/ usr / lib / presto - current / bin / launcher  -- config =/ usr
 / lib / presto - current / etc / worker - config . properties
 stop
```

Perform bulk operations on components using CLI

Write script commands to perform bulk operations for all worker (core) nodes. In an EMR cluster, all worker nodes that run in the hadoop account and the hdfs account are accessible from the master node using SSH.

For example, you can run the following commands to stop the NodeManager components for all worker nodes. Assume that the number of worker nodes is 10.

```
for   i   in  ` seq   1   10 `; do   ssh   emr - worker -$ i  /
usr / lib / hadoop - current / sbin / yarn - daemon . sh   stop
nodemanage  r ; done
```