Huawei Big Data Certification Training

HCIA-Big Data Lab Guide for Big Data Engineers

ISSUE:3.0



HUAWEI TECHNOLOGIES CO., LTD.

Copyright © Huawei Technologies Co., Ltd. 2020. All rights reserved.

No part of this document may be reproduced or transmitted in any form or by any means without prior written consent of Huawei Technologies Co., Ltd.

Trademarks and Permissions

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

Notice

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

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

Huawei Technologies Co., Ltd.

Address: Huawei Industrial Base Bantian, Longgang Shenzhen 518129 People's Republic of China

Website: http://e.huawei.com

Huawei Certificate System

Huawei Certification follows the "platform + ecosystem" development strategy, which is a new collaborative architecture of ICT infrastructure based on "Cloud-Pipe-Terminal". Huawei has set up a complete certification system consisting of three categories: ICT infrastructure certification, Platform and Service certification and ICT vertical certification, and grants Huawei certification the only all-range technical certification in the industry.

Huawei offers three levels of certification: Huawei Certified ICT Associate (HCIA), Huawei Certified ICT Professional (HCIP), and Huawei Certified ICT Expert (HCIE).

Huawei Certified ICT Associate-Big Data (HCIA-Big Data) is designed for train and certify engineers who are capable of using Huawei MRS big data development platform.

The HCIA-Big Data certificate system introduces you the technical principles and architectures of common and important big data components, and enables you to stand atop the Big Data frontiers.





Contents

1 About This Document	4
1.1 Introduction	4
1.2 Content Description	4
1.3 Precautions	4
1.4 References	4
1.5 MRS Architecture	5
2 HDFS Practice	6
2.1 Background	6
2.2 Objectives	6
2.3 Tasks	6
2.3.1 Task 1: Understating Common HDFS Commands	6
2.3.2 Task 2: Using the Recycle Bin	15
2.4 Summary	15
3 Hive Data Warehouse Practice	16
3.1 Background	16
3.2 Objectives	16
3.3 Tasks	16
3.3.1 Task 1: Creating Hive Tables	16
3.3.2 Task 3: Performing Basic Hive Queries	18
3.3.3 Task 3: Performing Hive Join Operations	22
3.3.4 Task 4: Using Hue to Execute HQL	29
3.4 Summary	
4 HBase Columnar Database Practice	35
4.1 Background	35
4.2 Objectives	35
4.3 Tasks	35
4.3.1 Task 1: Performing Common HBase Operations	35
4.3.2 Task 2: Pre-splitting Regions During Table Creation	41
4.3.3 Task 3: Using Filters	45
4.4 Summary	45
5 MapReduce Data Processing Practice	46
5.1 Background	46
5.2 Objectives	46
5.3 Tasks	46



Page	2
	_

5.3.1 Task 1: MapReduce Shell Practice	46
5.3.1 (Optional) Task 2: MapReduce Java Practice: Collecting Statistics on Online Duration	51
5.4 Summary	56
6 Spark Memory Computing Practice	57
6.1 Background	57
6.2 Objectives	57
6.3 Tasks	57
6.3.1 Task 1: Spark RDD Programming	57
6.3.2 Task 2: RDD Shell Operations	59
6.3.3 (Optional) Task 3: RDD Code Programming — Java Programming	66
6.3.4 Task 4: Spark SQL DataFrame Programming	69
6.3.5 Task 5: Spark SQL DataSet Programming	76
6.4 Summary	78
7 Flink Real-Time Processing System Practice	79
7.1 Background	79
7.2 Objectives	79
7.3 Tasks	79
7.3.1 Task 1: Importing a Flink Sample Project	79
7.3.2 Task 2: Exercising the Asynchronous CheckPoint Mechanism	80
7.3.3 Task 3: Obtaining Top N Hot-Selling Offerings in Flink in Real Time	89
7.4 Summary	96
8 Kafka Message Subscription Practice	97
8.1 Background	97
8.2 Objectives	97
8.3 Tasks	97
8.3.1 Task 1: Producing and Consuming Kafka Messages on the Shell Side	97
8.3.2 Task 2: Using Kafka Consumer Groups	99
8.4 Summary	104
9 Flume Data Collection Practice	105
9.1 Background	105
9.2 Objectives	105
9.3 Tasks	105
9.3.1 Task 1: Installing the Flume Client	105
9.3.2 Task 2: Using SpoolDir to Collect and Upload Data to HDFS	110
9.3.3 Task 3: Using SpoolDir to Collect and Upload Data to Kafka	120
9.4 Summary	123
10 Loader Data Import and Export Practice	124
10.1 Background	124



10.2 Objectives	
10.3 Tasks	
10.3.1 Task 1: Preparing MySQL Data	
10.3.2 Task 2: Configuring the MySQL Driver of Loader	
10.3.3 Task 3: Creating a Loader Link	
10.3.4 Task 4: Importing MySQL Data to HDFS	
10.3.5 Task 5: Importing MySQL Data to Hive	
10.3.6 Task 6: Importing HDFS Data to HBase	
10.4 Summary	
11 Comprehensive Exercise: Hive Data Warehouse	150
11.1 Background	
11.2 Objectives	
11.3 Tasks	
11.3.1 Preparing MySQL Data	
11.3.2 Importing MySQL Data to Hive	
11.3.3 Processing Hive Data	
11.3.4 Importing HDFS Data to HBase	
11.3.5 Querying Data in HBase in Real Time	
11.4 Summary	
12 Appendix: Environment Preparations and Commands	
12.1 (Optional) Preparing the Java Environment	
12.1.1 Installing JDK	
12.1.2 Installing Apache Maven	
12.1.3 Installing Eclipse	
12.1.4 Importing an MRS 2.0 Sample Project to Eclipse	
12.2 Binding an EIP to an ECS	
12.3 Viewing the IP address of ZooKeeper	
12.4 Viewing the IP Address of a Kafka Broker Instance	
12.5 Common Linux Commands	
12.6 HDFS Commands	
12.7 Yarn Application Operation Commands	



About This Document

1.1 Introduction

This document uses HUAWEI CLOUD MapReduce Service (MRS) as the exercise environment to guide trainees through related tasks and help them understand how to use big data components of MRS.

1.2 Content Description

This document consists of eight exercises and illustrates how to use important big data components.

The exercises include HUAWEI CLOUD MRS application practice, HDFS practice, Loader data import and export practice, Flume data collection practice, Kafka message subscription practice, Hive data warehouse practice, HBase database practice, and comprehensive cluster practice.

1.3 Precautions

A HUAWEI CLOUD account and real-name authentication are required.

It is recommended that each trainee uses one exercise environment so that they do not affect each other.

1.4 References

To obtain the MapReduce help documents, visit https://support.huaweicloud.com/intl/enus/mrs/index.html.



1.5 MRS Architecture



Figure 1-1



2 HDFS Practice

2.1 Background

HDFS is the basis of big data components. Hive data, MapReduce and Spark computing data, and regions of HBase are all stored in HDFS. On the HDFS shell client, you can perform various operations, such as uploading, downloading, and deleting data, and managing file systems. Learning HDFS will help you better understand and master big data knowledge.

2.2 Objectives

- Understand command HDFS operations.
- Understand HDFS management operations.

2.3 Tasks

2.3.1 Task 1: Understating Common HDFS Commands

Run the following command to set environment variables before running commands to operate the MRS components:

source /opt/client/bigdata_env

Step 1 Run the **help** command.

This command is used to view the command help document.

hdfs dfs -help



```
[root@node-master1duzY ~]# hdfs dfs -help
Usage: hadoop fs [generic options]
        [-appendToFile <localsrc> ... <dst>]
        [-cat [-ignoreCrc] <src> ...]
        [-checksum <src> ...]
        [-chgrp [-R] GROUP PATH...]
        [-chmod [-R] <MODE[,MODE]... | OCTALMODE> PATH...]
        [-chown [-R] [OWNER][:[GROUP]] PATH...]
         [-copyFromLocal [-f] [-p] [-l] [-d] [-t <thread count>] <localsrc> ... <dst>]
        [-copyToLocal [-f] [-p] [-ignoreCrc] [-crc] <src> ... <localdst>]
        [-count [-q] [-h] [-v] [-t [<storage type>]] [-u] [-x] [-e] <path> ...]
        [-cp [-f] [-p | -p[topax]] [-d] <src> ... <dst>]
[-createSnapshot <snapshotDir> [<snapshotName>]]
        [-deleteSnapshot <snapshotDir> <snapshotName>]
        [-df [-h] [<path> ...]]
        [-du [-s] [-h] [-v] [-x] <path> ...]
        [-expunge]
        [-find <path> ... <expression> ...]
        [-get [-f] [-p] [-ignoreCrc] [-crc] <src> ... <localdst>]
         [-getfacl [-R] <path>]
        [-getfattr [-R] {-n name | -d} [-e en] <path>]
         [-getmerge [-nl] [-skip-empty-file] <src> <localdst>]
        [-head <file>]
```

```
Figure 2-1
```

Check how to use the **ls** command.

```
hdfs dfs -help ls
```

```
[root@node-master1duzY ~]# hdfs dfs -help ls
-ls [-C] [-d] [-h] [-q] [-R] [-t] [-S] [-r] [-u] [-e] [<path> ...] :
List the contents that match the specified file pattern. If path is not
specified, the contents of /user/<currentUser> will be listed. For a directory a
list of its direct children is returned (unless -d option is specified).
Directory entries are of the form:
    permissions - userId groupId sizeOfDirectory(in bytes)
modificationDate(yyyy-MM-dd HH:mm) directoryName
and file entries are of the form:
    permissions numberOfReplicas userId groupId sizeOfFile(in bytes)
modificationDate(yyyy-MM-dd HH:mm) fileName
    -C Display the paths of files and directories only.
```

Figure 2-2

Step 2 Run the **ls** command.

This command is used to display the directory information.

hdfs dfs -ls /



[root@node-m	aster1HaCu	u ~]# hdfs	dfs -ls /
2020-04-06 1	7:13:35,8	10 INFO obs	s.OBSFileSystem: This Filesystem GC-ful, clear resource.
Found 11 ite	ms		
drwxrwxrwx	- hdfs	hadoop	0 2020-04-06 13 : 29 /app-logs
drwxrwxrwx	- hive	hive	0 2020-04-06 13:36 /apps
drwxrwxrwx	- hdfs	hadoop	0 2020-04-06 13:29 /ats
drwxr-xr-x	- hdfs	hadoop	0 2020-04-06 13:29 /datasets
drwxr-xr-x	- hdfs	hadoop	0 2020-04-06 13:29 /datastore
drwxrwxrwx	- flink	hadoop	0 2020-04-06 13:31 /flink
drwxr-xr-x	- hbase	hadoop	0 2020-04-06 13:33 /hbase
drwxrwxrwx	- mapred	hadoop	0 2020-04-06 13:29 /mr-history
drwxrwxrwt	- spark	hadoop	0 2020-04-06 13:43 /sparkJobHistory
drwxrwxrwx	- hdfs	hadoop	0 2020-04-06 13:42 /tmp
drwxrwxrwx	- hdfs	hadoop	0 2020-04-06 13:39 /user
[root@node-m	aster1HaCi	u ~l#	

Figure 2-3

Step 3 Run the **mkdir** command.

This command is used to create directories in HDFS.

To create the **stu01** folder in the **user** folder of the **root** directory, view the content in the **user** folder, create the folder, and then run the **ls** command. The **stu01** folder is displayed.

hdfs dfs -mkdir /user/stu01

```
[root@node-master1HaCu ~]# hdfs dfs
                                    -mkdir /user/stu01
2020-04-06 17:24:58,832 INFO obs.OBSFileSystem: This Filesystem GC-ful, clear resource.
[root@node-master1HaCu ~]# hdfs dfs -ls /user
2020-04-06 17:26:16,824 INFO obs.OBSFileSystem: This Filesystem GC-ful, clear resource.
Found 6 items
drwxrwxrwx
           - hive
                     hive
                                     0 2020-04-06 13:29 /user/hive
            - loader hadoop
                                    0 2020-04-06 13:35 /user/loader
drwxr-x---
drwxr-xr-x - mapred hadoop
                                    0 2020-04-06 13:29 /user/mapred
                                     0 2020-04-06 13:43 /user/omm
drwxrwxrwx
            - omm
                     hadoop
           - omm
                                     0 2020-04-06 13:39 /user/spark
drwxrwxrwx
                     hadoop
                    hadoop
drwxr-xr-x - root
                                     0 2020-04-06 17:24 /user/stu01
[root@node-master1HaCu ~]#
```



Step 4 Run the **put** command.

This command is used to upload a file in the Linux system to a specified HDFS directory. Before executing this command, run the following command to edit a file in a local Linux host:

vi stu01.txt

[root@node-master1duzY ~]# vi stu01.txt

Figure 2-5

Press i to enter the editing mode, enter the content, and press **Esc**. Then, press **Shift** and a colon (:), and enter **wq** to save the settings and exit. The following is a file content example:



HCIA-Big Data V3.0 Lab Guide

🛃 root@node-master1HaCu:~
1234
5678
hadoop
~

Figure 2-6

Run the hdfs dfs -put stu01.txt /user/stu01/ command to upload the file.

```
[root@node-master1HaCu ~]# hdfs dfs -put stu01.txt /user/stu01/
2020-04-06 17:41:22,430 INFO obs.OBSFileSystem: This Filesystem GC-ful, clear resource.
[root@node-master1HaCu ~]# hdfs dfs -ls /user/stu01
2020-04-06 17:43:16,663 INFO obs.OBSFileSystem: This Filesystem GC-ful, clear resource.
Found 1 items
-rw-r--r- 1 root hadoop 17 2020-04-06 17:41 /user/stu01/stu01.txt
[root@node-master1HaCu ~]# []
```

Figure 2-7

Run the **ls** command to check whether the **stu01.txt** file has been uploaded to the **/user/stu01** directory.

Step 5 Run the cat command.

This command is used to display the file content.

```
hdfs dfs -cat /user/stu01/stu01.txt
```

```
[root@node-master1duzY ~]# hdfs dfs -cat /user/stu01/stu01.txt
1234
5678
hadoop
[root@node-master1duzY ~]#
```

```
Figure 2-8
```

Step 6 Run the **text** command.

This command is used to show the content of a file in character format.

hdfs dfs -text /user/stu01/stu01.txt

```
[root@node-master1duzY ~]# hdfs dfs -text /user/stu01/stu01.txt
1234
5678
hadoop
[root@node-master1duzY ~]#
```

Figure 2-9

Step 7 Run the **moveFromLocal** command.

This command is used to cut and paste data from the local PC to HDFS.



Run the **vi** command to create a data file, for example, **stu01_2.txt**, on a local Linux host. The following figure shows the content of the file.



Figure 2-10

Run the following command:

hdfs dfs -moveFromLocal stu01_2.txt /user/stu01/

```
[root@node-master1HaCu ~]# vi stu01_2.txt
[root@node-master1HaCu ~]# hdfs dfs -moveFromLocal stu01_2.txt /user/stu01/
2020-04-06 17:57:53,914 INFO obs.OBSFileSystem: This Filesystem GC-ful, clear resource.
[root@node-master1HaCu ~]# hdfs dfs -ls /user/stu01
2020-04-06 17:57:59,608 INFO obs.OBSFileSystem: This Filesystem GC-ful, clear resource.
Found 2 items
-rw-r-r-- 1 root hadoop 17 2020-04-06 17:41 /user/stu01/stu01.txt
-rw-r-r-- 1 root hadoop 12 2020-04-06 17:57 /user/stu01/stu01_2.txt
```

Figure 2-11

The **stu01_2.txt** file has been uploaded to the **/user/stu01** directory on the HDFS. Run the **ls** command to check the Linux local host. The **stu01_2.txt** file does not exist, indicating that the file is cut and pasted to the destination HDFS directory. Comparatively, after the **put** command is executed, the local file is only copied to the HDFS and still exists on the Linux host.

Step 8 Run the appendToFile command.

This command is used to add a file to the end of an existing file.

Run the **vi** command to edit data file **stu01_3.txt** on a local Linux host. The file content is as follows:

[root@node-master1duzY ~]# vi stu01_3.txt
[root@node-master1duzY ~]# cat stu01_3.txt
www.huawei.com

Figure 2-12

Run the following command:

hdfs dfs -appendToFile stu01_3.txt /user/stu01/stu01_2.txt

Add the content of the **stu01_3.txt** file to the **stu01_2.txt** file in the HDFS.

Run the **cat** command to view the result. The following information is displayed:



```
[root@node-master1HaCu ~]# vi stu01_3.txt
[root@node-master1HaCu ~]# hdfs dfs -appendToFile stu01_3.txt /user/stu01/stu01_2.txt
2020-04-06 18:01:17,023 INFO obs.OBSFileSystem: This Filesystem GC-ful, clear resource.
[root@node-master1HaCu ~]# hdfs dfs -cat /user/stu01/stu01_2.txt
2020-04-06 18:01:29,846 INFO obs.OBSFileSystem: This Filesystem GC-ful, clear resource.
hadoop
hive
www.huawei.com
[root@node-master1HaCu ~]#
```

Figure 2-13

Step 9 Run the **cp** command.

This command is used to copy a file from one HDFS path to another HDFS path.

Run the **vi** command to edit the **stu01_4.txt** file on a local Linux host, and run the **put** command to upload the file to the HDFS **root** directory, as shown in the following figure:

```
[root@node-master1duzY ~]# vi stu01_4.txt
[root@node-master1duzY ~]# hdfs dfs -put stu01_4.txt /
[root@node-master1duzY ~]#
```

Figure 2-14

```
Run the hdfs dfs -cp /stu01_4.txt /user/stu01/ command.
```

[root@node-master1HaCu ~]# vi stu01 4.txt [root@node-master1HaCu ~]# hdfs dfs -put stu01_4.txt 2020-04-06 18:03:35,115 INFO obs.OBSFileSystem: This Filesystem GC-ful, clear resource. [root@node-master1HaCu ~]# hdfs dfs -cp /stu01_4.txt /user/stu01/ 2020-04-06 18:03:42,513 INFO obs.OBSFileSystem: This Filesystem GC-ful, clear resource. [root@node-master1HaCu ~] # hdfs dfs -ls /user/stu01 2020-04-06 18:03:57,522 INFO obs.OBSFileSystem: This Filesystem GC-ful, clear resource. Found 3 items 17 2020-04-06 17:41 /user/stu01/stu01.txt -rw-r--r-- 1 root hadoop -rw-r--r-- 1 root hadoop -rw-r--r-- 1 root ' 27 2020-04-06 18:01 /user/stu01/stu01_2.txt 18 2020-04-06 18:03 /user/stu01/stu01 4.txt [root@node-master1HaCu ~]# hdfs dfs -ls / 2020-04-06 18:04:09,980 INFO obs.OBSFileSystem: This Filesystem GC-ful, clear resource. Found 12 items 0 2020-04-06 13:29 /app-logs drwxrwxrwx - hdfs hadoop 0 2020-04-06 13:36 /apps drwxrwxrwx - hive hive – hdfs hadoop – hdfs hadoop 0 2020-04-06 13:29 /ats drwxrwxrwx 0 2020-04-06 13:29 /datasets drwxr-xr-x drwxr-xr-x - hdfs hadoop 0 2020-04-06 13:29 /datastore drwxrwxrwx - flink hadoop drwxr-xr-x - hbase hadoop 0 2020-04-06 13:31 /flink 0 2020-04-06 13:33 /hbase drwxrwxrwx - mapred hadoop 0 2020-04-06 13:29 /mr-history drwxrwxrwt - spark drwxrwxrwt - spark 0 2020-04-06 13:43 /sparkJobHistory – spark hadoop 18 2020-04-06 18:03 /stu01_4.txt ficommon drwxrwxrwx - hdfs 0 2020-04-06 13:42 /tmp hadoop - hdfs hadoop 0 2020-04-06 17:24 /user drwxrwxrwx [root@node-master1HaCu ~]#

Figure 2-15

The stu01_4.txt file exists in the /user/stu01 directory and the root directory.

Step 10 Run the **mv** command.

This command is used to move files in the HDFS directory.

Run the **vi** command to edit the **stu01_5.txt** file on the local Linux host, and run the **put** command to upload the file to the HDFS **root** directory, as shown in the following figure:



[root@node-m	aster1dCr0	C ~]# vi st	u01_5.txt				
[root@node-m	aster1dCr0	C ∼]# hdfs	dfs -put :	stu01_5.txt	/		
2020-04-06 1	9:53:54,80	53 INFO obs	.OBSFileS	ystem: This	Filesy	stem GC-ful, cl	ear resource.
[root@node-m	aster1dCr0	C ∼]# hdfs	dfs -ls /				
2020-04-06 1	9:54:05,13	32 INFO obs	.OBSFileS	ystem: This	Filesy	stem GC-ful, cl	ear resource.
Found 12 ite	ms						
drwxrwxrwx	- hdfs	hadoop	0	2020-04-06	18:24	/app-logs	
drwxrwxrwx	- hive	hive	0	2020-04-06	18:27	/apps	
drwxrwxrwx	- hdfs	hadoop	0	2020-04-06	18:24	/ats	
drwxr-xr-x	- hdfs	hadoop	0	2020-04-06	18:24	/datasets	
drwxr-xr-x	- hdfs	hadoop	0	2020-04-06	18:24	/datastore	
drwxrwxrwx	- flink	hadoop	0	2020-04-06	18:25	/flink	
drwxr-xr-x	- hbase	hadoop	0	2020-04-06	18:26	/hbase	
drwxrwxrwx	- mapred	hadoop	0	2020-04-06	18:24	/mr-history	
drwxrwxrwt	 spark 	hadoop	0	2020-04-06	18:29	/sparkJobHistor	У
-rw-rr	1 root	ficommon	21	2020-04-06	19:53	/stu01 5.txt	
drwxrwxrwx	- hdfs	hadoop	0	2020-04-06	18:28	/tmp	
drwxrwxrwx	- hdfs	hadoop	0	2020-04-06	18:28	/user	

Figure 2-16

Run the hdfs dfs -mv /stu01_5.txt /user/stu01/ command.

[root@node-	-master1dCr(C ∼]# hdfs dfs	-n	w /stu01 !	5.txt	/user/stu01/
2020-04-06	19:58:05,82	22 INFO obs.OBS	File	System: Th	is File	esystem GC-ful, clear resource.
[root@node-	-master1dCr	C ∼]# hdfs dfs	-ls	/user/stu0	1	
2020-04-06	19:58:19,43	34 INFO obs.OBS	File	System: Th:	is File	esystem GC-ful, clear resource.
Found 1 ite	ems					
-rw-rr	1 root fi	icommon	21	2020-04-06	19:53	/user/stu01/stu01 5.txt
[root@node-	-master1dCr(C ∼]# hdfs dfs	-ls	/		
2020-04-06	19:58:28,69	93 INFO obs.OBS	File	System: Th:	is File	esystem GC-ful, clear resource.
Found 11 it	cems					
drwxrwxrwx	- hdfs	hadoop	0	2020-04-06	18:24	/app-logs
drwxrwxrwx	- hive	hive	0	2020-04-06	18:27	/apps
drwxrwxrwx	- hdfs	hadoop	0	2020-04-06	18:24	/ats
drwxr-xr-x	- hdfs	hadoop	0	2020-04-06	18:24	/datasets
drwxr-xr-x	- hdfs	hadoop	0	2020-04-06	18:24	/datastore
drwxrwxrwx	- flink	hadoop	0	2020-04-06	18:25	/flink
drwxr-xr-x	– hbase	hadoop	0	2020-04-06	18:26	/hbase
drwxrwxrwx	- mapred	hadoop	0	2020-04-06	18:24	/mr-history
drwxrwxrwt	- spark	hadoop	0	2020-04-06	18:29	/sparkJobHistory
drwxrwxrwx	- hdfs	hadoop	0	2020-04-06	18:28	/tmp
drwxrwxrwx	- hdfs	hadoop	0	2020-04-06	19:56	/user
[root@node-	-master1dCr(2~]#				

Figure 2-17

The **stu01_5.txt** file exists in the **/user/stu01** folder, but it has been removed from the **root** directory.

Step 11 Run the **get** command.

Similar to copyToLocal, this command is used to download files from the HDFS to a local host.

Delete the **stu01_5.txt** file from the Linux host.

```
[root@node-master1duzY ~]# ls
env_file stu01_2.txt stu01_3.txt stu01_4.txt stu01_5.txt stu01.txt
[root@node-master1duzY ~]# rm -rf stu01_5.txt
[root@node-master1duzY ~]# ls
env_file stu01_2.txt stu01_3.txt stu01_4.txt stu01.txt
[root@node-master1duzY ~]#
```

Figure 2-18

Run the hdfs dfs -copyToLocal /user/stu01/stu01_5.txt command.



```
[root@node-master1duzY ~]# hdfs dfs -copyToLocal /user/stu01/stu01_5.txt
[root@node-master1duzY ~]# ls
env_file stu01_2.txt stu01_3.txt stu01_4.txt stu01_5.txt stu01.txt
[root@node-master1duzY ~]#
```

Figure 2-19

The **stu01_5.txt** file exists on the Linux host.

Note the period at the end of the HDFS command, which indicates the current directory. If you specify another directory, you can specify the path for saving the file.

Step 12 Run the **getmerge** command.

This command is used to download a combination of multiple files.

Run the **ls** to view the files in the **/user/stu01/** directory.

```
[root@node-master1dCrC ~]# hdfs dfs -ls /user/stu01
2020-04-06 20:00:29,828 INFO obs.OBSFileSystem: This Filesystem GC-ful, clear resource.
Found 3 items
-rw-r--r- 1 root hadoop 16 2020-04-06 20:00 /user/stu01/stu01_3.txt
-rw-r--r- 1 root hadoop 12 2020-04-06 20:00 /user/stu01/stu01_4.txt
-rw-r--r- 1 root ficommon 21 2020-04-06 19:53 /user/stu01/stu01_5.txt
[root@node-master1dCrC ~]#
```

Figure 2-20

Run the hdfs dfs -getmerge /user/stu01/* ./merge.txt command.

```
[root@node-master1dCrC ~]# hdfs dfs -getmerge /user/stu01/* ./merge.txt
2020-04-06 20:01:08,357 INFO obs.OBSFileSystem: This Filesystem GC-ful, clear resource.
[root@node-master1dCrC ~]# ls
env_file merge.txt stu01_3.txt stu01_4.txt stu01_5.txt
[root@node-master1dCrC ~]# cat merge.txt
www.huawei.com
hive
hadoop
41324124
41341241234
[root@node-master1dCrC ~]#
```

Figure 2-21

The **merge.txt** file is generated in the current directory. The content in the file is a combination of the files in **/user/stu01/**.

Step 13 Run the **rm** command.

This command is used to delete an HDFS file or folder.

Run the hdfs dfs -rm /user/stu01/stu01_5.txt command.



[[root@node-	-masteridere ~]# ndi	s dis -1s /user/studi
2020-04-06	20:02:10,508 INFO c	bs.OBSFileSystem: This Filesystem GC-ful, clear resource.
Found 3 ite	ems	
-rw-rr	1 root hadoop	16 2020-04-06 20:00 /user/stu01/stu01_3.txt
-rw-rr	1 root hadoop	12 2020-04-06 20:00 /user/stu01/stu01_4.txt
-rw-rr	1 root ficommon	21 2020-04-06 19:53 /user/stu01/stu01_5.txt
[root@node-	-master1dCrC ~]# hdf	s dfs -rm /user/stu01/stu01_5.txt
2020-04-06	20:02:24,735 INFO o	bs.OBSFileSystem: This Filesystem GC-ful, clear resource.
2020-04-06	20:02:25,159 INFO f	s.TrashPolicyDefault: Moved: 'hdfs://hacluster/user/stu01/stu01_5.txt' to
trash at:	hdfs://hacluster/us	er/root/.Trash/Current/user/stu01/stu01 5.txt
[root@node-	-master1dCrC ~]# hdf	s dfs -ls /user/stu01
2020-04-06	20:02:33,592 INFO c	bs.OBSFileSystem: This Filesystem GC-ful, clear resource.
Found 2 ite	ems	
-rw-rr	1 root hadoop	16 2020-04-06 20:00 /user/stu01/stu01 3.txt
-rw-rr	1 root hadoop	12 2020-04-06 20:00 /user/stu01/stu01 4.txt
F		_

Figure 2-22

The stu01_5.txt file does not exist in /user/stu01/.

Step 14 Run the **df** command.

This command is used to collect information on the available space of a file system.

Run the **hdfs dfs -df -h** / command.

[root@node-master1duzY ~]# hdfs dfs -df -h / Filesystem Size Used Available Use% hdfs://hacluster 98.3 G 627.9 M 91.8 G 1% [root@node-master1duzY ~]#

Figure 2-23

Step 15 Run the **du** command.

This command is used to collect information on the folder size.

Run the hdfs dfs -du -s -h /user/stu01 command.

```
[root@node-master1duzY ~]# hdfs dfs -ls /user/stu01
Found 3 items
-rw-r--r-- 1 root hive 17 2019-03-25 13:47 /user/stu01/stu01.txt
-rw-r--r-- 1 root hive 27 2019-03-25 14:07 /user/stu01/stu01_2.txt
-rw-r--r-- 1 root hive 21 2019-03-25 15:59 /user/stu01/stu01_4.txt
[root@node-master1duzY ~]# hdfs dfs -du -s -h /user/stu01
[root@node-master1duzY ~]#
```

Figure 2-24

Step 16 Run the **count** command.

This command is used to collect information on the number of file nodes in a specified directory.

Run the hdfs dfs -count -v /user/stu01 command.

```
[root@node-masterlduzY ~]# hdfs dfs -count -v /user/stu01
DIR_COUNT FILE_COUNT CONTENT_SIZE PATHNAME
1 3 65 /user/stu01
[root@node-masterlduzY ~]#
```

```
Figure 2-25
```



2.3.2 Task 2: Using the Recycle Bin

Files may be deleted by mistake in daily work. In this case, you can find the deleted files in the recycle bin of HDFS. By default, the deleted files are retained in the recycle bin for seven days. For example, after the **/user/stu01/stu01_5.txt** file is deleted, the **stu01_5.txt** file is moved to the recycle bin

Run the hdfs dfs -ls /user/root/.Trash/Current/user/stu01/ command.

The **stu01_5.txt** file in the recycle bin is displayed.



Figure 2-26

Note that deleted data is retained for seven days by default.

Run the mv command to move the file back to the /user/stu01/ directory.

hdfs dfs -mv /user/root/.Trash/Current/user/stu01/stu01_5.txt /user/stu01

```
[root@node-master1dCrC ~]# hdfs dfs -mv /user/root/.Trash/Current/user/stu01/stu01_5.txt /user/stu01
2020-04-06 20:04:47,513 INFO obs.OBSFileSystem: This Filesystem GC-ful, clear resource.
[root@node-master1dCrC ~]# hdfs dfs -ls /user/stu01
2020-04-06 20:04:59,144 INFO obs.OBSFileSystem: This Filesystem GC-ful, clear resource.
Found 3 items
-rw-r--r- 1 root hadoop 16 2020-04-06 20:00 /user/stu01/stu01_3.txt
-rw-r--r- 1 root hadoop 12 2020-04-06 20:00 /user/stu01/stu01_4.txt
-rw-r--r- 1 root ficommon 21 2020-04-06 19:53 /user/stu01/stu01_5.txt
```



2.4 Summary

This exercise mainly describes common operations on HDFS. After completing this exercise, you will be able to perform common HDFS operations.



3 Hive Data Warehouse Practice

3.1 Background

Hive is a data warehouse tool and plays an important role in data mining, data aggregation, and statistics analysis. In telecom services, Hive can be used to collect statistics on users' data usage and phone bills, and mine user consumption models to help carriers better design packages.

3.2 Objectives

- Understand common Hive operations.
- Learn how to run HQL on Hue.

3.3 Tasks

3.3.1 Task 1: Creating Hive Tables

3.3.1.1 Viewing Statements for Creating Tables

CREATE [EXTERNAL] TABLE [IF NOT EXISTS] table_name [(col_name data_type [COMMENT col_comment], ...)] [COMMENT table_comment] [PARTITIONED BY (col_name data_type [COMMENT col_comment], ...)] [CLUSTERED BY (col_name, col_name, ...) [SORTED BY (col_name [ASC|DESC], ...)] INTO num_buckets BUCKETS] [ROW FORMAT row_format] [STORED AS file_format] [LOCATION hdfs_path]

3.3.1.2 Creating Database Tables

Set environment variable using **source /opt/client/bigdata_env**.

Enter **beeline** and press **Enter** to go to Hive.

Note that all statements in Hive must end with a semicolon (;). Otherwise, the statements cannot be executed.

Reference: Run the following command to filter the output of INFO logs:

beeline --hiveconf hive.server2.logging.operation.level=NONE



[root@node-master1dCrC ~]# beeline Connecting to jdbc:hive2://192.168.0.195:2181/;serviceDiscoveryMode=zooKeeper;zooKeeperN amespace=hiveserver2 Connected to: Apache Hive (version 3.1.0-mrs-2.0) Driver: Hive JDBC (version 3.1.0-mrs-2.0) Transaction isolation: TRANSACTION_REPEATABLE_READ Beeline version 3.1.0-mrs-2.0 by Apache Hive 0: jdbc:hive2://192.168.0.195:2181/> []

Figure 3-1

Statement for creating database tables (If multiple users share the same environment, it is recommended that the name of each table contain the first letters of the user's last and first names to differentiate tables.)

create table **cx_**stu01(name string,gender string ,age int) row format delimited fields terminated by ',' stored as textfile;

No rows selected (0.057 seconds)
0: jdbc:hive2://192.168.0.195:2181/> create table cx_stu01(name string,gender string ,age int) row fo
No rows affected (0.087 seconds)
0: jdbc:hive2://192.168.0.195:2181/> show tables;
+------+
| tab_name |
+-----+
| cx_stu01 |
+----+
1 row selected (0.034 seconds)
0: jdbc:hive2://192.168.0.195:2181/>

Figure 3-2

The **show** tables command is used to display all tables.

3.3.1.3 Creating Foreign Tables

Run the following command:

create external table cx_stu02(name string,gender string ,age int) row format delimited fields terminated by ',' stored as textfile ;

1 row selected (0.034 seconds)
0: jdbc:hive2://192.168.0.195:2181/> create external table cx_stu02(name string,gender string, age int)
row format delimited fields terminated by ',' stored as textfile ;
No rows affected (0.079 seconds)
0: jdbc:hive2://192.168.0.195:2181/> show tables;
+-----+
| tab_name |
+----+
| cx_stu01 |
| cx_stu02 |
+----+
2 rows selected (0.035 seconds)
0: jdbc:hive2://192.168.0.195:2181/>

Figure 3-3

3.3.1.4 Loading HDFS Data

Press **Ctrl+C** to exit Hive (or open a new shell window), and edit the **cx_stu01.txt** file on the local Linux host. The file content is as follows:



[root@node-master1dCrC	~]#	vi cx_stu01.txt
[root@node-master1dCrC	~]#	cat cx_stu01.txt
tom,male,19		
hanmeimei,female,20		
jack,female,22		
lilei,female,18		
Lily,male,23		
[root@node-master1dCrC	~]#	

Figure 3-4

Run the following **put** command to upload data to the **/user/stu01/** directory of the HDFS:

hdfs dfs -put cx_stu01.txt /user/stu01/

```
[root@node-master1dCrC ~]# hdfs dfs -put cx_stu01.txt /user/stu01/
2020-04-06 20:18:38,196 INFO obs.OBSFileSystem: This Filesystem GC-ful, clear resource.
[root@node-master1dCrC ~]# hdfs dfs -ls /user/stu01/
2020-04-06 20:18:56,431 INFO obs.OBSFileSystem: This Filesystem GC-ful, clear resource.
Found 4 items
-rw-r-r-- 1 root hadoop 76 2020-04-06 20:18 /user/stu01/cx_stu01.txt
-rw-r-r-- 1 root hadoop 16 2020-04-06 20:00 /user/stu01/stu01_3.txt
-rw-r-r-- 1 root hadoop 12 2020-04-06 20:00 /user/stu01/stu01_4.txt
-rw-r-r-- 1 root ficommon 21 2020-04-06 19:53 /user/stu01/stu01_5.txt
```

Figure 3-5

Run the **beeline** command to go to Hive and run the following command to load data and import data to the foreign table:

	load data i	npath '/user/stu0)1/cx_stu01.txt'	into table cx_st	u02;	
0 0 N 0	: jdbc:hive2://1 2; o rows affected : jdbc:hive2://1	92.168.0.195:2181/> (0.35 seconds) 92.168.0.195:2181/>	load data inpose	ath '/user/stu01/cx	_stu01.txt' into t	able cx_stu
+	cx_stu02.name	cx_stu02.gender	+ cx_stu02.age	+ -		
	tom hanmeimei jack lilei Lily	male female female female male	19 20 22 18 23			
5 0	rows selected (0.276 seconds) 92.168.0.195:2181/>				

Figure 3-6

3.3.2 Task 3: Performing Basic Hive Queries

3.3.2.1 Fuzzy Queries

Run the **show tables like 'cx_stu*'**; statement.



```
0: jdbc:hive2://192.168.0.195:2181/> show tables like 'cx_stu*';
+-----+
| tab_name |
+-----+
| cx_stu01 |
| cx_stu02 |
+-----+
2 rows selected (0.05 seconds)
0: jdbc:hive2://192.168.0.195:2181/>
```

Figure 3-7

3.3.2.2 Simple Queries

Step 1 Run the following Limit statement:

select * from cx_stu02 limit 2;

0: jdbc:hive2://192.168.0.195:2181/> select * from cx_stu02 limit 2; +-----+ | cx_stu02.name | cx_stu02.gender | cx_stu02.age | +----+ | tom | male | 19 | | hanmeimei | female | 20 | +----+ 2 rows selected (0.064 seconds) 0: jdbc:hive2://192.168.0.195:2181/> []

Figure 3-8

Step 2 Run the following Where statement:

select * from cx_stu02 where gender ='male' limit 2;

0: jdbc:hive2://19	92.168.0.195:2181/>	select *	from	cx_stu02	where	gender ='male'	limit	2;
+	+	+	+					
+			ige i					
tom	male	19	Í					
Lily	male	23	1					
+	+	+	+					
0: idbc:hive2://19	02.168.0.195:2181/>							
J: Jube:hivez://is	92.100.0.195:2101/>							

Figure 3-9

Step 3 Run the following Order statement:

select * from cx_stu02 where gender ='female' order by age limit 2;





3.3.2.3 Complex Queries

Step 1 Use the **vi** editor to edit the **cx_stu03.txt** file on the local Linux host. The file content is as follows:



Figure 3-11

Step 2 Upload data to the HDFS.

