

## 实验二 Hadoop 1.x 部署 \*

### 2.1 实验目的

- 学习 Hadoop 1.x 部署，理解单机集中式部署、单机伪分布式部署两种部署方式之间的区别
- 学会通过查找系统日志中的错误来解决系统部署中遇到的问题
- 学会基本的 HDFS Shell 操作命令
- 通过系统部署理解 Hadoop 1.x 的体系架构，以及 HDFS 和 MapReduce 之间的关系

### 2.2 实验任务

- 完成 Hadoop 1.x 的单机集中式部署，并运行 WordCount 的示例程序
- 完成 Hadoop 1.x 的单机伪分布式部署，并运行 WordCount 的示例程序

### 2.3 实验环境

- 操作系统：Ubuntu 18.04
- JDK 版本：1.8
- Hadoop 版本：1.2.1

### 2.4 实验步骤

#### 2.4.1 单机集中式部署

注意：在单机集中式部署 hadoop-1.2.1 时，需要将其它版本 Hadoop 的相关进程都关闭，否则将会产生冲突。

##### MapReduce

###### (1) 准备工作

- 登录 dase-local 用户
- 在用户目录下，下载 Hadoop 1.2.1 并解压

```
1 cd ~
2 wget https://archive.apache.org/dist/hadoop/common/hadoop-1.2.1/hadoop-1.
   .2.1.tar.gz (Ucloud 课程镜像中已经下载好了，位于 /software_prepare/ 目录下)
3 tar -xzf hadoop-1.2.1.tar.gz #解压下载好的的安装包
```

```

4 cd ~/hadoop-1.2.1
5 ./bin/hadoop version #查看Hadoop版本信息

```

## (2) 运行 MapReduce 应用程序

- 提交 jar 包并查看运行结果

运行 Grep 示例：

```

1 cd ~/hadoop-1.2.1
2 mkdir /home/dase-local/input #在用户目录下新建input文件夹
3 cp ./conf/*.xml /home/dase-local/input #拷贝文件至输入文件夹中
4 ./bin/hadoop jar hadoop-examples-1.2.1.jar grep /home/dase-local/input
   /home/dase-local/output/grep 'dfs[a-z.]+' # hadoop jar [jar包名称]
   [按Java程序内容给定参数]
5 cat /home/dase-local/output/grep/* #查看输出结果

```

运行成功后，结果如图2.1所示。

图 2.1 Grep 示例运行结果

- 在运行过程中查看进程

- 下载测试数据集 pd.train，并保存在 ~/input/ 中

```

1 cd ~(Ucloud 课程镜像中已经下载好了，位于 /software_prepare/ 目录下，名称为 Chinese-Cloze-RC-master.zip)
2 wget https://github.com/ymcui/Chinese-Cloze-RC/archive/master.zip
3 unzip master.zip
4 unzip Chinese-Cloze-RC-master/people_daily/pd.zip
5 mv pd/pd.train ~/input

```

- 运行 WordCount 示例，并且在执行该作业过程中查看系统启动的进程

```

1 cd ~/hadoop-1.2.1
2 ./bin/hadoop jar hadoop-examples-1.2.1.jar wordcount
   /home/dase-local/input/pd.train /home/dase-local/output/wordcount

