目录

[1. 实验一：通过命令监控大数据平台运行状态 2](#_Toc22396)

[1.1. 实验目的 2](#_Toc21830)

[1.2. 实验要求 2](#_Toc7530)

[1.3. 实验环境 2](#_Toc9513)

[1.4. 实验过程 2](#_Toc16217)

[1.4.1. 实验任务一：通过命令查看大数据平台状态 2](#_Toc9907)

[1.4.2. 实验任务二： 通过命令查看Hadoop状态 9](#_Toc12376)

[2. 实验二 通过命令监控大数据平台资源状态 11](#_Toc28672)

[2.1. 实验目标 11](#_Toc2100)

[2.2. 实验要求 11](#_Toc13120)

[2.3. 实验环境 11](#_Toc11149)

[2.4. 实验过程 11](#_Toc6403)

[2.4.1. 实验任务一：通过命令查看YARN状态 11](#_Toc17786)

[2.4.2. 实验任务二：通过命令查看HDFS状态 12](#_Toc14781)

[2.4.3. 实验任务三：通过命令查看HBase状态 13](#_Toc10094)

[2.4.4. 实验任务四：通过命令查看Hive状态 16](#_Toc6299)

[3. 实验三 通过命令监控大数据平台服务状态 20](#_Toc2374)

[3.1. 实验目标 20](#_Toc20964)

[3.2. 实验要求 20](#_Toc12953)

[3.3. 实验环境 20](#_Toc378)

[3.4. 实验过程 20](#_Toc8137)

[3.4.1. 实验任务一： 通过命令查看ZooKeeper状态 20](#_Toc18994)

[3.4.2. 实验任务二： 通过命令查看Sqoop状态 24](#_Toc2154)

[3.4.3. 实验任务三： 通过命令查看Flume状态 26](#_Toc27930)

# 实验一：通过命令监控大数据平台运行状态

## 实验目的

完成本实验，您应该能够：

* 掌握大数据平台的运行状况
* 掌握查看大数据平台运行状况的命令

## 实验要求

* 熟悉查看大数据平台运行状态的方式
* 了解查看大数据平台运行状况的命令

## 实验环境

本实验所需之主要资源环境如表1-1所示。

表1-1 资源环境

|  |  |
| --- | --- |
| **服务器集群** | 3个以上节点，节点间网络互通，各节点最低配置：双核CPU、8GB内存、100G硬盘 |
| **运行环境** | CentOS 7.4 |
| **大数据平台** | H3C DataEngine-E0104-RHEL6-X86\_64 |
| **服务和组件** | 完成前面章节的实验，其他服务及组件根据实验需求安装 |