Run the hdfs dfs -put cx_stu03.txt /user/stu01/ command.

```
[root@node-master1dCrC ~]# hdfs dfs -put cx_stu03.txt
                                                            /user/stu01/
2020-04-06 20:50:55,148 INFO obs.OBSFileSystem: This Filesystem GC-ful, clear resource.
[root@node-master1dCrC ~]# hdfs dfs -ls
                                            /user/stu01/
2020-04-06 20:51:04,395 INFO obs.OBSFileSystem: This Filesystem GC-ful, clear resource.
Found 4 items
-rw-r--r--
             1 root hadoop
                                     183 2020-04-06 20:50 /user/stu01/cx stu03.txt
-rw-r--r--
                                     16 2020-04-06 20:00 /user/stu01/stu01 3.txt
            1 root hadoop
           1 root hadoop
1 root ficommo<u>n</u>
-rw-r--r--
                                      12 2020-04-06 20:00 /user/stu01/stu01_4.txt
-rw-r--r--
                                      21 2020-04-06 19:53 /user/stu01/stu01_5.txt
[root@node-master1dCrC ~]#
```

Figure 3-12

Step 3 Create a table and import data to the table.

Run the **beeline** command to go to Hive and enter the following table creation statement:

create external table cx_table_stu03(id int,name string ,subject string,score float) row format delimited fields terminated by ',' stored as textfile ;

Run the following statement to import data:

load data inpath '/user/stu01/cx_stu03.txt' into table cx_table_stu03;



0: jdbc:hive2://192.1 tring,score float) No rows affected (0.0 0: jdbc:hive2://192.1 ble_stu03; No rows affected (0.2 0: jdbc:hive2://192.1	68.0.195:2181/> create e row format delimited fie 55 seconds) 68.0.195:2181/> load da 71 seconds) 68.0.195:2181/> select	external table cx_table_sti elds terminated by ',' sto: ata inpath '/user/stu01/o * from cx_table_stu03;	u03(id int,name string ,subje red as textfile ; cx_stu03.txt' into table c	ct s x_ta
cx_table_stu03.id	cx_table_stu03.name	cx_table_stu03.subject	cx_table_stu03.score	
1001	 Jack	Chinese	78.0	
1002	Jack	English	82.0	
1003	Jack	Math	87.0	
1004	Mark	Chinese	69.0	
1005	Mark	English	89.0	
1006	Mark	Math	73.0	
1007	Hanke	Chinese	89.0	
1008	Hanke	English	85.0	
1009	Hanke	Math	75.0	
+ 9 rows selected (0.11 0: jdbc:hive2://192.1	+ 6 seconds) 68.0.195:2181/>		++	

Figure 3-13

Step 4 Perform the sum operation.

To calculate the total score of each student, run the following statement:

select name ,sum(score) total_score from cx_table_stu03 group by name ;

```
0: jdbc:hive2://192.168.0.195:2181/> select name ,sum(score) total_score from cx_table_stu03 grou

p by name;

------+

| name | total_score |

+-----+

| Hanke | 249.0 |

| Jack | 247.0 |

| Mark | 231.0 |

+-----+

3 rows selected (5.453 seconds)

0: jdbc:hive2://192.168.0.195:2181/>
```

Figure 3-14

To calculate the total score of each student and filter out students whose total score is greater than 230, run the following statement:

select name ,sum(score) total_score from cx_table_stu03 group by name having total_score > 235;

```
0: jdbc:hive2://192.168.0.195:2181/> select name ,sum(score) total_score from cx_table_stu03 grou

p by name having total_score > 235;

+------+

| name | total_score |

+-----+

| Hanke | 249.0 |

| Jack | 247.0 |

+-----+

2 rows selected (5.429 seconds)

0: jdbc:hive2://192.168.0.195:2181/>
```

Figure 3-15

Step 5 Perform the max operation.

To view the highest score of each course, run the following statement:

select subject,max(score) from cx_table_stu03 group by subject;



Figure 3-16

Step 6 Perform the count operation.

To calculate the number of trainees taking the exam of each course, run the following statement:

select subject,count(1) from cx_table_stu03 group by subject;

0: jdbc:hive2://192.168.0.195:2181/> select subject,count(1) from cx_table_stu03 group by subject; +-----+ | subject | _c1 | +-----+ | chinese | 3 | | English | 3 | | Math | 3 | +------+ 3 rows selected (4.649 seconds) 0: jdbc:hive2://192.168.0.195:2181/>

Figure 3-17

3.3.3 Task 3: Performing Hive Join Operations

Hive supports common SQL join statements, such as INNER JOIN, LEFT OUTER JOIN, RIGHT OUTER JOIN, and map-side JOIN.

Step 1 Create a table and import data to the table.

Create three tables: **cx_table_employee** (employee table), **cx_table_department** (department table), and **cx_table_salary** (salary table). Import data to the three tables. For details about how to import data, see the previous content.

Statements for creating **cx_table_employee**:

create table if not exists cx_table_employee(
 user_id int,
 username string,
 dept_id int)
 row format delimited fields terminated by ','
 stored as textfile;

```
0: jdbc:hive2://192.168.0.195:2181/> create table if not exists cx_table_employee(user_id int,username
string,dept_id int) row format delimited fields terminated by ' ' lines terminated by '\n';
No rows affected (0.052 seconds)
0: jdbc:hive2://192.168.0.195:2181/>
```

Figure 3-18

Statements for creating cx_table_department:

```
create table if not exists cx_table_department(
  dept_id int,
```



Page 23

dept_name string) row format delimited fields terminated by ',' stored as textfile ;

```
0: jdbc:hive2://192.168.0.195:2181/> create table if not exists cx_table_department(dept_id int,dept_na
me string) row format delimited fields terminated by ' ' lines terminated by '\n';
No rows affected (0.053 seconds)
0: jdbc:hive2://192.168.0.195:2181/>
```

Figure 3-19

Statements for creating **cx_table_salary**:

create table if not exists cx_table_salary(
userid int,
dept_id int,
salarys double)
row format delimited fields terminated by ','
stored as textfile ;

```
0: jdbc:hive2://192.168.0.195:2181/> create table if not exists cx_table_salary(userid int,dept_id int, salarys double) row format delimited fields terminated by ' ' lines terminated by '\n';
No rows affected (0.051 seconds)
0: jdbc:hive2://192.168.0.195:2181/>
```

Figure 3-20

The data in the three tables is as follows:

cx_table_ employee (employee table):

1,zhangsas,1

2,lisi,2

3,wangwu,3

4,tom,1

5,lily,2

6,amy,3

7,lilei,1

8,hanmeimei,2

9,poly,3

cx_table_department (department table):

1,Technical

2,sales

3,HR

4, marketing

cx_table_salary (salary table):

1,1,20000

2,2,16000

3,3,20000

4,1,50000



5,2,18900 6,3,12098 7,1,21900

Step 2 Perform INNER JOIN.

When INNER JOIN join is performed on multiple tables, only the data that matches the on condition in all tables is displayed. For example, the following SQL statement implements the join between the employee table and the department table. The on condition is dept_id. Only data with the same dept_id is matched and displayed.

Run the following statement:

select e.username,e.dept_id,d.dept_name,d.dept_id from cx_table_employee e join cx_table_department d on e.dept_id = d.dept_id;

0: jdbc:hive2: ployee e join	<pre>//192.168.0. cx_table_dep</pre>	195:2181/> seled artment d on e.d	ct e.username dept_id = d.d
+	+	-+	-+
e.username	e.dept_id	d.dept_name	d.dept_id
zhangsas	1	Technical	1
lisi	2	sales	2
wangwu	3	HR	3
tom	1	Technical	1
lily	2	sales	2
amy	3	HR	3
lilei	1	Technical	1
hanmeimei	2	sales	2
poly	3	HR	3
+	+	-+	-+

Figure 3-21

You can join two or more tables. Run the following statement to query the employee names, departments, and salaries:

select e.username,d.dept_name,s.salarys from cx_table_employee e join cx_table_department d on e.dept_id = d.dept_id join cx_table_salary s on e.user_id = s.userid;

0: jdbc:hive2: able_departmen	//192.168.0.19 t d on e.dept_	5:2181/> se id = d.dept	lect e.username,d.dept_name,s.salarys from cx_table_employee e join cx_t _id join cx_table_salary s on e.user_id = s.userid;
e.username	d.dept_name	s.salary	+ 3 +
tom	Technical	50000.0	
lilei	Technical	21900.0	
amy	HR	12098.0	
zhangsas	Technical	20000.0	
lily	sales	18900.0	
lisī	sales	16000.0	
wangwu	HR	20000.0	
+	+	-+	+
7 rows selecte	d (30.69 secon	ds)	
0: jdbc:hive2:	//192.168.0.19	5:2181/>	

Figure 3-22

Generally, a MapReduce job is generated for a join. If more than two tables are joined, Hive associates the tables from left to right. For the preceding SQL statement, a MapReduce job is started to connect the employee and department tables, and then the second MapReduce job is started to connect the output of the first MapReduce job to the salary table. This is contrary to the standard SQL, which performs the join operation from



Page 25

right to left. Therefore, in Hive SQL, small tables are written on the left to improve the execution efficiency.

Hive supports the **/*+STREAMTALBE*/** syntax to specify which table is a large table. For example, in the following SQL statement, **dept** is specified as a large table. If the **/+STREAMTALBE/** syntax is not used, Hive considers the rightmost table as a large table.

Run the following statement:

select /*+STREAMTABLE(d)*/ e.username,e.dept_id,d.dept_name,d.dept_id from cx_table_employee e
join cx_table_department d on e.dept_id = d.dept_id;

e.username	e.dept_id	d.dept_name	d.dept_id	
zhangsas	1	Technical	1	
lisi	2	sales	2	
wangwu	3	HR	3	
tom	1	Technical	1	
lily	2	sales	2	
amy	3	HR	3	
lilei	1	Technical	1	
hanmeimei	2	sales	2	
poly	3	HR	3	

Figure 3-23

Generally, the number of MapReduce jobs to be started is the same as the number of tables to be joined. However, if the join keys of the on condition are the same, only one MapReduce job is started.

Step 3 Perform LEFT OUTER JOIN.

LEFT OUTER JOIN, same as the standard SQL statement, uses the left table as a baseline. If the right table matches the on condition, the data is displayed. Otherwise, NULL is displayed.

Run the following statement:

select e.user_id,e.username,s.salarys from cx_table_employee e left outer join cx_table_salary s on e.user_id = s.userid;

oin cx_table	_salary s on e +	user_id = s.	userid; /
e.user_id	e.username	s.salarys	1
1	zhangsas	20000.0	T
2	lisi	16000.0	
3	wangwu	20000.0	
4	tom	50000.0	
5	lily	18900.0	
6	amy	12098.0	
7	lilei	21900.0	
8	hanmeimei	NULL	1
9	poly	NULL	

Figure 3-24



As shown in the preceding figure, all records in the employee table on the left are displayed, and the data that meets the on condition in the salary table on the right is displayed. The data that does not meet the on condition is displayed as NULL.

Step 4 Perform RIGHT OUTER JOIN.

LEFT OUTER JOIN is opposite to LEFT OUTER JOIN. It uses the table on the right as a baseline. If the table on the left matches the on condition, the data is displayed. Otherwise, NULL is displayed.

Hive is a component for processing big data. It is often used to process hundreds of GB or even TB-level data. Therefore, you are advised to use the where condition to filter out data that does not meet the condition when compiling SQL statements. However, for LEFT and RIGHT OUTER JOINs, the where condition is executed after the on condition is executed. Therefore, to optimize the Hive SQL execution efficiency, use subqueries in scenarios where OUTER JOINs are required and use the where condition to filter out data that does not meet the conditions in the subqueries.

Run the following statement:

select e1.user_id,e1.username,s.salarys from (select e.* from cx_table_employee e where e.user_id < 8) e1 left outer join cx_table_salary s on e1.user_id = s.userid;

0: jdbc:hive2: loyee e where	//192.168.0.195 e.user_id < 8) (:2181/> selec el left outer	t el.user_id,el.username,s.salarys from (select e.* from cx_table_emp ; join cx_table_salary s on el.user_id = s.userid;
e1.user_id	el.username	s.salarys	+ +
1	zhangsas	20000.0	
2	lisi	16000.0	
3	wangwu	20000.0	
4	tom	50000.0	
5	lily	18900.0	
6	amy	12098.0	
7	lilei	21900.0	
+	+	+	+
7 rows selected	d (6.117 second	5)	
0: jdbc:hive2:	//192.168.0.195	:2181/>	

Figure 3-25

In the preceding SQL statement, the data whose **user_id** is greater than or equal to **8** is filtered out in the subquery.

Step 5 Perform FULL OUTER JOIN.

FULL OUTER JOIN returns all the data that meets the where condition in the table. The data that does not meet the where condition is replaced with NULL.

Run the following statement:

select e.user_id,e.username,s.salarys from cx_table_employee e full outer join cx_table_salary s on e.user_id = s.userid where e.user_id > 0;



0: jdbc:hive2: join cx_table_	://192.168.0.19 _salary s on e	95:2181/> se .user_id = s	<pre>lect e.user_id,e.username,s.salarys from cx_table_employee e full outer .userid where e.user_id > 0; </pre>
e.user_id	e.username	s.salarys	
1	zhangsas	20000.0	
3	1151 wangwu	20000.0	
4 5	tom lily	50000.0 18900.0	
16	amy lilei	12098.0 21900.0	
8	hanmeimei	NULL	
++	+		-+
<pre>9 rows selecte 0: jdbc:hive2:</pre>	ed (7.032 secon ://192.168.0.19	nds) 95:2181/> 📕	

Figure 3-26

The results of FULL OUTER JOIN and LEFT OUTER JOIN are the same.

Step 6 Perform LEFT SEMI JOIN.

LEFT SEMI JOIN is used to query only the data that meets the requirements of the left table.

Run the following statement:

select e.* from cx_table_employee e LEFT SEMI JOIN cx_table_salary s on e.user_id=s.userid;



Figure 3-27

LEFT SEMI JOIN is evolved from INNER JOIN. When a data record in the left table exists in the right table, Hive stops scanning. Therefore, the efficiency is higher than that of INNER JOIN. However, only the fields in the left table can be displayed behind the select and where keywords in the LEFT SEMI JOIN. Hive does not support RIGHT SEMI JOIN.

Step 7 Perform CARTESIAN JOIN.

The result of Cartesian product join is to multiply the data in the left table by the data in the right table.

Run the following statement:

select e.user_id,e.username,s.salarys from cx_table_employee e join cx_table_salary s;



HCIA-Big Data V3.0 Lab Guide

0: jdbc:hive2 le_salary s;	://192.168.0.1	<pre>>>:2181/> select e.user_id,e.username,s.salarys from cx_table_employee e join cx_tab</pre>	
e.user_id	e.username	s.salarys	
1	zhangsas	2000.0	
1	zhangsas	21900.0	
1	zhangsas	12098.0	
1	zhangsas	18900.0	
1	zhangsas	5000.0	
1	zhangsas	2000.0	
1	zhangsas	16000.0	
2	lisi	2000.0	
2	lisi	21900.0	
2	lisi	12098.0	
2	lisi	18900.0	
2	lisi	5000.0	
2	lisi	2000.0	
2	lisi	16000.0	
3	wangwu	2000.0	
3	wangwu	21900.0	
3	wangwu	12098.0	
3	wangwu	18900.0	
3	wangwu	50000.0	
3	wangwu	2000.0	
3	wangwu	16000.0	
4	tom	20000.0	
4	tom	21900.0	
4	tom	12098.0	
4	tom		
4	tom	50000.0	
4	tom		
4	tom		
5	111Y		
5	1 111Y		
15	1 1117		
15	1 1117 1 1117		
15	1 111V		
15	1 111V		
16	I ⊤⊤⊤∑ I ⊃mu		
16	l cruth	21900.0	
	l cruth		
	∣ amy		
6		50000	
16	l amy		
16	l cruth		
17	l anny I liloi		
17	l liloi	21900.0	
17	l liloi		
1 1	I TITET	12090.0	_

Figure 3-28

The execution result of the preceding SQL statement is the number of records in the employee table multiplied by the number of records in the salary table.

Step 8 Perform map-side JOIN.

Map-side JOIN is an optimization of Hive SQL. Hive converts SQL statements into MapReduce jobs. Therefore, the map-side JOIN corresponds to the map-side JOIN in the Hadoop Join. Small tables are loaded to the memory to improve the Hive SQL execution speed. You can use either of the following methods to use map-side JOIN of Hive SQL. The first method is to use **/*+ MAPJOIN*/**:

Run the following statement:

select /*+ MAPJOIN(d)*/ e.username,e.dept_id,d.dept_name,d.dept_id from cx_table_employee e join
cx_table_department d on e.dept_id = d.dept_id;



able_employee	//192.168.0.1 e e join cx_t	able_department	d on e.dept_	(a) / e.username,e.aept_1a,a.aept_name,a.aept_1a from id = d.dept_id;
e.username	e.dept_id	d.dept_name	d.dept_id	
zhangsas	1	Technical	1	
lisi	2	sales	2	1
wangwu	3	HR	3	
tom	1	Technical	1	
lily	2	sales	2	
amy	3	HR	3	
lilei	1	Technical	1	
hanmeimei	2	sales	2	
vloq	3	HR	3	

Figure 3-29

The second one is to set hive.auto,convert.JOIN to true.

3.3.4 Task 4: Using Hue to Execute HQL

Step 1 Log in to MRS Manager.

On the **Services** page, click **Hue**. On the displayed page, click **Hue (Active)**. The Hue page is displayed.

Da	ashboard	Services	Hosts Alarms	
Pe	riod Real ti	me 🔻	🖸 View 📑 Export	
	Service Sum	mary	_ ^ĸ	ы
			4	
	HDFS	Sood 🕑	Zkfc: 1 DataNode:	
	Hive	< Good	MetaStore: 1 Web	
E	Hue	🕑 Good	Hue: 1	
	Kafka	Sood	Broker: 1 MirrorM	
	KrbServer	Sood	KerberosServer: 1	-



Service Hue > Service Status	
Service Status Instance	Service Configuration Resou
• Start Service • Store	op Service 🛃 Download Client
Hue Summary	
Health Status	< Good
Configuration Status	Synchronized
Version	3.11.0
Hue Web UI	Hue (Active)

Figure 3-31

Click the **Query Editor** and select **Hive**.

H)Ue 🖀	Query Editor 🗸	Data Browsers 🗸
🖗 Hive 🛛 🕯	Hive	cription
< 🛢 default	Mala Impala	
Tables	(6) Q Z	
⊞cx_stu01		1
⊞cx_stu02		
⊞ cx_table_departme	nt	
⊞ cx_table_employee		
⊞cx_table_salary		
⊞ cx_table_stu03		
		Query History Q Saved Queries Q

Figure 3-32





Figure 3-33

Step 2 Compile HQL.

Edit the HQL statement in the blank area.

select *,row_number() over(order by totalscore desc) rank from (select name,sum(score) totalscore
from cx_table_stu03 group by name) a;

	Query Editors 🗸	Data Browsers 🗸
🖗 Hive	Add a name Ad	dd a description
🔇 🛢 default	<	
Tables	(3) Q 🔁	
⊞cx_stu01		<pre>1 select *, 2 row_number() over(order by totalscore desc) rank</pre>
⊞cx_table_stu15		3 from 4 (select name,
⊞ stu15		<pre>5 sum(score) totalscore 6 from stu15 7 group by name) a; 1 -</pre>
		Query History Q Saved Queries Q
		You don't have any saved query.

Figure 3-34



H)Ue 🕯	Query Editors 🗸	Data Browsers 🗸
🖗 Hive	Add a name Ad	ld a description
🔇 🛢 default	<	
Tables	(3) Q 2	
⊞cx_stu01		<pre>1 select *, 2 row_number() over(order by totalscore desc) rank</pre>
⊞cx_table_stu15		3 from 4 (select name.
⊞stu15		<pre>5 sum(score) totalscore 6 from cx_table_stu03 7 group by name) a; 10 -</pre>
		Query History Q Saved Queries Q
		You don't have any saved query.

Figure 3-35

Step 3 Query data.

Click the triangle button to execute HQL.

tien €	Query Editors 🗸	Data Browsers 🗸
🖗 Hive	Add a name Add a	a description
 ✓ ■ default Tables 	(3) Q C EXCUTE HOL	<pre>1 select *, 2 row_number() over(order by totalscore desc) rank 3 from 4 (select name, 5 sum(score) totalscore 6 from stu15 7 group by name) a;</pre>
		Query History Q Saved Queries Q
		You don't have any saved query.

Figure 3-36


	Query Editors 🗸	Data Browsers 🗸
🖗 Hive	Add a name Ad	dd a description
C € default Tables ⊞ cx_stu01 ⊞ cx_table_stu15 ⊞ stu15	(3) Q 2	<pre>1 select *, 2</pre>
		Query History Q Saved Queries Q You don't have any saved query.

Figure 3-37

Step 4 View the result.



Figure 3-38



HUE A	Query Editors 🗸	Data Browsers 🗸
🖗 Hive	Add a name Ad	ld a description
🕻 🛢 default	<	
Tables ⊞ cx_stu01 ⊞ cx_table_stu15 ⊞ stu15	(3) Q B	<pre>1 select *, 2</pre>
		Query History Q Saved Queries Q You don't have any saved query.

Figure 3-39

3.4 Summary

This exercise describes the add, delete, modify, and query operations of the Hive data warehouse and introduces multiple join methods to help trainees understand the join types and differences. This exercise aims to help trainees better understand and use Hive.



4 HBase Columnar Database Practice

4.1 Background

The HBase database is an important big data component and is the most commonly used NoSQL database in the industry. Banks can store new customer information in HBase and update or delete out-of-date data in HBase.

4.2 Objectives

• Understand common HBase operations, region operations, and filter usage.

4.3 Tasks

4.3.1 Task 1: Performing Common HBase Operations

Run the **source /opt/client/bigdata_env** command to set environment variables. Run the **hbase shell** command to access the HBase shell client.

[root@node-master1duzY ~]# source /opt/client/bigdata_env
[root@node-master1duzY ~]# hbase shell

Figure 4-1

4.3.1.1 Creating Common Tables

Run the create 'cx_table_stu01' , 'cf1' command.

```
hbase(main):011:0> create 'cx_table_stu01', 'cf1'
2020-04-07 09:51:31,723 INFO [main] client.HBaseAdmin: Operation: CREATE, Table Name: default:cx_t
able_stu01, procId: 15 completed
Created table cx_table_stu01
Took 2.2636 seconds
=> Hbase::Table - cx_table_stu01
hbase(main):012:0> list
TABLE
cx_table_stu01
1 row(s)
Took 0.0112 seconds
=> ["cx_table_stu01"]
hbase(main):013:0>
```

Figure 4-2

list: displays all tables.



4.3.1.2 Adding Data

Run the following commands:

```
      put
      'cx_table_stu01','20200001','cf1:name','tom'

      put
      'cx_table_stu01','20200001', 'cf1:gender','male'

      put
      'cx_table_stu01','20200001', 'cf1:age','20'

      put
      'cx_table_stu01','20200002', 'cf1:name','hanmeimei'

      put
      'cx_table_stu01','20200002', 'cf1:name','hanmeimei'

      put
      'cx_table_stu01','20200002', 'cf1:gender','female'

      put
      'cx_table_stu01','20200002', 'cf1:gender','female'
```

```
hbase(main):021:0> put
                      'cx table stu01','20200001','cf1:name','tom'
Took 0.0120 seconds
                      'cx_table_stu01','20200001','cf1:gender','male'
hbase(main):022:0> put
Took 0.0105 seconds
hbase(main):023:0> put
                      'cx_table_stu01','20200001', 'cf1:age','20'
Took 0.0110 seconds
hbase(main):024:0> put 'cx table stu01','20200002', 'cf1:name','hanmeimei'
Took 0.0053 seconds
hbase(main):025:0> put 'cx table stu01','20200002', 'cf1:gender','female'
Took 0.0044 seconds
Took 0.0074 seconds
hbase(main):027:0> scan 'cx table stu01'
                        COLUMN+CELL
ROW
20200001
                        column=cf1:age, timestamp=1586224622452, value=20
20200001
                        column=cf1:gender, timestamp=1586224622397, value=male
                        column=cf1:name, timestamp=1586224622340, value=tom
20200001
                        column=cf1:age, timestamp=1586224622561, value=19
20200002
                        column=cf1:gender, timestamp=1586224622531, value=female
20200002
                        column=cf1:name, timestamp=1586224622492, value=hanmeimei
20200002
2 row(s)
Took 0.0142 seconds
hbase(main):028:0>
```

Figure 4-3

4.3.1.3 Querying Data in Scan Mode

Run the following commands:

```
scan 'cx_table_stu01',{COLUMNS=>'cf1'} #Queries only the data in the cf1 column family.
scan 'cx_table_stu01',{COLUMNS=>'cf1:name'} #Queries only the name information in the cf1
column family.
```

```
hbase(main):034:0> scan 'cx table stu01',{COLUMNS=>'cf1'}
ROW
                            COLUMN+CELL
20200001
                            column=cf1:age, timestamp=1586224622452, value=20
20200001
                            column=cf1:gender, timestamp=1586224622397, value=male
                            column=cf1:name, timestamp=1586224622340, value=tom
column=cf1:age, timestamp=1586224622561, value=19
 20200001
20200002
20200002
                            column=cf1:gender, timestamp=1586224622531, value=female
20200002
                            column=cf1:name, timestamp=1586224622492, value=hanmeimei
2 row(s)
Took 0.0190 seconds
hbase(main):035:0> scan 'cx table stu01',{COLUMNS=>'cf1:name'}
ROW
                            COLUMN+CELL
20200001
                            column=cf1:name, timestamp=1586224622340, value=tom
20200002
                            column=cf1:name, timestamp=1586224622492, value=hanmeimei
2 row(s)
Took 0.0116 seconds
hbase(main):036:0>
```



4.3.1.4 Querying Data in Get Mode

In Get mode, data is queried based on the row key.

Run the following commands:

```
get 'cx_table_stu01','20200001'
get 'cx_table_stu01','20200001','cf1:name'
```

hbase(main):038:0> get COLUMN	'cx_table_stu01','20200001' CELL
cf1:age	timestamp=1586224622452, value=20
cf1:gender	timestamp=1586224622397, value=male
cf1:name	timestamp=1586224622340, value=tom
1 row(s)	
Took 0.0095 seconds	
hbase(main):039:0> get	'cx_table_stu01','20200001','cf1:name'
COLUMN	CELL
cf1:name	timestamp=1586224622340, value=tom
1 row(s)	
Took 0.0082 seconds	
hbase(main)•040•05	

Figure 4-5

4.3.1.5 Querying Data by Specified Criteria

Run the following commands:

scan 'cx_table_stu01',{STARTROW=>'20200001','LIMIT'=>2,STOPROW=>'20200002'}
scan 'cx_table_stu01',{STARTROW=>'20200001','LIMIT'=>2,COLUMNS=>'cf1:name'}

```
hbase(main):007:0> scan 'cx_table_stu01',{STARTROW=>'20200001','LIMIT'=>2,STOPROW=>'20200002'}
ROW
                               COLUMN+CELL
 20200001
                               column=cf1:age, timestamp=1586224622452, value=20
 20200001
                               column=cf1:gender, timestamp=1586224622397, value=male
                               column=cf1:name, timestamp=1586224622340, value=tom
 20200001
1 row(s)
Took 0.0250 seconds
hbase(main):008:0> scan 'cx_table_stu01',{STARTROW=>'20200001','LIMIT'=>2,COLUMNS=>'cf1:name'}
ROW
                               COLUMN+CELL
                               column=cf1:name, timestamp=1586224622340, value=tom
column=cf1:name, timestamp=1586224622492, value=hanmeimei
 20200001
20200002
2 row(s)
Took 0.0086 seconds
hbase(main):009:0>
```

Figure 4-6

Note: In addition to column (COLUMNS) modifiers, HBase supports Limit (limiting the number of rows in the query results) and STARTROW (ROWKEY start row. The system locates the region based on the key and then scans the region backwards.), STOPROW (end row), TIMERANGE (timestamp range), VERSIONS (the number of versions), and FILTER (filtering rows based on conditions).

4.3.1.6 Querying Multiversion Data

HBase can store data of historical versions. You can set **VERSIONS** to specify the number of versions to be stored.

Add data.

put 'cx_table_stu01','20200001','cf1:name','ZhangSan' put 'cx_table_stu01','20200001','cf1:name','LiSi'



put 'cx_table_stu01','20200001','cf1:name','WangWu'

Scan the table to view the result.

hbase(main):001:0> put	<pre>'cx_table_stu01','20200001','cf1:name','ZhangSan'</pre>
Took 0.3839 seconds	
hbase(main):002:0> put	: 'cx_table_stu01','20200001','cf1:name','LiSi'
Took 0.0028 seconds	
hbase(main):003:0> put	: 'cx_table_stu01','20200001','cf1:name','WangWu'
Took 0.0030 seconds	
hbase(main):004:0> sca	an 'cx_table_stu01'
ROW	COLUMN+CELL
20200001	column=cf1:age, timestamp=1586224622452, value=20
20200001	column=cf1:gender, timestamp=1586224622397, value=male
20200001	column=cf1:name, timestamp=1586239656833, value=WangWu
20200002	column=cf1:age, timestamp=1586224622561, value=19
20200002	column=cf1:gender, timestamp=1586224622531, value=female
20200002	<pre>column=cf1:name, timestamp=1586224622492, value=hanmeimei</pre>
2 row(s)	
Took 0.0189 seconds	
hbase(main):005:0>	

Figure 4-7

Specify multiple versions to be queried.

get 'cx_table_stu01','20200001',{COLUMNS=>'cf1',VERSIONS=>5}

```
hbase(main):009:0> get 'cx_table_stu01','20200001',{COLUMNS=>'cf1',VERSIONS=>5}
COLUMN CELL
cf1:age timestamp=1586224622452, value=20
cf1:gender timestamp=1586224622397, value=male
cf1:name timestamp=1586239656833, value=WangWu
1 row(s)
Took 0.0309 seconds
hbase(main):010:0>
```

Figure 4-8

The version is specified during the search, but the last record is still displayed. Although **VERSIONS** is added, only one record is returned after the get operation. This is because the default value of **VERSIONS** is **1** during table creation.

Run the **desc'cx_table_stu01'** statement to view the table attributes.

```
hbase(main):011:0> desc 'cx_table_stu01'
Table cx_table_stu01 is ENABLED
cx_table_stu01
COLUMN FAMILIES DESCRIPTION
{NAME => 'cf1', VERSIONS => '1', EVICT_BLOCKS_ON_CLOSE => 'false', NEW_VERSION_B
EHAVIOR => 'false', KEEP_DELETED_CELLS => 'FALSE', CACHE_DATA_ON_WRITE => 'false
', DATA_BLOCK_ENCODING => 'NONE', TTL => 'FOREVER', MIN_VERSIONS => '0', REPLICA
TION_SCOPE => '0', BLOOMFILTER => 'ROW', CACHE_INDEX_ON_WRITE => 'false', IN_MEM
ORY => 'false', CACHE_BLOOMS_ON_WRITE => 'false', PREFETCH_BLOCKS_ON_OPEN => 'fa
lse', COMPRESSION => 'NONE', BLOCKCACHE => 'true', BLOCKSIZE => '65536'}
1 row(s)
Took 0.0136 seconds
hbase(main):012:0>
```

Figure 4-9

To view data of multiple versions, run the following statement to change the value of **VERSIONS** of the table or specify the value when creating the table:



alter 'cx_table_stu01',{NAME=>'cf1','VERSIONS'=>5}

alter 'cx_table_stu01',{NAME=>'cf1','VERSIONS'=>5}

Then, insert multiple data records.

 put
 'cx_table_stu01','20200001','cf1:name','ZhangSan'

 put
 'cx_table_stu01','20200001','cf1:name','LiSi'

 put
 'cx_table_stu01','20200001','cf1:name','WangWu'

The value of name has multiple versions.

```
hbase(main):022:0> alter 'cx_table_stu01', {NAME=>'cf1', 'VERSIONS'=>5}
Updating all regions with the new schema...
1/1 regions updated.
Done.
Took 3.3525 seconds
hbase(main):023:0> get 'cx table stu01','20200001',{COLUMNS=>'cf1',VERSIONS=>5}
COLUMN
                      CELL
                      timestamp=1586224622452, value=20
cf1:age
cf1:gender
                      timestamp=1586224622397, value=male
cf1:name
                      timestamp=1586239656833, value=WangWu
cf1:name
                      timestamp=1586224622340, value=tom
1 \operatorname{row}(s)
Took 0.0148 seconds
hbase(main):024:0> put 'cx table stu01','20200001','cf1:name','ZhangSan'
Took 0.0082 seconds
hbase(main):025:0> put 'cx_table stu01','20200001','cf1:name','LiSi'
Took 0.0034 seconds
                        'cx_table_stu01','20200001','cf1:name','WangWu'
hbase(main):026:0> put
Took 0.0032 seconds
hbase(main):027:0> get 'cx table stu01','20200001',{COLUMNS=>'cf1',VERSIONS=>5}
COLUMN
                      CELL
                     timestamp=1586224622452, value=20
cf1:age
                     timestamp=1586224622397, value=male
 cf1:gender
                     timestamp=1586240738902, value=WangWu
cf1:name
cf1:name
                     timestamp=1586240738884, value=LiSi
                     timestamp=1586240738860, value=ZhangSan
cf1:name
 cf1:name
                      timestamp=1586239656833, value=WangWu
                      timestamp=1586224622340, value=tom
cf1:name
1 row(s)
Took 0.0336 seconds
hbase(main):028:0>
```

Figure 4-10

4.3.1.7 Deleting Data

Run the **delete 'cx_table_stu01','20200002','cf1:age'** command to delete data from a column family.



```
002'
```

```
hbase(main):028:0> get 'cx table stu01','20200002'
COLUMN
                       CELL
 cf1:age
                       timestamp=1586224622561, value=19
 cf1:gender
                       timestamp=1586224622531, value=female
                       timestamp=1586224622492, value=hanmeimei
 cf1:name
1 \operatorname{row}(s)
Took 0.0117 seconds
hbase(main):029:0> delete 'cx table stu01','20200002','cf1:age'
Took 0.0065 seconds
hbase(main):030:0> get 'cx table stu01','20200002'
COLUMN
                       CELL
 cf1:gender
                       timestamp=1586224622531, value=female
 cf1:name
                       timestamp=1586224622492, value=hanmeimei
1 row(s)
Took 0.0088 seconds
hbase(main):031:0>
```

Figure 4-11

Run the **deleteall 'cx_table_stu01','20200002'** command to delete a row of data.

```
hbase(main):031:0> deleteall 'cx_table_stu01','20200002'
Took 0.0030 seconds
hbase(main):032:0> get 'cx_table_stu01','20200002'
COLUMN CELL
0 row(s)
Took 0.0080 seconds
hbase(main):033:0>
```

Figure 4-12

4.3.1.8 Deleting Tables

You can run the **drop** command to delete a table. However, you must disable a table before deleting it.

Step 1 Run the **disable** '*table name*' command.

Step 2 Run the **drop** 'table name' command.

```
hbase(main):033:0> disable 'cx_table_stu01'
2020-04-07 14:32:54,815 INFO [main] client.HBaseAdmin: Started disable of cx ta
ble stu01
2020-04-07 14:32:57,068 INFO [main] client.HBaseAdmin: Operation: DISABLE, Tabl
e Name: default:cx_table_stu01, procId: 22 completed
Took 2.2688 seconds
hbase(main):034:0> drop 'cx table stu01'
2020-04-07 14:33:06,518 INFO [main] client.HBaseAdmin: Operation: DELETE, Table
Name: default:cx_table_stu01, procId: 24 completed
Took 0.2700 seconds
hbase(main):035:0> list
TABLE
0 row(s)
Took 0.0145 seconds
=> []
hbase(main):036:0> 🚪
```

Figure 4-13



4.3.2 Task 2: Pre-splitting Regions During Table Creation

By default, HBase creates a table with only one region. The row key of the region has no boundary, that is, there is no start key or end key. All data is written to the default region. As the data volume increases, the region cannot handle the increasing data. Therefore, the region is split into two regions. During this process, the following problems may occur:

- 1. When data is written to a region, data hotspots may occur.
- 2. Region splitting consumes valuable cluster I/O resources.

To resolve the preceding problems, create multiple empty regions during table creation, and determine the start and end row keys of each region. In this way, as long as the row key can evenly hit each region, the write hotspot problem does not exist, and the probability of splitting is greatly reduced. HBase provides two pre-splitting algorithms: HexStringSplit and UniformSplit. HexStringSplit applies to the row key of hexadecimal characters, and UniformSplit applies to the row key of random byte arrays.