```

在作业执行时，另起一个终端，输入 jps 查看运行中的进程，详细信息如图2.2所示。

由图2.2(b)和图2.2(c)可知：在单机集中式部署方式下，系统只启动 RunJar 进程来运行整个 MapReduce 作业。此时也不会输出任何日志到本地，即生成 ~/hadoop-1.2.1/logs/ 目录和目录下日志文件。

```

21/02/02 20:45:50 INFO mapred.JobClient: Job complete: job_local1736613275_0001
21/02/02 20:45:50 INFO mapred.JobClient: Counters: 20
21/02/02 20:45:50 INFO mapred.JobClient: Map-Reduce Framework
21/02/02 20:45:50 INFO mapred.JobClient:     Spilled Records=99892012
21/02/02 20:45:50 INFO mapred.JobClient:     Map output materialized bytes=26272
21/02/02 20:45:50 INFO mapred.JobClient:     Reduce input records=25128639
21/02/02 20:45:50 INFO mapred.JobClient:     Virtual memory (bytes) snapshot=0
21/02/02 20:45:50 INFO mapred.JobClient:     Map input records=14892939
21/02/02 20:45:50 INFO mapred.JobClient:     SPLIT_RAW_BYTES=6435
21/02/02 20:45:50 INFO mapred.JobClient:     Map output bytes=3974172730
21/02/02 20:45:50 INFO mapred.JobClient:     Reduce shuffle bytes=0
21/02/02 20:45:50 INFO mapred.JobClient:     Physical memory (bytes) snapshot=0
21/02/02 20:45:50 INFO mapred.JobClient:     Reduce input groups=23997207
21/02/02 20:45:50 INFO mapred.JobClient:     Combine output records=51627283
21/02/02 20:45:50 INFO mapred.JobClient:     Reduce output records=23997207
21/02/02 20:45:50 INFO mapred.JobClient:     Map output records=171038044
21/02/02 20:45:50 INFO mapred.JobClient:     Combine input records=197536688
21/02/02 20:45:50 INFO mapred.JobClient:     CPU time spent (ms)=0
21/02/02 20:45:50 INFO mapred.JobClient:     Total committed heap usage (bytes)=
40115896320
21/02/02 20:45:50 INFO mapred.JobClient: File Input Format Counters
21/02/02 20:45:50 INFO mapred.JobClient:     Bytes Read=2173827625
21/02/02 20:45:50 INFO mapred.JobClient: FileSystemCounters
21/02/02 20:45:50 INFO mapred.JobClient:     FILE_BYTES_WRITTEN=123143343192
21/02/02 20:45:50 INFO mapred.JobClient:     FILE_BYTES_READ=139076075504
21/02/02 20:45:50 INFO mapred.JobClient: File Output Format Counters
21/02/02 20:45:50 INFO mapred.JobClient:     Bytes Written=2503965986

```

(a) 作业执行结束

5440 Jps  
3706 RunJar

(b) 运行过程中执行jps

bin	hadoop-ant-1.2.1.jar	ivy	sbin
build.xml	hadoop-client-1.2.1.jar	ivy.xml	share
c++	hadoop-core-1.2.1.jar	lib	src
CHANGES.txt	hadoop-examples-1.2.1.jar	libexec	webapps
conf	hadoop-minicluster-1.2.1.jar	LICENSE.txt	
contrib	hadoop-test-1.2.1.jar	NOTICE.txt	
docs	hadoop-tools-1.2.1.jar	README.txt	

(c) 执行完成后的目录内容

图 2.2 单机集中式部署方式下的作业运行情况

## 2.4.2 单机伪分布式部署

### HDFS

#### (1) 准备工作

- 登录 dase-local 用户

#### (2) 修改 HDFS 配置

配置文件位于 `~/hadoop-1.2.1/conf/` 目录下，修改其中的 `core-site.xml`、`hdfs-site.xml`

和 hadoop-env.sh 这 3 个文件。依次定位到这 3 个文件所在位置，右键单击文件，选择查看所有应用程序，找到文本编辑器并单击，点击选择后开始编辑文件内容。

- 修改 core-site.xml 文件

```

1 <configuration>
2   <!-- 指定数据存放目录 -->
3   <property>
4     <name>hadoop.tmp.dir</name>
5     <value>/home/dase-local/tmp-1.2.1</value>
6   </property>
7   <!-- 指定Hadoop所使用的文件系统schema (URI) , NameNode的地址 -->
8   <property>
9     <name>fs.default.name</name>
10    <value>hdfs://localhost:9000</value>
11  </property>
12
13 </configuration>
```