## 实验过程

### 实验任务一：通过命令查看大数据平台状态

#### 步骤一：查看Linux系统的信息（uname -a）

[root@master ~]# uname -a

**Linux master 3.10.0-693.el7.x86\_64 #1 SMP Tue Aug 22 21:09:27 UTC 2017 x86\_64 x86\_64 x86\_64 GNU/Linux**

结果显示，该Linux节点名称为master，内核发行号为3.10.0-693.el7.x86\_64。

#### 步骤二：查看硬盘信息

（1）查看所有分区（fdisk -l）

[root@master ~]# fdisk -l

**磁盘 /dev/sda：42.9 GB, 42949672960 字节，83886080 个扇区**

**Units = 扇区 of 1 \* 512 = 512 bytes**

**扇区大小(逻辑/物理)：512 字节 / 512 字节**

**I/O 大小(最小/最佳)：512 字节 / 512 字节**

**磁盘标签类型：dos**

**磁盘标识符：0x0009e895**

**设备 Boot Start End Blocks Id System**

**/dev/sda1 \* 2048 2099199 1048576 83 Linux**

**/dev/sda2 2099200 83886079 40893440 8e Linux LVM**

**磁盘 /dev/mapper/centos-root：39.7 GB, 39720058880 字节，77578240 个扇区**

**Units = 扇区 of 1 \* 512 = 512 bytes**

**扇区大小(逻辑/物理)：512 字节 / 512 字节**

**I/O 大小(最小/最佳)：512 字节 / 512 字节**

**磁盘 /dev/mapper/centos-swap：2147 MB, 2147483648 字节，4194304 个扇区**

**Units = 扇区 of 1 \* 512 = 512 bytes**

**扇区大小(逻辑/物理)：512 字节 / 512 字节**

**I/O 大小(最小/最佳)：512 字节 / 512 字节**

结果显示，硬盘空间为42.9GB。

（2）查看所有交换分区（swapon -s）

[root@master ~]# swapon -s

**文件名 类型 大小 已用 权限**

**/dev/dm-1 partition 2097148 0 -1**

结果显示，交换分区为2097148。

（3）查看文件系统占比（df -h）

[root@master ~]# df -h

**文件系统 容量 已用 可用 已用% 挂载点**

**/dev/mapper/centos-root 37G 4.3G 33G 12% /**

**devtmpfs 3.9G 0 3.9G 0% /dev**

**tmpfs 3.9G 0 3.9G 0% /dev/shm**

**tmpfs 3.9G 8.7M 3.9G 1% /run**

**tmpfs 3.9G 0 3.9G 0% /sys/fs/cgroup**

**/dev/sda1 1014M 143M 872M 15% /boot**

**tmpfs 781M 0 781M 0% /run/user/0**

结果显示，挂载点“/”的容量为37G，已使用4.3G。

#### 步骤三：查看网络IP地址（ifconfig）

[root@master ~]# ifconfig

**ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500**

**inet 192.168.1.6 netmask 255.255.255.0 broadcast 192.168.1.255**

**inet6 fe80::6b63:dc78:878e:35f3 prefixlen 64 scopeid 0x20<link>**

**inet6 fe80::2e35:1d99:a67d:6df9 prefixlen 64 scopeid 0x20<link>**

**inet6 fe80::84a9:35d5:e08d:bfeb prefixlen 64 scopeid 0x20<link>**

**ether 00:0c:29:f9:05:0e txqueuelen 1000 (Ethernet)**

**RX packets 373 bytes 41380 (40.4 KiB)**

**RX errors 0 dropped 0 overruns 0 frame 0**

**TX packets 452 bytes 50188 (49.0 KiB)**

**TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0**

**lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536**

**inet 127.0.0.1 netmask 255.0.0.0**

**inet6 ::1 prefixlen 128 scopeid 0x10<host>**

**loop txqueuelen 1 (Local Loopback)**

**RX packets 0 bytes 0 (0.0 B)**

**RX errors 0 dropped 0 overruns 0 frame 0**

**TX packets 0 bytes 0 (0.0 B)**

**TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0**

结果显示ens33的IP地址为192.168.1.6，子网掩码为255.255.255.0；回环地址为127.0.0.1，子网掩码为255.0.0.0。

#### 步骤四：查看所有监听端口（netstat -lntp）

[root@master ~]# netstat -lntp

**Active Internet connections (only servers)**

**Proto Recv-Q Send-Q Local Address Foreign Address State PID/Program name**

**tcp 0 0 0.0.0.0:22 0.0.0.0:\* LISTEN 932/sshd**

**tcp6 0 0 :::22 :::\* LISTEN 932/sshd**

**tcp6 0 0 :::3306 :::\* LISTEN 1074/mysqld**

[root@master ~]#

结果显示，在监听的端口分别为22、3306。

#### 步骤五：查看所有已经建立的连接（netstat -antp）

[hadoop@master bin]$ netstat -antp

**(Not all processes could be identified, non-owned process info**

**will not be shown, you would have to be root to see it all.)**

**Active Internet connections (servers and established)**

**Proto Recv-Q Send-Q Local Address Foreign Address State PID/Program name**

**tcp 0 0 0.0.0.0:50070 0.0.0.0:\* LISTEN 1453/java**

**tcp 0 0 0.0.0.0:22 0.0.0.0:\* LISTEN -**

**tcp 0 0 192.168.1.6:9000 0.0.0.0:\* LISTEN 1453/java**

**tcp 0 0 0.0.0.0:50090 0.0.0.0:\* LISTEN 1628/java**

**tcp 0 0 192.168.1.6:50608 192.168.1.6:9000 TIME\_WAIT -**

**tcp 0 0 192.168.1.6:22 192.168.1.1:49438 ESTABLISHED -**

**tcp 0 0 192.168.1.6:9000 192.168.1.8:39198 ESTABLISHED 1453/java**

**tcp 0 0 192.168.1.6:9000 192.168.1.7:49666 ESTABLISHED 1453/java**

**tcp6 0 0 192.168.1.6:3888 :::\* LISTEN 3925/java**

**tcp6 0 0 :::22 :::\* LISTEN -**

**tcp6 0 0 192.168.1.6:8088 :::\* LISTEN 1821/java**

**tcp6 0 0 192.168.1.6:8030 :::\* LISTEN 1821/java**

**tcp6 0 0 192.168.1.6:8031 :::\* LISTEN 1821/java**

**tcp6 0 0 192.168.1.6:8032 :::\* LISTEN 1821/java**

**tcp6 0 0 192.168.1.6:8033 :::\* LISTEN 1821/java**

**tcp6 0 0 :::2181 :::\* LISTEN 3925/java**

**tcp6 0 0 :::40648 :::\* LISTEN 3925/java**

**tcp6 0 0 :::3306 :::\* LISTEN -**

**tcp6 0 0 192.168.1.6:8031 192.168.1.7:51526 ESTABLISHED 1821/java**

**tcp6 0 0 192.168.1.6:8031 192.168.1.8:42024 ESTABLISHED 1821/java**

结果显示，已经连接上的本地端口分别为50608、22、9000、8031等。

#### 步骤六：实时显示进程状态（top），该命令可以查看进程对CPU、内存的占比等。

[root@master ~]# top

**top - 21:32:44 up 1:02, 2 users, load average: 0.00, 0.02, 0.05**

**Tasks: 112 total, 1 running, 111 sleeping, 0 stopped, 0 zombie**

**%Cpu(s): 0.0 us, 0.0 sy, 0.0 ni,100.0 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st**

**KiB Mem : 7994076 total, 7441732 free, 320652 used, 231692 buff/cache**

**KiB Swap: 2097148 total, 2097148 free, 0 used. 7401476 avail Mem**

**PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND**

**685 root 20 0 305296 6300 4924 S 0.3 0.1 0:08.90 vmtoolsd**

**1 root 20 0 190736 3780 2488 S 0.0 0.0 0:04.80 systemd**

**2 root 20 0 0 0 0 S 0.0 0.0 0:00.01 kthreadd**

**3 root 20 0 0 0 0 S 0.0 0.0 0:00.03 ksoftirqd/0**

**5 root 0 -20 0 0 0 S 0.0 0.0 0:00.00 kworker/0:0H**

**6 root 20 0 0 0 0 S 0.0 0.0 0:00.27 kworker/u256:0**

**7 root rt 0 0 0 0 S 0.0 0.0 0:00.49 migration/0**

**8 root 20 0 0 0 0 S 0.0 0.0 0:00.00 rcu\_bh**

**9 root 20 0 0 0 0 S 0.0 0.0 0:00.63 rcu\_sched**

**10 root rt 0 0 0 0 