4.3.2.1 Splitting into Four Regions Randomly by Row Key

Run the **create 'cx_table_stu02','cf2', {NUMREGIONS => 4 , SPLITALGO => 'UniformSplit'}** to create a table.

```
hbase(main):037:0> create 'cx_table_stu02','cf2', {NUMREGIONS => 4 , SPLITALGO =
> 'UniformSplit'}
2020-04-07 15:10:11,345 INFO [main] client.HBaseAdmin: Operation: CREATE, Table
Name: default:cx_table_stu02, procId: 25 completed
Created table cx_table_stu02
Took 2.2548 seconds
=> Hbase::Table - cx_table_stu02
hbase(main):038:0>
```

Figure 4-14

Region name format: [table],[region start key],[region id]

Log in to the HBase WebUI and check the table partitions.

Log in to MRS Manager, choose Services > HBase.



Dashboard	Services	Hosts	Alarms	Audit
🕹 Download	d Client Mo	re ▼		
Service 🗘	Operating	Status 💠 Healt	h Status 🗢	Configuration St
DBService	Started Started	🕑 Go	ood	Synchronized
Flink	Started	🕑 Go	ood	Synchronized
Flume	Started	🕑 Go	ood	Synchronized
HBase	Started	🕑 Go	ood	Synchronized
HDFS	Started	🕑 Go	ood	Synchronized

Figure 4-15

Click **HMaster (Active)**. The HMaster WebUI is displayed.



Service HBase > Service St	atus
Service Status Instar	nce Service Configuration
Start Service	Stop Service 🕹 Downlo
HBase Summary	
Health Status	Sood
Configuration Status	Synchronized
Version	2.1.1.0101-mrs-2.0
Requests	0
Flush Queue Size	0
HMaster Web UI	HMaster (Active)

Figure 4-16

Click **cx_table_stu02** on the **User Tables** tab page. The **Tables Regions** page is displayed.



HBAS	Ē	Home	Table Details	Procedures &	Locks Pro	ocess Metrics	Local Logs	Log Level	Debug Dump	Metrics	Dump
Total:1											0
Backup	Mas	sters									
ServerName						Port			Start Time		
Total:0											
Tables											
User Tables	Syster	n Tables	Snapshots								
1 table(s) in set.	[Details]										
Namespace	◆ Name	•	♦ State	•			Regio	ns			\$
Humespace	•		v outo	OPEN ¢	OPENING	CLOSED	CLOSING	OFFLINE	+ FAILED +	SPLIT ¢	Other +
default	cx_ta	ble_stu02	ENABLED	4	0	0	0	0	0	0	0

Figure 4-17

The **cx_table_stu02** table has four partitions.

F	IBASE	Home	Table Details	Procedures & Locks	Process Metrics	Local Logs	Log Level	Debu	ıg Dump	Metrics Dump	HBase Confi	iguration Logo	ut
1	Table Regio	ons											
	Name(4)					Re	egion Server	ReadR (0)	lequests	WriteRequests (0)	StorefileSize (0 B)	Num. Storefiles (0)	MemSize (0B)
	cx_table_stu02,,15862	43409101.0	3b29ef6425c28fce	23ca3f8db90bae2a.		nc co	ode-ana- prevbbe:16030	0		0	0 B	0	0 B
	cx_table_stu02,@\x00	\x00\x00\x0	0\x00\x00\x00,158	36243409101.ceab0a3e8	e8b23c23d2f36b7d9f	39898. no co	ode-ana- prevbbe:16030	0		0	0 B	0	0 B
	cx_table_stu02,\x80\x0	00\x00\x00\	x00\x00\x00\x00,1	586243409101.229e5a3	3702cd95590c6dfe9c3	3fc3c87. no co	ode-ana- prevbbe:16030	0		0	0 B	0	0 B
	cx_table_stu02,\xC0\x	00x/00x/00	x00\x00\x00\x00,1	586243409101.272e8e4	279dde26f5e7c7dcfa1	13a0c26. no co	ode-ana- prevbbe:16030	0		0	0 B	0	0 B

Figure 4-18

4.3.2.2 Viewing the Start Key and End Key of Specified Regions

Run the **create 'cx_table_stu03', 'cf3', SPLITS => ['10000', '20000', '30000']** command to create a table.

Check the table partitions.

Table Regions										
Name(4)	Region Server	ReadR (0)	Requests	WriteRequests (0)	StorefileSize (0 B)	Num. Storefiles (0)	MemSize (0 B)	Locality	Start Key	End Key
cx_table_stu03,,1586244846025.4c3bf78e9278b51523823f0183280374.	node-ana- corevbbe:16030	0		0	0 B	0	0 B	0.0		10000
cx_table_stu03,10000,1586244846025.362d24cc14311da56e7859bc7ab85425.	node-ana- corevbbe:16030	0		0	0 B	0	0 B	0.0	10000	20000
cx_table_stu03,20000,1586244846025.1baf17169dbf8bf81fa010d9fed09a6a.	node-ana- corevbbe:16030	0		0	0 B	0	0 B	0.0	20000	30000
cx_table_stu03,30000,1586244846025.933f85411312301b5810ec45f315648f.	node-ana- corevbbe:16030	0		0	0 B	0	0 B	0.0	30000	

Figure 4-19



4.3.3 Task 3: Using Filters

If the **cx_table_stu01** table is deleted in the previous practice, recreate the table and insert data.

Run the following commands:

```
scan 'cx_table_stu01',{FILTER=>"ValueFilter(=,'binary:20')"}
scan 'cx_table_stu01',{FILTER=>"ValueFilter(=,'binary:tom')"}
scan 'cx_table_stu01',FILTER=>"ColumnPrefixFilter('gender')"
scan 'cx_table_stu01',{FILTER=>"ColumnPrefixFilter('name') AND ValueFilter(=,'binary:hanmeimei')"}
```

Took 0.0025 seconds	
hbase(main):043:0>]	put 'cx_table_stu01','20200001', 'cf1:age','20'
Took 0.0032 seconds	
hbase(main):044:0>]	put 'cx_table_stu01','20200002', 'cf1:name','hanmeimei'
Took 0.0027 seconds	
hbase(main):045:0>]	put 'cx_table_stu01','20200002', 'cf1:gender','female'
Took 0.0026 seconds	
hbase(main):046:0>]	put 'cx_table_stu01','20200002', 'cf1:age','19'
Took 0.0029 seconds	
hbase(main):047:0>	<pre>scan 'cx_table_stu01', {FILTER=>"ValueFilter(=,'binary:20')"}</pre>
ROW	COLUMN+CELL
20200001	column=cf1:age, timestamp=1586245100377, value=20
1 row(s)	
Took 0.0410 seconds	
hbase(main):048:0>	<pre>scan 'cx_table_stu01', {FILTER=>"ValueFilter(=, 'binary:tom')"}</pre>
ROW	COLUMN+CELL
20200001	column=cil:name, timestamp=1586245100345, value=tom
1 row(s)	
Took 0.0040 seconds	
hbase(main):049:0>	<pre>scan 'cx table stu01',FILTER=>"ColumnPrefixFilter('gender')"</pre>
ROW	COLUMN+CELL
20200001	column=cil:gender, timestamp=1586245100362, value=male
20200002	column=cil:gender, timestamp=1586245100406, value=iemale
2 row(s)	
Took 0.0085 seconds	
nbase(main):050:0>	Scan 'CX_table_stuul', {FILTER=>"ColumnPrelixFilter('name') AN
D valueFilter (=, 'bil	ary:nanmelmel')"}
20200002	COLUMN+CELL
20200002	corumn=crr:name, trmestamp=rs86245100391, value=nanmermer
I row(S)	
TOOK U.UI4U SECONDS	
npase(main):051:0>	

Figure 4-20

4.4 Summary

This exercise describes how to create and delete HBase tables and add, delete, modify, and query data, how to pre-split regions, and how to use filters to query data. After completing this exercise, you will be able to know how to use HBase.



5 MapReduce Data Processing Practice

5.1 Background

This section mainly introduces how to use MR to count words.

5.2 Objectives

Understand the principles of MapReduce programming.

5.3 Tasks

5.3.1 Task 1: MapReduce Shell Practice

Step 1 Log in to an ECS.

Use PuTTY to log in to the ECS and set environment variables.

Run the **source /opt/client/bigdata_env** command.

```
proot@node-master1dCrC:~
login as: root
root@116.63.53.83's password:
Last login: Tue Apr 7 14:06:58 2020 from 119.3.119.19
[root@node-master1dCrC ~]# source /opt/client/bigdata_env
[root@node-master1dCrC ~]#
```

Figure 5-1

Step 2 Edit a data file on the local Linux host.

The file content is as follows:



```
[root@node-master1dCrC ~]# vi cx_wd.txt
[root@node-master1dCrC ~]# more cx_wd.txt
hadoop hive hadoop
hbase spark hive hadoop
spark
[root@node-master1dCrC ~]#
```

Figure 5-2

Step 3 Upload the file to the HDFS.

Figure 5-3

Step 4 Run the following command to execute the JAR file program:

yarn jar /opt/client/Yarn/hadoop/share/hadoop/mapreduce /hadoop-mapreduce-examples-3.1.1-mrs-2.0.jar wordcount /user/stu01/cx_wd.txt /user/st u01/output01



root@node-master1dCrC mapreduce]# yarn jar /opt/client/Yarn/hadoop/share/hadoop/mapredu
e/hadoop-mapreduce-examples-3.1.1-mrs-2.0.jar wordcount /user/stu01/cx wd.txt /user/st
101/output01
2020-04-11 17:04:00,452 INFO obs.OBSFileSystem: This Filesystem GC-ful, clear resource.
2020-04-11 17:04:01,256 INFO client.AHSProxy: Connecting to Application History server a
t /0.0.0.0:10200
2020-04-11 17:04:01,479 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for
path: hdfs://hacluster/tmp/hadoop-varn/staging/root/.staging/job 1586168696337 0101
2020-04-11 17:04:01.725 INFO input FileInputFormat: Total input files to process : 1
2020-04-11 17:04:04.249 INFO mapreduce.JobSubmitter: number of splits:1
2020-04-11 17:04:04.298 INFO Configuration.deprecation: varn.resourcemanager.system-metr
ics-publisher.enabled is deprecated. Instead, use varn.system-metrics-publisher.enabled
2020-04-11 17:04:05 603 INFO manyreduce. Jobsyubmitter: Submitter Submitter for job: job 1586168696337 0101
2020-04-11 17:04:05 605 INFO mapreduce JobSubmitter: Executing with tokens: []
2020-04-11 17:04:05 858 INFO configuration: resource-times yml not found
2020-04-11 17:04:05 858 INFO resource ResourceHils: Unable to find 'resource-types yml'
2020-04-11 17:04:05,000 into resource.uso
2020-04-11 17:04:06,114 INFO Imperial content in the unit to track the job, https://pdoi.material.com/op/04/04/
tion 1506169606337 0101/
allon_1506160696537_0101/ 2000_04_11 17:04:06 146 INFO maproduce Tob: Bunning job: job 1596169696337_0101
2020-04-11 17:04:04;140 INFO mapreduce Job. Ruming JOB. JOB 10010000057_0101
2020-04-11 17.04.13,223 INFO Mapleduce. ob. 000 job_job_totototototototototototi fumining in uber mode . faise
2020-04-11 17.04.15,224 INFO Mapleduce.ob. map 1006 reduce 05
2/20-04+11 = 17.04:16,264 INFO mapheduce.ob): map 1000 reduce 100
2020-04-11 17:04:25,263 INFO mapreduce.ob: Tab icb 1501(06)223 0101 completed evenessfully
2020-04-11 17:04:26,302 INFO mapreduce.Job: 300 1506160696337_0101 completed successfully
2020-04-11 17:04:26,380 INFO Mapreduce.Job: Counters: 53
File System counters
FILE: Number of bytes read=67
File: Number of Dyles Written=325279
File: Number of read operations=0
File: Number of large read operations=0
File: Number of write operations=0
HDFS: Number of bytes read=151
HDFS: Number of bytes written=32
HDFS: Number of read operations=6
HDFS: Number of large read operations=0
HDFS: Number of write operations=2
Job Counters
Launched map tasks=1
Launched reduce tasks=1
Data-local map tasks=1
Total time spent by all maps in occupied slots (ms)=13488
Total time spent by all reduces in occupied slots (ms)=17370
Total time spent by all map tasks (ms)=3372
Total time spent by all reduce tasks (ms)=2895
Total vcore-milliseconds taken by all map tasks=3372
Total vcore-milliseconds taken by all reduce tasks=2895

Figure 5-4

Note: This JAR package is a sample JAR package built-in the Hadoop framework. The default file separator is the Tab key. The **output01** folder does not exist. The program automatically creates the folder.

Step 5 View statistics results.

The result file is saved in the **output01** folder. The system automatically generates a **partr-00000** file.

```
[root@node-master1dCrC mapreduce]# hdfs dfs -ls /user/stu01/output01
2020-04-11 17:04:44,916 INFO obs.OBSFileSystem: This Filesystem GC-ful, clear resource.
Found 2 items
-rw-r-r-- 1 root hadoop 0 2020-04-11 17:04 /user/stu01/output01/_SUCCESS
-rw-r-r-- 1 root hadoop 32 2020-04-11 17:04 /user/stu01/output01/part-r-00000
[root@node-master1dCrC mapreduce]# hdfs dfs -cat /user/stu01/output01/part-r-00000
2020-04-11 17:04:58,887 INFO obs.OBSFileSystem: This Filesystem GC-ful, clear resource.
hadoop 3
hbase 1
hive 2
spark 2
[root@node-master1dCrC mapreduce]#
```

Figure 5-5

The file statistics are complete.

Step 6 Parse the source code of the WordCount JAR package.

package com.huawei.bigdata.mapreduce.examples;



```
import java.io.IOException;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.LongWritable;
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.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
public class WordCountDemo {
public static class MyMapper extends Mapper<LongWritable, Text, Text, LongWritable>{
     @Override
     protected void map(LongWritable key, Text value, Mapper<LongWritable, Text, Text,
LongWritable>.Context context)
               throws IOException, InterruptedException {
                    String line = value.toString();
                    String[] splited = line.split("\t");
                    for (String word : splited) {
                              Text k2 = new Text(word);
                              LongWritable v2 = new LongWritable(1);
                              context.write(k2, v2);
                   }
              }
}
public static class MyReducer extends Reducer<Text, LongWritable, Text, LongWritable> {
     @Override
     protected void reduce(Text k2, Iterable<LongWritable> v2s,
               Reducer<Text, LongWritable, Text, LongWritable>.Context context)
               throws IOException, InterruptedException {
               long count = 0L;
               for (LongWritable times : v2s) {
                        count += times.get();
               }
               LongWritable v3 = new LongWritable(count);
               context.write(k2, v3);
    }
}
public static void main(String[] args) throws Exception {
     Configuration conf = new Configuration();
    Job job = Job.getInstance(conf, WordCountDemo.class.getSimpleName());
     // Mandatory
    job.setJarByClass(WordCountDemo.class);
    // Specify where data comes from.
     FileInputFormat.setInputPaths(job, args[0]);
     // Specify where the custom mapper is.
    job.setMapperClass(MyMapper.class);
     // Specify the type of <k2,v2> output by the mapper.
    job.setMapOutputKeyClass(Text.class);
```



```
job.setMapOutputValueClass(LongWritable.class);
    // Specify where the custom reducer comes from.
    job.setReducerClass(MyReducer.class);
    // Specify the type of <k3,v3> output by reducer.
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(LongWritable.class);
    // Specify where data is written.
    FileOutputFormat.setOutputPath(job, new Path(args[1]));
    // true indicates that information such as running progress is sent to users in time.
    job.waitForCompletion(true);
}
}
package com.huawei.bigdata.mapreduce.examples;
import java.io.IOException;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.LongWritable;
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.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
public class WordCountDemo {
public static class MyMapper extends Mapper<LongWritable, Text, LongWritable>{
    @Override
    protected void map(LongWritable key, Text value, Mapper<LongWritable, Text, Text,
LongWritable>.Context context)
              throws IOException, InterruptedException {
                    String line = value.toString();
                   String[] splited = line.split("\t");
                   for (String word : splited) {
                             Text k2 = new Text(word);
                             LongWritable v2 = new LongWritable(1);
                             context.write(k2, v2);
                   }
              }
}
public static class MyReducer extends Reducer<Text, LongWritable, Text, LongWritable> {
    @Override
    protected void reduce(Text k2, Iterable<LongWritable> v2s,
              Reducer<Text, LongWritable, Text, LongWritable>.Context context)
              throws IOException, InterruptedException {
              long count = 0L;
              for (LongWritable times : v2s) {
                        count += times.get();
              }
              LongWritable v3 = new LongWritable(count);
              context.write(k2, v3);
    }
```







5.3.1 (Optional) Task 2: MapReduce Java Practice: Collecting Statistics on Online Duration

Prerequisites: The Java development environment has been installed and the MRS2.0 sample project has been imported. For details, see *Appendix 1*.

Step 1 Check the imported sample project.

The directory structure of the imported sample project is as follows:



⊿ 🚰 mapreduce-examples
4 进 src/main/java
a 🌐 com.huawei.bigdata.examples.tools
FileUploader.java
LocalRunner.java
I TarManager.java
a 🌐 com.huawei.bigdata.examples.util
JarFinderUtil.java
a 🌐 com.huawei.bigdata.mapreduce.examples
FemaleInfoCollector.java
MultiComponentExample.java
a 🌐 com.huawei.hadoop.security
LoginUtil.java
🃂 src/test/java
🗅 🛋 Maven Dependencies
⊳ 🛋 JRE System Library [jdk1.8.0_181]
🖻 🗁 conf
🖻 🗁 src
📂 target
pom.xml

Figure 5-6

Step 2 Understand the scenario.

Develop a MapReduce application to perform the following operations on logs about Time On Page (TP) of netizens for shopping online.

- 1. Collect statistics on female netizens whose TP for online shopping is more than 2 hours on the weekend.
- 2. The first column in the log file records names, the second column records gender, and the third column records the TP in the unit of minute. Three columns are separated by comma (,).

log1.txt: logs collected on Saturday.

LiuYang,female,20 YuanJing,male,10 GuoYijun,male,5 CaiXuyu,female,50 Liyuan,male,20 FangBo,female,50 LiuYang,female,20 YuanJing,male,10 GuoYijun,male,50 CaiXuyu,female,50 FangBo,female,60

log2.txt: logs collected on Sunday.



LiuYang,female,20	
YuanJing,male,10	
CaiXuyu,female,50	
FangBo,female,50	
GuoYijun,male,5	
CaiXuyu,female,50	
Liyuan,male,20	
CaiXuyu,female,50	
FangBo,female,50	
LiuYang,female,20	
YuanJing,male,10	
FangBo,female,50	
GuoYijun,male,50	
CaiXuyu,female,50	
FangBo,female,60	

Step 3 Plan data.

Save the original log files in the HDFS.

- Create two text files on the local host, copy the content in log1.txt to cx_input_data1.txt, and copy the content in log2.txt to cx_input_data2.txt.
- 2. Create folder **/user/stu01/input** in the HDFS and upload **cx_input_data1.txt** and **cx_input_data2.txt** to the directory.
 - a Run the **hdfs dfs -mkdir /user/stu01/input** command on the HDFS client in the Linux system.
 - b Run the hdfs dfs -put local_filepath /user/stu01/input command twice.

After the operation is complete, the files in the corresponding HDFS directory are as follows:



Figure 5-7

Step 4 Understand the development approaches.

Collect statistics on female netizens whose TP is more than 2 hours on the weekend.

To achieve the objective, the process is as follows:

- 1. Read the original file data.
- 2. Filter data about the TP of the female netizens.
- 3. Summarize the total TP of each female.



- Page 54
- 4. Filter information about female netizens whose TP for online shopping is more than two hours.

Parse sample code. The class in the sample project is FemaleInfoCollector.java.

Collect statistics on female netizens whose TP for online shopping is more than 2 hours on the weekend.

To achieve the objective, the process is as follows:

- 1. Filter the TP of female netizens in original files using the CollectionMapper class inherited from the Mapper abstract class.
- 2. Summarize the TP of each female netizen, and output information about female netizens whose TP is more than 2 hours using the CollectionReducer class inherited from the Reducer abstract class.
- 3. Use the main method to create a MapReduce job and submit the MapReduce job to the Hadoop cluster.

Step 5 Run the MR packaging program.

Open the cmd window, go to the directory where the project is located, and run the **mvn package** command to package the project.



Figure 5-8

Run the **mvn package** command to generate a JAR package and obtain it from the target directory in the project directory, for example, **mapreduce-examples-mrs-2.0.jar**.

2020/4/11 22:16
2020/4/11 22:16
2020/4/11 22:16
2020/4/11 22:16
2020/4/11 21:00
2020/4/11 22:16

Figure 5-9

Step 6 Use WinSCP to log in to an ECS.

Upload the JAR package to the **/root** directory.



HCIA-Big Data V3.0 Lab Guide

Name	Size	Туре	Changed	/root/
۴		Parent directory	6/24/2020 1:03:13 PM	Name
classes		File folder	6/24/2020 12:54:54 PM	L
generated-sources		File folder	6/24/2020 9:29:37 AM	ssh
maven-archiver		File folder	6/24/2020 9:29:52 AM	oracle ire usage
📜 maven-status		File folder	6/24/2020 9:29:37 AM	<pre> mapreduce-examples-mrs-2.0.iar </pre>
-				

Figure 5-10

Step 7 Use PuTTY to log in to the ECS and run the MR program.

Run the source /opt/client/bigdata_env command.

[root@node-master2-BWgLh	~]#	source	- /opt/client/bigdata env
[root@node-master2-BWgLh	~]#		—

Figure 5-11

Run the mapreduce program.

yarn jar /root/mapreduce-examples-mrs-2.0.jar com.huawei.bigdata.mapreduce.examples.FemaleInfoCollector /user /stu01/input /user/stu01/output2

Note: /output2 must not exist.



Figure 5-12

Step 8 View the result. The MR output result is stored in the **/output2** directory. A result file **part-r-00000** is generated. Run the **cat** command to view the result.



```
2020-04-11 22:54:10,139 INFO obs.OBSFileSystem: This Filesystem GC-ful, clear resource.

Found 2 items

-rw-r--r-- 1 root hadoop 0 2020-04-11 22:29 /user/stu01/output2/_SUCCESS

-rw-r--r-- 1 root hadoop 23 2020-04-11 22:29 /user/stu01/output2/part-r-00000

[root@node-master1dCrC ~]# hdfs dfs -cat /user/stu01/output2/part-r-00000

2020-04-11 22:54:13,781 INFO obs.OBSFileSystem: This Filesystem GC-ful, clear resource.

CaiXuyu 300

FangBo 320

[root@node-master1dCrC ~]#
```

Figure 5-13

There are two persons whose TP exceeds 2 hours.

5.4 Summary

This exercise describes the MapReduce programming process in shell and Java modes and explains the source code to help trainees quickly get started with MapReduce.



6 Spark Memory Computing Practice

6.1 Background

Spark is implemented in the Scala language, and uses Scala as its application framework. Different from Hadoop, Spark can be tightly integrated with Scala. Scala can operate Resilient Distributed Datasets (RDDs) so easily as operating local combined objects. This exercise describes how to use Scala to operate Spark RDD and Spark SQL.

6.2 Objectives

Understand Spark programming by exercising Spark RDD and Spark SQL.

6.3 Tasks

6.3.1 Task 1: Spark RDD Programming

This exercise introduces Spark RDD programming to help you understand the working principles and core mechanism of Spark Core.

The process is as follows:

- Understand how to create an RDD.
- Understand the common operator of RDD.
- Understand how to use Scala project code to complete RDD operations.

Step 1 Load data from a file system to create an RDD.

Spark uses the textFile() method to load data from a file system to create an RDD.

This method takes the URI of the file as a parameter, which can be the address of the local file system, the address of the HDFS, the address of Amazon S3, or more.

Connect to the cluster, start PuTTY or another connection software, load environment variables, and enter spark-shell.

source /opt/client/bigdata_env spark-shell



2020-04-11 22:58:10,874 | WARN | main | The configuration key 'spark.yarn.access. the new key 'spark.yarn.access.hadoopFileSystems' instead. | org.apache.spark.int 2020-04-11 22:58:10,880 | WARN | main | The configuration key 'spark.yarn.access the new key 'spark.yarn.access.hadoopFileSystems' instead. | org.apache.spark.int 2020-04-11 22:58:10,881 | WARN | main | The configuration key 'spark.yarn.access. the new key 'spark.yarn.access.hadoopFileSystems' instead. | org.apache.spark.int 2020-04-11 22:58:13,712 | WARN | main | load mapred-default.xml, HIVE_CONF_DIR er tate.java:1101) Spark context Web UI available at http://node-master1dCrC:22688 Spark context available as 'sc' (master = local[*], app id = local-1586617091054). Spark session available as 'spark'. Welcome to version 2.3.2-mrs-2.0 Using Scala version 2.11.8 (OpenJDK 64-Bit Server VM, Java 1.8.0_212) Type in expressions to have them evaluated. Type :help for more information. scala> 🚪

Figure 6-1

1. Load data from a Linux local file system.

scala> val lines = sc.textFile("file:///home/data/log1.txt")

scala> val line = sc.textFile("file:///root/cx_input_data1.txt")
line: org.apache.spark.rdd.RDD[String] = file:///root/cx_input_data1.txt MapPartitionsRDD[3] at textFil
e at <console>:24
scala> line.count()
res3: Long = 11
scala> line.collect()
res4: Array[String] = Array(LiuYang,female,20, YuanJing,male,10, GuoYijun,male,5, CaiXuyu,female,50, Li
yuan,male,20, FangBo,female,50, LiuYang,female,20, YuanJing,male,10, GuoYijun,male,50, CaiXuyu,female,5
0, FangBo,female,60)

scala>

Figure 6-2

2. Load data from the HDFS. If the file does not exist, put it again.

scala> val lines1 = sc.textFile("hdfs://hacluster/user/stu01/cx_input_data1.txt")
scala> val lines2 = sc.textFile("/user/stu01/cx_input_data1.txt ")

You can use either of the statements but the second one is recommended.





Figure 6-3

Step 2 Create an RDD using a parallel set (array).

You can call the parallelize method of SparkContext to create an RDD on an existing set (array) in Driver.

scala> val array = Array(1,2,3,4,5)
array: Array[Int] = Array(1, 2, 3, 4, 5)
scala> val rdd = sc.parallelize(array)
rdd: org.apache.spark.rdd.RDD[Int] = ParallelCollectionRDD[14] at parallelize at <console>:26
scala> rdd.collect()
res9: Array[Int] = Array(1, 2, 3, 4, 5)

Alternatively, you can create an RDD as follows:

```
scala> val list = List(1,2,3,4,5)
list: List[Int] = List(1, 2, 3, 4, 5)
scala> val rdd = sc.parallelize(list)
rdd: org.apache.spark.rdd.RDD[Int] = ParallelCollectionRDD[15] at parallelize at <console>:26
scala> rdd.collect()
res10: Array[Int] = Array(1, 2, 3, 4, 5)
scala>
```

6.3.2 Task 2: RDD Shell Operations

Common transformations

Table 6-1

Transformation	Meaning
map(func)	Returns a new RDD formed by passing each element of the source through a function <i>func</i> .
filter(func)	Returns a new RDD formed by selecting those elements of the source on which <i>func</i> returns true .
flatMap(func)	Similar to map, but each input item can be mapped to 0 or more output items (so <i>func</i> should return a



Transformation	Meaning			
	Seq instead of a single item).			
mapPartitions(func)	Similar to map, but runs separately on each partition (block) of the RDD, so <i>func</i> must be of type Iterator <t> => Iterator<u> when running on an RDD of type T.</u></t>			
mapPartitionsWithIndex(func)	Similar to mapPartitions, but also provides <i>func</i> with an integer value representing the index of the partition, so <i>func</i> must be of type (Int, Iterator <t>) => Iterator<u> when running on an RDD of type T.</u></t>			
union(otherDataset)	Returns a new RDD that contains the union of the elements in the source RDD and the argument.			
intersection(otherDataset)	Returns a new RDD that contains the intersection of the elements in the source RDD and the argument.			
distinct([numTasks]))	Returns a new RDD that contains the distinct elements of the source RDD.			
groupByKey([numTasks])	When called on an RDD of (K, V) pairs, returns an RDD of (K, Iterable <v>) pairs.</v>			
reduceByKey(func, [numTasks])	When called on an RDD of (K, V) pairs, returns an RDD of (K, V) pairs where the values for each key are aggregated using the given reduce function <i>func</i> , which must be of type $(V,V) => V$. Like in groupByKey, the number of reduce tasks is configurable through an optional second argument.			
sortByKey([ascending], [numTasks])	When called on an RDD of (K, V) pairs where K implements Ordered, returns an RDD of (K, V) pairs sorted by keys in descending order.			
sortBy(func,[ascending], [numTasks])	Similar to sortByKey, but more flexible.			
join(otherDataset, [numTasks])	When called on RDDs of type (K, V) and (K, W), returns an RDD of (K, (V, W)) pairs with all pairs of elements for each key.			
cogroup(otherDataset, [numTasks])	When called on RDDs of type (K, V) and (K, W), returns an RDD of (K, (Iterable <v>, Iterable<w>)) tuples.</w></v>			
coalesce(numPartitions)	Decreases the number of partitions in the RDD to numPartitions.			
repartition(numPartitions)	Reshuffles the data in the RDD randomly to create			



Transformation	Meaning		
	either more or fewer partitions and balance it across them.		
repartitionAndSortWithinPartitio ns(partitioner)	Repartitions the RDD according to the given partitioner and, within each resulting partition, sorts records by their keys.		

Common actions

Action	Meaning	
reduce(func)	Aggregates the elements of the RDD using a function which takes two arguments and returns one.	
collect()	Returns all the elements of the RDD as an array at the driver program.	
count()	Returns the number of elements in the RDD.	
first()	Returns the first element of the RDD, which is similar to take(1).	
take(n)	Returns an array with the first <i>n</i> elements of the RDD.	
takeOrdered(n, [ordering])	Returns the first <i>n</i> elements of the RDD using either their natural order or a custom comparator.	
saveAsTextFile(path)	Writes the elements of the RDD as a text file (or set of text files) in a given directory in the local filesystem, HDFS or any other Hadoop-supported file system. Spark will call toString on each element to convert it to a line of text in the file.	
saveAsSequenceFile(path)	Writes the elements of the RDD as a Hadoop SequenceFile in a given path in the local filesystem, HDFS or any other Hadoop-supported file system.	
saveAsObjectFile(path)	Writes the elements of the RDD in a simple format using Java serialization.	
countByKey()	Only available on RDDs of type (K, V). Returns a hashmap of (K, Int) pairs with the count of each key.	
foreach(func)	Runs a function <i>func</i> on each element of the RDD.	
foreachPartition(func)	Runs a function <i>func</i> on each partition of the RDD.	



Step 1 Use map and filter.

Generate RDDs in parallel.

val rdd1 = sc.parallelize(List(5, 6, 4, 7, 3, 8, 2, 9, 1, 10))
//Multiply each element in rdd1 by 2 and sort the results.
val rdd2 = rdd1.map(_ * 2).sortBy(x => x, true)
//Filter elements greater than or equal to 5.
val rdd3 = rdd2.filter(_ >= 5)
//Display elements on the client in array mode.

The result is as follows:

```
scala> val rdd1 = sc.parallelize(List(5, 6, 4, 7, 3, 8, 2, 9, 1, 10))
rdd1: org.apache.spark.rdd.RDD[Int] = ParallelCollectionRDD[16] at parallelize at <console>:24
scala> val rdd2 = rdd1.map(_ * 2).sortBy(x => x, true)
rdd2: org.apache.spark.rdd.RDD[Int] = MapPartitionsRDD[22] at sortBy at <console>:25
scala> val rdd3 = rdd2.filter(_ >= 5)
rdd3: org.apache.spark.rdd.RDD[Int] = MapPartitionsRDD[23] at filter at <console>:25
scala> rdd3.collect
res11: Array[Int] = Array(6, 8, 10, 12, 14, 16, 18, 20)
scala>
```

Figure 6-4

Step 2 Use flatMap.

val rdd1 = sc.parallelize(Array("a b c", "d e f", "h i j"))
//Divide each element in rdd1 and flatten the elements.
val rdd2 = rdd1.flatMap(_.split(" "))
rdd2.collect

The result is as follows:

```
scala> val rdd1 = sc.parallelize(Array("a b c", "d e f", "h i j"))
rdd1: org.apache.spark.rdd.RDD[String] = ParallelCollectionRDD[24] at parallelize at <console>:24
scala> val rdd2 = rdd1.flatMap(_.split(" "))
rdd2: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[25] at flatMap at <console>:25
scala> rdd2.collect
res12: Array[String] = Array(a, b, c, d, e, f, h, i, j)
scala> П
```

Figure 6-5

Step 3 Use intersection and union.

```
val rdd1 = sc.parallelize(List(5, 6, 4, 3))
val rdd2 = sc.parallelize(List(1, 2, 3, 4))
//Obtain the union set.
val rdd3 = rdd1.union(rdd2)
//Obtain the intersection.
val rdd4 = rdd1.intersection(rdd2)
//Deduplicate data.
```



rdd3.distinct.collect rdd4.collect

The result is as follows:

```
scala> val rdd1 = sc.parallelize(List(5, 6, 4, 3))
rdd1: org.apache.spark.rdd.RDD[Int] = ParallelCollectionRDD[0] at parallelize at <console>:24
scala> val rdd2 = sc.parallelize(List(1, 2, 3, 4))
rdd2: org.apache.spark.rdd.RDD[Int] = ParallelCollectionRDD[1] at parallelize at <console>:24
scala> // Get Union Set
scala> val rdd3 = rdd1.union(rdd2)
rdd3: org.apache.spark.rdd.RDD[Int] = UnionRDD[2] at union at <console>:27
scala> // GetIntersection Set
scala> val rdd4 = rdd1.intersection(rdd2)
rdd4: org.apache.spark.rdd.RDD[Int] = MapPartitionsRDD[8] at intersection at <console>:27
scala> // Deduplication
scala> rdd3.distinct.collect
res0: Array[Int] = Array(1, 2, 3, 4, 5, 6)
scala> rdd4.collect
```

Figure 6-6

res1: Array[Int] = Array(3, 4) scala>

Figure 6-7

Step 4 Use join and groupByKey.

```
val rdd1 = sc.parallelize(List(("tom", 1), ("jerry", 3), ("kitty", 2)))
val rdd2 = sc.parallelize(List(("jerry", 2), ("tom", 1), ("shuke", 2)))
//Obtain the join.
val rdd3 = rdd1.join(rdd2)
rdd3.collect
//Obtain the union set.
val rdd4 = rdd1 union rdd2
rdd4.collect
//Group by key.
val rdd5=rdd4.groupByKey
rdd5.collect
```

Step 5 Use cogroup.

```
val rdd1 = sc.parallelize(List(("tom", 1), ("tom", 2), ("jerry", 3), ("kitty", 2)))
val rdd2 = sc.parallelize(List(("jerry", 2), ("tom", 1), ("jim", 2)))
//cogroup
val rdd3 = rdd1.cogroup(rdd2)
//Pay attention to the difference between cogroup and groupByKey.
rdd3.collect
```

Step 6 Use reduce.

val rdd1 = sc.parallelize(List(1, 2, 3, 4, 5))



```
//Reduce aggregation.
val rdd2 = rdd1.reduce(_ + _)
rdd2
```

```
scala> val rdd1 = sc.parallelize(List(1, 2, 3, 4, 5))
rdd1: org.apache.spark.rdd.RDD[Int] = ParallelCollectionRDD[31] at parallelize at <console>:24
scala> //reduce aggregation
scala> val rdd2 = rdd1.reduce(_ + _)
rdd2: Int = 15
scala> rdd2
res12: Int = 15
scala>
```

Figure 6-8

Step 7 Use reduceByKey and sortByKey.

```
val rdd1 = sc.parallelize(List(("tom", 1), ("jerry", 3), ("kitty", 2), ("shuke", 1)))
val rdd2 = sc.parallelize(List(("jerry", 2), ("tom", 3), ("shuke", 2), ("kitty", 5)))
val rdd3 = rdd1.union(rdd2)
//Aggregate by key.
val rdd4 = rdd3.reduceByKey(_ + _)
rdd4.collect
//Sort by value in descending order.
val rdd5 = rdd4.map(t => (t._2, t._1)).sortByKey(false).map(t => (t._2, t._1))
rdd5.collect
```

Figure 6-9

```
scala> val rdd1 = sc.parallelize(List(("tom", 1), ("jerry", 3), ("kitty", 2), ("shuke", 1)))
rdd1: org.apache.spark.rdd.RDD[(String, Int)] = ParallelCollectionRDD[32] at parallelize at <console>:24
scala> val rdd2 = sc.parallelize(List(("jerry", 2), ("tom", 3), ("shuke", 2), ("kitty", 5)))
rdd2: org.apache.spark.rdd.RDD[(String, Int)] = ParallelCollectionRDD[33] at parallelize at <console>:24
scala> val rdd3 = rdd1.union(rdd2)
rdd3: org.apache.spark.rdd.RDD[(String, Int)] = UnionRDD[34] at union at <console>:27
scala> // Aggregation by key
scala> val rdd4 = rdd3.reduceByKey(_ + _)
rdd4: org.apache.spark.rdd.RDD[(String, Int)] = ShuffledRDD[35] at reduceByKey at <console>:25
scala> rdd4.collect
res13: Array[(String, Int)] = Array((tom,4), (shuke,3), (kitty,7), (jerry,5))
scala> // Sort by value in descending order
scala> val rdd5 = rdd4.map(t => (t._2, t._1)).sortByKey(false).map(t => (t._2, t._1))
rdd5: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[40] at map at <console>:25
scala> rdd5.collect
res14: Array[(String, Int)] = Array((kitty,7), (jerry,5), (tom,4), (shuke,3))
scala>
```

Figure 6-10

Step 8 Understand the lazy mechanism.