- 修改 hdfs-site.xml 文件

```

1 <configuration>
2   <!-- 指定HDFS上的文件副本数 -->
3   <property>
4     <name>dfs.replication</name>
5     <value>1</value>
6   </property>
7   <!-- 指定NameNode存储其元数据和编辑日志的目录的URI -->
8   <property>
9     <name>dfs.namenode.name.dir</name>
10    <value>file:/home/dase-local/tmp-1.2.1/dfs/name</value>
11  </property>
12  <!-- 指定DataNode存储块所在目录的URI -->
13  <property>
14    <name>dfs.datanode.name.dir</name>
15    <value>file:/home/dase-local/tmp-1.2.1/dfs/data</value>
16  </property>
17 </configuration>
```

- 修改 hadoop-env.sh 文件

找到 `# export JAVA_HOME=/usr/lib/j2sdk1.5-sun` 这一行，将此行修改为：`export JAVA_HOME=/usr/local/jdk1.8` 即可。

### (3) 启动 HDFS 服务

- 格式化 NameNode

注意：仅在第一次启动 HDFS 时才需要格式化 NameNode，如果是重启 HDFS，那么跳过“格式化 NameNode”这一步，直接执行下一步“启动 HDFS 服务”即可。此外，在进行 NameNode 格式化之前，如果 `~/hadoop-1.2.1/tmp-1.2.1` 文件夹已存在，那么需要删除该文件夹后再执行以下格式化命令。

```
1 cd ~/hadoop-1.2.1
2 ./bin/hadoop namenode -format
```

- 启动 HDFS 服务

```
1 cd ~/hadoop-1.2.1
2 ./bin/start-dfs.sh #启动HDFS服务进程
```

#### (4) 查看 HDFS 服务信息

- 使用 jps 查看进程，验证 HDFS 是否成功启动

正常启动结果如图2.3所示。如启动失败，可参照下面的查看进程日志来排查原因。

```
14529 Jps
14437 SecondaryNameNode
14007 NameNode
14221 DataNode
```

图 2.3 jps 运行结果

- 查看 NameNode、DataNode、SecondaryNameNode 进程日志

本实验的进程日志记录在 `~/hadoop-1.2.1/logs/` 路径下，后缀为.log 的文件中。一般日志都是不断追加在日志文件末尾。因此，在文件末尾可以查看最近日志记录，通过查看记录的时间就能找到指定的日志信息。

- NameNode 进程日志

默认位置：`~/hadoop-1.2.1/logs/hadoop-* Namenode-* .log`

- DataNode 进程日志

默认位置：`~/hadoop-1.2.1/logs/hadoop-* DataNode-* .log`

- SecondaryNameNode 进程日志

默认位置：`~/hadoop-1.2.1/logs/hadoop-* SecondaryNameNode-* .log`

- 访问 HDFS Web 页面

通过 NameNode 的 Web 页面 `http://localhost:50070`，查看 HDFS 信息。如

图2.4所示，可以看到目前集群中 Live Nodes 显示为 1（即有 1 个 DataNode 节点）。

点击 Browse the filesystem 可以查看文件目录信息。

## NameNode 'localhost:9000'

Started:	Tue Feb 02 20:55:44 CST 2021							
Version:	1.2.1, r1503152							
Compiled:	Mon Jul 22 15:23:09 PDT 2013 by mattf							
Upgrades:	There are no upgrades in progress.							
<a href="#">Browse the filesystem</a>								
<a href="#">Namenode Logs</a>								
<b>Cluster Summary</b>								
<b>1 files and directories, 0 blocks = 1 total. Heap Size is 240 MB / 889 MB (26%)</b>								
Configured Capacity	: 103.51 GB							
DFS Used	: 28 KB							
Non DFS Used	: 85.12 GB							
DFS Remaining	: 18.39 GB							
DFS Used%	: 0 %							
DFS Remaining%	: 17.77 %							
Live Nodes	: 1							
Dead Nodes	: 0							
Decommissioning Nodes	: 0							
Number of Under-Replicated Blocks	: 0							
<b>NameNode Storage:</b>								
<table border="1"> <thead> <tr> <th>Storage Directory</th> <th>Type</th> <th>State</th> </tr> </thead> <tbody> <tr> <td>/home/dase-dis/tmp-1.2.1/dfs/name</td> <td>IMAGE_AND_EDITS</td> <td>Active</td> </tr> </tbody> </table>			Storage Directory	Type	State	/home/dase-dis/tmp-1.2.1/dfs/name	IMAGE_AND_EDITS	Active
Storage Directory	Type	State						
/home/dase-dis/tmp-1.2.1/dfs/name	IMAGE_AND_EDITS	Active						
This is Apache Hadoop release 1.2.1								