S 0.0 0.0 0:00.02 watchdog/0**

**11 root rt 0 0 0 0 S 0.0 0.0 0:00.01 watchdog/1**

**12 root rt 0 0 0 0 S 0.0 0.0 0:00.77 migration/1**

**13 root 20 0 0 0 0 S 0.0 0.0 0:00.09 ksoftirqd/1**

**15 root 0 -20 0 0 0 S 0.0 0.0 0:00.00 kworker/1:0H**

**16 root rt 0 0 0 0 S 0.0 0.0 0:00.01 watchdog/2**

**17 root rt 0 0 0 0 S 0.0 0.0 0:00.76 migration/2**

**18 root 20 0 0 0 0 S 0.0 0.0 0:00.14 ksoftirqd/2**

**20 root 0 -20 0 0 0 S 0.0 0.0 0:00.00 kworker/2:0H**

**21 root rt 0 0 0 0 S 0.0 0.0 0:00.01 watchdog/3**

**22 root rt 0 0 0 0 S 0.0 0.0 0:00.68 migration/3**

**23 root 20 0 0 0 0 S 0.0 0.0 0:00.02 ksoftirqd/3**

**25 root 0 -20 0 0 0 S 0.0 0.0 0:00.00 kworker/3:0H**

**27 root 20 0 0 0 0 S 0.0 0.0 0:00.01 kdevtmpfs**

**28 root 0 -20 0 0 0 S 0.0 0.0 0:00.00 netns**

**29 root 20 0 0 0 0 S 0.0 0.0 0:00.00 khungtaskd**

**30 root 0 -20 0 0 0 S 0.0 0.0 0:00.00 writeback**

**31 root 0 -20 0 0 0 S 0.0 0.0 0:00.00 kintegrityd**

#### 步骤七：查看CPU信息（ cat /proc/cpuinfo）

[root@master ~]# cat /proc/cpuinfo

**processor : 0**

**vendor\_id : GenuineIntel**

**cpu family : 6**

**model : 85**

**model name : Intel(R) Xeon(R) Gold 5118 CPU @ 2.30GHz**

**stepping : 4**

**microcode : 0x2000050**

**cpu MHz : 2294.123**

**cache size : 16896 KB**

**physical id : 0**

**siblings : 2**

**core id : 0**

**cpu cores : 2**

**apicid : 0**

**initial apicid : 0**

**fpu : yes**

**fpu\_exception : yes**

**cpuid level : 22**

**wp : yes**

**flags : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts mmx fxsr sse sse2 ss ht syscall nx pdpe1gb rdtscp lm constant\_tsc arch\_perfmon pebs bts nopl xtopology tsc\_reliable nonstop\_tsc aperfmperf eagerfpu pni pclmulqdq ssse3 fma cx16 pcid sse4\_1 sse4\_2 x2apic movbe popcnt tsc\_deadline\_timer aes xsave avx f16c rdrand hypervisor lahf\_lm abm 3dnowprefetch epb fsgsbase tsc\_adjust bmi1 hle avx2 smep bmi2 invpcid rtm rdseed adx smap xsaveopt dtherm ida arat pln pts hwp hwp\_act\_window hwp\_epp hwp\_pkg\_req**

**bogomips : 4589.21**

**clflush size : 64**

**cache\_alignment : 64**

**address sizes : 42 bits physical, 48 bits virtual**

**power management:**

结果显示，该CPU为4核，Intel(R) Xeon(R) Gold 5118 CPU @ 2.30GHz。

#### 步骤八：查看内存信息（ cat /proc/meminfo），该命令可以查看总内存、空闲内存等信息。

[root@master ~]# cat /proc/meminfo

**MemTotal: 7994076 kB**

**MemFree: 7441996 kB**

**MemAvailable: 7401740 kB**

**Buffers: 2112 kB**

**Cached: 176408 kB**

**SwapCached: 0 kB**

**Active: 265072 kB**

**Inactive: 137936 kB**

**Active(anon): 224980 kB**

**Inactive(anon): 8332 kB**

**Active(file): 40092 kB**

**Inactive(file): 129604 kB**

**Unevictable: 0 kB**

**Mlocked: 0 kB**

**SwapTotal: 2097148 kB**

**SwapFree: 2097148 kB**

**Dirty: 0 kB**

**Writeback: 0 kB**

**AnonPages: 224516 kB**

**Mapped: 29664 kB**

**Shmem: 8824 kB**

**Slab: 53172 kB**

**SReclaimable: 22956 kB**

**SUnreclaim: 30216 kB**

**KernelStack: 4464 kB**

**PageTables: 3948 kB**

**NFS\_Unstable: 0 kB**

**Bounce: 0 kB**

**WritebackTmp: 0 kB**

**CommitLimit: 6094184 kB**

**Committed\_AS: 780596 kB**

**VmallocTotal: 34359738367 kB**

**VmallocUsed: 191112 kB**

**VmallocChunk: 34359310332 kB**

**HardwareCorrupted: 0 kB**

**AnonHugePages: 180224 kB**

**HugePages\_Total: 0**

**HugePages\_Free: 0**

**HugePages\_Rsvd: 0**

**HugePages\_Surp: 0**

**Hugepagesize: 2048 kB**

**DirectMap4k: 81728 kB**

**DirectMap2M: 3063808 kB**

**DirectMap1G: 7340032 kB**

### 实验任务二： 通过命令查看Hadoop状态

#### 步骤一：切换到hadoop用户

若当前的用户为root，请切换到hadoop用户进行操作。

[root@master ~]# su - hadoop

#### 步骤二：切换到Hadoop的安装目录

[hadoop@master ~]$ cd /usr/local/src/hadoop/

#### 步骤三：启动Hadoop

[hadoop@master hadoop]$ start-all.sh

**This script is Deprecated. Instead use start-dfs.sh and start-yarn.sh**

**Starting namenodes on [master]**

**master: starting namenode, logging to /usr/local/src/hadoop/logs/hadoop-hadoop-namenode-master.out**

**192.168.1.7: starting datanode, logging to /usr/local/src/hadoop/logs/hadoop-hadoop-datanode-slave1.out**

**192.168.1.8: starting datanode, logging to /usr/local/src/hadoop/logs/hadoop-hadoop-datanode-slave2.out**

**Starting secondary namenodes [0.0.0.0]**

**0.0.0.0: starting secondarynamenode, logging to /usr/local/src/hadoop/logs/hadoop-hadoop-secondarynamenode-master.out**

**starting yarn daemons**

**starting resourcemanager, logging to /usr/local/src/hadoop/logs/yarn-hadoop-resourcemanager-master.out**

**192.168.1.8: starting nodemanager, logging to /usr/local/src/hadoop/logs/yarn-hadoop-nodemanager-slave2.out**

**192.168.1.7: starting nodemanager, logging to /usr/local/src/hadoop/logs/yarn-hadoop-nodemanager-slave1.out**

#### 步骤四：关闭Hadoop

[hadoop@master hadoop]$ stop-all.sh

**This script is Deprecated. Instead use stop-dfs.sh and stop-yarn.sh**

**Stopping namenodes on [master]**

**master: stopping namenode**

**192.168.1.8: stopping datanode**

**192.168.1.7: stopping datanode**

**Stopping secondary namenodes [0.0.0.0]**

**0.0.0.0: stopping secondarynamenode**

**stopping yarn daemons**

**stopping resourcemanager**

**192.168.1.7: stopping nodemanager**

**192.168.1.8: stopping nodemanager**

**no proxyserver to stop**

# 实验二 通过命令监控大数据平台资源状态

## 实验目标

完成本实验，您应该能够：

* 掌握大数据平台资源的运行状况
* 掌握查看大数据平台资源运行状况的命令

## 实验要求

* 熟悉查看大数据平台资源运行状态的方式
* 了解查看大数据平台资源运行状况的命令

## 实验环境

本实验所需之主要资源环境如表1-1所示。

表1-1 资源环境

|  |  |
| --- | --- |
| **服务器集群** | 3个以上节点，节点间网络互通，各节点最低配置：双核CPU、8GB内存、100G硬盘 |
| **运行环境** | CentOS 7.4 |
| **大数据平台** | H3C DataEngine-E0104-RHEL6-X86\_64 |
| **服务和组件** | 完成前面章节的实验，其他服务及组件根据实验需求安装 |