The lazy mechanism means that the entire transformation process only records the track of the transformation and does not trigger real calculation. Only when an operation is performed, real calculation is triggered from the beginning to the end.



A simple statement is provided to explain the lazy mechanism of Spark. The **data.txt** file does not exist, but the first two statements are executed successfully. An error occurs only when the third action statement is executed.

```
scala> val lines = sc.textFile("data.txt")
scala> val lineLengths = lines.map(s => s.length)
scala> val totalLength = lineLengths.reduce((a, b) => a + b)
```

scala> val lines = sc.textFile("data.txt")
lines: org.apache.spark.rdd.RDD[String] = data.txt MapPartitionsRDD[42] at textFile at <console>:24</console>
<pre>scala> val lineLengths = lines.map(s => s.length)</pre>
lineLengths: org.apache.spark.rdd.RDD[Int] = MapPartitionsRDD[43] at map at <console>:25</console>
scala> val totallengtn = linelengtns.requce $((a - b) = b - a + b)$
org.abache.nadoop.mapred.invalidinputException: input path does not exist: ndis://nacluster/user/root/data.txt
at org.apache.nadoop.mapred.HileInputFormat.SingleThreadedListStatus(HileInputFormat.JaVa:297)
at org.apache.hadoop.mapred.FileInputFormat.Tiststatus(FileInputFormat.Java:239)
at org.apache.hadoop.mapred.fileinputrormat.getspiits(fileinputrormat.java:325)
at org.apache.spark.rud.nadoopkdb.getPartitions(nadoopkdb.scata:200)
at org.apache.spark.rud.kbbssahoniunspartitions2.apply(Rbb.scala:23)
at org.apache.spark.rud.kbb3yahohi heispartition3y2.appry(kbb.scala.251)
at scale.option.getorbise(option.scale.izi)
at org apache spark rdd ManDartitions/DND Scalazzi)
at org.apache.spark.tud.mapraititionshor.getratitions(mapraititionshor.scala.40)
at organache spark idd PDDSanonfunspartitionsS2 apply(RDD.Scala.23)
at org.apache.spark.turk.bb/yahohimurk.apache.com/y.appry(hb/.scala.251)
at organache snark rdd RDD nartificns (RDD scala 251)
at org anache snark rdd ManDartitionsRDD getDartitions(ManDartitionsRDD scala-46)
at org anache snak rdd BDRSanonfushartitions(2 ann)u(BDR scala:253)
at org apache spark rdd RDDSSanonfunSpartitionsS2 apply(RDD scala:251)
at scala.ontion.geforElse(Ontion.scala:121)
at org.apache.spark.rdd.BDD.partitions(BDD.scala:251)
at org.apache.spark.SparkContext.runJob (SparkContext.scala:2137)
at org.apache.spark.rdd.RDD\$\$anonfun\$reduce\$1.apply(RDD.scala:1035)
at org.apache.spark.rdd.RDDOperationScopeS.withScope(RDDOperationScope.scala:151)
at org.apache.spark.rdd.RDDOperationScopeS.withScope(RDDOperationScope.scala:112)
at org.apache.spark.rdd.RDD.withScope(RDD.scala:363)
at org.apache.spark.rdd.RDD.reduce(RDD.scala:1017)
49 elided
scala>

Figure 6-11

Step 9 Perform persistence operations.

The following is an example of calculating the same RDD for multiple times:

scala> val list = List("Hadoop","Spark","Hive")
list: List[String] = List(Hadoop, Spark, Hive)
scala> val rdd = sc.parallelize(list)
rdd: org.apache.spark.rdd.RDD[String] = ParallelCollectionRDD[22] at parallelize at <console>:29
scala> println(rdd.count())
//Action operation, which triggers a real start-to-end calculation.
3
scala> println(rdd.collect().mkString(","))
//Action operation, which triggers a real start-to-end calculation.
Hadoop,Spark,Hive

After the preceding instance is added, the execution process after a persistence statement is added is as follows:

```
scala> val list = List("Hadoop","Spark","Hive")
list: List[String] = List(Hadoop, Spark, Hive)
scala> val rdd = sc.parallelize(list)
rdd: org.apache.spark.rdd.RDD[String] = ParallelCollectionRDD[22] at parallelize at <console>:29
scala> rdd.cache()
```



3

//Persist(MEMORY_ONLY) is called. However, when the statement is executed, the RDD is not cached because the RDD has not been calculated and generated. scala> println(rdd.count())

//The first action triggers a real start-to-end calculation. In this case, the preceding rdd.cache() is executed and the RDD is stored in the cache.

scala> println(rdd.collect().mkString(","))

//The second action does not need to trigger a start-to-end calculation. Only the RDD in the cache needs to be reused. Hadoop,Spark,Hive

6.3.3 (Optional) Task 3: RDD Code Programming — Java Programming

Step 1 Understand the scenario.

Same as the MapReduce exercise background, this exercise requires you to calculate the TP.

Develop a Spark application to perform the following operations on logs about the TP of netizens for online shopping on a weekend:

- 1. Collect statistics on female netizens whose TP for online shopping is more than 2 hours on the weekend.
- 2. The first column in the log file records names, the second column records gender, and the third column records the TP in the unit of minute. Three columns are separated by comma (,).

log1.txt: logs collected on Saturday.

LiuYang,female,20
YuanJing,male,10
GuoYijun,male,5
CaiXuyu,female,50
Liyuan,male,20
FangBo,female,50
LiuYang,female,20
YuanJing,male,10
GuoYijun,male,50
CaiXuyu,female,50
FangBo,female,60

log2.txt: logs collected on Sunday.

LiuYang,female,20	
YuanJing,male,10	
CaiXuyu,female,50	
FangBo,female,50	
GuoYijun,male,5	
CaiXuyu,female,50	
Liyuan,male,20	
CaiXuyu,female,50	
FangBo,female,50	



LiuYang,female,20		
YuanJing,male,10		
FangBo,female,50		
GuoYijun,male,50		
CaiXuyu,female,50		
FangBo,female,60		

Step 2 Plan data.

Upload two Internet access log files to the **/user/stu01/input** directory of the HDFS. If the log files already exist, you do not need to upload it.

Step 3 Start a Spark sample project.

Based on the MRS 2.0 sample project imported during environment installation, start the **FemaleInfoCollection** project, which is the Spark Core project. The folder in the MRS 2.0 sample project package is **SparkJavaExample**.



Figure 6-12

Step 4 Package the project.

Open the cmd window, go to the directory where the project is located, and run the **mvn package** command to package the project.

[[NFO] Building jar: D:\eclipseworkspace2020\huaweicloud-mrs-example-mrs-2.0\hua
foCollection-mrs-2.0.jar
[INF0]
[INFO] BUILD SUCCESS
[INFO]
[INF0] Finished at: 2020-04-12T00:38:08+08:00
[INFO] Final Memory: 30M/698M [INFO]
D: \eclipseworkspace2020\huaweicloud-mrs-example-mrs-2.0\huaweicloud-mrs-example- mrs-2.0\src\spark-examples\SparkJavaExample>_ 半:

Figure 6-13

Step 5 Use WinSCP to log in to an ECS.

Upload the JAR package to the **/root** directory.



C:\Users\mwx711840\Documents\			📲 🖳 Download 👻 📝 Edit 👻 🗶 🎵 Properties 🖆 New 🕶 🗄 🕂 💌						
Name	Size	Туре	Changed	/root/					
μ.		Parent directory	6/24/2020 10:16:44 AM	Name	Size	Changed	Rights	Owner	Γ
Custom Office Templa		File folder	6/24/2020 10:16:44 AM	₹		6/24/2020 9:37:10 AM	rwxr-xr-x	root	
My Music		File folder	6/20/2020 3:38:30 PM	.oracle_jre_usage		11/13/2019 8:29:35 PM	rwxr-xr-x	root	
My Pictures		File folder	6/20/2020 3:38:30 PM	📜 .ssh		6/22/2020 8:38:41 AM	rwx	root	
My Videos		File folder	6/20/2020 3:38:30 PM	.bash_history	1 KB	6/24/2020 10:30:05 AM	rw	root	
🔄 desktop.ini	1 KB	Configuration setti	6/20/2020 3:38:35 PM	.bash_logout	1 KB	11/15/2016 6:43:34 PM	rw-rr	root	
				.bash_profile	1 KB	11/15/2016 6:43:34 PM	rw-rr	root	
				.bashrc	1 KB	11/13/2019 8:17:37 PM	rw-rr	root	
				.cshrc	1 KB	11/15/2016 6:43:34 PM	rw-rr	root	
				history	0 KB	11/13/2019 8:29:30 PM	rw	root	
				ostackrc 🗋	2 KB	11/13/2019 8:11:30 PM	rwxr-xr-x	root	
				.rnd	1 KB	11/13/2019 8:25:23 PM	rw-r	root	
				tcshrc .	1 KB	11/15/2016 6:43:34 PM	rw-rr	root	
				env_file	1 KB	11/13/2019 8:19:59 PM	rw-rr	root	
				FemaleInfoCollection	26,388 KB	6/24/2020 9:30:13 AM	rw-r	root	

Figure 6-14

Step 6 Use PuTTY to log in to the ECS and run the Spark program.

Run the **source /opt/client/bigdata_env** command.

[root@node-master2-BWgLh	~]#	source	/opt/client/bigdata_env
[root@node-master2-BWgLh	~]#		

Figure 6-15

Execute the Spark program.

/opt/client/Spark/spark/bin/spark-subm	itclass				
com.huawei.bigdata.spark.examples.FemaleInfoCollection			yarn	deploy-mode	client
/root/FemaleInfoCollection-mrs-2.0.jar	/user/stu01/input				


gInfo(Logging.scala:54) 2020-04-12 00:50:52,546 | INFO | dispatcher-event-loop-7 | Starting task 2.0 in stage .apache.spark.internal.Logging\$class.logInfo(Logging.scala:54) 2020-04-12 00:50:52,551 | INFO | task-result-getter-3 | Finished task 1.0 in stage 1.0 l.Logging\$class.logInfo(Logging.scala:54) 2020-04-12 00:50:52,567 | INFO | task-result-getter-0 | Finished task 0.0 in stage 1.0 l.Logging\$class.logInfo(Logging.scala:54) 2020-04-12 00:50:52,570 | INFO | task-result-getter-1 | Finished task 2.0 in stage 1.0 .Logging\$class.logInfo(Logging.scala:54) 2020-04-12 00:50:52,570 | INFO | task-result-getter-1 | Removed TaskSet 1.0, whose tas ogging.scala:54) 2020-04-12 00:50:52,571 | INFO | dag-scheduler-event-loop | ResultStage 1 (collect at g\$class.logInfo(Logging.scala:54) 2020-04-12 00:50:52,578 | INFO | main | Job 0 finished: collect at FemaleInfoCollection <u>scala:54</u>) CaiXuyu**,**300 FangBo, 320 2020-04-12 00:50:52,592 | INFO | main | Stopped Spark web UI at http://node-master1dCr 2020-04-12 00:50:52,625 | INFO | Yarn application state monitor | Interrupting monitor 2020-04-12 00:50:52,647 | INFO | main | Shutting down all executors | org.apache.spark 2020-04-12 00:50:52,648 | INFO | dispatcher-event-loop-0 | Asking each executor to shu 2020-04-12 00:50:52,659 | INFO | main | Stopping SchedulerExtensionServices (serviceOption=None, services=List(), started=false) | org.apache.spark.internal.Logging\$class.logInfo(Logging.scala:54) 2020-04-12 00:50:52,660 | INFO | main | Stopped | org.apache.spark.internal.Logging\$cla 2020-04-12 00:50:52,664 | INFO | dispatcher-event-loop-1 | MapOutputTrackerMasterEndpo 2020-04-12 00:50:52,693 | INFO | main | MemoryStore cleared | org.apache.spark.interna 2020-04-12 00:50:52,694 | INFO | main | BlockManager stopped | org.apache.spark.interna 2020-04-12 00:50:52,702 | INFO | main | BlockManagerMaster stopped | org.apache.spark. 2020-04-12 00:50:52,705 | INFO | dispatcher-event-loop-1 | OutputCommitCoordinator stop 2020-04-12 00:50:52,734 | INFO | main | Successfully stopped SparkContext | org.apache 2020-04-12 00:50:53,738 | INFO | pool-1-thread-1 | Shutdown hook called | org.apache.sp pool-1-thread-1 | Shutdown hook called | org.apache.sp 2020-04-12 00:50:53,740 | INFO | pool-1-thread-1 | Deleting directory /tmp/spark-71e360 ging.scala:54) 2020-04-12 00:50:53,740 | INFO | pool-1-thread-1 | Deleting directory /tmp/spark-58370ging.scala:54) [root@node-master1dCrC spark]# 🗌

Figure 6-16

The total TP of the two persons exceeds 2 hours.

6.3.4 Task 4: Spark SQL DataFrame Programming

In versions earlier than Spark 2.0, SQLContext in Spark SQL is the entry for creating DataFrames and executing SQL statements. You can use HiveContext to operate Hive table data through Hive SQL statements. HiveContext is compatible with Hive operations and is inherited from SQLContext. In versions later than Spark 2.0, all these functions are integrated into SparkSession. SparkSession encapsulates SparkContext and SQLContext. You can obtain SparkConetxt and SQLContext objects through SparkSession.



Spark context Web UI available at http://node-master1jhOP:22615 Spark context available as 'sc' (master = local[*], app id = local-1586920279462). Spark session available as 'spark'. Welcome to version 2.3.2-mrs-2.0 Using Scala version 2.11.8 (OpenJDK 64-Bit Server VM, Java 1.8.0 212) Type in expressions to have them evaluated. Type :help for more information.

Figure 6-17

Step 1 Edit a data file.

Create the **cx_person.txt** file on the local Linux host. The file contains three columns: id, name, and age. The three columns are separated by space. The content of the **cx_person.txt** file is as follows:

1 zhangsan 20	
2 lisi 29	
3 wangwu 25	
4 zhaoliu 30	
5 tianqi 35	
6 kobe 40	

Step 2 Upload the data file to a directory in the HDFS.

hdfs dfs -put cx_person.txt

```
[root@node-master1jhOP ~]# vi cx_person.txt
[root@node-master1jhOP ~]# hdfs dfs -put cx person.txt
2020-04-15 11:08:57,886 INFO obs.OBSFileSystem: This Filesystem GC-ful, clear re
source.
[root@node-master1jhOP ~]# hdfs dfs -ls /
2020-04-15 11:09:09,512 INFO obs.OBSFileSystem: This Filesystem GC-ful, clear re
source.
Found 12 items
                                       0 2020-04-15 10:09 /app-logs
drwxrwxrwx - hdfs hadoop
drwxrwxrwx
            - hive
                     hive
                                       0 2020-04-15 10:12 /apps
drwxrwxrwx - hdfs hadoop
                                       0 2020-04-15 10:09 /ats
-rw-r--r-- 1 root ficommon
                                     72 2020-04-15 11:08 /cx_person.txt
                                      0 2020-04-15 10:09 /datasets
drwxr-xr-x - hdfs hadoop
drwxr-xr-x - hdfs hadoop
drwxrwxrwx - flink hadoop
                                       0 2020-04-15 10:09 /datastore
                                       0 2020-04-15 10:10 /flink
drwxr-xr-x - hbase hadoop
                                      0 2020-04-15 10:11 /hbase
                                      0 2020-04-15 10:09 /mr-history
drwxrwxrwx - mapred hadoop
drwxrwxrwt - spark hadoop
drwxrwxrwx - hdfs hadoop
                                       0 2020-04-15 10:14 /sparkJobHistory
                                      0 2020-04-15 10:14 /tmp
drwxrwxrwx - hdfs hadoop
                                       0 2020-04-15 10:13 /user
[root@node-master1jhOP ~]#
```

Figure 6-18

Run the **spark-shell** command to go to Spark. Then run the following command to read data and separate data in each row using column separators:



val lineRDD= sc.textFile("/cx_person.txt").map(_.split(" "))

```
scala> val lineRDD= sc.textFile("/cx_person.txt").map(_.split(" "))
lineRDD: org.apache.spark.rdd.RDD[Array[String]] = MapPartitionsRDD[2] at map at <console>:2
4
scala>
```

Figure 6-19

Step 3 Define a case class.

A class is equivalent to a schema of a table.

case class Person(id:Int, name:String, age:Int)

```
scala> case class Person(id:Int, name:String, age:Int)
defined class Person
scala>
```

Figure 6-20

Step 4 Associate an RDD with the case class.

val personRDD = lineRDD.map(x => Person(x(0).toInt, x(1), x(2).toInt))

```
scala> val personRDD = lineRDD.map(x => Person(x(0).toInt, x(1), x(2).toInt))
personRDD: org.apache.spark.rdd.RDD[Person] = MapPartitionsRDD[3] at map at <console>:27
scala>
```

Figure 6-21

Step 5 Transform the RDD into DataFrame.

val personDF = personRDD.toDF

```
scala> val personDF = personRDD.toDF
personDF: org.apache.spark.sql.DataFrame = [id: int, name: string ... 1 more field]
scala>
```

Figure 6-22

Step 6 View information about DataFrame.

personDF.show





```
scala> personDF.show
 --+---+
 id
        name | age |
 --+---+
  1|zhangsan| 20|
I
        lisi| 29|
  21
  3| wungwu| 25|
I
  4| zhaoliu| 20|
  5| tianqi| 35|
       sunba| 40|
  61
     ----+
```

Figure 6-23

personDF.printSchema

Figure 6-24

Step 7 Use the domain-specific language (DSL).

DataFrame provides the DSL to operate structured data. View the data of the **name** field.

personDF.select(personDF.col("name")).show



Figure 6-25

Check another format of the **name** field.

personDF.select("name").show



<pre>scala> personDF.select("name").show</pre>
++
name
++
zhangsan
lisi
wungwu
zhaoliu
tianqi
sunba
++
scala>

Figure 6-26

Step 8 Check the data of the **name** and **age** fields.

personDF.select(col("name"), col("age")).show



Figure 6-27

Step 9 Query all names and ages and increase the value of **age** by 1.

personDF.select(col("id"), col("name"), col("age") + 1).show



```
scala> personDF.select(col("id"), col("name"), col("age") + 1).show
     -----+
+---+-
| id|
      name|(age + 1)|
     ----+
  -+-
  1|zhangsan|
                 21|
  21
       lisi
                 30|
  3 wungwu
                 261
  4| zhaoliu|
                 21
  5| tiangi|
                 361
  61
     sunba|
                  41|
     -----+----+
```



You can also perform the following operation:

personDF.select(personDF("id"), personDF("name"), personDF("age") + 1).show

Step 10 Use the filter method to filter the records where age is greater than or equal to 25.

personDF.filter(col("age") >= 25).show

<pre>scala> personDF.filter(col("age") >= 25).show</pre>
id _ name age
+++
2 lisi 29
3 wungwu 25
5 tianqi 35
6 sunba 40
+++

Figure 6-29

Step 11 Count the number of people who are older than 30.

personDF.filter(col("age")>30).count()

```
scala> personDF.filter(col("age")>30).count()
res10: Long = 2
```

Figure 6-30

Step 12 Group people by age and collect statistics on the number of people of the same age.

personDF.groupBy("age").count().show



```
scala> personDF.groupBy("age").count().show
  --+---+
|age|count|
  --+---+
  201
         21
  401
         11
  351
         11
         11
  291
         1|
          -+
```



Step 13 Use SQL.

251

A powerful feature of DataFrame is that it can be regarded as a relational data table. You can use spark.sql() in the program to execute SQL statements for query. The result is returned as a DataFrame.

If the SQL is used, you need to register DataFrame as a table in the following way:

personDF.registerTempTable("cx_t_person")

Run the following command to display the schema information of the table:

spark.sql("desc cx_t_person ").show

scala> spark.sql("desc cx t person ").show ----+ |col name|data type|comment| ____+ idl int| null string name null int null agel ---+----+----+

Figure 6-32

Step 14 Query the two oldest people.

spark.sql("select * from cx_t_person order by age desc limit 2").show



```
scala> spark.sql("select * from cx_t_person order by age desc limit 2").show
+---+
| id| name|age|
 ---+----+---+
  6| sunba| 40|
 5|tianqi| 35|
  --+---+---+
```

Figure 6-33

Step 15 Query information about people older than 30.

spark.sql("select * from cx_t_person where age > 30 ").show

```
scala> spark.sql("select * from cx_t_person where age > 30 ").show
 --+---+
| id| name|age|
  --+----+---+
+-
  5|tianqi| 35|
  6| sunba| 40|
L
   -+---+
```

Figure 6-34

6.3.5 Task 5: Spark SQL DataSet Programming

Step 1 Create a dataset using spark.createDataset.

```
val ds1 = spark.createDataset(1 to 5)
ds1.show
```

```
scala> val ds1 = spark.createDataset(1 to 5)
ds1: org.apache.spark.sql.Dataset[Int] = [value: int]
scala> ds1.show
----+
|value|
  ---+
    11
    21
    31
     41
     51
    --+
```

Figure 6-35

Step 2 Create a dataset using a file.

```
val ds2 = spark.createDataset(sc.textFile("/cx_person.txt"))
ds2.show
```



```
scala> val ds2 = spark.createDataset(sc.textFile("/person.txt"))
ds2: org.apache.spark.sql.Dataset[String] = [value: string]
scala> ds2.show
+-----+
| value|
+-----+
|1 zhangsan 20|
| 2 lisi 29|
| 3 wangwu 25|
| 4 zhaoliu 30|
| 5 tianqi 35|
| 6 kobe 40|
+-----+
```

Figure 6-36

Step 3 Create a dataset using the **toDS** method.

```
case class Person2(id:Int, name:String, age:Int)
val data = List(Person2(1001,"liubei",20),Person2(1002,"guanyu",30))
val ds3 = data.toDS
ds3.show
```

```
scala> case class Person2(id:Int, name:String, age:Int)
defined class Person2
scala> val data = List(Person2(1001,"liubei",20),Person2(1002,"guanyu",30))
data: List[Person2] = List(Person2(1001,liubei,20), Person2(1002,guanyu,30))
scala> val ds3 = data.toDS
ds3: org.apache.spark.sql.Dataset[Person2] = [id: int, name: string ... 1 more field]
scala> ds3.show
+----+---+
| id| name|age|
+---++---+
| 1001|liubei| 20|
|1002|guanyu| 30|
+---++---++
```

Figure 6-37

Step 4 Create a database using DataFrame and as[Type].

Perform transformation based on the DataFrame of personDF in task 1. Note that the **person** object fields in Person2 and personDF must be the same.

```
val ds4= personDF.as[Person2]
ds4.show
```



```
scala> val ds4= personDF.as[Person2]
ds4: org.apache.spark.sql.Dataset[Person2] = [id: int, name: string ... 1 more field]
scala> ds4.show
  --+----+---+
| id| name|age|
  --+----+---+
  1|zhangsan| 20|
  21
       lisi| 29|
  3| wungwu| 25|
  4| zhaoliu| 20|
  5| tianqi| 35|
  61
     sunba| 40|
   -+-
       ----+
```

Figure 6-38

Step 5 Collect statistics on the number of people older than 30 in the dataset.

ds4.filter(col("age") >= 25).show

```
scala> ds4.filter(col("age") >= 25).show
+---+---+
| id| name|age|
+---+---+
| 2| lisi| 29|
| 3|wungwu| 25|
| 5|tianqi| 35|
| 6| sunba| 40|
+---+---+
```

Figure 6-39

6.4 Summary

This exercise introduces RDD-based Spark Core programming and DataFrame- and DataSet-based Spark SQL programming, and enables trainees to understand the basic operations of Spark programming.



7 Flink Real-Time Processing System Practice

7.1 Background

Flink is a unified computing framework that supports both batch processing and stream processing. It provides a stream data processing engine that supports data distribution and parallel computing.

Flink provides high-concurrency pipeline data processing, millisecond-level latency, and high reliability, making it extremely suitable for low-latency data processing.

7.2 Objectives

The asynchronous CheckPoint mechanism and real-time hot-selling product statistics of Flink help you understand the core ideas of Flink and how to use Flink to solve problems.

7.3 Tasks

7.3.1 Task 1: Importing a Flink Sample Project

Step 1 Download Flink sample code.

Visit https://support.huaweicloud.com/en-us/devgmrs/mrs_06_0002.html#mrs_06_0002__section336726849219.

Click the sample project of HUAWEI CLOUD MRS 1.8 for download.

Obtaining a Sample Project

- For versions earlier than MRS 1.8, you can download the sample project at https://mapreduceservice.obswebsite.cn-north-1.myhwclouds.com/.
- For MRS 1.8, you can download the sample project at https://github.com/huaweicloud/huaweicloud-mrsexample/tree/mrs-1.8.
- For MRS 2.0, you can download the sample project at https://github.com/huaweicloud/huaweicloud-mrsexample/tree/mrs-2.0.

Figure 7-1

Step 2 Import the sample project.



For details about how to import the MRS sample project, navigate to the Appendix to refer to the instructions on how to import an MRS sample project in Eclipse. After the import, the project automatically downloads related dependency packages.



Figure 7-2

The preceding figure shows the project code structure.

7.3.2 Task 2: Exercising the Asynchronous CheckPoint Mechanism

Assume that you want to collect data volume in a 4-second time window every other second and the status of operators must be strictly consistent. That is, if an application recovers from a failure, the status of all operators must the same.

7.3.2.1 Data Planning

- 1. A custom operator generates about 10,000 pieces of data per second.
- 2. The generated data is a quadruple (Long, String, String, Integer).
- 3. After the statistics are collected, the statistical result is displayed on the terminal.
- 4. The output data is of the Long type.

7.3.2.2 Exercise Process

- 1. The source operator sends 10,000 pieces of data every second and injects the data into the window operator.
- 2. The window operator calculates the data generated in the last 4 seconds every second.
- 3. The statistical result is displayed on the terminal every second.
- 4. A checkpoint is triggered every 6 seconds and saved to the HDFS.

7.3.2.3 Procedure

Step 1 Write the snapshot data code.



The snapshot data is used to store the number of data pieces recorded by operators during snapshot creation.

Create a class named **UDFState** in the **com.huawei.flink.example.common** package of the sample project. The code is as follows:

```
import java.io.Seriablizale;
// This class is part of the snapshot and is used to save UDFState.
public class UDFState implements Serializable {
    private long count;
    // Initialize UDFState.
    public UDFState() {
        count = 0L;
    }
    // Set UDFState.
    public void setState(long count) {
        this.count = count;
    }
    // Obtain UDFState.
    public long geState() {
        return this.count;
    }
}
```

Step 2 Compile a data source with a checkpoint.

The code snippet of a source operator pauses 1 second every time after sending 10,000 pieces of data. When a snapshot is created, the code saves the total number of sent data pieces in UDFState. When the snapshot is used for restoration, the number of sent data pieces saved in UDFState is read and assigned to the count variable.

Create the SimpleSourceWithCheckPoint class in the common package. The code is as follows:

```
import org.apache.flink.api.java.tuple.Tuple4;
import org.apache.flink.streaming.api.checkpoint.ListCheckpointed;
import org.apache.flink.streaming.api.functions.source.SourceFunction;
import java.util.ArrayList;
import java.util.List;
import java.util.Random;
public class SimpleSourceWithCheckPoint implements SourceFunction<Tuple4<Long, String, String,
Integer>>, ListCheckpointed<UDFState> {
    private long count = 0;
    private boolean isRunning = true;
    private String alphabet = "justtest";
    @Override
    public List<UDFState> snapshotState(long l, long l1) throws Exception
        UDFState udfState = new UDFState();
        List<UDFState> udfStateList = new ArrayList<UDFState>();
        udfState.setState(count);
```





Step 3 Define a window with a checkpoint.

This code snippet is about a window operator and is used to calculate the number or tuples in the window.

Create the WindowStatisticWithChk class in the common package. The code is as follows:







Step 4 Develop application code.

The code is about the definition of StreamGraph and is used to implement services. The processing time is used as the time for triggering the window.

Create the FlinkProcessingTimeAPIChkMain class in the common package. The code is as follows:

```
import org.apache.flink.api.java.utils.ParameterTool;
import org.apache.flink.runtime.state.StateBackend;
import org.apache.flink.runtime.state.filesystem.FsStateBackend;
import org.apache.flink.streaming.api.CheckpointingMode;
import org.apache.flink.streaming.api.environment.StreamExecutionEnvironment;
import org.apache.flink.streaming.api.windowing.assigners.SlidingProcessingTimeWindows;
import org.apache.flink.streaming.api.windowing.time.Time;
public class FlinkProcessingTimeAPIChkMain {
    public static void main(String[] args) throws Exception
        String chkPath = ParameterTool.fromArgs(args).get("chkPath",
"hdfs://hacluster/flink/checkpoints/");
        StreamExecutionEnvironment env =
StreamExecutionEnvironment.getExecutionEnvironment();
        env.setStateBackend((StateBackend) new FsStateBackend((chkPath)));
        env.enableCheckpointing(6000, CheckpointingMode.EXACTLY_ONCE);
        env.addSource(new SimpleSourceWithCheckPoint())
                 .keyBy(0)
```





The code is compiled.

Step 5 Package the program.

Open the cmd window, go to the directory where the project is located, and run the **mvn package** command to package the project.



Figure 7-3

Run the **mvn package** command to generate a JAR package and obtain it from the target directory in the project directory, for example, **mapreduce-examples-mrs-2.0.jar**.

is PC > Desktop > huaweicloud-mrs-example-mrs-1.8 > src > flink-examples > target >				
Name	Date modified	Туре	Size	
L classes	6/24/2020 12:54 PM	File folder		
generated-sources	6/24/2020 9:29 AM	File folder		
📜 maven-archiver	6/24/2020 9:29 AM	File folder		
📜 maven-status	6/24/2020 9:29 AM	File folder		
test-classes	6/24/2020 1:47 AM	File folder		
🕌 flink-examples-1.0.jar	6/24/2020 1:03 PM	Executable Jar File	61,890 KB	
🖆 flink-examples-1.0-shaded.jar	6/24/2020 1:03 PM	Executable Jar File	61,890 KB	
🕌 original-flink-examples-1.0.jar	6/24/2020 1:03 PM	Executable Jar File	35,538 KB	
enginar mini enampres nonjar	0, 2 1, 2020 1100 1 111		00,000 112	

Figure 7-4

Step 6 Use WinSCP to log in to an ECS.

Upload the JAR package to the **/root** directory.



C:\Users\mwx711840\Deskte	op\huaweiclo	oud-mrs-example-mrs-	-1.8\src\flink-examples\target\	🛯 🔛 Download 👻 📝 Edi	t 🗝 🗙 📝 l	🕁 Properties 🛛 🞽 New 🔻	+ - 4	
Name	Size	Туре	Changed	/root/				
🔁		Parent directory	6/24/2020 1:03:13 PM	Name	Size	Changed	Rights	Owner
classes		File folder	6/24/2020 12:54:54 PM	<u>₹</u>		6/24/2020 9:37:10 AM	rwxr-xr-x	root
generated-sources		File folder	6/24/2020 9:29:37 AM	.oracle_jre_usage		11/13/2019 8:29:35 PM	rwxr-xr-x	root
maven-archiver		File folder	6/24/2020 9:29:52 AM	ssh .		6/22/2020 8:38:41 AM	rwx	root
📜 maven-status		File folder	6/24/2020 9:29:37 AM	.bash_history	1 KB	6/24/2020 10:30:05 AM	rw	root
test-classes		File folder	6/24/2020 1:47:09 AM	.bash_logout	1 KB	11/15/2016 6:43:34 PM	rw-rr	root
🔬 flink-examples-1.0.jar	61,890 KB	Executable Jar File	6/24/2020 1:03:26 PM	.bash_profile	1 KB	11/15/2016 6:43:34 PM	rw-rr	root
🔬 flink-examples-1.0-sh	61,890 KB	Executable Jar File	6/24/2020 1:03:26 PM	.bashrc	1 KB	11/13/2019 8:17:37 PM	rw-rr	root
🔬 original-flink-example	35,538 KB	Executable Jar File	6/24/2020 1:03:26 PM	.cshrc	1 KB	11/15/2016 6:43:34 PM	rw-rr	root
				history	0 KB	11/13/2019 8:29:30 PM	rw	root
				.ostackrc	2 KB	11/13/2019 8:11:30 PM	rwxr-xr-x	root
				.rnd	1 KB	11/13/2019 8:25:23 PM	rw-r	root
				.tcshrc	1 KB	11/15/2016 6:43:34 PM	rw-rr	root
				env_file	1 KB	11/13/2019 8:19:59 PM	rw-rr	root
				FemaleInfoCollection	26,388 KB	6/24/2020 9:30:13 AM	rw-r	root
				🔬 flink-examples-1.0.jar	61,890 KB	6/24/2020 1:03:26 PM	rw-r	root

Figure 7-5

Step 7 Start the Flink cluster.

Use PuTTY to log in to the ECS and run the **source /opt/client/bigdata_env** command.

[root@node-master2-BWgLh [root@node-master2-BWgLh	~]# ~]#	source	/opt/client/bigdata_env

Figure 7-6

Start the Flink cluster before running the Flink applications on Linux. Run the **yarn session** command on the Flink client to start the Flink cluster. The following is a command example:

/opt/client/Flink/flink/bin/yarn-session.sh -n 3 -jm 1024 -tm 1024

Note: **yarn-session** starts a running Flink cluster on Yarn. Once the session is successfully created, you can use the bin/flink tool to submit tasks to the cluster. The system uses the **conf/flink-conf.yaml** configuration file by default.

Parameters in the yarn-session command:

Mandatory:

-n,--container <arg>: number of Yarn containers(= number of taskmanagers)

Optional:

-D <arg>: dynamic attribute -d,--detached: independent running -jm,--jobManagerMemory <arg>: JobManager memory [in MB] -nm,--name: sets a name for a user-defined application on Yarn. -q,--query: displays the available resources (memory and the number of CPU cores) on Yarn. -qu,--queue <arg>: specifies the Yarn queue. -s,--slots <arg>: number of slots used by each TaskManager -tm,--taskManagerMemory <arg>: memory of each TaskManager [in MB] -z,--zookeeperNamespace <arg>: creates a namespace in the ZooKeeper in HA mode.

After the command is executed, the following result is displayed:





Figure 7-7

The IP address of the JobManager web page is an intranet IP address. Log in to MRS Manager, find the server, and bind an EIP to it. For details about how to bind an EIP, see the related operations in the MRS documents. Use the EIP to replace the intranet IP address and access the server. For example, if the bound IP address is 119.3.4.47, the access address is http://119.3.4.47:42552.

← → C ⁽ Not secure 119.3.4.47:42552/#/running-jobs			
Apache Flink Dashboard	Running Jobs		
8 Overview	Start Time	End Time	
Running Jobs			
Completed Jobs			
🛔 Task Managers			
ob Manager			
▲ Submit new Job			

Figure 7-8

Step 8 Run the JAR package of Flink.

Save the checkpoint snapshot information to HDFS

Press **Ctrl+C** to close the cluster command window (the cluster is still running in the background), or start PuTTY and run the following command:

/opt/client/Flink/flink/bin/flink run --**class** com.huawei.flink.example.checkpoint.FlinkProcessingTimeAPIChkMain /root/flink-examples-1.0.jar --chkPath hdfs://hacluster/flink/checkpoints/

Parameter description: **class** is followed by the full path of the main program entry class and then the JAR package of the program. **chkPath** is the path for storing the checkpoint file. In cluster mode, Flink stores the checkpoint file in HDFS.

The **run** parameter can be used to compile and run a program.

Usage: run [OPTIONS] <jar-file> <arguments>



Run parameters:

-c,--class <classname>: If the entry class is not specified in the JAR package, this parameter is used to specify the entry class.

-m,--jobmanager <host:port>: specifies the address of the JobManager (active node) to be connected. This parameter can be used to specify a JobManager that is different from that in the configuration file.

-p,--parallelism <parallelism>: specifies the degree of parallelism of a program. The default value in the configuration file can be overwritten.

Execution result:

```
[root@node-master1JRhg ~]# source /opt/client/bigdata_env
[root@node-master1JRhg ~]# /opt/client/Flink/flink/bin/flink run --class com.hua
wei.flink.example.checkpoint.FlinkProcessingTimeAPIChkMain /opt/Flink_test/flink
-examples-1.0.jar --chkPath hdfs://hacluster/flink-checkpoint/
^C[root@node-master1JRhg ~]# /opt/client/Flink/flink/bin/flink run --class com.h
wei.flink.example.checkpoint.FlinkProcessingTimeAPIChkMain /root/flink-examples-
1.0.jar --chkPath hdfs://hacluster/flink-checkpoint/
YARN properties set default parallelism to 9
2020-04-16 12:47:22,625 | WARN | [main] | The short-circuit local reads feature
cannot be used because libhadoop cannot be loaded. | org.apache.hadoop.hdfs.sho
rtcircuit.DomainSocketFactory (DomainSocketFactory.java:116)
2020-04-16 12:47:22,625 | WARN | [main] | The short-circuit local reads feature
cannot be used because libhadoop cannot be loaded. | org.apache.hadoop.hdfs.sho
rtcircuit.DomainSocketFactory (DomainSocketFactory.java:116)
2020-04-16 12:47:22,625 | WARN | [main] | The short-circuit local reads feature
cannot be used because libhadoop cannot be loaded. | org.apache.hadoop.hdfs.sho
rtcircuit.DomainSocketFactory (DomainSocketFactory.java:116)
Starting execution of program
```

Figure 7-9

On the Flink management panel, one more running Flink job is displayed.