图 2.4 集群环境信息

### (5) 常用的 HDFS Shell 命令

- 命令中的路径说明

Hadoop 命令中若出现 HDFS 中的路径时，可以使用路径的全写或者缩写，二者是等价的。全写格式如：`hdfs://user/dase-local/xxx`。缩写格式则是从用户（如 `dase-local`）目录下开始，如：`xxx` 或者 `./xxx`。

- 目录操作

HDFS 中的目录操作包括：新建目录、查看目录内容、删除目录等。示例如下：

注意：第一次使用 HDFS 时，需要在 HDFS 中创建用户目录。

```

1 cd ~/hadoop-1.2.1
2 ./bin/hadoop fs -mkdir /user/dase-local
   #在HDFS中为当前dase-local用户创建一个用户目录
3 ./bin/hadoop fs -ls . #显示 hdfs:///user/dase-local 下的文件
4 ./bin/hadoop fs -ls /user/dase-local #显示 hdfs:///user/dase-local 下的文件

```

```

5 ./bin/hadoop fs -mkdir input #新建 hdfs:///user/dase-local/input 目录
6 ./bin/hadoop fs -rmr /user/dase-local/input
    #删除hdfs:///user/dase-local/input 目录
7 ./bin/hadoop fs -mkdir /input #新建 hdfs:///input 目录
8 ./bin/hadoop fs -rmr /input #删除 hdfs:///input 目录

```

- 文件操作

HDFS 中文件操作包括：上传文件、下载文件、移动文件等。示例如下：

- 从本地文件系统向 HDFS 中上传文件

```

1 cd ~/hadoop-1.2.1
2 ./bin/hadoop fs -mkdir input #新建 hdfs:///user/dase-local/input 目录
3 ./bin/hadoop fs -put README.txt input/
    #把本地文件系统的“~/hadoop-1.2.1/README.txt”上传到HDFS
    中当前用户目录的input目录下
4 ./bin/hadoop fs -ls input/ #查看文件是否成功上传到HDFS上
5 ./bin/hadoop fs -cat input/README.txt
    #查看上传到HDFS上的README.txt的文件内容

```

将之前下载保存至 `~/input/` 的文件 `pd.train` 上传至 HDFS。

```

1 ./bin/hadoop fs -put ~/input/pd.train input/
    #将本地文件系统的“~/input/pd.train”上传至HDFS
    中当前用户目录的input目录下

```

此时另起一个终端，在上传过程中运行 `jps`，查看出现的 `FsShell` 进程。如图2.5所示。

- 把 HDFS 中的文件下载到本地文件系统中

```

1 cd ~/hadoop-1.2.1
2 ./bin/hadoop fs -get input/README.txt ~/input
    #把HDFS上用户目录下input文件夹中的README.txt下载到~/input/目录下。
3 cd ~/input #切换到当前用户目录下的input/目录下
4 ls #列出所有文件
5 cat README.txt #查看从HDFS下载得到的README.txt文件内容

```

- 把文件从 HDFS 中的一个目录拷贝至 HDFS 中的另外一个目录

```

1 cd ~/hadoop-1.2.1
2 ./bin/hadoop fs -cp input/README.txt /
    #把刚刚上传的README.txt文件拷贝到HDFS的根目录/下
3 ./bin/hadoop fs -ls / #查看拷贝的文件