## 实验过程

### 实验任务一：通过命令查看YARN状态

#### 步骤一：确认切换到目录 /usr/local/src/hadoop

[hadoop@master hadoop]$cd /usr/local/src/hadoop

#### 步骤二：返回主机界面在Master主机上执行 start-all.sh

master节点启动zookeeper

[hadoop@master hadoop]$ zkServer.sh start

slave1节点启动zookeeper

[hadoop@slave1 hadoop]$ zkServer.sh start

slave2节点启动zookeeper

[hadoop@slave2 hadoop]$ zkServer.sh start

master节点

[hadoop@master hadoop]$ start-all.sh

#### 步骤三：执行JPS命令，发现Master上有NodeManager进程和ResourceManager进程，则YARN启动完成。

[hadoop@master hadoop]$jps

执行结果如下，说明YARN已启动。

[hadoop@master hadoop]$ jps

**2817 NameNode**

**3681 ResourceManager**

**3477 NodeManager**

**3909 Jps**

**2990 SecondaryNameNode**

[hadoop@master hadoop]$

### 实验任务二：通过命令查看HDFS状态

#### 步骤一：目录操作

切换到hadoop目录，执行cd /usr/local/src/hadoop命令

[hadoop@master hadoop]$ cd /usr/local/src/hadoop

查看HDFS目录

[hadoop@master hadoop]$ ./bin/hdfs dfs –ls /

#### 步骤二：查看HDSF的报告，执行命令： bin/hdfs dfsadmin -report

[hadoop@master hadoop]$ bin/hdfs dfsadmin -report

**Configured Capacity: 79401328640 (73.95 GB)**

**Present Capacity: 75129376768 (69.97 GB)**

**DFS Remaining: 75129131008 (69.97 GB)**

**DFS Used: 245760 (240 KB)**

**DFS Used%: 0.00%**

**Under replicated blocks: 8**

**Blocks with corrupt replicas: 0**

**Missing blocks: 0**

**Missing blocks (with replication factor 1): 0**

**-------------------------------------------------**

**Live datanodes (2):**

**Name: 192.168.1.8:50010 (slave2)**

**Hostname: slave2**

**Decommission Status : Normal**

**Configured Capacity: 39700664320 (36.97 GB)**

**DFS Used: 122880 (120 KB)**

**Non DFS Used: 2135302144 (1.99 GB)**

**DFS Remaining: 37565239296 (34.99 GB)**

**DFS Used%: 0.00%**

**DFS Remaining%: 94.62%**

**Configured Cache Capacity: 0 (0 B)**

**Cache Used: 0 (0 B)**

**Cache Remaining: 0 (0 B)**

**Cache Used%: 100.00%**

**Cache Remaining%: 0.00%**

**Xceivers: 1**

**Last contact: Mon May 04 21:54:13 CST 2020**

**Name: 192.168.1.7:50010 (slave1)**

**Hostname: slave1**

**Decommission Status : Normal**

**Configured Capacity: 39700664320 (36.97 GB)**

**DFS Used: 122880 (120 KB)**

**Non DFS Used: 2136649728 (1.99 GB)**

**DFS Remaining: 37563891712 (34.98 GB)**

**DFS Used%: 0.00%**

**DFS Remaining%: 94.62%**

**Configured Cache Capacity: 0 (0 B)**

**Cache Used: 0 (0 B)**

**Cache Remaining: 0 (0 B)**

**Cache Used%: 100.00%**

**Cache Remaining%: 0.00%**

**Xceivers: 1**

**Last contact: Mon May 04 21:54:13 CST 2020**

#### 步骤三：查看HDFS空间情况，执行命令：hdfs dfs -df

[hadoop@master hadoop]$ hdfs dfs -df /

**Filesystem Size Used Available Use%**

**hdfs://192.168.1.6:9000 79401328640 262144 75129102336 0%**

### 实验任务三：通过命令查看HBase状态

#### 步骤一：启动运行HBase

切换到HBase安装目录/usr/local/src/hbase，命令如下：

[hadoop@master hadoop]$cd /usr/local/src/hbase

[hadoop@master src]$ hbase version

**HBase 1.2.1**

**Source code repository git://asf-dev/home/busbey/projects/hbase revision=8d8a7107dc4ccbf36a92f64675dc60392f85c015**

**Compiled by busbey on Wed Mar 30 11:19:21 CDT 2016**

**From source with checksum f4bb4a14bb4e0b72b46f729dae98a772**

结果显示HBase1.2.1，说明HBase正在运行，版本号为1.2.1。

如果没有启动，则执行命令start-hbase.sh启动HBase。

[hadoop@master hbase]$ start-hbase.sh

**starting master, logging to /usr/local/src/hbase/logs/hbase-hadoop-master-master.out**

**Java HotSpot(TM) 64-Bit Server VM warning: ignoring option PermSize=128m; support was removed in 8.0**

**Java HotSpot(TM) 64-Bit Server VM warning: ignoring option MaxPermSize=128m; support was removed in 8.0**

**master: starting regionserver, logging to /usr/local/src/hbase/logs/hbase-hadoop-regionserver-master.out**

**slave1: starting regionserver, logging to /usr/local/src/hbase/logs/hbase-hadoop-regionserver-slave1.out**

**slave2: starting regionserver, logging to /usr/local/src/hbase/logs/hbase-hadoop-regionserver-slave2.out**

**master: Java HotSpot(TM) 64-Bit Server VM warning: ignoring option PermSize=128m; support was removed in 8.0**

**master: Java HotSpot(TM) 64-Bit Server VM warning: ignoring option MaxPermSize=128m; support was removed in 8.0**

**slave1: Java HotSpot(TM) 64-Bit Server VM warning: ignoring option PermSize=128m; support was removed in 8.0**

**slave1: Java HotSpot(TM) 64-Bit Server VM warning: ignoring option MaxPermSize=128m; support was removed in 8.0**

**slave2: Java HotSpot(TM) 64-Bit Server VM warning: ignoring option PermSize=128m; support was removed in 8.0**

**slave2: Java HotSpot(TM) 64-Bit Server VM warning: ignoring option MaxPermSize=128m; support was removed in 8.0**

#### 步骤二：查看HBase版本信息

执行命令hbase shell，进入HBase命令交互界面。

[hadoop@master hadoop]$ hbase shell

**SLF4J: Class path contains multiple SLF4J bindings.**

**SLF4J: Found binding in [jar:file:/usr/local/src/hbase/lib/slf4j-log4j12-1.7.5.jar!/org/slf4j/impl/StaticLoggerBinder.class]**

**SLF4J: Found binding in [jar:file:/usr/local/src/hadoop/share/hadoop/common/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBinder.class]**