👌 Apache Flink Dashboard	Running Jobs				
8 Overview	Start Time	End Time	Duration	Job Name	Job ID
Running Jobs	2020-04-16, 17:24:04	2020-04-16, 17:24:08	35	Flink Streaming Job	dc9639f998074926587c72081c2e8599
Completed Jobs					
📥 Task Managers					
😂 Job Manager					
1 Submit new Job					

Figure 7-10

Step 9 View the output.

On the **Task Manager** page of the Flink management panel, click **Stdout** to view the output.



HCIA-Big Data V3.0 Lab Guide

Apache Flink Dashboard	Task Manager	Last Heartbeat: 2020-04-16, 18:10:02	akka.tcp://flink@node-ana-corekxim:32326/user/taskmanager_0
R Overview	Metrics Logs	Stdout	
Running Jobs	Task Manager Outpu	ut	
Completed Jobs	1 2		
Task Managers	3		
Job Manager	5		
🛓 Submit new Job	7 8 9		
	10		
	11		
	13		
	14		
	15		
	16		
	17		
	18		
	20		
	21		
	22		
	23		
	24		
	25		
	26		
	2/		

Figure 7-11

Step 10 View checkpoints.

Start PuTTY and run the HDFS command to view the /flink/checkpoints directory.

[root@node-master1]Rhg ~]# hdfs dfs -ls /flink/checkpoints
2020-04-16 18:16:49,765 INFO obs.OBSFileSystem: This Filesystem GC-ful, clear resource.
Found 4 items
drwxr-xr-x - root hadoop 0 2020-04-16 17:21 /flink/checkpoints/86d77ce7fc06776dae2d6c41419216ef
drwxr-xr-x - root hadoop 0 2020-04-16 18:16 /flink/checkpoints/a83182851598f6632d2a4b55be324f28
drwxr-xr-x - root hadoop 0 2020-04-16 14:00 /flink/checkpoints/cb16da625337a397af5fe263ba9f1da2
drwxr-xr-x - root hadoop 0 2020-04-16 17:59 /flink/checkpoints/dc9639f998074926587c72081c2e8599
[root@node-master1]Rhg ~]#



Step 11 Kill a Flink job.

Run the /opt/client/Flink/flink/bin/flink list command to view the Flink task list.

Figure 7-13

Specify the obtained job ID and run the following command to kill the job:

/opt/client/Flink/flink/bin/flink cancel dc9639f998074926587c72081c2e8599



<pre>[root@node-master1]Rhg ~]# /opt/client/Flink/flink/bin/flink cancel dc9639f998074926587c72081c2e8599</pre>
Cancelling job dc9639f998074926587c72081c2e8599.
YARN properties set default parallelism to 9
2020-04-16 17:59:39,234 WARN [main] The short-circuit local reads feature cannot be used because libhadoop
ortcircuit.DomainSocketFactory (DomainSocketFactory.java:116)
2020-04-16 17:59:39,234 WARN [main] The short-circuit local reads feature cannot be used because libhadoop
ortcircuit.DomainSocketFactory (DomainSocketFactory.java:116)
Cancelled job dc9639f998074926587c72081c2e8599.
[root@node-master1]Rhg ~]#

Figure 7-14

7.3.3 Task 3: Obtaining Top N Hot-Selling Offerings in Flink in Real Time

This exercise illustrates how to develop a complex Flink application for obtaining bestselling offerings in real time.

7.3.3.1 Tasks

- 1. How is data processed based on EventTime and how is Watermark specified?
- 2. How are Window APIs with flexible Flink used?
- 3. When and how is State used?
- 4. How is ProcessFunction used to Implement the TopN function?

7.3.3.2 Exercise Process

- 1. Extract the service timestamp and enable the Flink framework to create a window based on the service time.
- 2. Filter click behavior data.
- 3. Collect statistics on the size of the sliding window every five minutes and aggregate the sliding windows.
- 4. Aggregate by window and output the top *N* offerings with the most clicks in each window.

7.3.3.3 Procedure

Step 1 Prepare a Flink project.

Import the Flink sample project.

Create the **com.huawei.flink.example.goods** package in **src/main/java**.





Figure 7-15

Step 2 Prepare data.

Data file attached to the exercise manual: UserBehavior.csv

This dataset contains all operation data (including click, purchase, add-on, and favorites) of one million users at random on e-commerce websites every day. Each row in the dataset indicates a user behavior, consists of the user ID, offering ID, offering category ID, behavior type, and timestamp, and is separated by comma (,). Each column in the dataset is described as follows:

Table	7-1
-------	-----

Column	Description
User ID (integer type)	Encrypted user ID
Offering ID (integer type)	Encrypted offering ID
Offering category ID (integer type)	ID of the category to which an encrypted offering belongs
Behavior type (enumeration type)	Character string, including pv , buy , cart , and fav
Timestamp	Timestamp when a behavior occurs, in seconds

Create the **resources** folder in the **src/main** directory of the project and save the data file to the folder.



This PC > Desktop > huaweicloud-mrs-example-mrs-1.8 > src > flink-examples > src > main > resources									
^	Name	Date modified	Туре	Size					
	SerBehavior.csv	6/24/2020 10:36 AM	Microsoft Exc	137,664 KB					

Figure 7-16



Figure 7-17

The preceding figure shows the project directory.

Step 3 Compile the HotGoods.Java class.

Create the HotGoods class in the **goods** package. The code is as follows:

package com.huawei.flink.example.goods; import java.io.File; import java.net.URL; import java.sql.Timestamp; import java.util.ArrayList; import java.util.Comparator; import java.util.List; import org.apache.flink.api.common.functions.AggregateFunction; import org.apache.flink.api.common.functions.FilterFunction; import org.apache.flink.api.common.state.ListState;



import org.apache.flink.api.common.state.ListStateDescriptor; import org.apache.flink.api.java.io.PojoCsvInputFormat; **import** org.apache.flink.api.java.tuple.Tuple; import org.apache.flink.api.java.tuple.Tuple1; **import** org.apache.flink.api.java.typeutils.PojoTypeInfo; **import** org.apache.flink.api.java.typeutils.TypeExtractor; **import** org.apache.flink.configuration.Configuration; import org.apache.flink.core.fs.Path; **import** org.apache.flink.streaming.api.TimeCharacteristic; import org.apache.flink.streaming.api.environment.StreamExecutionEnvironment; import org.apache.flink.streaming.api.functions.KeyedProcessFunction; import org.apache.flink.streaming.api.functions.timestamps.AscendingTimestampExtractor; import org.apache.flink.streaming.api.functions.windowing.WindowFunction; **import** org.apache.flink.streaming.api.windowing.time.Time; import org.apache.flink.streaming.api.windowing.windows.TimeWindow; **import** org.apache.flink.util.Collector; public class HotGoods { public static void main(String[] args) throws Exception { // Step 1 Create the execution environment. StreamExecutionEnvironment env = StreamExecutionEnvironment.getExecutionEnvironment(); // Start processing based on EventTime. env.setStreamTimeCharacteristic(TimeCharacteristic. *EventTime*); // To keep the results shown on the console in order, configure concurrency as 1. Changing the concurrency value does not affect the result accuracy. env.setParallelism(1); // The /root directory in the Linux file system of UserBehavior.csv URL fileUrl = HotGoods.class.getClassLoader().getResource("/root/UserBehavior.csv"); Path filePath = Path.fromLocalFile(new File(fileUrl.toURI())); // To extract TypeInformation of UserBehavior, which is PojoTypeInfo PojoTypeInfo<UserBehavior> pojoType = (PojoTypeInfo<UserBehavior>) TypeExtractor.*createTypeInfo*(UserBehavior.class); // To show the sequence of the fields in a specified file because the sequence of fields extracted by Java is uncertain String[] fieldOrder = new String[]{"userId", "itemId", "categoryId", "behavior", "timestamp"}; // To create PojoCsvInputFormat PojoCsvInputFormat<UserBehavior> csvInput = new PojoCsvInputFormat<>(filePath, pojoType, fieldOrder); env // To create data source and obtain DataStream of the UserBehavior type .createInput(csvInput, pojoType) // To extract time and generate watermark .assignTimestampsAndWatermarks(new AscendingTimestampExtractor<UserBehavior>() { @Override public long extractAscendingTimestamp(UserBehavior userBehavior) { // Convert the unit of the source data from seconds to millisecond return userBehavior.timestamp * 1000; } }) // To filter the click data



```
.filter(new <a href="https://www.serBehaviorscomestimationscomestimationscomestimationscomestimationscomestimationscomestimationscomestimationscomestimationscomestimationscomestimationscomestimationscomestimationscomestimationscomestimationscomestimationscomestimationscomestimationscomestimationscomestimationscomestimationscomestimationscomestimationscomestimationscomestimationscomestimationscomestimationscomestimationscomestimationscomestimationscomestimationscomestimationscomestimationscomestimationscomestimationscomestimationscomestimationscomestimationscomestimationscomestimationscomestimationscomestimationscomestimationscomestimationscomestimationscomestimationscomestimationscomestimationscomestimationscomestimationscomestimationscomestimationscomestimationscomestimationscomestimationscomestimationscomestimationscomestimationscomestimationscomestimationscomestimationscomestimationscomestimationscomestimationscomestimationscomestimationscomestimationscomestimationscomestimationscomestimationscomestimationscomestimationscomestimationscomestimationscomestimationscomestimationscomestimationscomestimationscomestimationscomestimationscomestimationscomestimationscomestimationscomestimationscomestimationscomestimationscomestimationscomestimationscomestimationscomestimationscomestimationscomestimationscomestimationscomestimationscomestimationscomestimationscomestimationscomestimationscomestimationscomestimationscomestimationscomestimationscomestimationscomestimationscomestimationscomestimationscomestimationscomestimationscomestimationscomestimationscomestimationscomestimationscomestimationscomestimationscomestimationscomestimationscomestimationscomestimationscomestimationscomestimationscomestimationscomestimationscomestimationscomestimationscomestimationscomestimationscomestimationsco
                               @Override
                               public boolean filter(UserBehavior userBehavior) throws Exception {
                                          // To filter the click data
                                         return userBehavior.behavior.equals("pv");
                               }
                    })
                     .keyBy("itemId")
                     .timeWindow(Time.minutes(60), Time.minutes(5))
                     .aggregate(new CountAgg(), new WindowResultFunction())
                    .keyBy("windowEnd")
                     .process(new TopNHotItems(3))
                     .print();
          env.execute("Hot Items Job");
}
/** To obtain the top N hot items in a window. key indicates the timestamp of the window. The
output is a character string of TopN. */
public static class TopNHotItems extends KeyedProcessFunction<Tuple, ItemViewCount, String> {
          private final int topSize;
          public TopNHotItems(int topSize) {
                    this.topSize = topSize;
          }
          // To save the states of the stored items and click count, and calculate TopN when all data in a
window is collected
          private ListState<ItemViewCount> itemState;
          @Override
          public void open(Configuration parameters) throws Exception {
                    super.open(parameters);
                    ListStateDescriptor<ItemViewCount> itemsStateDesc = new ListStateDescriptor<>(
                               "itemState-state",
                               ItemViewCount.class);
                    itemState = getRuntimeContext().getListState(itemsStateDesc);
         }
          @Override
          public void processElement(
                    ItemViewCount input,
                     Context context,
                    Collector<String> collector) throws Exception {
                    // Each data record is saved in the item state.
                    itemState.add(input);
                    // To register EventTime Timer in windowEnd+1. When triggered, it indicates that all item
data in the windowEnd window has been collected.
                    context.timerService().registerEventTimeTimer(input.windowEnd + 1);
         }
          @Override
          public void onTimer(
```



}

}

```
long timestamp, OnTimerContext ctx, Collector<String> out) throws Exception {
         // To obtain the click count of all received items
         List<ItemViewCount> allItems = new ArrayList<>();
         for (ItemViewCount item : itemState.get()) {
              allItems.add(item);
         }
         // To clear data in state in advance to release space
         itemState.clear();
         // To sort by click count in descending order
         allItems.sort(new Comparator<ItemViewCount>() {
              @Override
              public int compare(ItemViewCount o1, ItemViewCount o2) {
                   return (int) (o2.viewCount - o1.viewCount);
              }
         });
         // To format the ranking information to String for easy display
         StringBuilder result = new StringBuilder();
         result.append("time: ").append(new Timestamp(timestamp-1)).append("\n");
                        for (int i=0; i<allItems.size() && i < topSize; i++) {</pre>
              ItemViewCount currentItem = allItems.get(i);
              // No1: item ID=12224 view count=2413
              result.append("No").append(i).append(":")
                   .append(" Item ID=").append(currentItem.itemId)
.append(" View count=").append(currentItem.viewCount)
                   .append("\n");
         }
         // To control the output frequency and simulate the real-time scrolling result
         Thread.sleep(1000);
         out.collect(result.toString());
    }
/** To output the result of the window */
public static class WindowResultFunction implements WindowFunction<Long, ItemViewCount, Tuple,
TimeWindow> {
    @Override
    public void apply(
         Tuple key, // Primary key of the window, that is, itemId
         TimeWindow window, // Window
         Iterable<Long> aggregateResult, // Result of the aggregate function, that is, count value
         Collector<ItemViewCount> collector // Output type: ItemViewCount
    ) throws Exception {
         Long itemId = ((Tuple1<Long>) key).f0;
         Long count = aggregateResult.iterator().next();
         collector.collect(ItemViewCount.of(itemId, window.getEnd(), count));
    }
/** COUNT of aggregate function implementation. The value increases by 1 each time a record is
generated. */
```



```
public static class CountAgg implements AggregateFunction<UserBehavior, Long, Long> {
     @Override
     public Long createAccumulator() {
          return 0L;
     }
     @Override
     public Long add(UserBehavior userBehavior, Long acc) {
          return acc + 1;
    }
     @Override
     public Long getResult(Long acc) {
          return acc;
    }
     @Override
     public Long merge(Long acc1, Long acc2) {
          return acc1 + acc2;
     }
}
/** Item click count (output type of the window operation) */
public static class ItemViewCount {
                            // Item ID
     public long itemId;
     public long windowEnd; // Window end timestamp
     public long viewCount; // Item click count
     public static ItemViewCount of(long itemId, long windowEnd, long viewCount) {
          ItemViewCount result = new ItemViewCount();
          result.itemId = itemId;
          result.windowEnd = windowEnd;
          result.viewCount = viewCount;
          return result;
    }
}
/** User behavior data structure **/
public static class UserBehavior {
     public long userId; // User ID
     public long itemId;
                               // Item ID
     public int categoryId; // Item category ID
     public String behavior;
                               // User behavior, including ("<u>pv</u>", "buy", "cart", "<u>fav</u>")
                                // Timestamp when the behavior occurs, in seconds
     public long timestamp;
}
}
```

Step 4 Run the program.

Flink can run on a single server or even a single Java virtual machine (VM). This mechanism enables users to test or debug Flink programs locally. Now, run the Flink program locally. You can also refer to task 2 to run the Flink program in the cluster.



Right-click **Run as** and choose **Java Application** from the shortcut menu. Run the main function. The hot-selling offering IDs at each time point are displayed.



Figure 7-18

7.4 Summary

This exercise describes two cases of implementing the asynchronous CheckPoint mechanism and real-time hot-selling offerings, and helps trainees learn multiple core concepts and API usage of Flink, including how to use EventTime, Watermark, State, Window API, and TopN. It is expected that this exercise can deepen your understanding of Flink and help you resolve real-world problems.



8 Kafka Message Subscription Practice

8.1 Background

The Kafka message subscription system plays an important role in big data services, especially in real-time services. The typical Taobao You May Like service uses Kafka to store page clickstream data. After the streaming analysis, the analysis result is pushed to users.

8.2 Objectives

• Understand how to use Kafka shell producers and consumers to generate and consume data in real time.

8.3 Tasks

8.3.1 Task 1: Producing and Consuming Kafka Messages on the Shell Side

Step 1 Log in to Kafka.

Use PuTTY to log in to a server and run the source command to set environment variables.

source /opt/client/bigdata_env

```
[root@node-master1jicC ~]# source /opt/client/bigdata_env
[root@node-master1jicC ~]#
```

Figure 8-1

Run the **cd /opt/client/Kafka/kafka/** command to go to the Kafka directory.

```
[root@node-master1jicC ~]# cd /opt/client/Kafka/kafka/
[root@node-master1jicC kafka]# ls
bin config libs LICENSE logs NOTICE site-docs
[root@node-master1jicC kafka]#
```

Figure 8-2

Step 2 Create a Kafka topic.



Page 98

Run the following command:

bin/kafka-topics.sh --create --zookeeper 192.168.0.151:2181/kafka --partitions 1 --replicationfactor 1 --topic cx_topic2

```
[root@node-master1bBdj kafka]# bin/kafka-topics.sh --create --zookeeper 192.168.0.151:2181/kafka --pa
rtitions 1 --replication-factor 1 --topic cx_topic2
WARNING: Due to limitations in metric names, topics with a period ('.') or underscore ('_') could collid
e. To avoid issues it is best to use either, but not both.
Created topic "cx_topic2".
[root@node-master1bBdj kafka]#
```

Figure 8-3

Note: For details about how to obtain the ZooKeeper IP address, see the related content in the Appendix.

Step 3 View the topic.

Run the following command:

bin/kafka-topics.sh --list --zookeeper 192.168.0.151:2181/kafka

```
[root@node-master1bBdj kafka]# bin/kafka-topics.sh --list --zookeeper 192.168.0.151:2181/kafka
__consumer_offsets
cx_topic1
cx_topic2
[root@node-master1bBdj kafka]#
```

Figure 8-4

Step 4 Create a console consumer.

Run the following command:

bin/kafka-console-consumer.sh --topic cx_topic2 --bootstrap-server 192.168.0.152:9092 --new-consumer --consumer.config config/consumer.properties

```
[root@node-master1bBdj kafka]# bin/kafka-console-consumer.sh --topic cx_topic2 --bootstrap-server 192.16
8.0.152:9092 --new-consumer --consumer.config config/consumer.properties
The --new-consumer option is deprecated and will be removed in a future major release.The new consumer i
s used by default if the --bootstrap-server option is provided.
```

Figure 8-5

Note: The IP address of **bootstrap-server** is the IP address of the Kafka broker. You can obtain the IP address by referring to the related content in the Appendix.

After this command is executed, the **cx_topic2** data is consumed. Do not perform other operations in this window or close the window.

Step 5 Create a console producer.

Log in to PuTTY again, run the source command to obtain the environment variables, and go to the Kafka directory.



[root@node-master1bBdj ~]# source /opt/client/bigdata_env [root@node-master1bBdj ~]# cd /opt/client/Kafka/kafka/ [root@node-master1bBdj kafka]# ls bin config libs LICENSE logs NOTICE site-docs [root@node-master1bBdj kafka]#

Figure 8-6

Run the following command to create a producer:

bin/kafka-console-producer.sh --broker-list 192.168.0.152:9092 --topic cx_topic2 -producer.config config/producer.properties

After the command is executed, enter any data.

```
[root@node-master1bBdj kafka]# bin/kafka-console-producer.sh --broker-list 192.168.0.152:9092 --topic
cx_topic2 --producer.config config/producer.properties
[2020-04-18 13:49:02,887] WARN The configuration 'producer.type' was supplied but isn't a known config.
(org.apache.kafka.clients.producer.ProducerConfig)
[2020-04-18 13:49:02,887] WARN The configuration 'request.required.acks' was supplied but isn't a known
config. (org.apache.kafka.clients.producer.ProducerConfig)
[2020-04-18 13:49:02,888] WARN The configuration 'serializer.class' was supplied but isn't a known confi
g. (org.apache.kafka.clients.producer.ProducerConfig)
>11111
>test
>
```

Figure 8-7

Note: The IP address of **broker-list** is the broker address of Kafka. For details about how to obtain the IP address, see the related content in the Appendix.

Step 6 Test the producer and consumer.

Switch to the shell of the consumer. The console data output is displayed.

```
[root@node-master1bBdj kafka]# bin/kafka-console-consumer.sh --topic cx_topic2 --bootstrap-server 192.16
8.0.152:9092 --new-consumer --consumer.config config/consumer.properties
The --new-consumer option is deprecated and will be removed in a future major release.The new consumer i
s used by default if the --bootstrap-server option is provided.
11111
test
```

Figure 8-8

You can continue to enter data on the producer for testing.

8.3.2 Task 2: Using Kafka Consumer Groups

The consumer group is a very interesting design of Kafka. In terms of high concurrency, multiple servers can be placed in the same consumer group to ensure that all consumers do not pull the same message and the message is complete, thereby improving execution efficiency of the consumers.

Step 1 Create a topic.

Create a topic named **cx_topic3**. For details, see Task 1.



[root@node-master1bBdj kafka]# bin/kafka-topics.shcreatezookeeper 192.168.0.151:2181/kafkapa
rtitions <u>3</u> replication-factor 1topic cx_topic3
WARNING: Due to limitations in metric names, topics with a period ('.') or underscore ('_') could collid
e. To avoid issues it is best to use either, but not both.
Created topic "cx_topic3".
<pre>[root@node-master1bBdj kafka]# bin/kafka-topics.shlistzookeeper 192.168.0.151:2181/kafka</pre>
consumer_offsets
cx_topic1
cx_topic2
cx_topic3
[root@node-master1bBdj kafka]#

Figure 8-9

Note that the topic partition is set to 3. Different value settings will lead to different effects.

For example, to delete the topic, run the bin/kafka-topics.sh --delete --topic cx_*** -- zookeeper 192.168.0.151:2181/kafka command.

Step 2 Create a producer and consumer.

Start the producer.

bin/kafka-console-producer.sh --broker-list 192.168.0.152:9092 --topic cx_topic3 -producer.config config/producer.properties

Open three PuTTY windows, set environment variables, go to the Kafka directory, and run the following command to start three consumers:

bin/kafka-console-consumer.sh --topic cx_topic3 --bootstrap-server 192.168.0.152:9092 --consumer-property group1 --consumer.config config/consumer.properties

Note that you can add **--consumer-property group1** to specify consumer group **group1**.

Step 3 Configure three consumers.

Send the following six messages in sequence in the producer window:



Figure 8-10

Switch to the three consumer windows. It is found that each window consumes two messages evenly.



5

6

[root@node-master1bBdj kafka]# bin/kafka-console-consumer.sh --topic cx_topic3 --bootstrap-server 192.16 8.0.152:9092 --consumer-property group1 --consumer.config config/consumer.properties [2020-04-18 14:35:05,662] WARN The configuration 'group1' was supplied but isn't a known config. (org.ap ache.kafka.clients.consumer.ConsumerConfig) 2

First consumer

Figure 8-11

[root@node-master1bBdj kafka]# bin/kafka-console-consumer.sh --topic cx_topic3 --bootstrap-server 192.16 8.0.152:9092 --consumer-property group1 --consumer.config config/consumer.properties [2020-04-18 14:35:09,096] WARN The configuration 'group1' was supplied but isn't a known config. (org.ap ache.kafka.clients.consumer.ConsumerConfig) 3

Second consumer

Figure 8-12

Figure 8-13

The three consumers evenly consume six messages. Each consumer processes two messages. This ensures data integrity.

Step 4 Start the fourth consumer.

Open another PuTTY window, set environment variables, go to the Kafka directory, and run the following command to start the fourth consumer:

bin/kafka-console-consumer.sh --topic cx_topic3 --bootstrap-server 192.168.0.152:9092 --consumer-property group1 --consumer.config config/consumer.properties

Specify the same consumer group group1.

[root@node-master1bBdj kafka]# bin/kafka-console-consumer.sh --topic cx_topic3 --bootstrap-server 192.16
8.0.152:9092 --consumer-property group1 --consumer.config config/consumer.properties
[2020-04-18 14:44:27,115] WARN The configuration 'group1' was supplied but isn't a known config. (org.ap
ache.kafka.clients.consumer.ConsumerConfig)
Fourth consumer

Figure 8-14

Step 5 Configure four consumers.

Continue to send six messages in sequence in the producer window.



5

а d

6 b



Figure 8-15

Switch to the four consumer windows:

[root@node-master1bBdj kafka]# bin/kafka-console-consumer.sh --topic cx_topic3 --bootstrap-server 192.16 2:9092 --consumer-property group1 --consumer.config config/consumer.properties [2020-04-18 14:35:05,662] WARN The configuration 'group1' was supplied but isn't a known config. (org.ap ache.kafka.clients.consumer.ConsumerConfig) 2

First consumer

Figure 8-16

[root@node-master1bBdj kafka]# bin/kafka-console-consumer.sh --topic cx_topic3 --bootstrap-server 192.1 8.0.152:9092 --consumer-property group1 --consumer.config config/consumer.properties [2020-04-18 14:35:09,096] WARN The configuration 'group1' was supplied but isn't a known config. (org.ap ache.kafka.clients.consumer.ConsumerConfig) 3

Second consumer

Figure 8-17

[root@node-master1bBdj kafka]# bin/kafka-console-consumer.sh --topic cx_topic3 --bootstrap-server 192.16 0.152:9092 --consumer-property group1 --consumer.config config/consumer.properties [2020-04-18 14:35:12,509] WARN The configuration 'group1' was supplied but isn't a known config. (org.ap ache.kafka.clients.consumer.ConsumerConfig) Third consumer

Figure 8-18

[root@node-master1bBdj kafka]# bin/kafka-console-consumer.sh --topic cx_topic3 --bootstrap-server 192.16 8.0.152:9092 --consumer-property group1 --consumer.config config/consumer.properties [2020-04-18 14:44:27,115] WARN The configuration 'group1' was supplied but isn't a known config. (org.ap ache.kafka.clients.consumer.ConsumerConfig)

Fourth consumer

Figure 8-19

As shown in the preceding figure, a consumer does not have a corresponding partition. Therefore, the consumer cannot obtain messages. Therefore, when creating topics, you can create more partitions to ensure that multiple consumers can correspond to the



partitions, preventing consumers from being wasted. To add partitions, you can run the **kafka-reassign-partitions.sh** command.

Step 6 Configure two consumers.

Disable two consumers by pressing Ctrl+C and retain the remaining two consumers.

Enter the following six messages in sequence in the producer window:



Figure 8-20

Check the status of the two consumer windows.

[root@node-master1bBdj kafka]# bin/kafka-console-consumer.sh --topic cx_topic3 --bootstrap-server 192.16
8.0.152:9092 --consumer-property group1 --consumer.config config/consumer.properties
[2020-04-18 14:35:05,662] WARN The configuration 'group1' was supplied but isn't a known config. (org.ap
ache.kafka.clients.consumer.ConsumerConfig)
2
5
a First consumer
d
111
444

Figure 8-21

[root@node-master1bBdj kafka]# bin/kafka-console-consumer.sh --topic cx_topic3 --bootstrap-server 192.16
8.0.152:9092 --consumer-property group1 --consumer.config config/consumer.properties
[2020-04-18 14:35:09,096] WARN The configuration 'group1' was supplied but isn't a known config. (org.ap
ache.kafka.clients.consumer.ConsumerConfig)
3
6
b
e
Second consumer
222
333
555
666

Figure 8-22

When two consumers are used, the unexpected phenomenon also occurs. That is, messages are not evenly distributed. Instead, they are divided into four messages and two messages. The reason is that one consumer corresponds to two partitions, and the other corresponds to one partition. Messages are consumed based on partitions.

Check the partitions of the **cx_topic3** topic.



View the consumer group list.

bin/kafka-consumer-groups.sh --bootstrap-server 192.168.0.152:9092 --list

```
[root@node-master1bBdj kafka]# bin/kafka-consumer-groups.sh --bootstrap-server 192.168.0.152:9092 --list
Note: This will not show information about old Zookeeper-based consumers.
example-group1
[root@node-master1bBdj kafka]#
```

Figure 8-23

View the details about the consumer group.

bin/kafka-consumer-groups.sh --bootstrap-server 192.168.0.152:9092 --describe --group example-group1

[root@node-master1bBdj kafka]# bin/kafka-consumer-groups.shbootstrap-server 192.168.0.152:9092describegroup example-group1 Note: This will not show information about old Zookeeper-based consumers.											
TOPIC	PARTITION	CURRENT-OFFSET	LOG-END-OFFSET	LAG	CONSUMER-ID	HOST	CLIENT-ID				
cx_topic3	0	10	10	0	consumer-1-404604b8-be64-4a1d-9a15-42	b7bf5f475f /192.168.0.151	consumer-1				
cx_topic3	1	11	11	0	consumer-1-404604b8-be64-4a1d-9a15-42	b7bf5f475f /192.168.0.151	consumer-1				
cx_topic3	2	10	10	0	consumer-1-9667f89c-4b0f-4b89-bbe8-c4	745c80f004 /192.168.0.151	consumer-1				
cx topic1	0	10	10	0	-	-	-				
cx topic2	0	4	4	0	-	-	-				
[root@node-master1bBdj kafka]#											

Figure 8-24

As the figure shows, the **consumer_id** values of **partition0** and **partition1** are both **consumer-1-404604b8-be64-4a1d-9a15-42b7bf5f475f**.

Sometimes, the data consumption sequences of different partitions are different. This is because Kafka messages are stored by partition, and only messages in the same partition are pulled in sequence.

8.4 Summary

This exercise describes how to generate and consume data in real time the shell end and enables trainees to have a deeper understanding of Kafka. Multiple consumers in the same consumer group are equivalent to one consumer, which improves consumption efficiency.


9 Flume Data Collection Practice

9.1 Background

Flume is an important data collection tool in the big data components, and is often used to collect data from various data sources for other components to analyze. In the log analysis service, server logs are collected to analyze whether servers are running properly. In real-time services, data is usually collected to the Kafka for analysis and processing by real-time components such as streaming and Spark. Flume is an important application in big data services.

9.2 Objectives

• Configure and use Flume to collect data.

9.3 Tasks

9.3.1 Task 1: Installing the Flume Client

Step 1 Open the Flume service page.

Access the MRS Manager cluster management page and choose **Services** > **Flume**.



Da	shboard	Service	es	Hosts	Alarm	IS	Audit	
Per	riod Real t	ime	•	Q View	🖸 Ехро	rt		
9	Service Sum	mary					-	к Ч
	Service 💠		Health S	Status 🗘	Roles			^
	DBService		🕑 Good	ł	DBServer:	1		
	Flink		🕑 Good	ł	FlinkReso	urce: 1		
	Flume		🕑 Good	ł	Flume: 2	Monit	orServer: 1	
r: 2	RESTServer:	1 HMaste	r: 1 Thr	iftServer1: 0	RegionSe	rver: 1	ThriftServ	

Figure 9-1

Step 2 Click Download Client.

Dashboard	Services	Hosts	Alarms	Audit	Te	
Service Flume >	Service Flume > Service Status					
Service Stat	us Instance	Service Conf	iguration Re	source Distributio	on	
O Start Serv	ice Stop	Service 🛓	Download Client	More •		
Flume Sumr	mary				•	
Health Statu	IS	🥑 G	ood			
Configuratio	on Status	🥑 Sj	ynchronized			
Version		1.6.0				

Figure 9-2



Click OK	and wait	for the	download.
-----------------	----------	---------	-----------

Download Client			
Warning: Generating a client will occupy a large number of disk I/Os. You are advised not to download a client when the cluster is being installed, started, and patched, or in other unstable states.			
* Client Type O All client files Only configuration files			
* Download to 💿 Server 🗌 Remote host			
Save to only the following path on the server. If the client file already exists under the path, it will be replaced.			
Client Path /tmp/MRS-client			
OK Cancel			

Figure 9-3

After the download is complete, a dialog box is displayed, indicating the server (Master node) to which the file is downloaded. The path is **/tmp/MRS-client**.



Figure 9-4



Step 3 Decompress the Flume client installation package.

Use MobaXterm to log in to the ECS of the preceding step and go to the **/tmp/MRS-client** directory.

```
[root@node-master1jicC ~]# cd /tmp/MRS-client/
[root@node-master1jicC MRS-client]# ls
MRS_Flume_Client.tar
[root@node-master1jicC MRS-client]#
```

Figure 9-5

Run the following command and decompress the package to obtain the verification file and client configuration packages:

tar -xvf MRS_Flume_Client.tar

```
[root@node-master1jicC MRS-client]# tar -xvf MRS_Flume_Client.tar
MRS_Flume_ClientConfig.tar
MRS_Flume_ClientConfig.tar.sha256
[root@node-master1jicC MRS-client]#
```

Figure 9-6

Step 4 Verify the file package.

Run the sha256sum -c MRS_Flume_ClientConfig.tar.sha256 command.

If the following information is displayed, the file package is successfully verified:

MRS_Flume_ClientConfig.tar: OK

Step 5 Decompress MRS_Flume_ClientConfig.tar.

Run the tar -xvf MRS_Flume_ClientConfig.tar command.

```
[root@node-master1jicC MRS-client]# tar -xvf MRS Flume ClientConfig.tar
MRS_Flume ClientConfig/
MRS_Flume_ClientConfig/switchuser.py
MRS Flume ClientConfig/jython-standalone-2.7.0.jar
MRS Flume_ClientConfig/Flume/
MRS_Flume_ClientConfig/install.sh
MRS_Flume_ClientConfig/hosts
MRS_Flume_ClientConfig/ca.crt
MRS_Flume_ClientConfig/bigdata_env.sample
MRS_Flume_ClientConfig/conf.py
MRS_Flume_ClientConfig/log4j.properties
MRS_Flume_ClientConfig/install.bat
MRS Flume ClientConfig/JDK/
MRS Flume ClientConfig/application.properties
MRS_Flume_ClientConfig/refreshConfig.sh
MRS_Flume_ClientConfig/JDK/install.sh
MRS_Flume_ClientConfig/JDK/component_env.sample
MRS Flume ClientConfig/JDK/jdk.tar.gz
MRS Flume ClientConfig/Flume/FusionInsight-Flume-1.6.0.tar.gz
[root@node-master1jicC MRS-client]# |
```

Figure 9-7

Step 6 Install the Flume environment variables.



Run the following command to install the client running environment to the new directory **/opt/Flumeenv**.

The directory is generated automatically during installation.

sh /tmp/MRS-client/MRS_Flume_ClientConfig/install.sh /opt/Flumeenv

Check the command output. If the following information is displayed, the client running environment has been successfully installed:

Components client installation is complete.

```
[19-03-27 20:55:12]: Install JDK begin ...
[19-03-27 20:55:12]: Decompress jdk.tar.gz to /opt/Flumeenv/JDK.
/tmp/MRS-client/MRS_Flume_ClientConfig/JDK
[19-03-27 20:55:17]: Create JRE env file "/opt/Flumeenv/JDK/component_env".
[19-03-27 20:55:17]: JDK installation is complete.
[19-03-27 20:55:17]: Components client installation is complete.
[root@node-master1jicC MRS-client]#
```

Figure 9-8

Step 7 Configure the environment variables.

Run the source /opt/Flumeenv/bigdata_env command.

Step 8 Decompress the Flume client.

Run the following commands:

cd /tmp/MRS-client/MRS_Flume_ClientConfig/Flume tar -xvf FusionInsight-Flume-1.6.0.tar.gz

```
[root@node-master1jicC MRS-client]# cd /tmp/MRS-client/MRS_Flume_ClientConfig/Flume
[root@node-master1jicC Flume]# ls
FusionInsight-Flume-1.6.0.tar.gz
[root@node-master1jicC Flume]# tar -xvf FusionInsight-Flume-1.6.0.tar.gz
flume/
flume/CHANGELOG
flume/DEVNOTES
flume/LICENSE
flume/NOTICE
```

Figure 9-9

Step 9 Install the Flume client.

Install Flume to the new directory **/opt/FlumeClient**. The directory is automatically generated during installation.

Run the following command:

sh /tmp/MRS-client/MRS_Flume_ClientConfig/Flume/install.sh -d /opt/FlumeClient

```
[root@node-master1jicC Flume]# sh /tmp/MRS-client/MRS_Flume_ClientConfig/Flume/in
-d /opt/FlumeClient
CST 2019-03-27 21:00:24 [flume-client install]: install flume client successfully.
[root@node-master1jicC Flume]#
```

Figure 9-10



-d: The Flume client installation path.

If the following information is displayed, the client running environment is successfully installed:

install flume client successfully.

Step 10 Copy the HDFS configuration file.

Run the following commands to copy the HDFS configuration file to the **conf** directory of Flume:

cp /opt/client/HDFS/hadoop/etc/hadoop/hdfs-site.xml /opt/FlumeClient/fusioninsight-flume-1.6.0/conf/ cp /opt/client/HDFS/hadoop/etc/hadoop/core-site.xml /opt/FlumeClient/fusioninsight-flume-1.6.0/conf/

Step 11 Restart the Flume service.

Go to the **/opt/FlumeClient/fusioninsight-flume-1.6.0** directory and restart the Flume.

Run the following commands:

cd /opt/FlumeClient/fusioninsight-flume-1.6.0 sh bin/flume-manage.sh restart

```
[root@node-master1jicC fusioninsight-flume-1.6.0]# sh bin/flume-manage.sh restart
Stop Flume PID=15340 successful.
Start flume successfully,pid=6815.
[root@node-master1jicC fusioninsight-flume-1.6.0]#
```

Figure 9-11

9.3.2 Task 2: Using SpoolDir to Collect and Upload Data to HDFS

Flume uses SpoolDir to monitor folders in a specified path and then collects and uploads the data in the folders to HDFS. Check that the HDFS and HBase clients are installed. Flume is mainly used for data collection. Therefore, you need to configure the Flume based on service requirements.