```

```
21076 Jps
14437 SecondaryNameNode
14007 NameNode
14221 DataNode
20878 FsShell
```

图 2.5 文件上传过程中的进程

#### (6) 停止 HDFS 服务

- 停止命令

```
1 cd ~/hadoop-1.2.1
2 ./bin/stop-dfs.sh
```

- 查看进程，验证是否成功停止服务

使用 jps 查看进程，不再出现 NameNode、SecondaryNameNode、DataNode 进程则表示服务已停止。

### MapReduce

#### (1) 修改 MapReduce 配置

修改 mapred-site.xml 文件，文件位置: `~/hadoop-1.2.1/conf/`。

```
1 <configuration>
2   <!-- JobTracker 的地址 -->
3   <property>
4     <name>mapred.job.tracker</name>
5     <value>localhost:9001</value>
6   </property>
7 </configuration>
```

#### (2) 启动 MapReduce 服务

注意：这里读者需要体会 HDFS 和 MapReduce 的关系，由于在我们的实验中，MapReduce 程序的输入来自 HDFS 中，所以在启动 MapReduce 服务的同时，需要开启 HDFS 服务。之后的实验同理。

- 启动 MapReduce 服务

```
1 cd ~/hadoop-1.2.1
2 ./bin/start-mapred.sh
```

- 启动 HDFS 服务

```
1 ./bin/start-dfs.sh
```

#### (3) 查看 MapReduce 服务信息

- 使用 jps 查看进程，验证是否成功启动服务

正常启动结果如图2.6所示。如果启动后缺少某些进程，说明启动过程中出现错误。参考下一步查看进程日志来排查原因，根据之日中的出错信息，在网上搜索相关解决方法。

```
14437 SecondaryNameNode
24118 TaskTracker
14007 NameNode
24219 Jps
14221 DataNode
23901 JobTracker
```

图 2.6 HDFS 和 MapReduce 启动后，运行中的进程

- 查看 JobTracker、TaskTracker 进程日志

- JobTracker 进程日志

默认位置：`~/hadoop-1.2.1/logs/*-jobtracker-*.``log`

- TaskTracker 进程日志

默认位置：`~/hadoop-1.2.1/logs/*-tasktracker-*.``log`

- 访问 JobTracker 的 Web 界面

通过 `http://localhost:50030/jobtracker.jsp` 查看 JobTracker 信息，如图2.7所示。

#### (4) 运行 MapReduce 应用程序

- 提交 jar 包并查看运行结果

运行 Grep 示例：

```
1 cd ~/hadoop-1.2.1
2 ./bin/hadoop fs -mkdir input
3 ./bin/hadoop fs -put conf/* input/ #将文件上传至HDFS作为应用程序的输入文件
4 ./bin/hadoop fs -rmr output/grep
#MapReduce不会覆盖输出，如果输出路径已存在，则在运行前删除
5 ./bin/hadoop jar hadoop-examples-1.2.1.jar grep input/ output/grep
'dfs[a-z.]+'
6 ./bin/hadoop fs -cat output/grep/p* #查看输出结果 结果内容取决于文件内容
```

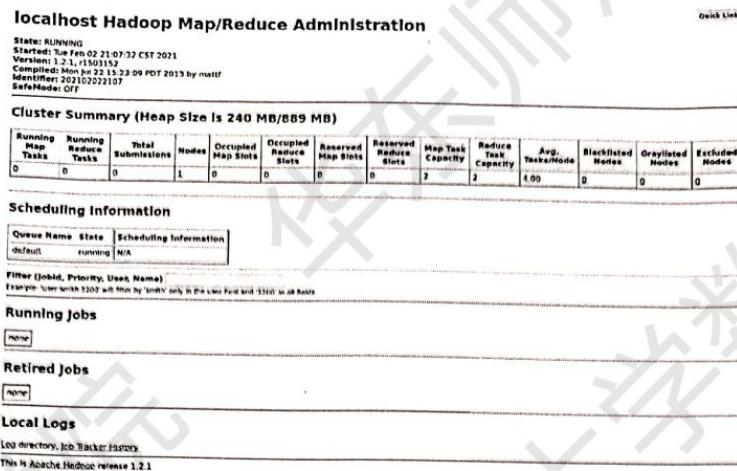


图 2.7 JobTracker Web 界面

执行成功后，运行结果如图2.8所示。

```
1          dfs.datanode.name.dir
1          dfs.namenode.name.dir
1          dfs.replication
1          dfs.server.namenode.
1          dfsadmin
```