**SLF4J: See http://www.slf4j.org/codes.html#multiple\_bindings for an explanation.**

**SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]**

**HBase Shell; enter 'help<RETURN>' for list of supported commands.**

**Type "exit<RETURN>" to leave the HBase Shell**

**Version 1.2.1, r8d8a7107dc4ccbf36a92f64675dc60392f85c015, Wed Mar 30 11:19:21 CDT 2016**

hbase(main):001:0>

输入version，查询HBase版本

hbase(main):001:0> version

**1.2.1, r8d8a7107dc4ccbf36a92f64675dc60392f85c015, Wed Mar 30 11:19:21 CDT 2016**

结果显示HBase版本为1.2.1。

#### 步骤三：查询HBase状态，在HBase命令交互界面，执行status命令

hbase(main):002:0> status

**1 active master, 0 backup masters, 3 servers, 0 dead, 0.6667 average load**

查询结果显示，1台活动master，0台备份masters，共3台服务主机，平均加载时间为0.6667秒。

我们还可以“简单”查询HBase的状态，执行命令status 'simple'

hbase(main):003:0> status 'simple'

**active master: master:16000 1589125905790**

**0 backup masters**

**3 live servers**

**master:16020 1589125908065**

**requestsPerSecond=0.0, numberOfOnlineRegions=1, usedHeapMB=28, maxHeapMB=1918, numberOfStores=1, numberOfStorefiles=1, storefileUncompressedSizeMB=0, storefileSizeMB=0, memstoreSizeMB=0, storefileIndexSizeMB=0, readRequestsCount=5, writeRequestsCount=1, rootIndexSizeKB=0, totalStaticIndexSizeKB=0, totalStaticBloomSizeKB=0, totalCompactingKVs=0, currentCompactedKVs=0, compactionProgressPct=NaN, coprocessors=[MultiRowMutationEndpoint]**

**slave1:16020 1589125915820**

**requestsPerSecond=0.0, numberOfOnlineRegions=0, usedHeapMB=17, maxHeapMB=440, numberOfStores=0, numberOfStorefiles=0, storefileUncompressedSizeMB=0, storefileSizeMB=0, memstoreSizeMB=0, storefileIndexSizeMB=0, readRequestsCount=0, writeRequestsCount=0, rootIndexSizeKB=0, totalStaticIndexSizeKB=0, totalStaticBloomSizeKB=0, totalCompactingKVs=0, currentCompactedKVs=0, compactionProgressPct=NaN, coprocessors=[]**

**slave2:16020 1589125917741**

**requestsPerSecond=0.0, numberOfOnlineRegions=1, usedHeapMB=15, maxHeapMB=440, numberOfStores=1, numberOfStorefiles=1, storefileUncompressedSizeMB=0, storefileSizeMB=0, memstoreSizeMB=0, storefileIndexSizeMB=0, readRequestsCount=4, writeRequestsCount=0, rootIndexSizeKB=0, totalStaticIndexSizeKB=0, totalStaticBloomSizeKB=0, totalCompactingKVs=0, currentCompactedKVs=0, compactionProgressPct=NaN, coprocessors=[]**

**0 dead servers**

**Aggregate load: 0, regions: 2**

显示更多的关于Master、Slave1和Slave2主机的服务端口、请求时间等详细信息。

如果需要查询更多关于HBase状态，执行命令 help 'status'

hbase(main):004:0> help 'status'

**Show cluster status. Can be 'summary', 'simple', 'detailed', or 'replication'. The**

**default is 'summary'. Examples:**

**hbase> status**

**hbase> status 'simple'**

**hbase> status 'summary'**

**hbase> status 'detailed'**

**hbase> status 'replication'**

**hbase> status 'replication', 'source'**

**hbase> status 'replication', 'sink'**

hbase(main):005:0> quit

结果显示出所有关于status的命令。

#### 步骤四 停止HBase服务

停止HBase服务，则执行命令stop-hbase.sh。

[hadoop@master hbase]$ stop-hbase.sh

**stopping hbasecat.........**

没有错误提示，显示$提示符时，即停止了HBase服务。

### 实验任务四：通过命令查看Hive状态

#### 步骤一：启动Hive

切换到/usr/local/src/hive目录，输入hive，回车。

[hadoop@master hadoop]$ cd /usr/local/src/hive

[hadoop@master hive]$ hive

**SLF4J: Class path contains multiple SLF4J bindings.**

**SLF4J: Found binding in [jar:file:/usr/local/src/hive/lib/hive-jdbc-2.0.0-standalone.jar!/org/slf4j/impl/StaticLoggerBinder.class]**

**SLF4J: Found binding in [jar:file:/usr/local/src/hive/lib/log4j-slf4j-impl-2.4.1.jar!/org/slf4j/impl/StaticLoggerBinder.class]**

**SLF4J: Found binding in [jar:file:/usr/local/src/hadoop/share/hadoop/common/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBinder.class]**

**SLF4J: See http://www.slf4j.org/codes.html#multiple\_bindings for an explanation.**

**SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]**

**Logging initialized using configuration in jar:file:/usr/local/src/hive/lib/hive-common-2.0.0.jar!/hive-log4j2.properties**

**Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.e. spark, tez) or using Hive 1.X releases.**

hive>

当显示hive>时，表示启动成功，进入到了Hive shell状态。

#### 步骤二：Hive操作基本命令

注意：Hive命令行语句后面一定要加分号。

（1）查看数据库

hive> show databases;

**OK**

**default**

**Time taken: 0.011 seconds, Fetched: 1 row(s)**

显示默认的数据库default。

（2）查看default数据库所有表

hive> use default;

hive> show tables;

**OK**

**test**

**Time taken: 0.026 seconds**

显示default数据中没有任何表。

（3）创建表stu，表的id为整数型，name为字符型

hive> create table stu(id int,name string);

**OK**

**Time taken: 0.53 seconds**

（4）为表stu插入一条信息，id号为001，name为张三

hive> insert into stu values (1001,"zhangsan");

**WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.e. spark, tez) or using Hive 1.X releases.**

**Query ID = hadoop\_20200515102811\_1bccf3d2-88e3-4403-b25b-1e51e6e215b5**

**Total jobs = 3**

**Launching Job 1 out of 3**

**Number of reduce tasks is set to 0 since there's no reduce operator**

**Starting Job = job\_1588987665170\_0001, Tracking URL = http://master:8088/proxy/application\_1588987665170\_0001/**

**Kill Command = /usr/local/src/hadoop/bin/hadoop job -kill job\_1588987665170\_0001**

**Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 0**

**2020-05-15 10:34:16,557 Stage-1 map = 0%, reduce = 0%**

**2020-05-15 10:34:37,656 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 5.63 sec**

**MapReduce Total cumulative CPU time: 5 seconds 630 msec**

**Ended Job = job\_1588987665170\_0001**

**Stage-4 is selected by condition resolver.**

**Stage-3 is filtered out by condition resolver.**

**Stage-5 is filtered out by condition resolver.**

**Moving data to: hdfs://192.168.1.6:9000/user/hive/warehouse/stu/.hive-staging\_hive\_2020-05-15\_10-33-51\_327\_8147862916316704428-1/-ext-10000**