Step 1 Download the Flume configuration planning tool.

Visit https://support.huawei.com/enterprise/en/doc/EDOC1000113257.

Step 2 Enable macros.

After the decompression, start the Flume configuration planning tool. If macros are disabled, enable them. Otherwise, the tool does not work.



Figure 9-12

Step 3 Configure parameters.



In the **Flume Name** row of the first sheet, select **client**. Then, switch to the **Flume Configuration** row of the second sheet.

FusionInsight V100R002C60U10 Flume Configuration Planning Tool				
, Tool Version	V1.0.0			
Language Selection	English			
Applicable for	V100R002C60U10			
Function Description	1. Supports to generate all configuration files required for Flume.			
Flume Name	client			
Remarks	 The required configuration items cannot be empty. The generated configuration file is in the same level directory of this tool. 			

Figure 9-13

Step 4 Configure the source.

Click Add Source and configure the parameters as follows:

SourceName: s1. The value cannot be empty.

type: spooldir. In this exercise, data in the static folder is collected and uploaded to the HDFS.

spoolDir: /tmp/flume_spooldir, which is the folder monitored by Flume.

channel: s-c1. The value cannot be empty.

Retain the default values for other parameters, as shown in the following figure:



	Add Source	
Source configuration	Source configuration	Source configuration
SourceName	Source name. The value cannot be	s1
type	Source type. The value can be any	spooldir
spoolDir	Directory where the file to be collected.	
fileSuffix	Suffix added to the file after the collection is	. COMPLETED
deletePolicy	Source file deletion strategy after file	never
trackerDir	Path for storing the metadata of files collected	.flumespool
ignorePattern	File name regular expression for ignoring.	^\$
batchSize	Number of events that Flume sends (number of data	1000
inputCharset	Encoding method used when Flume reads files	UTF-8
deserializer	Way in which Flume reads files.	LINE
selector.type	Way in which Flume sends data to the channel	replicating
fileHeaderKey	Saves the key name for the absolute path of	file
fileHeader	Whether to save the absolute path for the file in	false
basenameHeader	Whether to save the current file	true

Figure 9-14



HCIA-Big Data V3.0 Lab Guide

basenameHeader Key	name of the current file name	basename
deserializer.m axBatchLine	Specifies the number of rows merged into an event when Flume	1
deserializer.m axLineLength	Specifies the number of characters to be	2048
channels	that the current source reads to a specified	c1

Figure 9-15



	Add Source	
configurat:	io configuration	configuration
SourceName	Source name. The value	s1
type	Source type. The value can	spooldir
spoolDir	Directory where the file to be	/tmp/flume_spool dir
fileSuffix	Suffix added to the file after	. COMPLETED
deletePolicy	Source file deletion strategy	never
trackerDir	Path for storing the metadata of files collected	.flumespool
ignorePatter	File name regular n expression for ignoring	^\$
batchSize	Number of events that Flume sends (number of data	1000
inputCharset	Encoding method used when Flume	UTF-8
deserializer	Way in which Flume reads files.	LINE



selector.type	Way in which Flume sends data	replicating
fileHeaderKey	Saves the key name for the	file
fileHeader	Whether to save the absolute path	false
basenameHeader	Whether to save the current file	true
basenameHeader Key	Saves the key name of the	basename
deserializer.m axBatchLine	Specifies the number of rows merged into an <u>event when Flume</u>	1
deserializer.m axLineLength	Specifies the number of characters to be	2048
channels	Sends the data that the current source reads to a	c1

Figure 9-17

Step 5 Configure a channel.

Click Add Channel. Set ChannelName to c1, which corresponds to that in Source, set type to memory, and retain the default values for other parameters.



	Add Channel				
Channel	Channel	Channel			
configuration	configuration	configuration			
ChanelName	Chanel name. The value cannot be	c1			
type	Chanel type. The value can be any end of file and	memory			
capacity	Number of events in buffer.The walue cappot be	10000			
transactionCap acity	Transaction size: the number of events in a	1000			
channlefullcou nt	Number of times for channel full. An alarm is sent	10			
keep-alive	Timeout period of event deleting after an event is	3			
byteCapacity	Maximum memory occupied by events in buffer				
byteCapacityBu fferPercentage	Percentage of the remaining memory size When the	20			

Figure 9-18

Step 6 Configure a sink.

Click **Add Sink** and configure the parameters as follows:

SinkName: sh1

type: hdfs

hdfs.path: hdfs://hacluster/user/stu02/. The stu02 directory is automatically created by the system.

channel:-c1. The name is the same as the channel name.

Retain the default values for other parameters, as shown in the following figure:



٢

confi	Sink guration	Sink configuration description	Sink configuration content
SinkNa	ame	Sink name. The value cannot be empty and	shl
type		can be any one of	hdfs
hdfs.p	oath	HDFS data write directory. The	hdfs://hacluster/user /stu02/
hdfs.f x	filePrefi	File prefix after the file is written to the HDES	over_%{basename}
hdfs.f x	fileSuffi	File suffix after the file is written to the	
hdfs.i ix	nUsePref	Prefix of the file that is being written	
hdfs.i ix	nUseSuff	Suffix of the file that is being written	.tmp

Figure 9-19



hdfs.threadsPoo lSize	Thread pool size during HDFS writing.	10
hdfs.rollTimerP oolSize	Thread pool size for rolling file generation	1
hdfs.kerberosPr incipal	Kerberos authentication user. Must fill in the	
hdfs.kerberosKe ytab	KeyTab file path for Kerberos authentication Must	
hdfs.round	Whether to generate documents in accordance with the	false
hdfs.roundUnit	Time range unit.	second
hdfs.useLocalTi meStamp	Whether to use the local time.The value can be true or false	false
hdfs.failcount	Number of failures for consecutively data writings to the HDES	10
hdfs.fileCloseB yEndEvent	Whether to be closed the HDFS file that is being written based on	true
channel	Channel to which data read by Sink is sent. The perameter cappet	c1

Note: The MRS cluster is in non-security mode. Therefore, you do not need to configure Kerberos in the sink.

Step 7 Generate a configuration file.

Click **Generate a configuration file** in the upper right corner. A **properties.properties** configuration file is automatically generated in the Excel file and saved in the same directory as the configuration tool.

	Add Source		Add Channel		Add Sink		Add Channel		Generate	a configuration	n file
Source configuration	Source configuration	Source configuration	Channel configuration	Channel configuration	Channel configuration	Sink configuration	Sink configuration description	Sink configuration content			
SourceName	Source name. The value cannot be	sl	ChanelName	Chanel name. The value cannot be	c1	SinkName	Sink name. The value cannot be empty and	sh1			

Figure 9-21





Figure 9-22

Step 8 Create /tmp/flume_spooldir in Linux

The commands are as follows:

<pre>[root@node-master1jicC</pre>	~]# cd /tmp		
<pre>[root@node-master1jicC</pre>	tmp]# mkdir	flume	_spooldir/
[root@node-master1jicC	tmp]#		

```
Figure 9-23
```

Step 9 Upload the Flume configuration file.

Use WinSCP to upload **properties.properties** to the following directory:

Quick connect				
📧 🛓 Ŧ 🕘 🍋 📗 😣 🖹	1 🙏 🖿			
/opt/FlumeClient/fusioninsight-flume-1.6.0)/conf/			
Vame Name	Size (KB)	Last modified	Owner	Group
1				
儿 monitormanager		2019-03-27	root	root
📙 service		2019-03-27	root	root
dient.properties.properties	2	2019-03-27	root	root
ENV_VARS	1	2019-03-27	root	root
flume-check.properties	1	2019-03-27	root	root
🗧 🔀 flume-client.conf	1	2019-03-27	root	root
flume-conf.properties.template	1	2019-03-27	root	root
flume-env.ps1.template	1	2019-03-27	root	root
lume-env.sh	1	2019-03-27	root	root
flume-env.sh.template	1	2019-03-27	root	root
FlumeMetric.properties	1	2019-03-27	root	root
🖹 log4j.properties	3	2019-03-27	root	root
plugin.conf	1	2019-03-27	root	root
🖌 properties.properties	2	2019-03-27	root	root
server.properties.properties	3	2019-03-27	root	root



Note: The Flume client automatically loads the properties.properties file.

Step 10 Write a file for testing.

Go to the **/tmp/flume_spooldir** directory, run the **vi** command to create the **test1.txt** file, and enter any content.

[root@node-master1jicC flume_spooldir]# vi test1.txt
[root@node-master1jicC flume spooldir]#

Figure 9-25

Step 11 View the result.

```
[root@node-master1jicC flume_spooldir]# hdfs dfs -ls /user/stu02
Found 1 items
-rw-r--r- 1 root hive 26 2019-03-28 15:41 /user/stu02/over_test1.txt
[root@node-master1jicC flume_spooldir]#
```

Figure 9-26

The data is successfully collected and uploaded to the HDFS. You can continue to create a data file for testing.

9.3.3 Task 3: Using SpoolDir to Collect and Upload Data to Kafka

Flume uses SpoolDir to monitor folders in a specified path and then collects and uploads the data in the folders to Kafka. The consumers can read the data on the console.

Step 1 Modify the Flume configuration file.

On the tool description page of the configuration tool, change **server** to **client**.

r	
Flume Name	client
-	

Figure 9-27

If the FlumeServer is deployed in a cluster, set this parameter to **server**. If the FlumeServer is not deployed in a cluster, set this parameter to **client**.

Step 2 Modify Sink configurations.

In the Flume configuration planning tool, change **type** of sink to **kafka** and set the value of **kafka.bootstrap.servers**.



	Add Sink	
Sink configuration itom	Sink configuration description	Sink configuration content
SinkName	Sink name. The value cannot be	shl
type	Sink type. The value can be any execut UDES_UDES_	kafka
kafka.topic	default is "default-flume-	cx_topic1
flumeBatchSize	by Flume at a time (data	1000
kafka.producer.t ype	Methods for sending data by Flume (sync or async).	sync
kafka.bootstrap. servers	ports on Kafka. The	192.168.0.152:9092
kafka.security.pr otocol	the security Kafka, the	SASL_PLAINTEXT
requiredAcks	Number of ACKs returned from Kafka to Producer.	0
channel	read by Sink is sent. This	c1

kafka.topic: cx_topic1

kafka.bootstrap.servers: 192.168.0.152:9092. If there are multiple Kafka instances in the cluster, you need to configure all of them. If the Kafka is installed in the cluster and configuration has been synchronized, you do not need to configure this parameter.

kafka.security.protocol: PLAINTEXT. The cluster used in this exercise is a non-security cluster.

After the configuration is complete, click **Generate a configuration file**.

Step 3 Create a Kafka topic.

Go to the Kafka directory **cd /opt/client/Kafka/kafka** and run the following command:

bin/kafka-topics.sh --create --zookeeper 192.168.0.151:2181/kafka --partitions 1 --replicationfactor 1 --topic cx_topic1



^C[root@node-master1bBdj kafka]# bin/kafka-topics.shcreatezookeeper 192.168.0.151:2181/kafka
rtitions 1replication-factor 1topic cx_topic1
WARNING: Due to limitations in metric names, topics with a period ('.') or underscore ('_') could collid
e. To avoid issues it is best to use either, but not both.
Created topic "cx_topic1".
[root@node-master1bBdj kafka]#

Note: You can obtain the IP address of the ZooKeeper by referring to the related content in the Appendix.

Step 4 Upload the Flume configuration file.

Use WinSCP to upload **properties.properties** to the following directory:

/opt/FlumeClient/fusioninsight-flum	e-1.6.0/cor	nf/		
Quick connect]
) <i>»</i>			
/opt/FlumeClient/fusioninsight-flume-1.6.0	/conf/			v
5 Name	Size (KB)	Last modified	Owner	Group
00 IL				
nonitormanager		2019-03-27	root	root
 Image: Image: Ima		2019-03-27	root	root
dient.properties.properties	2	2019-03-27	root	root
B ENV_VARS	1	2019-03-27	root	root
👔 flume-check.properties	1	2019-03-27	root	root
flume-client.conf	1	2019-03-27	root	root
g flume-conf.properties.template	1	2019-03-27	root	root
flume-env.ps1.template	1	2019-03-27	root	root
flume-env.sh	1	2019-03-27	root	root
flume-env.sh.template	1	2019-03-27	root	root
FlumeMetric.properties	1	2019-03-27	root	root
Iog4j.properties	3	2019-03-27	root	root
Plugin.conf	1	2019-03-27	root	root
properties.properties	2	2019-03-27	root	root
server.properties.properties	3	2019-03-27	root	root
1				

Figure 9-30

Note: The Flume client automatically loads the **properties.properties** file.

Step 5 Create a console consumer.

Run the following command in the Kafka directory:

bin/kafka-console-consumer.sh --topic cx_topic1 --bootstrap-server 192.168.0.152:9092 --new-consumer --consumer.config config/consumer.properties



```
[root@node-masterljicC kafka]# bin/kafka-console-consumer.sh --topic topic-1001 --b
ootstrap-server 192.168.0.250:9092 --new-consumer --consumer.config config/consumer
.properties
The --new-consumer option is deprecated and will be removed in a future major relea
se.The new consumer is used by default if the --bootstrap-server option is provided
.
```

Note: The IP address of **bootstrap-server** corresponds to the IP address of the Kafka instance. You can obtain the IP address by referring to the related content in the Appendix.

After this command is executed, the **cx_topic1** data is consumed. Do not perform other operations in this window or close it.

Step 6 Test data.

On PuTTY, open a shell connection (do not close the consumer window that is just started) and go to the **/tmp/flume_spooldir** directory.

Run the **vi** command to edit the **testkafka.txt** file, enter any content, save the file, and exit.

```
[root@node-master1jicC flume_spooldir]# vi testkafka.txt
[root@node-master1jicC flume_spooldir]# cat testkafka.txt.COMPLETED
hello world
hello ketty
[root@node-master1jicC flume_spooldir]#
```

Figure 9-32

Step 7 View the result.

Switch back to the shell window of the consumer. The data output is displayed.

```
[root@node-master1bBdj kafka]# bin/kafka-console-consumer.sh --topic cx_topic1 --bootstrap-server 192.16
8.0.152:9092 --new-consumer --consumer.config config/consumer.properties
The --new-consumer option is deprecated and will be removed in a future major release.The new consumer i
s used by default if the --bootstrap-server option is provided.
hello world
hello ketty
```

Figure 9-33

9.4 Summary

This exercise mainly describes how to collect data using Flume SpoolDir and Avro sources. This exercise aims to help trainees better understand Flume by learning common offline and real-time data collection methods.



10 Loader Data Import and Export Practice

10.1 Background

Big data services often involve data migration, especially data migration between relational databases and big data components. Loader is often used to migrate data between MySQL and HDFS/HBase. The graphical operations of Loader make data migration easier.

10.2 Objectives

• Use Loader to migrate data in service scenarios.

10.3 Tasks

10.3.1 Task 1: Preparing MySQL Data

Step 1 Apply for the MySQL service.

Log in to the HUAWEI CLOUD website at https://www.huaweicloud.com/en-us/ and choose **Products > Database > RDS for MySQL**.





Figure 10-1

Click **Buy Now** and configure the database instance information as follows:

Billing Mode: Pay-per-use

Region: CN East-Shanghai2 (the same region as MRS)

DB Instance Name: Enter a custom name. In this exercise, **rds_loader** is used as an example. The instance name must start with a letter and contain 4 to 64 characters. Only letters, digits, hyphens (-), and underscores (_) are allowed.

DB Engine: MySQL

DB Engine Version: 5.6

DB Instance Type: Single

AZ: default value

Time Zone: default value

Instance Class: 1 vCPU | 2 GB

Storage Type: Ultra-high I/O

Storage Space: 40 GB

Disk Encryption: Disable

VPC: default value (the same network as MRS)

Security Group: default value (the same security group as MRS)

Administrator: root

Administrator Password: set the password as required.

Parameter Template: default value

Tag: not specified

Quantity: 1



Confirm the information and click **Next**.

HUAWEI	HUAWEI CLOUD Console	e 🔹 Shanghai1	¥		Search	Q Billing	Center Resource
≡	RDS	Relational Da	tabase Service ③				
(i) (i)	Instance Management	() We would	much appreciate it if you could complete our questionna	aire on RDS. Your feedback will	help us provide a bette	r user experience.	
00	Backup Management						
	Parameter Template Management	Renew	Unsubscribe Change to Yearly/Monthly	Reboot		All DB engi	nes 💌
	Task Center		Name/ID ↓Ξ	Description	DB Instanc ↓Ξ	DB Engine Version ↓Ξ	Status
0	Recycling Management		rds_loader 8f6980b5c96f4b38a0c12f8c6163a622in01 🗇		Primary/Standby	MySQL 5.6.47	:)): Creating

Figure 10-2

Step 2 Log in to MySQL.

After the RDS for MySQL DB instance is created, click **Log In** and enter username **root** and password to log in to the MySQL DB instance.

Relational D	atabase Service ⑦					⑦ Ala	rm Rule Setting 🏼 🛛 Usage W	izard 🕼 Help Guide 🛛 Buy DB Ir
🚺 We would	much appreciate it if you could complete our questionnal	re on RDS. Your feedback will	help us provide a better	r user experience.				
Renew	Unsubscribe Change to Yearly/Monthly	Reboot		All DB engines	5 v DB instance	name 🔻	Q	Search by Tag 😸 🖸 🔞
	Name/ID ↓Ξ	Description	DB Instanc ↓Ξ	DB Engine Version ↓Ξ	Status	Billing Mode	Floating IP Address	Operation
	rds_loader 8f699005c96f4b38a0c12f8c6163a622in01		Primary/Standby	MySQL 5.6.47	Available	Pay-per-use Created on Jun 23, 2020	192.168.0.124	Log In View Metric More 🗸

Figure 10-3

The MySQL data service management page is displayed.

Data Admin Servîce	SQL Operation	Database Management	Import Export
Home			
DB Instance Name: rd	ls-loader1 DB E	Engine Version: mysql 5.7.29	
Database List + Create Database			
Database Name 🖕		Table Quantity 🍦	Table Size 🍦

Figure 10-4

Step 3 Create a database.

Click **Create Database**, enter a database name, for example, **rdsdb**, set **Character Set** to **utf8**, and click **OK**.



Create Database ×				
* Db Name	rdsdb			
Character Set	utf8	\sim		
OK Cancel				

Figure 10-5

Step 4 Create a table.

In the list on the left, choose **rdsdb**. On the displayed page, click **Create Table**, name the table, and change the character set, as shown in the following figure:

1 Basic Information —		2 Column
* Table Name	cx_student	
Storage Engine	InnoDB	\sim
Character Set	utf8	\sim
Collation	utf8_general_ci	\sim
Comment		
Advanced Settings \lor		

Figure 10-6

Click Next, and then Add, and set the fields as follows:



1 Basic Information				2 Colu	mn			(
Add Insert Delete Move Up Move Down									
NO.	Column	Name	Туре		Length		Nullable	Primary Key	
1	id			\sim					
2	name			\sim			~		
3	gender			\sim			~		
4	age			\sim			~		



Set **id** to the primary key and click **Next**. Do not set the index and foreign key. Then click **Create Now**.

NOT NULL,			
nor norr,			
NILL			
D NULL			
U NOLL,			
= utf8			
l ci:			
,			
T	NULL, IED NULL, T = utf8 al_ci;	NULL, IED NULL, T = utf8 al_ci;	NULL, IED NULL, T = utf8 al_ci;

Figure 10-8

Click Execute.

Step 5 Insert data.

Click the SQL operation button in the upper part, select **SQL Window**, select the **rdsdb** database on the left, and enter the following statement in the SQL window:

insert into cx_student(id,name,gender,age) VALUES('1001','MacDonald','male','30'); insert into cx_student(id,name,gender,age) VALUES('1002','Calvin','male','25'); insert into cx_student(id,name,gender,age) VALUES('1003','Haley','female','18'); insert into cx_student(id,name,gender,age) VALUES('1004','Madonna','female','22'); insert into cx_student(id,name,gender,age) VALUES('1005','Randell','male','36');



Home	Database Management-rdsd	lb × SQL Window ×	
Curr	ent Database: I'dsdb ()	Master Switch SQL Exect	ation Node Instance Name: rds-loader IP Address: 192.168.0.161 Port: 3306
Dbnam Table	e: rdsdb \lor View	• Execute SQL(F8) analysis and diagnosis	SQL Tuning Format(F9) Format(F9) Go now Stored SQL V
Please	search by ke Q C	<pre>1 insert into cx_stud 2 insert into cx_stud 3 insert into cx_stud 4 insert into cx_stud 5 insert into cx_stud</pre>	<pre>ent(id,name,gender,age) VALUES('1001','MacDonald','male','30'); ent(id,name,gender,age) VALUES('1002','Calvin','male','25'); ent(id,name,gender,age) VALUES('1003','Haley','female','18'); ent(id,name,gender,age) VALUES('1004','Madonna','female','22'); ent(id_name,gender,age) VALUES('1005','Rendell','male','22');</pre>
	indexes	Executed SQL Statements	Message
		Time	SQL Statement

Figure 10-9

Click Execute SQL to insert data.

10.3.2 Task 2: Configuring the MySQL Driver of Loader

In the Loader service of MRS, the default MySQL connection JAR package is 5.1.12. Therefore, you need to update the MySQL connection JAR package.

Step 1 Download the MySQL driver package.

Visit http://mvnrepository.com/artifact/mysql/mysql-connector-java/5.1.21 to go to the maven repository and download the MySQL JDBC driver **mysql-connector-java-5.1.21.jar**. Make sure that the version number is the same. Click **jar** to download the driver.

← → C ☆ 🛈 mvnreposit	tory.com/artifact/mysql	/mysql-connector-java/5.1.21					
MVNREPOSITORY	Sea	rch for groups, artifacts, categories					
Indexed Artifacts (11.5M)	s (11.5M) Home » mysql » mysql-connector-java » 5.1.21						
5754k	JDBC Type 4	driver for MySQL					
0 2004 2018	License	GPL 2.0					
	Categories	MySQL Drivers					
Popular Categories	HomePage	http://dev.mysql.com/usingmysql/java/					
Aspect Oriented Date		(May 08, 201 <u>5)</u>					
Actor Frameworks	Files	pom (1 KB) jar (808 KB) View All					
Application Metrics	Repositories	Central Aspose					
Build Tools	Used By	2,708 artifacts					
Bytecode Libraries							
Command Line Parsers	Note: There is a new	version for this artifact					
Cache Implementations	New Yessien	0.0.12					
Cloud Computing	New Version	8.0.12					
Code Analyzers							
Collections	Maven Gradle S	BT Ivy Grape Leiningen Buildr					
Configuration Libraries	<pre><!-- https://mypreposi</pre--></pre>	tory com/artifact/mysal/mysal-connector-iava>					
Core Utilities		······································					
Date and Time Utilities	<pre></pre>	oupla/ connector-java					
Dependency Injection	<pre></pre>	ersion>					
Embedded SQL Databases							



Figure 10-10

Step 2 Upload the JAR file.

Start WinSCP, connect to the master node, and upload the MySQL JAR package to the **/opt/Bigdata/MRS_2.1.0/1_18_Sqoop/install/FusionInsight-Sqoop-1.99.7/server/jdbc** directory.

Name	Size (KB)	Last modified	Owner	Group
asidhc4-1-1-0 iar	452	2010-11-12		fcommon
asidbc4-V100R003C10SPC11	1	2019-11-13	omm	wheel
jdbc.properties	1	2019-11-13	omm	ficommon
🔬 mysql-connector-java-5.1.21.jar	808	2020-04-18	root	root
실 oracle-jdbc-11.2.0.4.jar	2 675	2019-11-13	omm	ficommon
🕌 postgresql-9.3-1103.jdbc4.jar	580	2019-11-13	omm	ficommon

Figure 10-11

Note: If the MRS cluster is highly available, upload the package to each master node. In this exercise, the HA function is not enabled for the MRS cluster. You only need to upload the package to one master node.

Step 3 Modify the properties of mysql-connector-java-5.1.21.jar.

Start PuTTY, go to the /opt/Bigdata/MRS_2.1.0/1_18_Sqoop/install/FusionInsight-Sqoop-1.99.7/server/jdbc directory, and change the owner of the mysql-connector-java-5.1.21.jar package to omm:wheel.

Run the following command:

chown omm:wheel mysql-connector-java-5.1.21.jar

After the modification, run the **ll** command to view the result.



Figure 10-12

Note: If the MRS cluster is highly available, you need to modify this attribute on each master node. In this exercise, the HA function is not enabled for the MRS cluster. You only need to modify the attribute for server master nodes.

Step 4 Modify the **jdbc.properties** configuration file on the master node.



Modify the **jdbc.properties** file in the folder in the previous step. Change the key value of MySQL to the name of the uploaded JDBC driver package **mysql-connector-java-5.1.21.jar**. If the name is already **mysql-connector-java-5.1.21.jar**, you do not need to change it.

```
[root@node-master1jicC jdbc]# more jdbc.properties
GAUSSDB=gsjdbc4-V100R003C10SPC115.jar
POSTGRESQL=postgresql-9.1-901.jdbc4.jar
MYSQL=mysql-connector-java-5.1.21.jar
ORACLE=oracle-jdbc-11.2.0.4.jar
MPPDB=gsjdbc4-1.1.0.jar
[root@node-master1jicC jdbc]#
```

Figure 10-13

Note: If the MRS cluster is in HA mode, you need to change the value of this parameter on each master node. In this exercise, the HA function is not enabled for the MRS cluster. You only need to change the value of this parameter on one master node.

Step 5 Restart the Loader service.

Log in to the MRS management page. On the **Services** tab page, click **Loader**.

Dashboard	Services	Hosts	Alarms	Audit	Tenant	
				-		
KrbServer	Started		< Good	Synchronized		
Kudu	Started		Sood	Synchronized		
LdapServer	Sta	arted	Sood	Synchronized		
Loader	 Started 		Sood	Synchronized		
Mapreduce	Started		Good	Syne Syne	chronized	
meta	🕑 Sta	arted	Good	Syne 🛛	chronized	

Figure 10-14

Choose **More > Restart Service**.



Dashboard	Services	Hosts	Alarms	Audit	Tenant	Sy			
Service Loader > Service Status									
Service Stat	Service Status Instance Service Configuration Resource Distribution								
	 Start Service Stop Service ▲ Download Client More ▼ 								
Loader Sum	mary			✓ Restart Serv	vice				
				Perform Ro	lling Service Resta	art			
Health Statu	IS	< Good		Start Service	e Health Check				
				View Service	e Health Check Re	eport			
Configuration Status		Synchronize	ed	Synchronize	Synchronize Configuration				
Version		200		Change Pas	sword				

Figure 10-15

Enter the verification password and click **OK**. In the **Restart Service** dialog box, select **Restart all upper-layer services**, and wait for the service to restart.

Rest	art Service	×
▲ Res	When the service is restarted, the upper-layer services that depend on the service may be temporarily unavailable. Before restarting the service, you can specify the upper-layer services you want to restart by selecting the following checkboxes. Restarting the service may affect service running. Are you sur- you want to restart the service?	2
🔽 Hu	e	
	OK Cancel	

Figure 10-16

Wait for the service to restart.



F	Restart Service								
		Step	Start Time	Progress		End Time			
	~	1. Verify request	06/23/2020 09:08:35 GM		100%	06/23/2020 09:08:35 GM			
	~	2. Stop Service	06/23/2020 09:08:35 GM		100%	06/23/2020 09:08:40 GM			
	~	3. Start Service	06/23/2020 09:08:40 GM		100%	06/23/2020 09:09:17 GM			
	~	4. Persist cluster c	06/23/2020 09:09:17 GM		100%	06/23/2020 09:09:17 GM			
•	C	Operation successful.							
	Finish								

Figure 10-17

10.3.3 Task 3: Creating a Loader Link

Step 1 Log in to Hue.

Log in to MRS Manager and choose **Service Hue**> **Service Status**. Click **Hue (Active)** to access the Hue page.



Service Hue > Service Status							
Service Status	Instance	Service	Configuration	Resou			
• Start Service	• Stop	Service	🛓 Download (Client			
Hue Summary				<			
Health Status		Sooc	ł				
Configuration Sta	atus	Synch	hronized				
Version		3.11.0					
Hue Web UI		Hue (Ac	tive)				

Figure 10-18

Step 2 Access Sqoop.

The open-source framework Sqoop is Loader in Huawei products. Click **Sqoop** in the **Data Browsers** drop-down list. The Sqoop page is displayed.

Ð	^	Query Editor 🗸	Dat	a Browsers 🗸	
👚 My d	locur	ments	▦	Metastore Ta	bles
			Sqoop		
My docu	iment	S	t	ZooKeeper	
,		_			

Figure 10-19





Figure 10-20

Step 3 Create a MySQL link.

In the upper right corner, choose **Manage links** > **New link**.



Figure 10-21

Name: cx_mysql_conn

Connector: generic-jdbc-connector

Database type: MySQL

Host: Enter the private IP address of the MySQL instance, as shown in the following figure:

Relational Da	atabase Service ⑦					⑦ Alarm Rule S	etting 🍳 Usage Wiz
Renew	Unsubscribe Change to Yearly	y/Monthly	Reboot Il DB engines	▼ DB instance	name 🔻		Q
	Name/ID ↓Ξ	Description	DB In ↓Ξ	DB Engine V ↓Ξ	Status	Billing Mode	Floating IP A
	rds-loader d242a6badd46492394b2ec70c26 🗇		Primary/Sta	MySQL 5.7.29	😔 Available	Pay-per-use Created on Ju	192.168.0.161

Figure 10-22

Port: 3306

Database: rdsdb

Username: root

Password: the password of user root set when you apply for the MySQL service



* Name	cx_mysql_conn
* Connector	generic-jdbc-connector
* Database type	MYSQL
	The database type. Example : MYSQL
* Host	192.168.0.78
	The database server's host name or IP. Example : 192.168.0.1
* Port	3306
	The database server's port. Example : 3306
* Database	rdsdb
	The database name. Example : DB_EXAMPLE
* Username	root
	Username to be used for connection to the database server. Example : sqoop
* Password	
	Password to be used for connection to the database server. Example : ********
	Show Senior Parameter
Cancel Test Save	

Figure 10-23



HCIA-Big Data V3.0 Lab Guide

	* Name	cx_mysql_conn
	* Connector	generic-idbc-connector
		genere juse connector
+ 1		MVS0I X
^ L	Database type	MYSQL
		The database type. Example : MYSQL
	* Host	192.168.0.161
		The database server's host name or IP. Example : 192.168.0.1
	* Port	2204
	FOIL	3300
		The database server's port. Example : 3306
	* Database	rdsdb
		The database name. Example : DB_EXAMPLE
	* Username	root
		Username to be used for connection to the database server. Example
	* Password	
		Password to be used for connection to the database server. Example
		Show Senior Parameter
•		
Cancel	Test Save	

Figure 10-24

After the configuration is complete, click **Test**. If the testing succeeds, click **Save**. The MySQL link is created.

Step 4 Create a HBase link.

Click **New link** and set the parameters as follows:

Name: cx_hbase_conn

Connector: hbase-connector

After the configuration is complete, click **Test**. If the testing succeeds, click **Save**.





Figure 10-25

Step 5 Create a Hive link.

Click **New link** and set the parameters as follows:

Name: cx_hive_conn

Connector: hive-connector

After the configuration is complete, click **Test**. If the testing succeeds, click **Save**.

* Name	cx_hive_conn
* Connector	hive-connector
Cancel Test Save	

Figure 10-26

Step 6 Create an HDFS link.

Click **New link** and set the parameters as follows:

Name: cx_hdfs_conn

Connector: hdfs-connector

After the configuration is complete, click **Test**. If the testing succeeds, click **Save**.





Figure 10-27

10.3.4 Task 4: Importing MySQL Data to HDFS

Step 1 Prepare MySQL data.

MySQL tables and data have been prepared in task 1. The data is as follows:

Executed SQL Sta	tements Message	Result Set1 ×			
The following FROM cx_stu	is the execution result : dent.	set of SELECT *	Click on the cell to edit th save the changes.	e data. After adding or editing, you need to subm	tit and Copy Row Copy Column Settings ✓
	id		name	gender	age
1	1001		MacDonald	male	30
2	1002		Calvin	male	25
3	1003		Haley	female	18
4	1004		Madonna	female	22
5	1005		Randell	male	36

Figure 10-28

Step 2 Log in to Hue and create a job.

On the Sqoop page of Hue, click **Create Job** and set the parameters as follows:



Step 1: Information	Step 2: From	Step 3: To	Step 4: Task Config
Connection			
* Na	me cx_job_	mysql_to_hdfs	
* From	link cx_mys	sql_conn	¥
* To	link cx_hdf	s_conn	•
	+ Add a	new link	

Figure 10-29

Click Next.

Step 3 Configure MySQL.

Set **Schema name** to **rdsdb**, **Table name** to **cx_student**, and **Partition column** to **id**, as shown in the following figure:

cx_mysql_conn	
* Schema name	rdsdb Schema or table space name if the table is not stored in
* Table name	cx_student Input table name from from which data will be retrieved.
* Partition column	id Input column that should be use to split the import into

Figure 10-30

Click **Next**.

Step 4 Configure HDFS.


Г

Set **Output directory** to the **/user/stu01/output2** directory. Retain the default values for other parameters, as shown in the following figure:

cx_hdfs_conn		
* Output directory	/user/stu01/output2 HDFS directory where trans	ferred data will be written to. Example : /user/sqoop/output
* File format	CSV_FILE File format that should be u	▼ sed for transferred data. Example : CSV_FILE
Compression codec	NONE Compression codec that sh	• ould be use to compress transferred data. Example : SNAPPY
Overwrite	True False	output directory already exists. If set to true then imported data
	Show Senior Parameter	

Figure 10-31

Note: If **output2** does not exist, the system automatically creates one.

Step 5 Configure a task.

Set Extractors to 1 and click Save and execute.

Task Config		
Extractors	1 Number of extractors when retrieving data.	Example : 3
	Show Senior Parameter	

Figure 10-32

The task is successfully run.

Name	Description	Creator	Activation	Last Execution	Use Time	Progress	Status
cx_job_mysql_to_hdfs	cx_mysql_conn->cx_hdfs_conn	admin	Enabled	2020/04/18 22:10:57	34s	100%	SUCCEEDED

Figure 10-33





Use PuTTY to log in to the master node and go to the HDFS directory to view data files.

Figure 10-34

10.3.5 Task 5: Importing MySQL Data to Hive

Step 1 Prepare a MySQL table.

Use the **cx_student** table data as the MySQL data.

Step 2 Create a table in Hive.

Use PuTTY to log in to a master node, go to Hive, and run the following statement to create a table:

create table cx_loader_stu01(id int,name string,gender string ,age int) row format delimited fields terminated by ',' stored as textfile ;

0: jdbc:hive2://192.168.0.151:2181/> create table cx_loader_stu01(id int,name string,gender string ,age int) row
format delimited fields terminated by ',' stored as textfile ;
INFO : Compiling command(queryId=omm_20200418221616_920bce3e-b989-4c1a-beee-aafa9496be79): create table cx_loade
r_stu01(id int,name string,gender string ,age int) row format delimited fields terminated by ',' stored as textfile
INFO : Concurrency mode is disabled, not creating a lock manager
INFO : Semantic Analysis Completed (retrial = false)
<pre>INFO : Returning Hive schema: Schema(fieldSchemas:null, properties:null)</pre>
INFO : EXPLAIN output for queryid omm_20200418221616_920bce3e-b989-4c1a-beee-aafa9496be79 : STAGE DEPENDENCIES:
Stage-0 is a root stage [DDL]

Figure 10-35

Step 3 Log in to Hue and create a job.

Click **Create** Job and set the parameters as follows:



* Name	cx_job_mysql_to_hive
* From link	cx_mysql_conn •
* To link	cx_hive_conn •
	♣ Add a new link

Figure 10-36

Click Next.

Step 4 Configure MySQL.

Set **Schema name** to **rdsdb**, **Table name** to **cx_student**, and **Partition column** to **id**, as shown in the following figure:

cx_mysql_conn	
* Schema name	rdsdb Schema or table space name if the table is not stored in a
* Table name	cx_student Input table name from from which data will be retrieved.
* Partition column	id Input column that should be use to split the import into in

Figure 10-37

Click Next.

Step 5 Configure Hive.

Retain the default database name **default** and set **Table** to **cx_loader_stu01**, as shown in the following figure:



cx_hive_conn		
* Database	default Specifies hive database. Example : default	
* Table	cx_loader_stu01 Specifies hive table. Example : TBL_EXAMPLE	



Step 6 Configure the field mapping.

Retain the default settings.

Source Field	Sample	Туре		Destination Field	Туре
id	1001	INT UNSIGNED	×	id	int
name	MacDonald	VARCHAR(32)		name	string
gender	male	VARCHAR(32)		gender	string
age	30	INT UNSIGNED		age	int
	+				+
Back Next					

Figure 10-39

Click Next.

Step 7 Configure a task.

Set Extractors to 1 and click Save and execute.



Task Config	
Extractors	1 Number of extractors when retrieving data. Example : 3
	Show Senior Parameter
Back Save Save	and execute

Figure 10-40

The task is successfully run.

Sqoop .	Jobs Search fo	r job name or link type						C Refres	h jobs 📰
Nar	me	Description	Creator	Activation	Last Execution	Use Time	Progress		Status
cx_	job_mysql_to_hdfs	cx_mysql_conn->cx_hdfs_conn	admin	Enabled	2020/04/18 22:10:57	34s	100%		SUCCEEDED
cx_	job_mysql_to_hive	cx_mysql_conn>cx_hive_conn	admin	Enabled	2020/04/18 22:21:01	32s	100%		SUCCEEDED

Figure 10-41

Step 8 View the result.

