

实验五 Spark 部署

5.1 实验目的

- 学习 Spark 的部署，简单使用 Spark-Shell
- 查看 Spark 的运行日志，体会与 MapReduce 运行过程中日志的区别
- 通过系统部署理解体系架构，体会 Spark 与 MapReduce 之间的区别

5.2 实验任务

- 完成 Spark 的单机集中式部署、单机伪分布式部署以及分布式部署
- 在单机伪分布式和分布式部署方式下，分别以 Client 和 Cluster 提交方式来运行示例程序

5.3 实验环境

- 操作系统：Ubuntu 18.04
- JDK 版本：1.8
- Spark 版本：2.4.7
- Hadoop 版本：2.10.1

5.4 实验步骤

5.4.1 单机集中式部署

(1) 准备工作

- 登录用户 dase-local
- 修改.bashrc 文件

说明：若终端配置的 TERM 环境变量为 xterm-256color，则会导致基于 Scala-2.11 构建的 Spark-2.4.7 中的 Spark-Shell 出现错误。对于该问题，需要将 TERM 环境变量设置为 xterm-color。而基于 Scala-2.12 构建的 Spark-2.4.7 不存在此问题，若采用 Scala-2.12 构建的 Spark-2.4.7，则无需修改 TERM 环境变量的配置。

```
vi ~/.bashrc
```

在文件结尾添加如下内容：

```
| export TERM=xterm-color
```

使.bashrc 配置生效：

```
1 | source ~/.bashrc
```

- 下载并安装 Spark

```
1 wget http://archive.apache.org/dist/spark/spark-2.4.7/spark-2.4.7-bin-without-hadoop.tgz # 下载安装包，可能较慢
2 tar -zxfv spark-2.4.7-bin-without-hadoop.tgz
3 mv ~/spark-2.4.7-bin-without-hadoop ~/spark-2.4.7
```

(2) 修改配置

- 修改 spark-env.sh 文件（文件路径：`~/dase-local/spark-2.4.7/conf/`）

将 spark-env.sh.template 文件重命名为 spark-env.sh：

```
mv ~/spark-2.4.7/conf/spark-env.sh.template ~/spark-2.4.7/conf/spark-env.sh
```

在 spark-env.sh 文件末尾添加如下内容：

```
1 # 因为下载的是 Hadoop Free 版本的 Spark, 所以需要配置 Hadoop 的路径  
2 HADOOP_HOME=/home/dase-local/hadoop-2.10.1  
3 export SPARK_DIST_CLASSPATH=$( $HADOOP_HOME/bin/hadoop classpath)  
4 export LD_LIBRARY_PATH=$HADOOP_HOME/lib/native:$LD_LIBRARY_PATH
```

(3) 运行 Spark 应用程序

- 通过 Spark-Shell 运行应用程序

- 执行以下命令进入 Spark-Shell。进入 Spark-Shell 后如图5.1所示

```
~/spark-2.4.7/bin/spark-shell --master local
```

```
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
Spark context Web UI available at http://ecnu01:4040
Spark context available as 'sc' (master = local, app id = local-1612254635858).
Spark session available as 'spark'.
Welcome to

    / \ / \ / \ / \ / \
   / \ \ / \ / \ / \ / \
  / \ / \ . / \ / \ / \ / \
 / \ / \ / \ / \ / \ / \ / \
version 2.4.7

Using Scala version 2.11.12 (Java HotSpot(TM) 64-Bit Server VM, Java 1.8.0_261)
Type in expressions to have them evaluated.
Type :help for more information.

scala> |
```

图 5.1 Spark-Shell

- 在 `scala>` 后输入 Scala 代码

```
1 sc.textFile("file:///home/dase-local/spark-2.4.7/RELEASE").flatMap(_.split
  (" ")).map((_, 1)).reduceByKey(_ + _).collect
```

此处执行的是统计 `/home/dase-local/spark-2.4.7/RELEASE` 文件中的单词数量，执行后应打印出如图 5.2 所示结果。

```
res0: Array[(String, Int)] = Array((-Psparkr,1), (-B,1), (Spark,1), (-Pkubernetes,1),
  (-Pyarn,1), (revision,1), (Build,1), (built,1), (-DzincPort=3038,1), (-Pfile,1),
  ((git,1), (2.6.5,1), (flags:,1), (-PMesos,1), (for,1), (-Pkafka-0-8,1),
  (-Phadoop-provided,1), (1421la1),1), (2.4.7,1), (Hadoop,1))
```

图 5.2 统计结果

- 在 `scala>` 后输入 “`:q`” 来退出 Spark-Shell
- 通过提交 jar 包运行应用程序

```
1 ~/spark-2.4.7/bin/spark-submit --master local --class org.apache.spark.
  examples.SparkPi ~/spark-2.4.7/examples/jars/spark-examples_2.11-
  2.4.7.jar
```

在运行过程中另起一个终端执行 `jps` 查看进程，此时只会出现 `SparkSubmit` 进程，应用程序运行结束后该进程消失，如图 5.3 所示。

```
dase-local@ecnu01:~$ jps
9484 Jps
9374 SparkSubmit
dase-local@ecnu01:~$ jps
9528 Jps
```