**Loading data to table default.stu**

**MapReduce Jobs Launched:**

**Stage-Stage-1: Map: 1 Cumulative CPU: 5.63 sec HDFS Read: 4177 HDFS Write: 78 SUCCESS**

**Total MapReduce CPU Time Spent: 5 seconds 630 msec**

**OK**

**Time taken: 47.769 seconds**

按照以上操作，继续插入两条信息：id和name分别为1002、1003和lisi、wangwu。

（5）插入数据后查看表的信息

hive> show tables;

**OK**

**stu**

**test**

**values\_\_tmp\_\_table\_\_1**

**Time taken: 0.019 seconds, Fetched: 3 row(s)**

（6）查看表stu结构

hive> desc stu;

**OK**

**id int**

**name string**

**Time taken: 0.031 seconds, Fetched: 2 row(s)**

（7）查看表stu的内容

hive> select \* from stu;

**OK**

**1001 zhangsan**

**2002 lisi**

**3003 wangwu**

**Time taken: 0.101 seconds, Fetched: 3 row(s)**

#### 步骤三：通过Hive 命令行界面查看文件系统和历史命令

（1）查看本地文件系统，执行命令 ! ls /usr/local/src;

hive> ! ls /usr/local/src;

**flume**

**hadoop**

**hbase**

**hive**

**java**

**sqoop**

**zookeeper**

（2）查看HDFS文件系统，执行命令dfs -ls /;

hive> dfs -ls /;

**Found 5 items**

**drwxr-xr-x - hadoop supergroup 0 2020-05-04 22:06 /bigdata**

**-rw-r--r-- 3 hadoop supergroup 12 2020-05-04 22:12 /bigdatafile**

**drwxr-xr-x - hadoop supergroup 0 2020-05-10 23:51 /hbase**

**drwx-wx-wx - hadoop supergroup 0 2020-05-15 10:33 /tmp**

**drwxrwxrwx - hadoop supergroup 0 2020-04-23 14:08 /user**

hive> exit;

（3）查看在Hive中输入的所有历史命令

进入到当前用户Hadoop的目录/home/hadoop，查看.hivehistory文件。

[hadoop@master home]$ cd /home/hadoop

[hadoop@master ~]$ cat .hivehistory

**create database sample;**

**show databases;**

**create database sample;**

**use sample;**

**create table student(number STRING, name STRING)**

**row format delimited**

**fields terminated by "|"**

**stored as textfile;**

**exit;**

**show databases;**

**use default;**

**show tables;**

**create table stu(id int,name string);**

**insert into stu values (1001,"zhangsan");**

**show tables;**

**desc stu;**

**select \* from stu;**

**! ls /usr/local/src;**

**dfs -ls /;**

**exit;**

结果显示，之前在Hive命令行界面下运行的所有命令（含错误命令）都显示了出来，有助于维护、故障排查等工作。

# 实验三 通过命令监控大数据平台服务状态

## 实验目标

完成本实验，您应该能够：

* 掌握大数据平台服务的运行状况
* 掌握查看大数据平台服务运行状况的命令

## 实验要求

* 熟悉查看大数据平台服务运行状态的方式
* 了解查看大数据平台服务运行状况的命令

## 实验环境

本实验所需之主要资源环境如表1-1所示。

表1-1 资源环境

|  |  |
| --- | --- |
| **服务器集群** | 3个以上节点，节点间网络互通，各节点最低配置：双核CPU、8GB内存、100G硬盘 |
| **运行环境** | CentOS 7.4 |
| **大数据平台** | H3C DataEngine-E0104-RHEL6-X86\_64 |
| **服务和组件** | 完成前面章节的实验，其他服务及组件根据实验需求安装 |

## 实验过程

### 实验任务一： 通过命令查看ZooKeeper状态

#### 步骤一: 查看ZooKeeper状态，执行命令zkServer.sh status，结果显示如下

[hadoop@master ~]$ zkServer.sh status

**ZooKeeper JMX enabled by default**

**Using config: /usr/local/src/zookeeper/bin/../conf/zoo.cfg**

**Mode: follower**

以上结果中，Mode:follower表示为ZooKeeper的跟随者。

#### 步骤二: 查看运行进程

QuorumPeerMain：QuorumPeerMain是ZooKeeper集群的启动入口类，是用来加载配置启动QuorumPeer线程的。

执行命令jps以查看进程情况。

[hadoop@master ~]$ jps

**3987 Jps**

**3925 QuorumPeerMain**

**1628 SecondaryNameNode**

**1453 NameNode**

**1821 ResourceManager**

此时QuorumPeerMain进程已启动。

#### 步骤四: 在成功启动ZooKeeper服务后，输入命令zkCli.sh，连接到ZooKeeper服务。

[hadoop@master ~]$ zkCli.sh

**Connecting to localhost:2181**

**2020-05-15 14:47:11,157 [myid:] - INFO [main:Environment@100] - Client environment:zookeeper.version=3.4.8--1, built on 02/06/2016 03:18 GMT**

**2020-05-15 14:47:11,160 [myid:] - INFO [main:Environment@100] - Client environment:host.name=master**

**2020-05-15 14:47:11,160 [myid:] - INFO [main:Environment@100] - Client environment:java.version=1.8.0\_152**

**2020-05-15 14:47:11,162 [myid:] - INFO [main:Environment@100] - Client environment:java.vendor=Oracle Corporation**

**2020-05-15 14:47:11,162 [myid:] - INFO [main:Environment@100] - Client environment:java.home=/usr/local/src/java/jre**

**2020-05-15 14:47:11,162 [myid:] - INFO [main:Environment@100] - Client environment:java.class.path=/usr/local/src/zookeeper/bin/../build/classes:/usr/local/src/zookeeper/bin/../build/lib/\*.jar:/usr/local/src/zookeeper/bin/../lib/slf4j-log4j12-1.6.1.jar:/usr/local/src/zookeeper/bin/../lib/slf4j-api-1.6.1.jar:/usr/local/src/zookeeper/bin/../lib/netty-3.7.0.Final.jar:/usr/local/src/zookeeper/bin/../lib/log4j-1.2.16.jar:/usr/local/src/zookeeper/bin/../lib/jline-0.9.94.jar:/usr/local/src/zookeeper/bin/../zookeeper-3.4.8.jar:/usr/local/src/zookeeper/bin/../src/java/lib/\*.jar:/usr/local/src/zookeeper/bin/../conf:.::/usr/local/src/java/lib:/usr/local/src/java/jre/lib:/usr/local/src/sqoop/lib**