Use PuTTY to log in to the master node, run the **beeline** command to go to Hive, and run the **select** statement to view the result.

select * from cx_loade	er_stu01;		
INFO : OK INFO : Concurrency mo	de is disabled, not crea	ting a lock manager	
cx_loader_stu01.id	cx_loader_stu01.name	cx_loader_stu01.gender	cx_loader_stu01.age
1001	MacDonald	 male	30
1002	Calvin	male	25
1003	Haley	female	18
1004	Madonna	female	22
1005	Randell	male	36
+ 5 rows selected (0.261 0: jdbc:hive2://192.16	+ seconds) 8.0.151:2181/>	+	++

Figure 10-42



10.3.6 Task 6: Importing HDFS Data to HBase

Step 1 Create a HBase table.

Use PuTTY to log in to the master node, run **hbase shell** to go to the HBase window, and run the following statement to create a table:

create 'cx_table_stu02','cf1'

```
hbase(main):001:0> create 'cx_table_stu02','cf1'
2020-04-18 22:25:54,656 INFO [main] client.HBaseAdmin: Operation:
Id: 9 completed
Created table cx_table_stu02
Took 4.7276 seconds
=> Hbase::Table - cx_table_stu02
hbase(main):002:0>
```

Figure 10-43

Step 2 Create a data file and upload it to the HDFS.

Edit data file cx_stu_info2.txt on the Linux PC. The file content is as follows:

```
[root@node-master1bBdj ~]# vi cx_stu_info2.txt
[root@node-master1bBdj ~]# cat cx_stu_info2.txt
2001,Jack,male,20
2002,Lucy,female,18
[root@node-master1bBdj ~]#
```

Figure 10-44

Run the following command to upload the file to the HDFS:

hdfs dfs -put cx_stu_info2.txt /user/stu01

```
[root@node-master1bBdj ~]# hdfs dfs -put cx_stu_info2.txt /user/stu01
2020-04-18 22:30:45,430 INFO obs.OBSFileSystem: This Filesystem GC-ful, clear resource.
[root@node-master1bBdj ~]# hdfs dfs -ls /user/stu01
2020-04-18 22:30:54,672 INFO obs.OBSFileSystem: This Filesystem GC-ful, clear resource.
Found 2 items
-rw-r--r-- 1 root hadoop 38 2020-04-18 22:30 /user/stu01/cx_stu_info2.txt
drwxrwxrwx - loader hadoop 0 2020-04-18 22:10 /user/stu01/output2
[root@node-master1bBdj ~]#
```

Figure 10-45

Step 3 Log in to the Hue page and create a job.

Click **Create Job** and set the parameters as follows:





Figure 10-46

Click Next.

Step 4 Configure the source path.

The input path is the path of the HDFS file to be imported. The configuration is as follows:

* Input directory or file	/user/stu01/cx_stu_info2.txt	
	Input directory containing files that should be trans	sferred. Or file path if only
* File format	CSV_FILE •	
	File format that should be used for transferred dat	a. Example : CSV_FILE

Figure 10-47

Click Next.

Step 5 Configure HBase information.

Set Table name to cx_stu_info2 and Method to PUTLIST, as shown in the following figure:

* Table name	cx_table_stu02	
	Specifies Hbase table. Example : TBL_EXAMPLE	
* Method	PUTLIST	
	The methods of loading data into HBase tables	xample : BULKLOAD

Figure 10-48

Click Next.



Step 6 Configure the field mapping.

The following figure shows the configuration information:

Field Mapp	ing			
Colume Num	Sample	Column Family	*Destination Field	*Row Key
1	2001	cf1	▼ id	✓
2	Jack	cf1	▼ name	
3	male	cf1	▼ gender	
4	20	cf1	▼ age	

Figure 10-49

Select **Row Key** in the first row, name the destination field, which the qualifier of the column in HBase, and click **Next**.

Step 7 Configure a task.

Set Extractors to 1 and click Save and execute.

Extractors	1
	Number of extractors when retrieving data. Example: 3
	Show Senior Parameter

Figure 10-50

The task is successfully run.

Name	Description	Creator	Activation	Last Execution	Use Time	Progress	Status
cx_job_mysql_to_hdfs	cx_mysql_conn	admin	Enabled	2020/04/18	34s	100%	SUCCEEDED
	>cx_hdfs_conn			22:10:57			
cx_job_mysql_to_hive	cx_mysql_conn	admin	Enabled	2020/04/18	32s	100%	SUCCEEDED
	>cx_hive_conn			22:21:01			
cx_job_hdfs_to_hbase	cx_hdfs_conn	admin	Enabled	2020/04/18	41s	100%	SUCCEEDED
	>cx_hbase_conn			22:38:02			

Figure 10-51

Step 8 View the result.

Log in to HBase and run the **scan** command to view the table data.



hbase(main):002:0> scan	'cx_table_stu02'
ROW	COLUMN+CELL
2001	column=cf1:age, timestamp=1587220677033, value=20
2001	column=cf1:gender, timestamp=1587220677033, value=male
2001	column=cf1:name, timestamp=1587220677033, value=Jack
2002	column=cf1:age, timestamp=1587220677033, value=18
2002	column=cf1:gender, timestamp=1587220677033, value=female
2002	column=cf1:name, timestamp=1587220677033, value=Lucy
2 row(s)	
Took 0.1269 seconds	
hbase(main):003:0>	

Figure 10-52

10.4 Summary

This exercise describes how to use Loader in multiple service scenarios. Trainees can perform data migration operations in actual services after completing this exercise. Note that tables must be created before table data is migrated among MySQL, HBase, and Hive. Otherwise, an error may occur and the exercise may fail.



11 Comprehensive Exercise: Hive Data Warehouse

11.1 Background

In the big data services, multiple components form a service system. The following two exercises involve these components.

The first one is a typical data analysis exercise. Loader periodically migrates MySQL database data to Hive. Because data in Hive is stored in HDFS, Loader is used to import data in the HDFS to HBase. Use HBase to query data in real time and use the big data processing capability of Hive to analyze related results.

The second one is to use Flume to collect incremental data, upload the data to HDFS, and use Hive to query and analyze the data.

11.2 Objectives

• Use big data components to convert and query data in real time.

11.3 Tasks

Data is imported from the MySQL database to Hive, and then imported from Hive to HBase for data analysis.

11.3.1 Preparing MySQL Data

Step 1 Log in to the MySQL database.

Go to the MySQL instance page and click **Log In**. You can use MySQL resources purchased in section 11.3. If no MySQL resource is available, purchase one.



Relational Dat	tabase Service ⑦				? Alar	rm Rule Setting	g Q Usage Wizard 🕞 Help Guide Buy DB Insta
Renew	Unsubscribe Change	to Yearly/Month gines	▼ DB inst	oot			Q Search by Tag 🛛 C 🕄
	Name/ID ↓Ξ	Descript	DB ↓Ξ	DB Engi ↓Ξ	Status	Billing	Floating Operation
	rds-loader d242a6badd46492394b2e 🗇		Primary/S	MySQL 5.7.29	🖯 Avai	Pay-per Created	192.168 Log In View Metric More 🗸

Figure 11-1

Step 2 Create the **cx_socker** table and set **timestr** as the primary key.

Create the **rdsdb** database (if there is no such a database) and create a table.

1 Basic Information		2	Column
* Table Name	cx_socker		
Storage Engine	InnoDB	\sim	
Character Set	utf8	\sim	
Collation	utf8_general_ci	\sim	
Comment			
Advanced Settings \lor			

Figure 11-2

Create a field.



1 Basic I	ormation 2 Column						
Add	Insert Delete	Move Up Mo	ove Down)			
NO.	Column Name	Туре		Length	Nullable	Primary Key	
1	timestr	varchar	\sim	32		~	
2	open	float	\sim		~		
3	high	float	\vee		~		
4	low	float	\sim		~		
5	close	float	\vee		~		
б	volumn	varchar	\sim	32	✓		
7	endprice	float	\vee		✓		

Figure 11-3

Click **Create Now** and execute the script.

- Step 3 Import data to cx_socker.
 - On the MySQL management page, choose **Import·Export** > **Import**.

Data Admin	Service MySQL	SQL Operation	Database Management	Import·Export	Structure 1
Home S	SQL Window X	Database Managemer	nt-rdsdb 🗙	Import	
		-		Export	
Current I	Database: rdsd	b Change IP Address	:: 192.168.0.161 Port:	3306 Character S	et: utf8
Objects	SQL Tuning	Metadata Collection	Alter Table: cx_socker ×		
1 The object	list data comes from	real-time query (up to 10000), v	which has a certain performan	ce consumption for your	database. It i
Table	+ Create 7	able			
View		Fable Name 🖕	Created 🌲	Rows (Estim	ated) 🌲
Procedure	+	cx_socker	2020-06-23 10:32:53	3 0	
Event	+	cx_student	2020-06-23 08:31:47	5	
Trigger	Total 2 Rows	< 1 > 10/pa	ge∨		
Function					

Figure 11-4

Click Create Task. By default, no bucket is available. Click Create OBS bucket.



Create Task			
Import Type	SQL	CSV	
File Souce	Upload File	Choose From OBS	
Attachment Storage 🕐	Creating an OBS bucket is f	` ree, but saving the file will i	No OBS bucket Create OBS bucket

Figure 11-5

Click **OK** to return to the import page. Select data file **sp500.csv** and upload it. The configuration is as follows:

Import Type	SQL	CSV	
File Souce	Upload File	Choose From OBS	
Attachment Storage 🕐	obs-8537	~	No OBS bucket?Create OBS bucket
	Creating an OBS bucket is f	ree, but saving the file will i	ncur a fee.
Attachment ⑦	Clie	k or drag the file here to u	pload the file
	Sp500_1592880074765.csv		
Database	rdsdb	\vee	
Table	cx_socker	~	
Data Position	Property First	Data First	
Charset	Auto Detetct	UTF8	GBK
Write Mode	INSERT INTO	INSERT IGNORE INT	O REPLACE INTO
Option	 Ignore errors, skip when S Delete uploaded file when Perform the TRUNCATE 	QL execution fails import completed opeartion before importing.	

Figure 11-6

Click **Create Import Task** and wait for the task to be executed.

Task ID	Created	Ty pe	Database & Table	File Name	Status
f58dff9a2f4f499b8dff9a2f4fe99	2020-06-23	C	Database: rdsdb	sp500_159288046	() Waiting

Figure 11-7



The execution succeeded.

Task ID	Created	Ty pe	Database & Table	File Name	Status	Execute Ti me	Importe d(rows)	Ingore E rror	Progress	Remark
6be6cf7f6c174cd2a6cf7f6c177c	2020-06-23	C	Database: rdsdb	sp500_159288098	⊘ Success	1 second	10022	No	100	%

Figure 11-8

Step 4 View data in **cx_socker**.

On the database management page, click **Query**.

+ Crea	ate Table						Enter a t
	Table Name 🖕	Created 🍦	Rows (Estimated) 🚔	Table Size 🍦	Index Size 🍦	Character Set	Operation
+	cx_socker	2020-06-23 10:32:53	10034	1.52MB	0B	utf8	Query Open

Figure 11-9

Detailed data is shown as follows:

Executed	SQL Statem	ents Message	Result Set1 \times					Append Mo
The fo	ollowing is the	e execution result s	et of select * from	Click on the cell to edit the data. Af save the changes.	ter adding or edi	ting, you need to submit and	Copy Row Copy Column V Column Settings V	
	t	imestr	open	high	low	close	volumn	~ ~
	1	1970-01-02	92.06	93.54	91.79	93.0	8050000	1
	2	1970-01-05	93.0	94.25	92.53	93.46	11490000	
	3	1970-01-06	93.46	93.81	92.13	92.82	11460000	
	4	1970-01-07	92.82	93.38	91.93	92.63	10010000	
	5	1970-01-08	92.63	93.47	91.99	92.68	10670000	
	6	1970-01-09	92.68	93.25	91.82	92.4	9380000	
	7	1970-01-12	92.4	92.67	91.2	91.7	8900000	, ·

Figure 11-10

11.3.2 Importing MySQL Data to Hive

Step 1 Create a table in Hive.

Log in to Hive and run the following command to create the **cx_hive_socker** table:

create table cx_hive_socker (timestr string,open float,high float,low float,close float,volume string,endprice float) row format delimited fields terminated by ',' stored as textfile;

```
0: jdbc:hive2://192.168.0.10:2181/> create table socker(timestr string,open floa
t,high float,low float,close float,volume string,endprice float) row format deli
mited fields terminated by ',' stored as textfile;
No rows affected (0.052 seconds)
0: jdbc:hive2://192.168.0.10:2181/>
```

Figure 11-11

Step 2 Log in to Hue and create a job in Sqoop.

Configure parameters as follows:



* Name	cx_job_mysql_to_hive	
* From link	cx_mysql_conn	•
* To link	cx_hive_conn	•
	+ Add a new link	

Figure 11-12

Step 3 Set the MySQL information for the job.

Configure parameters as follows:

cx_mysql_conn	
* Schema name	rdsdb Schema or table space name if the table is not stored in d
* Table name	cx_socker Input table name from from which data will be retrieved.
* Partition column	timestr Input column that should be use to split the import into in
	Show Senior Parameter

Figure 11-13

Step 4 Configure the Hive information for the job.

Configure parameters as follows:



cx_hive_conn		
* Database	default Specifies hive database. Example : default	
* Table	cx_hive_socker Specifies hive table. Example : TBL_EXAMPLE	

Figure 11-14

Step 5 Configure the field mapping.

Retain the default values.

F	ield Mapping				
	Source Field	Sample	Туре	Destination Field	Туре
	timestr	1970-01-02	VARCHAR(32)	timestr	string
	open	92.06	FLOAT UNSIGNED	open	float
	high	93.54	FLOAT UNSIGNED	high	float
	low	91.79	FLOAT UNSIGNED	low	float
	close	93	FLOAT UNSIGNED	close	float
	volumn	8050000	VARCHAR(32)	volume	string
	endprice	93	FLOAT UNSIGNED	endprice	float
		+			+



Click Next.

Step 6 Configure a task.

Set **Extractors** to **1** and click **Save and execute**.



Task Config	
Extractors	1 Number of extractors when retrieving data. Example : 3
	Show Senior Parameter
Back Save Save	and execute

Figure 11-16

The task is successfully run.

Sqoo	p Jobs Search	for job name or link type							C Refresh jobs	1	📕 Manag	je links
	Name	Description	Creator	Activation	Last Execution	Use Time	Progress		Stat	us	Opera	ate
	cx_job_mysql_to_hive	cx_mysql_conn->cx_hive_conn	admin	Enabled	2020/04/22 12:00:14	37s		100%	SUC	CEEDEI	D	2

Figure 11-17

Step 7 View data in Hive.

Run the **select * from cx_hive_socker limit 10** command in Hive.

cx_hive_socker.timestr cx_hive_socker.open cx_hive_socker.high cx_hive_socker.low cx_hive_socker.close (x_hive_socker.volume cx_hive_socker.endprice 1970-01-02 92.06 93.54 91.79 93.0 8050000 93.0 1970-01-02 92.06 93.54 91.79 93.0 8050000 93.0 1970-01-05 93.0 94.25 92.53 93.46 11490000 93.46 1970-01-06 93.46 91.73 92.82 11460000 92.63 1970-01-07 92.63 93.47 91.99 92.68 10670000 92.68 970-01-09 92.68 1970-01-09 92.68 93.25 91.82 92.4 938000 91.7 1970-01-12					
$\begin{array}{ c c c c c c c c c c c c c c c c c c c$	cx_hive_socker.timestr cx_hive_socker.volume	· cx_hive_socker.open cx_hive_socker.endprice	+ cx_hive_socker.high 	cx_hive_socker.low	<pre></pre>
$ \begin{vmatrix} 1970 - 01 - 02 & 92.06 & 93.54 & 91.79 & 93.0 \\ 805000 & 93.0 & \\ 1970 - 01 - 05 & 93.0 & 94.25 & 92.53 & 93.46 \\ 11490000 & 93.46 & 93.81 & 92.13 & 92.82 \\ 11460000 & 92.82 & \\ 1970 - 01 - 07 & 92.82 & 93.38 & 91.93 & 92.63 \\ 10010000 & 92.63 & \\ 1970 - 01 - 08 & 92.63 & 93.47 & 91.99 & 92.68 \\ 10670000 & 92.68 & \\ 1970 - 01 - 09 & 92.68 & \\ 1970 - 01 - 09 & 92.68 & \\ 1970 - 01 - 09 & 92.4 & \\ 1970 - 01 - 12 & 92.4 & 92.67 & 91.2 & 91.7 \\ 890000 & 91.7 & \\ 1970 - 01 - 12 & 91.7 & 92.61 & 90.99 & 91.92 \\ 9870000 & 91.7 & \\ 1970 - 01 - 14 & 91.92 & 92.4 & 90.88 & 91.65 \\ 10380000 & 91.65 & \\ 1970 - 01 - 15 & 91.65 & 92.35 & 90.73 & 91.68 \\ 11120000 & 91.68 & \\ 112000 & $	+	+	+ +	+	+
8050000 93.0 94.25 92.53 93.46 11490000 93.46 1 11970-01-05 93.46 93.81 92.13 92.82 11460000 92.82 1 1 1 1970-01-06 92.82 1 1 1 1970-01-07 92.82 93.38 91.93 92.63 10010000 92.63 1 1 1 1970-01-08 92.63 93.47 91.99 92.68 10670000 92.68 1 1 1 1970-01-09 92.68 93.25 91.82 92.4 1970-01-12 92.4 92.67 91.2 91.7 1970-01-12 92.4 92.61 90.99 91.92 1970-01-13 91.7 1 1 1 1970-01-13 91.7 92.61 90.99 91.92 1970-01-14 91.92 1 1 1 1 1970-01-15 91.65 1 1 1 1 1970-01-15 91.65 1 1	1970-01-02	92.06	93.54	91.79	93.0
$ \begin{vmatrix} 1970-01-05 & 93.0 & 94.25 & 92.53 & 93.46 \\ 1149000 & 93.46 & \\ 1970-01-06 & 93.46 & 93.81 & 92.13 & 92.82 \\ 11460000 & 92.82 & \\ 1970-01-07 & 92.82 & 93.38 & 91.93 & 92.63 \\ 1001000 & 92.63 & \\ 1970-01-08 & 92.63 & \\ 1970-01-08 & 92.68 & \\ 1970-01-09 & 92.68 & 93.25 & 91.82 & 92.4 \\ 1970-01-09 & 92.68 & 92.67 & 91.2 & 91.7 \\ 938000 & 92.4 & \\ 1970-01-12 & 92.4 & 92.67 & 91.2 & 91.7 \\ 890000 & 91.7 & \\ 1970-01-13 & 91.7 & 92.61 & 90.99 & 91.92 \\ 987000 & 91.92 & \\ 1970-01-13 & 91.7 & 92.61 & 90.99 & 91.92 \\ 987000 & 91.92 & \\ 1970-01-14 & 91.92 & 92.4 & 90.88 & 91.65 \\ 10380000 & 91.65 & \\ 1970-01-15 & 91.65 & 92.35 & 90.73 & 91.68 \\ 11120000 & 91.68 & \\ 112000 & 91.68 & \\ 11200 & $	8050000	93.0			
$\begin{array}{c c c c c c c c c c c c c c c c c c c $	1970-01-05	93.0	94.25	92.53	93.46
$ \begin{vmatrix} 1970-01-06 & 93.46 & 93.81 & 92.13 & 92.82 \\ 1146000 & 92.82 & \\ 1970-01-07 & 92.82 & 93.38 & 91.93 & 92.63 \\ 10010000 & 92.63 & \\ 1970-01-08 & 92.63 & 93.47 & 91.99 & 92.68 \\ 10670000 & 92.68 & \\ 1970-01-09 & 92.68 & \\ 1970-01-09 & 92.4 & 92.67 & 91.82 & 92.4 \\ 9380000 & 91.7 & \\ 1970-01-12 & 92.4 & 92.67 & 91.2 & 91.7 \\ 8900000 & 91.7 & \\ 1970-01-13 & 91.7 & 92.61 & 90.99 & 91.92 \\ 9870000 & 91.92 & \\ 1970-01-14 & 91.92 & \\ 1970-01-14 & 91.92 & 92.4 & 90.88 & 91.65 \\ 1038000 & 91.65 & \\ 1970-01-15 & 91.65 & 92.35 & 90.73 & 91.68 \\ 11120000 & 91.68 & \\ 1120000 &$	11490000	93.46			
$\begin{array}{c c c c c c c c c c c c c c c c c c c $	1970-01-06	93.46	93.81	92.13	92.82
$ \begin{vmatrix} 1970-01-07 & 92.82 & 93.38 & 91.93 & 92.63 \\ 10010000 & 92.63 & \\ 1970-01-08 & 92.63 & 93.47 & 91.99 & 92.68 \\ 10670000 & 92.68 & \\ 1970-01-09 & 92.68 & 93.25 & 91.82 & 92.4 \\ 9380000 & 92.4 & \\ 1970-01-12 & 92.4 & 92.67 & 91.2 & 91.7 \\ 890000 & 91.7 & \\ 1970-01-13 & 91.7 & 92.61 & 90.99 & 91.92 \\ 9870000 & 91.92 & \\ 1970-01-14 & 91.92 & 92.4 & 90.88 & 91.65 \\ 10380000 & 91.65 & \\ 1120000 & 91.65 & \\ 1120000 & 91.68 & \\ 112000 & 91.68 & \\ 1120000 & 91.68 & \\ 11$	11460000	92.82			
$\begin{array}{c c c c c c c c c c c c c c c c c c c $	1970-01-07	92.82	93.38	91.93	92.63
$ \begin{vmatrix} 1970-01-08 & 92.63 & 93.47 & 91.99 & 92.68 \\ 10670000 & 92.68 & \\ 1970-01-09 & 92.68 & 93.25 & 91.82 & 92.4 \\ 9380000 & 92.4 & \\ 1970-01-12 & 92.4 & 92.67 & 91.2 & 91.7 \\ 890000 & 91.7 & \\ 1970-01-13 & 91.7 & 92.61 & 90.99 & 91.92 \\ 9870000 & 91.92 & \\ 1970-01-14 & 91.92 & 92.4 & 90.88 & 91.65 \\ 10380000 & 91.65 & \\ 1970-01-15 & 91.65 & 92.35 & 90.73 & 91.68 \\ 11120000 & 91.68 & \\ 122000 & 91.68 & $	10010000	92.63	<u> </u>		
$\begin{array}{c c c c c c c c c c c c c c c c c c c $	1970-01-08	92.63	93.47	91.99	92.68
1970-01-09 92.68 93.25 91.82 92.4 9380000 92.4 1970-01-12 92.4 92.67 91.7 890000 91.7 1970-01-13 91.7 91.92 9870000 91.92 1970-01-13 91.92 91.65 1970-01-14 91.92 92.4 90.88 91.65 1970-01-15 91.65 1970-01-15 91.65	10670000	92.68	<u> </u>		
938000 92.4 1970-01-12 92.4 92.67 91.2 91.7 890000 91.7 1970-01-13 91.7 92.61 90.99 91.92 987000 91.92 1970-01-14 91.92 92.4 90.88 91.65 10380000 91.65 1970-01-14 91.65 1120000 91.68	1970-01-09	92.68	93.25	91.82	92.4
1970-01-12 92.4 92.67 91.2 91.7 890000 91.7 1970-01-13 91.7 92.61 90.99 91.92 9870000 91.92 1970-01-14 91.92 92.4 90.88 91.65 10380000 91.65 <	9380000	92.4	_ I		
8900000 91.7 1970-01-13 91.7 92.61 90.99 91.92 9870000 91.92 1970-01-14 91.92 92.4 90.88 91.65 10380000 91.65 1970-01-15 91.65	1970-01-12	92.4	92.67	91.2	91.7
1970-01-13 191.7 192.61 90.99 191.92 9870000 191.92 1 1 1 1970-01-14 191.92 1 90.88 1 10380000 191.65 1 1 1970-01-15 191.65 1 1 11120000 191.68 1 1	8900000	91.7		1	1
9870000 91.92 1970-01-14 91.92 92.4 90.88 91.65 1970-01-15 91.65 1970-01-15 91.65 92.35 90.73 91.68 11120000 91.68	1970-01-13	91.7	92.61	90.99	91.92
1970-01-14 191.92 192.4 190.88 191.65 10380000 191.65 1 1 1970-01-15 191.65 1 1 11120000 191.68 1 1	9870000	91.92			04.65
10380000 91.65 1970-01-15 91.65 92.35 90.73 91.68 11120000 91.68	1970-01-14		92.4	90.88	91.65
15/0-01-15 51.05 52.55 90.75 91.08 11120000 91.68	10380000	01 (5		00.73	01.68
11120000 51.05	11120000	01 (0	92.33	1 90.75	1 21.08
	11120000		I +		

Figure 11-18



11.3.3 Processing Hive Data

Obtain the latest stock growth data and save the result to a new Hive table.

Step 1 Create a table in Hive.

Run the following command in the Hive shell to create a table:

create table cx_up_hive_socker like cx_hive_socker;



Figure 11-19

Step 2 Run the following command to insert data:

insert into cx_up_hive_socker select * from cx_hive_socker where cx_hive_socker.endprice> cx_hive_socker.open sort by cx_hive_socker.endprice desc;

0: jdbc:hive2://192.168.0.187: e_socker.endprice desc; No rows affected (34.168 secon 0: jdbc:hive2://192.168.0.187: +	2181/> insert into cx_up_ ds) 2181/> show tables;	hive_socker select * from	cx_hive_socker where cx_h	ive_socker.endprice> cx_hiv	re_socker.open sort by cx_hiv
cx_hive_socker cx_up_hive_socker					
2 rows selected (0.064 seconds 0: jdbc:hive2://192.168.0.187:) 2181/> select * from cx_u	up_hive_socker limit 10;	+	+	+
+ cx_up_hive_socker.timestr cx_up_hive_socker.endprice	+ cx_up_hive_socker.open 	cx_up_hive_socker.high	cx_up_hive_socker.low	<pre>cx_up_hive_socker.close</pre>	cx_up_hive_socker.volume
++ +	+ 1553 18	1565-26	1551 82	+	2932040000
1565.15 2007-10-12	1555.41	1563.03	1554.09	1561.8	2788690000
1561.8 2007-10-05	1543.84	1561.91	1543.84	1557.59	2919030000
1557.59 2007-07-19 1553.08	1546.13	1555.2	1546.13	1553.08	3251450000
2007-07-13 1552.5	1547.68	1555.1	1544.85	1552.5	2801120000
2007-10-31 1549.38	1532.15	1552.76	1529.4	1549.38	3953070000
2007-07-12 1547.7	1518.74	1547.92	1518.74	1547.7	3489600000
2007-10-01 1547.04	1527.29	1549.02	1527.25	1547.04	3281990000
2007-10-04 1542.84	1539.91	1544.02	1537.63	1542.84	2690430000

Figure 11-20

Step 3 Run the following command to obtain the total number of the stocks that grow:

select count(*) from cx_hive_socker where cx_hive_socker.endprice> cx_hive_socker.open;



0: jdbc:hive2://192.168.0.187:2181/> select count(*) from cx_hive_socker where cx_hive_socker.endprice> cx_hive_socker.open; +-----+ | _c0 | +-----+ | 5228 | +-----+ 1 row selected (16.963 seconds) 0: jdbc:hive2://192.168.0.187:2181/>

Figure 11-21

11.3.4 Importing HDFS Data to HBase

Step 1 Create a table in HBase.

Access the hbase shell and run the following statement to create a table:

create 'cx_hbase_socker','cf1'

```
hbase(main):001:0> create 'cx_hbase_socker','cf1'
2020-04-22 13:44:37,594 INFO [main] client.HBaseAdmin: Operation: CREATE, Table Name: default:cx_hbase_socker, procId:
9 completed
Created table cx_hbase_socker
Took 2.8852 seconds
=> Hbase::Table - cx_hbase_socker
hbase(main):002:0>
```

Figure 11-22

Step 2 Create a Loader job.

Configure parameters as follows:

Connection	
* Name	cx_job_hdfs2hbase
* From link	cx_hdfs_conn •
* To link	cx_hbase_conn •
	+ Add a new link

Figure 11-23

Step 3 Configure the source path.

The input path is the path of the HDFS file to be imported. The address is the data in the Hive table **cx_hive_socker**, and is in the Hive data warehouse directory of the HDFS, as shown in the following figure:



[root@node-master1floZ ~]# hdfs	s dfs -ls /user/hive/warehouse/cx_hive_socker/
2020-04-22 13:49:36,461 INFO ot	os.OBSFileSystem: This Filesystem GC-ful, clear resource.
Found 1 items	
-rw-r 3 loader hive	567946 2020-04-22 12:00 /user/hive/warehouse/cx hive_socker/60f9ba12-3a37-472f-baaf-1b09
2c82740f	
[root@node-master1floZ ~]#	

Figure 11-24

To configure the HDFS address of the Loader job, you need to query the specific path. In this example, the HDFS path is as follows:

/user/hive/warehouse/cx_hive_socker/60f9ba12-3a37-472f-baaf-1b092c82740f

The HDFS configuration of the job is as follows:

cx_hdfs_conn	
* Input directory or file	/user/hive/warehouse/cx_hive_socker/60f9ba1
	Input directory containing files that should be transferred. Or file path if only
* File format	CSV_FILE •
	File format that should be used for transferred data. Example : CSV_FILE
	Show Senior Parameter

Figure 11-25

Click Next.

Step 4 Configure the HBase information.

Configure parameters as follows:

cx_hbase_conn		
* Table name	cx_hbase_socker Specifies Hbase table. Example : TBL_EXAMPLE	•
* Method	PUTLIST The methods of loading data into HBase tables	xample : BULKLOAD

Figure 11-26

Click Next.

Step 5 Configure the field mapping.



Configure parameters as follows:

Field Mapping					
Colume Num	Sample	Column Family		*Destination Field	*Row Key
1	1970-01-02	cf1	•	timestr	✓
2	92.06	cf1	•	open	
3	93.54	cf1	•	high	
4	91.79	cf1	•	low	
5	93	cf1	•	close	
6	8050000	cf1	•	volumn	
7	93	cf1	•	endprice	

Figure 11-27

Click **Next**.

Step 6 Configure a task.

Set **Extractors** to **1** and click **Save and execute**.

Task Config	
Extractors	1 Number of extractors when retrieving data. Example : 3
	Show Senior Parameter

Figure 11-28

The task is successfully run.

Sqoop Jobs Search for	or job name or link type							C Refres	n jobs 🛛 🛛	🗏 Mana	ge links
Name	Description	Creator	Activation	Last Execution	Use Time	Progress			Status	Ope	rate
cx_job_mysql_to_hive	cx_mysql_conn->cx_hive_conn	admin	Enabled	2020/04/22 12:00:14	37s		100%		SUCCEEDED	>	ත
cx_job_hdfs2hbase	cx_hdfs_conn->cx_hbase_conn	admin	Enabled	2020/04/22 13:54:43	40s		100%		SUCCEEDED	D •	

Figure 11-29

Step 7 View the result.

Log in to HBase and run the **scan** command to view the table data.



hbase(main):001:0> scan 'cx_hb	ase_socker'
ROW	COLUMN+CELL
1970-01-02	column=cf1:close, timestamp=1587534878585, value=93
1970-01-02	column=cf1:endprice, timestamp=1587534878585, value=93
1970-01-02	column=cf1:high, timestamp=1587534878585, value=93.54
1970-01-02	column=cf1:low, timestamp=1587534878585, value=91.79
1970-01-02	column=cf1:open, timestamp=1587534878585, value=92.06
1970-01-02	column=cf1:volumn, timestamp=1587534878585, value=8050000
1970-01-05	column=cf1:close, timestamp=1587534878585, value=93.46
1970-01-05	column=cf1:endprice, timestamp=1587534878585, value=93.46
1970-01-05	column=cf1:high, timestamp=1587534878585, value=94.25
1970-01-05	column=cf1:low, timestamp=1587534878585, value=92.53
1970-01-05	column=cf1:open, timestamp=1587534878585, value=93
1970-01-05	column=cf1:volumn, timestamp=1587534878585, value=11490000
1970-01-06	column=cf1:close, timestamp=1587534878585, value=92.82
1970-01-06	<pre>column=cf1:endprice, timestamp=1587534878585, value=92.82</pre>
1970-01-06	column=cf1:high, timestamp=1587534878585, value=93.81
1970-01-06	column=cf1:low, timestamp=1587534878585, value=92.13
1970-01-06	column=cf1:open, timestamp=1587534878585, value=93.46
1970-01-06	column=cf1:volumn, timestamp=1587534878585, value=11460000
1970-01-07	column=cf1:close, timestamp=1587534878585, value=92.63
1970-01-07	column=cf1:endprice, timestamp=1587534878585, value=92.63
1970-01-07	column=cf1:high, timestamp=1587534878585, value=93.38
1970-01-07	column=cf1:low, timestamp=1587534878585, value=91.93

Figure 11-30

11.3.5 Querying Data in HBase in Real Time

Step 1 Query specified records.

Run the following command in HBase:

get 'cx_hbase_socker','2009-09-15'

```
hbase(main):004:0> get 'cx_hbase_socker','2009-09-15'
COLUMN
                               CELL
cf1:close
                               timestamp=1587534879064, value=1052.63
cf1:endprice
                               timestamp=1587534879064, value=1052.63
                               timestamp=1587534879064, value=1056.04
 cf1:high
 cf1:low
                               timestamp=1587534879064, value=1043.42
                               timestamp=1587534879064, value=1049.03
cf1:open
                               timestamp=1587534879064, value=6185620000
cf1:volumn
1 row(s)
Took 0.0595 seconds
hbase(main):005:0>
```

Figure 11-31

Step 2 Query the number of records in a specified period.

Run the following command in HBase:

scan 'cx_hbase_socker',{COLUMNS=>'cf1:endprice',STARTROW=>'2009-08-15',STOPROW=>'2009-09-15'}



base(main):006:0> scan	<pre>'cx_hbase_socker',{COLUMNS=>'cf1:endprice',STARTROW=>'2009-08-15',STOPROW=>'2009-09-15'}</pre>
OW	COLUMN+CELL
2009-08-17	column=cf1:endprice, timestamp=1587534879064, value=979.73
2009-08-18	column=cf1:endprice, timestamp=1587534879064, value=989.67
2009-08-19	column=cf1:endprice, timestamp=1587534879064, value=996.46
2009-08-20	column=cf1:endprice, timestamp=1587534879064, value=1007.37
2009-08-21	column=cf1:endprice, timestamp=1587534879064, value=1026.13
2009-08-24	column=cf1:endprice, timestamp=1587534879064, value=1025.57
2009-08-25	column=cf1:endprice, timestamp=1587534879064, value=1028
2009-08-26	column=cf1:endprice, timestamp=1587534879064, value=1028.12
2009-08-27	column=cf1:endprice, timestamp=1587534879064, value=1030.98
2009-08-28	column=cf1:endprice, timestamp=1587534879064, value=1028.93
2009-08-31	column=cf1:endprice, timestamp=1587534879064, value=1020.62
2009-09-01	column=cf1:endprice, timestamp=1587534879064, value=998.04
2009-09-02	column=cf1:endprice, timestamp=1587534879064, value=994.75
2009-09-03	column=cf1:endprice, timestamp=1587534879064, value=1003.24
2009-09-04	column=cf1:endprice, timestamp=1587534879064, value=1016.4
2009-09-08	column=cf1:endprice, timestamp=1587534879064, value=1025.39
2009-09-09	column=cf1:endprice, timestamp=1587534879064, value=1033.37
2009-09-10	column=cf1:endprice, timestamp=1587534879064, value=1044.14
2009-09-11	column=cf1:endprice, timestamp=1587534879064, value=1042.73
2009-09-14	column=cf1:endprice, timestamp=1587534879064, value=1049.34
0 row(s)	
ook 0.0125 seconds	
base(main):007:0>	

Figure 11-32

Step 3 Queries all columns whose values are greater than a specified value. Values are compared as character strings.

Run the following command in HBase:

<pre>scan 'cx_hbase_socker',{FILTER => "ValueFilter(>,'binary:999')"}</pre>				
hbase(main):013:0> scan 'c	<pre>x_hbase_socker',{FILTER => "ValueFilter(>,'binary:999')"}</pre>			
ROW	COLUMN+CELL			
1970-10-22	column=cf1:volumn, timestamp=1587534878585, value=9e+06			
1998-02-04	column=cf1:low, timestamp=1587534878961, value=999.43			
2000-04-26	column=cf1:volumn, timestamp=1587534878961, value=999600000			
2003-08-18	column=cf1:close, timestamp=1587534878996, value=999.74			
2003-08-18	column=cf1:endprice, timestamp=1587534878996, value=999.74			
2003-08-19	column=cf1:open, timestamp=1587534878996, value=999.74			
2003-08-21	column=cf1:low, timestamp=1587534878996, value=999.33			
2003-08-29	column=cf1:low, timestamp=1587534878996, value=999.52			
2009-08-07	column=cf1:low, timestamp=1587534879044, value=999.83			
2009-08-07	column=cf1:open, timestamp=1587534879044, value=999.83			
2009-08-19	column=cf1:high, timestamp=1587534879064, value=999.61			
$9 \operatorname{row}(s)$				

Figure 11-33

Step 4 Query all information ending with endprice. The value of the character string must be greater than 999.