图 2.8 查看 grep 执行结果

- 查看运行过程中的进程

运行 WordCount 示例，并且查看系统执行该作业过程中启动的进程：

```
1 cd ~/hadoop-1.2.1
2 ./bin/hadoop fs -rmr output/wordcount
#MapReduce不会覆盖输出，如果输出路径已存在，则在运行前删除
3 ./bin/hadoop jar hadoop-examples-1.2.1.jar wordcount input/pd.train
output/wordcount
```

在以上程序运行时，另起一个终端执行 jps，如图2.9所示。

注意：如果运行过程出现 Java Heap Space 异常，那么说明进程的堆内存不足。

解决此问题可修改以下两个文件。

- mapred-site.xml

```
1 <property>
2   <name>mapred.child.java.opts</name>
3   <value>-Xmx400m</value>
4   <!-- 设置每个child进程最多使用400M堆内存 -->
```

```
21/02/02 21:38:43 INFO mapred.JobClient: Job complete: job_202102022132_0001
21/02/02 21:38:43 INFO mapred.JobClient: Counters: 29
21/02/02 21:38:43 INFO mapred.JobClient: Map-Reduce Framework
21/02/02 21:38:43 INFO mapred.JobClient:   Spilled Records=91975489
21/02/02 21:38:43 INFO mapred.JobClient:   Map output materialized bytes=2
623870563
21/02/02 21:38:43 INFO mapred.JobClient:   Reduce input records=24899877
21/02/02 21:38:43 INFO mapred.JobClient:   Virtual memory (bytes) snapshot
=73095450624
21/02/02 21:38:43 INFO mapred.JobClient:   Map input records=14892939
21/02/02 21:38:43 INFO mapred.JobClient:   SPLIT_RAW_BYTES=3795
21/02/02 21:38:43 INFO mapred.JobClient:   Map output bytes=3974172730
21/02/02 21:38:43 INFO mapred.JobClient:   Reduce shuffle bytes=2623870563
21/02/02 21:38:43 INFO mapred.JobClient:   Physical memory (bytes) snapshot
t=10129211392
21/02/02 21:38:43 INFO mapred.JobClient:   Reduce input groups=23997207
21/02/02 21:38:43 INFO mapred.JobClient:   Combine output records=51382096
21/02/02 21:38:43 INFO mapred.JobClient:   Reduce output records=23997207
21/02/02 21:38:43 INFO mapred.JobClient:   Map output records=171038044
21/02/02 21:38:43 INFO mapred.JobClient:   Combine input records=197520263
21/02/02 21:38:43 INFO mapred.JobClient:   CPU time spent (ms)=478370
21/02/02 21:38:43 INFO mapred.JobClient:   Total committed heap usage (bytes)=9275179008
21/02/02 21:38:43 INFO mapred.JobClient:   File Input Format Counters
21/02/02 21:38:43 INFO mapred.JobClient:     Bytes Read=2173696553
```

(a) 作业执行结束

```
14437 SecondaryNameNode
12358 JobTracker
14007 NameNode
13433 RunJar
12585 TaskTracker
13900 Child
14221 DataNode
13918 Child
13935 Child
14079 Jps
```