**2020-05-15 14:47:11,162 [myid:] - INFO [main:Environment@100] - Client environment:java.library.path=/usr/java/packages/lib/amd64:/usr/lib64:/lib64:/lib:/usr/lib**

**2020-05-15 14:47:11,162 [myid:] - INFO [main:Environment@100] - Client environment:java.io.tmpdir=/tmp**

**2020-05-15 14:47:11,163 [myid:] - INFO [main:Environment@100] - Client environment:java.compiler=<NA>**

**2020-05-15 14:47:11,163 [myid:] - INFO [main:Environment@100] - Client environment:os.name=Linux**

**2020-05-15 14:47:11,163 [myid:] - INFO [main:Environment@100] - Client environment:os.arch=amd64**

**2020-05-15 14:47:11,163 [myid:] - INFO [main:Environment@100] - Client environment:os.version=3.10.0-693.el7.x86\_64**

**2020-05-15 14:47:11,163 [myid:] - INFO [main:Environment@100] - Client environment:user.name=hadoop**

**2020-05-15 14:47:11,163 [myid:] - INFO [main:Environment@100] - Client environment:user.home=/home/hadoop**

**2020-05-15 14:47:11,163 [myid:] - INFO [main:Environment@100] - Client environment:user.dir=/usr/local/src/hadoop**

**2020-05-15 14:47:11,164 [myid:] - INFO [main:ZooKeeper@438] - Initiating client connection, connectString=localhost:2181 sessionTimeout=30000 watcher=org.apache.zookeeper.ZooKeeperMain$MyWatcher@42110406**

**Welcome to ZooKeeper!**

**2020-05-15 14:47:11,191 [myid:] - INFO [main-SendThread(localhost:2181):ClientCnxn$SendThread@1032] - Opening socket connection to server localhost/127.0.0.1:2181. Will not attempt to authenticate using SASL (unknown error)**

**JLine support is enabled**

**2020-05-15 14:47:11,249 [myid:] - INFO [main-SendThread(localhost:2181):ClientCnxn$SendThread@876] - Socket connection established to localhost/127.0.0.1:2181, initiating session**

**2020-05-15 14:47:11,260 [myid:] - INFO [main-SendThread(localhost:2181):ClientCnxn$SendThread@1299] - Session establishment complete on server localhost/127.0.0.1:2181, sessionid = 0x171f70f3bda20ea, negotiated timeout = 30000**

**WATCHER::**

**WatchedEvent state:SyncConnected type:None path:null**

**[zk: localhost:2181(CONNECTED) 0]**

**结果显示已经连接成功，系统输出ZooKeeper的相关环境配置信息，并在屏幕中输出“Welcome to ZooKeeper!”等信息。**

**输入help命令之后，屏幕会输出如下可用的ZooKeeper命令。**

**[zk: localhost:2181(CONNECTED) 0] help**

**ZooKeeper -server host:port cmd args**

**stat path [watch]**

**set path data [version]**

**ls path [watch]**

**delquota [-n|-b] path**

**ls2 path [watch]**

**setAcl path acl**

**setquota -n|-b val path**

**history**

**redo cmdno**

**printwatches on|off**

**delete path [version]**

**sync path**

**listquota path**

**rmr path**

**get path [watch]**

**create [-s] [-e] path data acl**

**addauth scheme auth**

**quit**

**getAcl path**

**close**

**connect host:port**

**[zk: localhost:2181(CONNECTED) 1]**

#### 步骤五: 使用Watch监听/hbase目录，一旦/hbase内容有变化，将会有提示。打开监视，执行命令get /hbase 1。

[zk: localhost:2181(CONNECTED) 0] get /hbase 1

**cZxid = 0x100000002**

**ctime = Thu Apr 23 16:02:29 CST 2020**

**mZxid = 0x100000002**

**mtime = Thu Apr 23 16:02:29 CST 2020**

**pZxid = 0x20000008d**

**cversion = 26**

**dataVersion = 0**

**aclVersion = 0**

**ephemeralOwner = 0x0**

**dataLength = 0**

**numChildren = 16**

**[zk: localhost:2181(CONNECTED) 1] set /hbase value-update**

**WATCHER::cZxid = 0x100000002**

**WatchedEvent state:SyncConnected type:NodeDataChanged path:/hbase**

**ctime = Thu Apr 23 16:02:29 CST 2020**

**mZxid = 0x20000c6d3**

**mtime = Fri May 15 15:03:41 CST 2020**

**pZxid = 0x20000008d**

**cversion = 26**

**dataVersion = 1**

**aclVersion = 0**

**ephemeralOwner = 0x0**

**dataLength = 12**

**numChildren = 16**

**[zk: localhost:2181(CONNECTED) 2] get /hbase**

**value-update**

**cZxid = 0x100000002**

**ctime = Thu Apr 23 16:02:29 CST 2020**

**mZxid = 0x20000c6d3**

**mtime = Fri May 15 15:03:41 CST 2020**

**pZxid = 0x20000008d**

**cversion = 26**

**dataVersion = 1**

**aclVersion = 0**

**ephemeralOwner = 0x0**

**dataLength = 12**

**numChildren = 16**

[zk: localhost:2181(CONNECTED) 3] quit

结果显示，当执行命令set /hbase value-update后，数据版本由0变成1，说明/hbase处于监控中。

### 实验任务二： 通过命令查看Sqoop状态

#### 步骤一: 查询Sqoop版本号，验证Sqoop是否启动成功。

首先切换到/usr/local/src/sqoop目录，执行命令：./bin/sqoop-version

[hadoop@master ~]$ cd /usr/local/src/sqoop

[hadoop@master sqoop]$ ./bin/sqoop-version

**Warning: /usr/local/src/sqoop/../hcatalog does not exist! HCatalog jobs will fail.**

**Please set $HCAT\_HOME to the root of your HCatalog installation.**

**Warning: /usr/local/src/sqoop/../accumulo does not exist! Accumulo imports will fail.**

**Please set $ACCUMULO\_HOME to the root of your Accumulo installation.**

**20/05/06 17:40:16 INFO sqoop.Sqoop: Running Sqoop version: 1.4.7**

**Sqoop 1.4.7**

**git commit id 2328971411f57f0cb683dfb79d19d4d19d185dd8**

**Compiled by maugli on Thu Dec 21 15:59:58 STD 2017**

结果显示：Sqoop 1.4.7，说明Sqoop版本号为1.4.7，并启动成功。

#### 步骤二: 测试Sqoop是否能够成功连接数据库

切换到Sqoop的目录，执行命令bin/sqoop list-databases --connect jdbc:mysql://master:3306/ --username root --password Password123$，命令中“master:3306”为数据库主机名和端口。