Run the following command in HBase:

scan 'cx_hbase_socker',{FILTER=>"ValueFilter(>,'binary:999') AND ColumnPrefixFilter('endprice')"}

```
hbase(main):014:0> scan 'cx_hbase_socker',{FILTER=>"ValueFilter(>,'binary:999') AND ColumnPrefixFilter('endprice')"}
ROW COLUMN+CELL
2003-08-18 column=cf1:endprice, timestamp=1587534878996, value=999.74
1 row(s)
Took 0.1199 seconds
hbase(main):015:0>
```

Figure 11-34



11.4 Summary

These exercises integrate the applications of each component, helping trainees better understand and use big data components.



12 Appendix: Environment Preparations and Commands

12.1 (Optional) Preparing the Java Environment

12.1.1 Installing JDK

Step 1 Download JDK.

Visit https://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html, select **Accept License Agreement**, and download the JDK of the Windows x64 version. If the operating system is 32-bit, select the x86 version.

Java SE Development Kit 8u241 This software is licensed under the Oracle Technology Network License Agreement for Oracle Java SE					
Product / File Description	File Size	Download			
Linux ARM 32 Hard Float ABI	72.94 MB	jdk-8u241-linux-arm32-vfp-hflt.tar.gz			
Linux ARM 64 Hard Float ABI	69.83 MB	°↓ jdk-8u241-linux-arm64-vfp-hflt.tar.gz			
Linux x86 RPM Package	171.28 MB	jdk-8u241-linux-i586.rpm			
Linux x86 Compressed Archive	186.1 MB	jdk-8u241-linux-i58ó.tar.gz			
Linux xó4 RPM Package	170.65 MB	jdk-8u241-linux-xó4.rpm			
Linux x64 Compressed Archive	185.53 MB	jdk-8u241-linux-xó4.tar.gz			
macOS x64	254.06 MB	jdk-8u241-macosx-x64.dmg			
Solaris SPARC 64-bit (SVR4 package)	133.01 MB	idk-8u241-solaris-sparcv9.tar.Z			
Solaris SPARC 64-bit	94.24 MB	jdk-8u241-solaris-sparcv9.tar.gz			
Solaris xó4 (SVR4 package)	133.8 MB	jdk-8u241-solaris-x64.tar.Z			
Solaris xó4	92.01 MB	[≜] ↓ jdk-8u241-solaris-x64.tar.gz			
Windows x86	200.86 MB	idk-8u241-windows-i586.exe			
Windows x64	210.92 MB	≝↓_ jdk-8u241-windows-xó4.exe			

Figure 12-1

Step 2 Double-click the downloaded **.exe** file and click **Next**.





Figure 12-2



😹 Java SE Development Kit 8 Update 201 (64-bit) - Custom Setup 🛛 🗡					
	_	_	_		
Select optional features to install from the lis installation by using the Add/Remove Program	t below. You can c ns utility in the Con	hange your choice of trol Panel	f features after		
		Feature Descripti	on		
Development Tools Source Code Public JRE		Java SE Developr 201 (64-bit), inclu SDK, a private JR Mission Control to will require 180M drive.	ment Kit 8 Update uding the JavaFX E, and the Java ools suite. This 18 on your hard		
C:\Program Eilos\]ava\idk1 8 0, 201\					
C. (Frogram Files pava (ukr.o.o_201)			<u>C</u> hange		
	< <u>B</u> ack	<u>N</u> ext >	Cancel		

Figure 12-3

Step 4 On the Change in License Terms page, click OK.



Change in License Terms	_		×
Important Information about Oracle Java SE Roadmap			
Changes are coming which will impact your access to future releases of Java SE fro	m Orac	le.	
Corporate users will be impacted with the April 2019 update			
These changes do not affect the version you are about to install.			
For additional guidance please follow the link below.			
More information			
		OK	

Figure 12-4

Step 5 Retain the default address and click Next.

Java Setup - Destination Folder		_		×
Destination Folder				
Click "Change" to install Java to a different folder.				
Install to: C:\Program Files\Java\jre1.&0_201		Cha	nge	
< B:	ack		Next >	

Figure 12-5

Wait for the installation to complete.



Java Setup - Progress —	\times
Status: Installing Java	
ATMs, Smartcards, POS Terminals, Blu-ray Players, PCs Set Top BELLING , Servers, Switches Routers, S BELLING , Devices Automote BELLING , Servers, Switches Devices Run Java Systems Devices Run Java Controls Devices Run Java	

Figure 12-6

After the installation is complete, click **Close**.

🖟 Java SE Development Kit 8 Update 201 (64-bit) - Complete	\times
Java SE Development Kit 8 Update 201 (64-bit) Successfully Installed	
Click Next Steps to access tutorials, API documentation, developer guides, release notes and more to help you get started with the JDK.	
<u>C</u> lose	

Figure 12-7

Step 6 Configure JDK environment variables.



Choose My Computer > Properties > Advanced system settings > Environment Variables.

Variable	Value		
OneDrive	C:\Users\mwx711840\OneDrive		
Path	C:\Users\mwx711840\AppData\Local\Microsoft\WindowsApps;		
TEMP	C:\Users\mwx711840\AppData\Local\Temp		
ТМР	C:\Users\mwx711840\AppData\Local\Temp		
	New Edit Delete		
stem variables			
Variable	Value		
Variable ComSpec	Value C:\windows\system32\cmd.exe		
Variable ComSpec DriverData	Value / C:\windows\system32\cmd.exe C:\Windows\System32\Drivers\DriverData		
Variable ComSpec DriverData NUMBER_OF_PROCESSOR:	Value ' C:\windows\system32\cmd.exe ' C:\Windows\System32\Drivers\DriverData ' S 4		
Variable ComSpec DriverData NUMBER_OF_PROCESSOR: OS	Value ' C:\windows\system32\cmd.exe ' C:\Windows\System32\Drivers\DriverData ' S 4 Windows_NT		
Variable ComSpec DriverData NUMBER_OF_PROCESSOR: OS Path	Value / C:\windows\system32\cmd.exe / C:\Windows\System32\Drivers\DriverData / S 4 Windows_NT / C:\Program Files (x86)\Common Files\Oracle\Java\javapath;C:		
Variable ComSpec DriverData NUMBER_OF_PROCESSOR: OS Path PATHEXT	Value C:\windows\system32\cmd.exe C:\Windows\System32\Drivers\DriverData S 4 Windows_NT C:\Program Files (x86)\Common Files\Oracle\Java\javapath;C: .COM;EXE;.BAT;.CMD;.VBS;.VBE;JS;JSE;.WSF;.WSF;.WSC		
Variable ComSpec DriverData NUMBER_OF_PROCESSOR: OS Path PATHEXT PROCESSOR_ARCHITECTU	Value C:\windows\system32\cmd.exe C:\Windows\System32\Drivers\DriverData S 4 Windows_NT C:\Program Files (x86)\Common Files\Oracle\Java\javapath;C, .COM;EXE;.BAT;.CMD;.VBS;.VBE;JS;JSE;.WSF;.WSH;.MSC AMD64		
Variable ComSpec DriverData NUMBER_OF_PROCESSOR: OS Path PATHEXT PROCESSOR_ARCHITECTU	Value / C:\windows\system32\cmd.exe ////////////////////////////////////		

Figure 12-8

Click **New** in the **System variables** area. Set **Variable name** to **JAVA_HOME** (all uppercase letters) and **Variable value** to the JDK installation path.

New System Variable	3	×
Variable <u>n</u> ame:	JAVA_HOME	
Variable <u>v</u> alue:	C:\Program Files\Java\jdk1.8.0_201	
Browse <u>D</u> irectory.	Browse <u>F</u> ile	OK Cancel

Figure 12-9

Find **Path** in the system variables and edit the variable.

Variable	Value	Edit System Variable	×
NUMBER_OF_PROCESSORS	4		
os	Windows_N1		
Path	C:\Program Files (x86)\Common Files\Oracle\Java\javapath;C:	Variable <u>n</u> ame:	Path
PATHEXT	.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC		
PROCESSOR_ARCHITECTU	AMD64	Variable value:	MROOT%\System32\WindowsPowerShell\V1.0\;%SYSTEMROOT%\System32\OpenSSH\;
PROCESSOR_IDENTIFIER	Intel64 Family 6 Model 94 Stepping 3, GenuineIntel		
PROCESSOR_LEVEL	6	Browse Directory.	Browse <u>F</u> ile OK Cancel
DROCESCOR REVISION	5.02		
	Ne <u>w</u> Ed <u>i</u> t De <u>l</u> ete		



Figure 12-10

Add a semicolon (;) at the end of the variable value, and then add %JAVA_HOME%\bin.

Edit System Variable		×
Variable name:	Path	
Variable value:	va\javapath <mark>%JAVA_HOME%\bin;</mark> %SystemRoot%;%SystemRoot%\system32;%Sy	ystemRo
Browse Directory.	Browse File OK Ca	ancel

Figure 12-11

Step 7 Check whether the JDK is installed successfully.

Choose **Start** > **Run**, enter **cmd**, and press **Enter**. In the displayed dialog box, enter **java** - **version**.

Command Prompt
Microsoft Windows [Version 10.0.18363.720]
(c) 2019 Microsoft Corporation. All rights reserved.
C:\Users`_____java -version
java version "1.8.0_201"
Java (TM) SE Runtime Environment (build 1.8.0_201-b09)
Java HotSpot(TM) 64-Bit Server VM (build 25.201-b09, mixed mode)
C:______

Figure 12-12

If the Java version information is displayed, the installation is successful.

12.1.2 Installing Apache Maven

Step 1 Install Apache Maven.

Maven is a software project management tool that manages project construction, reports, and documents through a small segment of description information. In short, Maven is one of the tools for managing Java projects.

After Maven is used, third-party JAR packages such as **spring.jar** and **hibernate.jar** do not need to be copied to the **lib** directory of the project each time. The Maven configuration file can be used to automatically import JAR packages to the project. Programmers do not need to manually copy the JAR packages.

Download the latest version at http://maven.apache.org/download.cgi and decompress it to the **D:\apache-maven-3.5.0** directory.



This PC > Data (D:) > apache-maven-3.5.2	>		
^ Name	Date modified	Туре	Size
📕 bin	6/23/2020 3:58 PM	File folder	
📜 boot	6/23/2020 3:58 PM	File folder	
📜 conf	6/23/2020 3:58 PM	File folder	
📕 lib	6/23/2020 3:58 PM	File folder	
	11/7/2019 8:32 PM	File	18 KB
☐ NOTICE	11/7/2019 8:32 PM	File	6 KB
README.txt	11/7/2019 8:32 PM	Text Document	3 KB

Figure 12-13

Add **MAVEN_HOME** or **M2_HOME** to the system environment variables. Its value is the Maven installation directory **D:\apache-maven-3.5.0**.

Environment Variable	25	<	
User variables for m	nWX711840		
Variable	Value		
OneDrive	OneDrive C:\Users\mwx711840\OneDrive		
Path	C:\Users\mwx711840\AppData\Local\Microsoft\WindowsApp		
TEMP	C:\Users\mwx711840\AppData\Local\Temp		
ТМР	MP C:\Users\mwx711840\AppData\Local\Temp		
Edit System Variable		;	
Variable <u>n</u> ame:	MAVEN_HOME		
Variable <u>v</u> alue:	D:\apache-maven-3.5.2		
Browse <u>D</u> irectory	Browse <u>F</u> ile OK Can	icel	
NUMBER_OF_PRO	DCESSORS 4		
OS	Windows_NT		
Path	%MAVEN_HOME%\bin;C:\Program Files (x86)\Common Files\		
PATHEXT	.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;JSE;.WSF;.WSH;.MSC		
	New Edit Delete		
	OK Cancel		



Figure 12-14

Step 2 Verify the Maven installation.

Press **Win+R** to open the **Run** window, enter **cmd**, and run the **mvn** -version command to check the version.

Command Prompt	_		×
(c) 2019 Microsoft Corporation. All rights reserved.			
C:\Users\mvn -version Apache Maven 3.6.3 (cecedd343002696d0abb50b32b541b8a6ba2883f) Maven home: D:\apache-maven-3.6.3\bin\ Java version: 1.8.0_201, vendor: Oracle Corporation, runtime: les\Java\jdk1.8.0_201\jre Default locale: en_US, platform encoding: GBK OS name: "windows 10", version: "10.0", arch: "amd64", family	C:\Pı : ‴win	rogram ndows″	Fi
C:\Users\			

Figure 12-15

12.1.3 Installing Eclipse

Step 1 Download Eclipse.

Visit https://www.eclipse.org/downloads/packages/ and click **Downloads** on the menu bar.



Figure 12-16

Step 2 Select a version to download.

Select 64-bit to download. If the computer is 32-bit, click **Download Packages** and select 32-bit to download.



Eclipse II	DE 2020-03 R Packages		
A	Eclipse IDE for Java Developers 196 MB 194,030 DOWNLOADS The essential tools for any Java developer, including a Java IDE, a Git client, XML Editor, Mylyn, Maven and Gradle integration	*	Windows 64-bit Mac Cocoa 64-bit Linux 64-bit
	Eclipse IDE for Enterprise Java Developers (includes Incubating components)		
۲	400 MB 159,940 DOWNLOADS Tools for developers creating Java Enterprise and Web applications, including a Java IDE, tools for Web Services, JPA and Data Tools, JSF, Mylyn, Maven and Gradle, Git, and more. Click here to file a bug against Eclipse Web Tools Platform. Click here to file a bug against Eclipse Platform. Click here to file a bug against Eclipse Web Tools Platform. Click here to file a bug against Eclipse Web Tools Platform. Click here to file a bug against Eclipse Web Tools Platform. Click here to file a bug against Eclipse Web Tools Platform. Click here to file a bug against Eclipse Web Tools Platform.	*	Windows <mark>64-bit</mark> Mac Cocoa 64-bit Linux 64-bit

Figure 12-17

Step 3 Decompress the downloaded package and go to the folder. The following figure shows the Eclipse startup program.



Figure 12-18

Double-click the program to start it. If you open the tool for the first time, you need to configure a workspace. You can select another location or use the default drive C. Then click **OK**.



Figure 12-19

Step 4 Choose Help > Eclipse Marketplace, search for Maven, and click Install to Install the Maven plug-in in Eclipse.



CE Eclipse Mark	tetplace					
Eclipse Marketplace Select solutions to install. Press Finish to proceed with installation. Press the information button to see a detailed overview and a link to more information.						
Search Rece	nt Popular Installed 🖓 Eclipse Newsletter: Boot Build Eclips					
Find: maven	🔍 🖉 🗚 Markets 🔹 🔹 All Categories 🔹 🗣 Go					
Maven Integration for Eclipse (Luna) 1.5.0						
m2e provides comprehensive Maven integration for Eclipse. You can use m2e to manage both simple and multi-module Maven projects, execute Maven builds via the <u>more info</u>						
	by <u>Eclipse.org</u> , EPL maven java build development					
* 100	Installs: 100K (935 last month)					
Java™ 8 support for m2e for Eclipse Kepler SR2						
Java [®] 8	This is a service release (version 1.4.1.20140328-1905) for m2e 1.4 that enables Java [™] 8 support for Kepler SR2-based Eclipse IDEs. Note that Java 8 Support in <u>more info</u>					
	by <u>The Eclipse Foundation</u> , EPL iava 8 m2e maven					

Figure 12-20

Step 5 Choose Window > Preferences > Maven > User Settings and configure the Maven in the installation directory.

Preferences		
type filter text	User Settings 🔶 🤝	>
 General Ant AspectJ Compiler Code Recommenders Help Install/Update Java JDT Weaving Maven Archetypes Discovery Errors/Warnings Installations Lifecycle Mapping Templates User Interface User Settings Mylyn Oomph Quick Search Run/Debug 	Global Settings (open file): D:\soft\maven\apache-maven-3.5.2-bin\apache-maven-3.5.2\conf\settings.xml User Settings (open file): D:\soft\maven\apache-maven-3.5.2-bin\apache-maven-3.5.2\conf\settings.xml Update Settings Local Repository (From merged user and global settings): D:\.m2\repository	Browse Browse Reindex
· · · · · · · · · · · · · · · · · · ·	ОК	Cancel

Figure 12-21

Step 6 Download the MRS2.0 sample code.

The address for downloading the sample project of MRS on HUAWEI CLOUD is https://github.com/huaweicloud/huaweicloud-mrs-example/tree/mrs-2.0.



Ex

imples for HUAWEI CLOUD MRS.							
-o- 3 commits	🐉 3 branches	🗊 0 packages	\bigcirc 0 releases	2 contributors			
ranch: mrs-2.0 - View #1]		Fine	d file Clone or download -			
This branch is 2 commits ahead, 1 commit behind master.							
chengdh90 modify hive example pom xml exclude gauss dependency			Use Git or checkout with SVN using the web URL.				
src	modify hive example pom xml e	exclude gauss dependency	https://github.com/huaweicloud/huaweiclo				
.gitignore	upload mrs-2.0 example code		Open in Desktop	Download ZIP			
README.md	Initial commit		13 months ago				
III README.md							
huaweicloud-mrs-example Examples for HUAWEI CLOUD MRS.							

Figure 12-22

Download the ZIP package and decompress it.



Figure 12-23

12.1.4 Importing an MRS 2.0 Sample Project to Eclipse

Step 1 Create a working set in Eclipse.

Start Eclipse, choose **File** > **New** > **Java Working Set**, enter the name, for example, **MRS2.0Demo**, and click **Finish**.



New Java Working Set								
Java Working Set ① No resources selected.	O							
Working set name:								
MRS2.0Demo								
Workspace content:	Working set <u>c</u> ontent:							
⊳ 🚮 > javasdkdemo [javasdkdemo ma	<u>A</u> dd ->							
	Add All ->							
	<- <u>R</u> emove							
	<- Remove All							
۰ III • •								
?		<u> </u>	Cancel					

Figure 12-24

Step 2 Import a sample project to Eclipse.

Decompress the package and start **Eclipse**. Then choose **File** > **Import**.


Import	
Select Import Existing Maven Projects	Ľ
Select an import source:	
type filter text	
 Gradle 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 Oomph Oomph Run/Debug Tasks Tasks XML 	



Click **Browse**, select the **huaweicloud-mrs-example-mrs-2.0** sample project folder in the decompressed package, select **Add project to working set**, select the **MRS2.0Demo** created in the previous step, and click **Finish**.



Import Maven Projects				
Maven Projects				
 Scanning errors (1): 1 Could not read pom.xml 				
Root Directory: D:\eclipseworkspace2020\huaweicloud-mrs-example-mrs-2.0	Browse			
Projects:				
/huaweicloud-mrs-example-mrs-2.0/src/hbase-examples/pom.xml hbase-exam	Select All			
/huaweicloud-mrs-example-mrs-2.0/src/hdfs-examples/pom.xml hdfs-example				
/huaweicloud-mrs-example-mrs-2.0/src/hive-examples/pom.xml hive-example				
/huaweicloud-mrs-example-mrs-2.0/src/katka-examples/pom.xml mrs-exampl	Select Tree			
/huaweicloud-mrs-example-mrs-2.0/src/mapreduce-examples/pom.xml_com.n_	Deselect Tree			
//uaweicloud-mrs-example-mrs-2.0/src/spark-examples/SparkHivetoHbaseJav				
//uaweicloud-mrs-example-mrs-2.0/src/spark-examples/SparkHivetoHbaseSca				
//uaweicloud mis example mis 2:0/stc/spark examples/sparkJavaExample/pc				
/huaweicloud-mrs-example-mrs-2.0/src/spark-examples/SparkLauncherJavaExa				
/huaweicloud-mrs-example-mrs-2.0/src/spark-examples/SparkLauncherScalaEx				
/huaweicloud-mrs-example-mrs-2.0/src/spark-examples/SparkOnHbaseJavaEx				
/hugunicloud mer grampla mer 20/crc/charle gramplac/SparleOnUbacaScalaE				
Add project(s) to working set	1			
MRS2.0Demo	-			
The second secon				
▶ Ad <u>v</u> anced				
(?) < <u>Back</u> <u>N</u> ext > <u>Finish</u>	Cancel			

Figure 12-26

Wait until the Maven dependency package is loaded.

🛿 Problems @ Javadoc 😣 Declaration 📮 Console 🖏 Progress 🕱	*	000	-	
Importing Maven projects				
nttps://repo.huaweicloud.com/repository/maven/huaweicloudsdk/org/eclipse/jetty/jetty-xml/9.3.19.v20170502/jetty-xml-9.3.19.v20170502.pom				

Figure 12-27

If an error is reported, ignore it and click **OK**.

Step 3 Modify the **pom** file.

Double-click the **pom** file in the **mapreduce-examples** project.



HCIA-Big Data V3.0 Lab Guide



Figure 12-28

Switch to the **pom.xml** page and add the following code, where the repositories code after the dependencies tag.

<repositories></repositories>	
<repository> <id>huaweicloudsdk</id> <id>huaweicloudsdk</id> <url>https://mirrors.huaweicloud.com/repository/maven/huaweicloudsdk/</url> <releases><enabled>true</enabled></releases> <snapshots><enabled>true</enabled></snapshots> </repository>	
<repository></repository>	

For details about the code, go to https://support.huaweicloud.com/en-us/devg-mrs/mrs_06_0002.html.

Configuration method 2
Add the following mirror warehouse address to the pom.xml file in the secondary development sample project.
<pre><repositories></repositories></pre>
<pre><repository></repository></pre>
<id>huaweicloudsdk</id>
<url>https://mirrors.huaweicloud.com/repository/maven/huaweicloudsdk/</url>
<releases><enabled>true</enabled></releases>
<pre><snapshots><enabled>true</enabled></snapshots></pre>
<repository></repository>
<id>central</id>
<name>Mavn Centreal</name>
<url>https://repol.maven.org/maven2/</url>

Figure 12-29



The modifications are as follows:



Figure 12-30

After saving the file, wait for Eclipse to download the JAR package and keep the network connection. Maven downloads the required JAR package from the Huawei image repository.

If the pom reports an error stating "Missing artifact jdk.tools:jdk.tools:jar:1.8", add the following information to the **pom.xml** file:



The following figure shows the content added to the **pom.xml** file:



	182⊜	<dependency></dependency>
exar	183	<pre><groupid>org.apache.hive</groupid></pre>
	184	<pre><artifactid>hive-service </artifactid></pre>
	185	<pre><version>1.3.0-mrs-1.6.0</version></pre>
	186	
	187⊝	<dependency></dependency>
	188	<groupid>jdk.tools</groupid>
	189	<pre><artifactid>jdk.tools</artifactid></pre>
	190	<version>1.8</version>
	191	<scope>system</scope>
	192	<systempath>\${JAVA_HOME}/lib/tools.jar</systempath>
	193	
	194	
	195 ⊝	<pre>krepositories></pre>
	196	
	197 ⊝	<repository></repository>
	100	zidahuawai alaudadkz/ida m
	Overview De	pendencies Dependency Hierarchy Effective POM pom.xml

Figure 12-31

Step 4 Modify the **pom** file.

Add the marked code to the **pom** file, indicating that the JAR package of the Gauss database is not introduced to the project.



Figure 12-32

<exclusion></exclusion>	<pre><groupid>com.huawei.gaussc10</groupid></pre>
<artifactid>gauss</artifactid>	

Step 5 Modify the Java dependency.

Right-click the project name and choose **Build Path** > **Configure Build Path** from the shortcut menu.

Select JRE System Library[J@SE-1.5], click Remove, click Add Library, select JRE System Library, and click Next.



Add Library	
Select the library type to add.	< → → < <
Order and Export	
JUnit Maven Managed Dependencies	Add JARs
Server Runtime User Library	Add External JARs
	Add Variable
	Add Library
bbA	External Class Folder
	Edit
	Remove
	Migrate JAR File
Image: Second	Apply

Figure 12-33

Select JDK1.8 from the Alternate JRE drop-down list and click Finish.

Add Library				
JRE System Library Select JRE for the project bu	ild path.			
System library				
© Execution environment:			-	Envir <u>o</u> nments
Alternate <u>J</u> RE:	jdk1.8.0_131		•	Installed JREs
© Workspace <u>d</u> efault JRE	jdk1.8.0_131)			
?	< <u>B</u> ack	<u>N</u> ext >	<u> </u>	Cancel

Figure 12-34



Select Java Compiler, set Compiler compliance level to 1.8, and click OK.

Propert	ies for spidertest			x
type filter	text	Java Compiler	⇔ - ⇒	- -
⊳ Resou Builde	irce irs	Enable project specific settings <u>Configure Wor</u>	kspace Set	tings
Java B	Build Path	JDK Compliance		
⊳ Java C	ode Style	Use compliance from execution environment on the <u>'Java Build Path'</u>		
⊳ Java C	Compiler	Compiler compliance level:	→ 1.8	-
⊳ Java E	ditor			
Javad	oc Location	Use default compliance settings		
⊳ Maver	n	Generated .class files compatibility:	1.8	
Projec	t Facets	Source competibility	1.8	-
Projec	t References	oodice companying.		
Refact	toring History	Disallow identifiers called 'assert':	Error	-
Run/D	ebug Settings	Disallow identifiers called 'enum'	Error	-
Server	r	District Mentiners carea endin.	LITOI	
⊳ Task F	Repository	Classfile Generation		
Tack I	Tags			

Figure 12-35

Select Yes.



Figure 12-36

The project architecture is as follows:



Figure 12-37

12.2 Binding an EIP to an ECS

Step 1 Access the cluster node management page.



Click the cluster name in the cluster list and click **Nodes**.

< mrs_hcia	Download Authentication C
Dashboard Nodes Components Alarms Patch	nes Files Jobs
Configure Task Node	
Node Group I	Node Type
✓ master_node_default_group I	Master
✓ core_node_analysis_group	Analysis Core
✓ core_node_streaming_group	Streaming Core

Figure 12-38



Click a node name under **core_node_streaming_group**, as shown in the following figure:

< mrs_hcia	Download
Dashboard Nodes Components Alarms	Patches Files
Node Group	Node Type
master_node_default_group	Master
 core_node_analysis_group 	Analysis Core
core_node_streaming_group	Streaming Core
Node J≡	IP 1≡
691b98cc-d5a7-466e-b6e4-3f4ae86618cc_node_str_corelonu	192.168.0.216

Figure 12-39



Select EIP and click **View EIP** to purchase an IP address. If you have purchased sufficient IP addresses when creating a cluster, click **Bind EIP**. Select **Pay-per-use**. After the purchase is complete, the Elastic Cloud Server page is displayed.

Billing Mode	Yearly/Monthly Pay-per-use
Region	CN East-Shanghait CN East-Shanghait An EIP can only be associated with a cloud resource in its same region. After the purchase, the region cannot be changed Exercise caution when selecting the region.
EIP Type	Dynamic BGP Static BGP Image: Comparison of the state of the stat
Billed By	Bandwidth 🙆 🕂 Traffic For Ispht/sharply fluctuating traffic For staggered traffic
	Billed based on usage duration and bandwidth size.
Bandwidth	1 2 5 10 100 200 ⑦ Custom - 5 + The bandwidth can be from 1 to 2,000 Mbbls. © Free Ants-DDoS protection
Bandwidth Name	bandwidth-224f
Advanced Settings *	Tag
Monitoring Price: ¥0.02/hour + Bandwidth P	Monitoring is enabled by default Free ince ¥0.252/hour Not

Figure 12-40

Step 3 Bind an IP address.

Click Bind EIP.

< 691b98cc-d5a7-466e-b6e4-3f4ae86618cc_node_str_corelonu					
Summary Disks NICs Security Groups EIPs Monitoring Tags					
Bind EIP View EIP					

Figure 12-41

Select an IP address and click **OK**.



Bind EIP									×
ECS Name	691b98cc-d5a7-466e-b6e4-3f4ae86618cc_node_str_corelonu								
Select NIC	NIC1(192.168.0.216) (Primary NIC)								
Select EIP	View EIP				En	ter an EIP.		Q	C
	EIP J≡	EIP Type ↓Ξ	Status JΞ	Bandw	1≡	Bandw	1≡	Bandwi	1≡
	116.63.37.33	Static BGP	Onbound	bandwidth	ı-22	Exclusive		5 Mbit/s	
			OK Cancel						

Figure 12-42

Refresh the page. You can see that the EIP is bound successfully.

< 691	< 691b98cc-d5a7-466e-b6e4-3f4ae86618cc_node_str_corelonu Start Stop Restart							
Summa	ary Disks NICs	Security Groups EIPs Monitoring Tags						
Bir	nd EIP View EIP							
^	116.63.37.33 192.168.0.216							
	EIP	116.63.37.33 Traffic Details	EIP Type	Static BGP				
	ID	45f67e5d-c24b-4a3e-bcd3-45edea37f564	Obtained	Jun 23, 2020 16:33:58 GMT+08:00				
	Status	Ø Bound	Bandwidth Name	bandwidth-224f				
	Bound Private IP Address	192.168.0.216	Bandwidth Size	5 Mbit/s				
	Details		Bandwidth ID	68255e91-ee01-4dbc-969a-15f40f2ea4d2 Traffic Details				
	Bandwidth Type	Exclusive	Expires On					
			Billing Mode	Pay-per-use				

Figure 12-43

12.3 Viewing the IP address of ZooKeeper

Step 1 Log in to the MRS cluster management page.



< mrs_hcia				Download Authentication Credential	Management Operations * Configur
Dashboard No	odes Components Alarms	Patches	Files Jobs Tenants	Backups & Restorations Bc	otstrap Actions Tags
Basic Information Lea	am more			_	
Cluster Name	mrs_hcia 🖉	Cluster Status	Running	MRS Manager	Manage
Billing Mode	Pay-per-use	Cluster Version	MRS 2.1.0	Cluster Type	Hybrid cluster
IAM User Sync 🔞	Not synchronized Click to synchronize	Cluster ID	691b98cc-d5a7-466e-b6e4-3f4ae86618cc	Created	Jun 22, 2020 08:55:13 GMT+08:00
AZ	AZ1	Subnet	subnet-cde6	VPC	vpc-zx
Data Connection	Manage	Agency	Manage Agency	EIP	116.63.42.182 ⑦ Add Security Group Rule
Kerberos Authentication	Disabled	Logging	Enabled	Streaming Core Node LVM	Disabled
Security Group	mrs_mrs_hcia_VIGg, full				

Figure 12-44

Step 2 Log in as user admin.

	User Login
MRS Manager	Username
	Login

Figure 12-45

Step 3 Check the status of the Zookeeper service.

Choose **Services** > **Service ZooKeeper** > **Instance**. The business IP address of ZooKeeper is displayed.



MRS M	lanager						
Dashboard	Services	Hosts	Alarm	s Audi	t	Tenant	Syst
Service ZooKee	per > Instance						
Service Statu	is Instance	Service Confi	guration	Resource Dist	ribution	I	
More -							
Role	÷	Host Name	\$	OM IP Address	\$	Business IP Addre	ss ‡
quorum	npeer	node-master	r1XFVm	192.168.0.6		192.168.0.6	

Figure 12-46

12.4 Viewing the IP Address of a Kafka Broker Instance

Step 1 Log in to the MRS cluster management page.

< mrs_hcia			Dor	wnload Authentication Credential	Management Operations * Configure
Dashboard No	odes Components Alarms	Patches	Files Jobs Tenants Ba	ackups & Restorations Bo	otstrap Actions Tags
Basic Information Lea	arn more			-	
Cluster Name	mrs_hcia 🖉	Cluster Status	Running	MRS Manager ⑦	Manage
Billing Mode	Pay-per-use	Cluster Version	MRS 2.1.0	Cluster Type	Hybrid cluster
IAM User Sync 🕐	Not synchronized Click to synchronize	Cluster ID	691b98cc-d5a7-466e-b6e4-3f4ae86618cc	Created	Jun 22, 2020 08:55:13 GMT+08:00
AZ	AZ1	Subnet	subnet-cde6	VPC	vpc-zx
Data Connection 🔞	Manage	Agency	Manage Agency	EIP	116.63.42.182 ② Add Security Group Rule
Kerberos Authentication	Disabled	Logging	Enabled	Streaming Core Node LVM	Disabled
Security Group	mrs_mrs_hcia_VIGg, full				

Figure 12-47

Step 2 Log in as user **admin**.



	User Login
	Username
MRS Manager	Password
	Login

Figure 12-48

Step 3 Check the status of the Zookeeper service.

Choose **Services** > **Service Kafka** > **Instance**. The business IP address of the Kafka Broker is displayed.

MRS M	lanager						
Dashboard	Services	Hosts	Alarms	Audit	t	Tenant	Syst
Service Kafka >	Instance						
Service Statu	s Instance	Service Config	uration Res	ource Disti	ribution	KafkaTopic N	/lonit
More -							
Role 🗧	;	Host Name 💲	OMI	P Address	* *	Business IP Addres	s ≑
Broker		node-str-corel	onu 192.1	68.0.216	Ľ	192.168.0.216	
Mirror	1aker	node-str-corel	onu 192.1	68.0.216		192.168.0.216	

Figure 12-49

12.5 Common Linux Commands

cd /home: to go to the /home directory.



cd..: to move one directory up. cd ../..: to move two directories up. cd: to go to the personal home directory. cd ~user1: to go to the personal home directory. cd -: to move to your previous directory. pwd: to show the current working directory you are in. ls: to view files in the directory. ls -F: to view files in a directory. ls -l: to show details about files and directories. ls -a: to show the hidden files. Is *[0-9]*: to show hidden file names and directory names that contain digits. tree: to show the contents of a directory in a tree-like format (1). lstree: to show the contents of a directory in a tree-like format (2). mkdir dir1: to create a directory named dir1. mkdir dir1 dir2: to create two directories at the same time. mkdir -p /tmp/. dir1/dir2: to create a directory tree. rm -f file1: to delete a file named file1. rmdir dir1: to delete a directory named **dir1**. rm -rf dir1: to delete a directory named dir1 and its content. rm -rf dir1 dir2: to delete two directories and their contents. mv dir1 new_dir: to rename/move a directory. cp file1 file2: to copy a file. cp dir/*: to copy all files in a directory to the current working directory. cp -a /tmp/dir1: to copy a directory to the current working directory. cp -a dir1 dir2: to copy a directory. In -s file1 lnk1: to create a soft link to a file or directory.

12.6 HDFS Commands

The fsck command is executed in HDFS to check data inconsistency. The fsck command can report file problems, such as block loss or lack of blocks.

The usage of the fsck command is as follows:

hdfs fsck <path> [-move -delete -openforwrite] [-files [-blocks [-locations -racks]]]</path>
<pre><path>: start directory to be checked</path></pre>
-move: to move the damaged file to /lost+found
-delete: to delete the damaged file
-openforwrite: to show the file that is being written
-files: to show all checked files
-blocks: to show the block report
-locations: to show the location of each block
-racks: to show the network topology of the DataNode

By default, fsck ignores files that are being written, and you can use the **-openforwrite** option to report such files.

Run the **hdfs fsck /1001/hive.log –racks** command to view the topology information of **/1001/hive.log**.



fi01host02:/tmp # hdfs fsck /10	01/hive.log -racks					
18/01/18 17:30:49 INFO hdfs.PeerCache: SocketCache disabled.						
Connecting to namenode via <u>http</u>	<u>s://fi01host02:25003/fsck?ugi=admin&racks=1&path=%2F1001%2Fhive.log</u>					
FSCK started by admin (auth:KER	BEROS SSL) from /192.168.225.12 for path /1001/hive.log at Thu Jan 18 17:30:49 GMT+08:00 2018					
.Status: HEALTHY	_					
Total size: 446 B						
Total dirs: 0						
Total files: 1						
Total symlinks:	0					
Total blocks (validated):	1 (avg. block size 446 B)					
Minimally replicated blocks:	1 (100.0 %)					
Over-replicated blocks:	0 (0.0 %)					
Under-replicated blocks:	0 (0.0 %)					
Mis-replicated blocks:	0 (0.0 %)					
Default replication factor:	3					
Average block replication:	3.0					
Corrupt blocks:	0					
Missing replicas:	0 (0.0 %)					
Number of data-nodes:	3					
Number of racks:	3					
FSCK ended at Thu Jan 18 17:30:	FSCK ended at Thu Jan 18 17:30:49 GMT+08:00 2018 in 2 milliseconds					
The filesystem under path '/100	1/hive.log' is HEALTHY					

Figure 12-50

hdfs fsck /1001/hive.log -files -blocks -locations -racks

The detailed information about each block in the file is displayed, including the rack information of the DataNode.

12.7 Yarn Application Operation Commands

1. Run the following command to check all applications on Yarn:

yarn application -list

In the Flink exercise, the **yarn-session.sh** script is used to start a Flink cluster. This is a Yarn application. Run the following command to view the Yarn application:



Figure 12-51

2. Run the following command to kill the Yarn application:

yarn application -kill application id

For example, to kill a Flink cluster application, run the **-list** command to view the ID, and then run the **kill** command.



[root@node-master1]Rhg ~]# yarn application -list				
2020-04-16 17:02:39,643 INFO client.AHSProxy: Connecting to Application History server at /0.0.0.0:10200				
Total number of applications (application-types: [], states: [SUBMITTED, ACCEPTED, RUNNING] and tags: []):2				
Application-Id	Application-Name	Application-Type	User	Queue-User
inal-State Progress	; ;	Tracking-URL		
application_1586998890444_0002	Spark-JDBCServer-192.168	8.0.99	SPARK	omm
UNDEFINED	10% http://no	ode-master1JRhg:4040		
application_1586998890444_0007	Flink session cluster	Apache Flink	root	root
UNDEFINED 100%	6 http://192.168	8.0.69:42552		
[root@node-master1JRhg ~]# yarn application -kill application_1586998890444_0007				
2020-04-16 17:06:40,126 INFO client.AHSProxy: Connecting to Application History server at /0.0.0.0:10200				
Killing application application_1586998890444_0007				
2020-04-16 17:06:40,444 INFO impl.YarnClientImpl: Killed application application_1586998890444_0007				
[root@node-master1JRhg ~]#				

Figure 12-52