(b) 运行过程中执行 jps

```
hadoop-dase-dis-datanode-ecnu01.log
hadoop-dase-dis-datanode-ecnu01.out
hadoop-dase-dis-jobtracker-ecnu01.log
hadoop-dase-dis-jobtracker-ecnu01.out
hadoop-dase-dis-jobtracker-ecnu01.out.1
hadoop-dase-dis-namenode-ecnu01.log
hadoop-dase-dis-namenode-ecnu01.out
hadoop-dase-dis-secondarynamenode-ecnu01.log
hadoop-dase-dis-secondarynamenode-ecnu01.out
hadoop-dase-dis-tasktracker-ecnu01.log
hadoop-dase-dis-tasktracker-ecnu01.out
hadoop-dase-dis-tasktracker-ecnu01.out.1
history
job_202102022107_0001_conf.xml
job_202102022107_0002_conf.xml
job_202102022132_0001_conf.xml
userlogs
```

(c) 执行完成后 logs 的目录内容

图 2.9 单机伪分布式部署作业运行情况

```
5 </property>
```

- hadoop-env.sh

```
1 # The maximum amount of heap to use, in MB. Default is 1000.  
2 export HADOOP_HEAPSIZE=2000
```

### (5) 查看 MapReduce 程序运行信息

- 访问 Hadoop Map/Reduce Administration 界面

通过 <http://localhost:50030/>，可以看到 Running Jobs 与 Retired Jobs 的相关信息。如图2.7所示。

- 查看 MapReduce 应用程序日志

- Task (即，Child 进程) 日志

默认位置: `~/hadoop-1.2.1/logs/userlogs/<jobid>/<attempt-id>`

- 查看 MapReduce 应用程序历史记录

通过 <http://localhost:50030/jobhistoryhome.jsp>，可以看到程序的历史运行记录。如图2.10所示。

### ecnu01 Hadoop Map/Reduce History Viewer

Filter (username:jobname)

Specify [user][:jobname keyword(s)][:MM/DD/YYYY]. Each of the three components is optional. Filter components are conjunctive.

Example: 'smith' will display jobs submitted by user 'smith'. 'smith:sort' will display jobs from user 'smith' having a 'sort' keyword in the jobname. ':07/04/2010' restricts to July 4, 2010

Available Jobs in History ( Displaying 3 jobs from 1 to 3 out of 3 jobs ) [[get more results](#)] [[show in one page](#)]  
[first page] [[last page](#)]

Job submit time	Job Id	Name	User
Tue Feb 02 21:34:00 CST 2021	job_202102022132_0001	word count	dase-dis
Tue Feb 02 21:18:30 CST 2021	job_202102022107_0002	grep-sort	dase-dis
Tue Feb 02 21:18:11 CST 2021	job_202102022107_0001	grep-search	dase-dis

图 2.10 JobHistory

### (6) 停止 MapReduce 服务

- 停止命令

```
1 cd hadoop-1.2.1  
2 ./bin/stop-all.sh
```

- 查看进程，验证是否成功停止服务

使用 `jps` 查看进程，不再出现 NameNode、SecondaryNameNode、DataNode、JobTracker、TaskTracker 等进程则服务停止。

## 2.5 思考题

- 1 在 mapred-site.xml 和 hadoop-env.sh 两个文件中所设置的内存大小分别是什么含义？
- 2 图2.5中的 FsShell 进程是什么？
- 3 图2.9(b)中哪些进程属于 HDFS，哪些进程属于 MapReduce？
- 4 如何查看一个文件有多少个分块？
- 5 在提交 MapReduce 作业的命令中通过 “mapred.map.tasks” 参数设置 Map 任务的数量，例如 “/bin/hadoop jar hadoop-examples-1.2.1.jar wordcount -Dmapred.map.tasks=2 input/pd.train output/wordcount”，观察 Map 任务数量是否改变？为什么？
- 6 在提交 MapReduce 作业的命令中通过 “mapred.reduce.tasks” 参数设置 Map 任务的数量，例如 “/bin/hadoop jar hadoop-examples-1.2.1.jar wordcount -Dmapred.reduce.tasks=2 input/pd.train output/wordcount”，观察 Reduce 任务数量是否改变？为什么？