[hadoop@master hadoop]$cd /usr/local/src/sqoop

[hadoop@master sqoop]$ bin/sqoop list-databases --connect jdbc:mysql://master:3306/ --username root --password Password123$

**Warning: /usr/local/src/sqoop/../hcatalog does not exist! HCatalog jobs will fail.**

**Please set $HCAT\_HOME to the root of your HCatalog installation.**

**Warning: /usr/local/src/sqoop/../accumulo does not exist! Accumulo imports will fail.**

**Please set $ACCUMULO\_HOME to the root of your Accumulo installation.**

**20/05/15 12:15:57 INFO sqoop.Sqoop: Running Sqoop version: 1.4.7**

**20/05/15 12:15:57 WARN tool.BaseSqoopTool: Setting your password on the command-line is insecure. Consider using -P instead.**

**20/05/15 12:15:57 INFO manager.MySQLManager: Preparing to use a MySQL streaming resultset.**

**Fri May 15 12:15:57 CST 2020 WARN: Establishing SSL connection without server's identity verification is not recommended. According to MySQL 5.5.45+, 5.6.26+ and 5.7.6+ requirements SSL connection must be established by default if explicit option isn't set. For compliance with existing applications not using SSL the verifyServerCertificate property is set to 'false'. You need either to explicitly disable SSL by setting useSSL=false, or set useSSL=true and provide truststore for server certificate verification.**

**information\_schema**

**hive**

**mysql**

**performance\_schema**

**sys**

结果显示，可以连接到MySQL，并查看到Master主机中MySQL的所有库实例，如information\_schema、hive、mysql、performance\_schema和sys等数据库。

#### 步骤三: 执行命令sqoop help，可以看到如下内容，代表Sqoop启动成功。

[hadoop@master sqoop]$ sqoop help

**Warning: /usr/local/src/sqoop/../hcatalog does not exist! HCatalog jobs will fail.**

**Please set $HCAT\_HOME to the root of your HCatalog installation.**

**Warning: /usr/local/src/sqoop/../accumulo does not exist! Accumulo imports will fail.**

**Please set $ACCUMULO\_HOME to the root of your Accumulo installation.**

**20/05/15 13:42:02 INFO sqoop.Sqoop: Running Sqoop version: 1.4.7**

**usage: sqoop COMMAND [ARGS]**

**Available commands:**

**codegen Generate code to interact with database records**

**create-hive-table Import a table definition into Hive**

**eval Evaluate a SQL statement and display the results**

**export Export an HDFS directory to a database table**

**help List available commands**

**import Import a table from a database to HDFS**

**import-all-tables Import tables from a database to HDFS**

**import-mainframe Import datasets from a mainframe server to HDFS**

**job Work with saved jobs**

**list-databases List available databases on a server**

**list-tables List available tables in a database**

**merge Merge results of incremental imports**

**metastore Run a standalone Sqoop metastore**

**version Display version information**

**See 'sqoop help COMMAND' for information on a specific command.**

结果显示了Sqoop的常用命令和功能，如下表所示。

|  |  |  |
| --- | --- | --- |
| 序号 | 命令 | 功能 |
| 1 | import | 将数据导入到集群 |
| 2 | export | 将集群数据导出 |
| 3 | codegen | 生成与数据库记录交互的代码 |
| 4 | create-hive-table | 创建Hive表 |
| 5 | eval | 查看SQL执行结果 |
| 6 | import-all-tables | 导入某个数据库下所有表到HDFS中 |
| 7 | job | 生成一个job |
| 8 | list-databases | 列出所有数据库名 |
| 9 | list-tables | 列出某个数据库下所有的表 |
| 10 | merge | 将HDFS中不同目录下数据合在一起，并存放在指定的目录中 |
| 11 | metastore | 记录Sqoop job的元数据信息，如果不启动metastore实例，则默认的元数据存储目录为：~/.sqoop |
| 12 | help | 打印Sqoop帮助信息 |
| 13 | version | 打印Sqoop版本信息 |

### 实验任务三： 通过命令查看Flume状态

#### 步骤一:检查Flume安装是否成功，执行flume-ng version命令，查看Flume的版本。

[hadoop@master sqoop]$ cd /usr/local/src/flume

[hadoop@master flume]$ flume-ng version

**Flume 1.6.0**

**Source code repository: https://git-wip-us.apache.org/repos/asf/flume.git**

**Revision: 2561a23240a71ba20bf288c7c2cda88f443c2080**

**Compiled by hshreedharan on Mon May 11 11:15:44 PDT 2015**

**From source with checksum b29e416802ce9ece3269d34233baf43f**

[hadoop@master flume]$

#### 步骤二:添加example.conf到/usr/local/src/flume

[hadoop@master flume]$ vim /usr/local/src/flume/example.conf

# 在文件中写入以下内容

# a1是agent名，r1,k1,c1是a1的三个组件

a1.sources=r1

a1.sinks=k1

a1.channels=c1

# 设置r1源文件的类型、路径和文件头属性

a1.sources.r1.type=spooldir

a1.sources.r1.spoolDir=/usr/local/src/flume/

a1.sources.r1.fileHeader=true

# 设置k1目标存储器属性

a1.sinks.k1.type=hdfs # 目标存储器类型hdfs

a1.sinks.k1.hdfs.path=hdfs://master:9000/flume # 目标存储位置

a1.sinks.k1.hdfs.rollsize=1048760 #临时文件达1048760 bytes时，滚动形成目标文件

a1.sinks.k1.hdfs.rollCount=0 #0表示不根据events数量来滚动形成目标文件

a1.sinks.k1.hdfs.rollInterval=900 # 间隔900秒将临时文件滚动形成目标文件

a1.sinks.k1.hdfs.useLocalTimeStamp=true # 使用本地时间戳

# 设置c1暂存容器属性

a1.channels.c1.type=file # 使用文件作为暂存容器

a1.channels.c1.capacity=1000

a1.channels.c1.transactionCapacity=100

# 使用c1作为源和目标数据的传输通道

a1.sources.r1.channels = c1

a1.sinks.k1.channel = c1

#### 步骤三:启动Flume Agent a1 日志控制台

[hadoop@master flume]$ /usr/local/src/flume/bin/flume-ng agent --conf ./conf --conf-file ./example.conf --name a1 -Dflume.root.logger=INFO,console

#### 步骤四: 查看结果

[hadoop@master flume]$ hdfs dfs -lsr /flume

**drwxr-xr-x - hadoop supergroup 0 2020-05-15 15:16 /flume/20200515**

**-rw-r--r-- 2 hadoop supergroup 11 2020-05-15 15:16 /flume/20200515/events-.1545376595231**