图 5.3 local 部署方式下运行过程中存在的进程

运行结果如图 5.4 所示，可以看到图片中输出的 π 的近似值。

```
21/02/02 16:43:13 INFO scheduler.DAGScheduler: ResultStage 0 (reduce at SparkPi.
  scala:38) finished in 0.483 s
21/02/02 16:43:13 INFO scheduler.DAGScheduler: Job 0 finished: reduce at SparkPi.
  scala:38, took 0.549308 s
Pi is roughly 3.1452557262786316
21/02/02 16:43:13 INFO server.AbstractConnector: Stopped Spark@27f9e982{HTTP/1.1
  , [http/1.1]}{0.0.0.0:4041}
21/02/02 16:43:13 INFO ui.SparkUI: Stopped Spark web UI at http://ecnu01:4041
21/02/02 16:43:13 INFO spark.MapOutputTrackerMasterEndpoint: MapOutputTrackerMas-
terEndpoint stopped!
```

图 5.4 local 部署方式下运行结果

5.4.2 单机伪分布式部署

(1) 准备工作

- 登录用户 dase-local
- 启动已部署的 HDFS 单机伪分布式服务

```
1 ~/hadoop-2.10.1/sbin/start-dfs.sh
```

(2) 修改配置

依次定位到如下文件所在位置，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 spark-env.sh 文件（文件路径：~/spark-2.4.7/conf/）

定位到文件位置，在末尾添加以下内容：

```
1 export SPARK_MASTER_HOST=localhost # 主节点主机名
2 export SPARK_MASTER_PORT=7077 # 端口号
```

- 修改 spark-defaults.conf 文件（文件路径：~/spark-2.4.7/conf/）

```
1 cp ~/spark-2.4.7/conf/spark-defaults.conf.template
~/spark-2.4.7/conf/spark-defaults.conf # 根据模板新建spark-defaults.conf
```

在 spark-defaults.conf 文件末尾添加以下内容：

```
1 # 通过配置以下参数开启日志聚集功能
2 spark.eventLog.enabled=true
3 spark.eventLog.dir=hdfs://localhost:9000/tmp/spark_history
4 spark.history.fs.logDirectory=hdfs://localhost:9000/tmp/spark_history
```

- 修改 spark-config.sh 文件（文件路径：~/spark-2.4.7/sbin/）

在文件末尾添加以下内容：

```
1 export JAVA_HOME=/usr/local/jdk1.8
```

- 在 HDFS 中建立目录 /tmp/spark_history

```
1 ~/hadoop-2.10.1/bin/hdfs dfs -mkdir -p /tmp/spark_history
```

(3) 启动服务

- 启动命令

```
1 ~/spark-2.4.7/sbin/start-all.sh # 启动 Spark
2 ~/spark-2.4.7/sbin/start-history-server.sh # 启动应用日志服务器
```

(4) 查看 Spark 服务信息

- 使用 jps 查看进程，验证是否成功启动服务

在单机伪分布式部署方式下，该节点既充当 Master，又充当 Worker，故该节点上会出现 Master 和 Worker 进程。如图5.5所示。

```
21040 DataNode
18753 Master
21572 HistoryServer
21687 Jps
21319 SecondaryNameNode
20794 NameNode
18986 Worker
```

图 5.5 启动 Spark 服务后存在的进程

- 查看 Master、Worker、HistoryServer 进程日志

本实验的进程日志记录在 `~/spark-2.4.7/logs/` 路径下，后缀为.out 的文件中。

- Master 进程日志: `spark-*-.org.apache.spark.deploy.master.Master-1-* .out`
- Worker 进程日志: `spark-*-.org.apache.spark.deploy.worker.Worker-1-* .out`
- HistoryServer 进程日志: `spark-*-.org.apache.spark.deploy.history.HistoryServer-1-* .out`

- 访问 Spark Web 界面

通过 `http://localhost:8080`，可看到 Master 和 Worker，如图5.6所示。

The screenshot shows the Spark Master UI at `http://localhost:8080`. It displays the following information:

- Global Metrics:**
 - URL: `spark://localhost:7077`
 - Alive Workers: 1
 - Cores in use: 4 Total, 0 Used
 - Memory in use: 14.6 GB Total, 0.0 B Used
 - Applications: 0 Running, 0 Completed
 - Drivers: 0 Running, 0 Completed
 - Status: ALIVE
- Workers (1):**

Worker Id	Address	State	Cores	Memory
worker-20210202171340-219.228.148.181-46437	219.228.148.181:46437	ALIVE	4 (0 Used)	14.6 GB (0.0 B Used)
- Running Applications (0):**

Application ID	Name	Cores	Memory per Executor	Submitted Time	User	State	Duration
- Completed Applications (0):**

Application ID	Name	Cores	Memory per Executor	Submitted Time	User	State	Duration

图 5.6 Spark Master 界面

(5) 运行 Spark 应用程序

- 通过 Spark-Shell 运行应用程序

- 准备输入文件

```
1 ~/hadoop-2.10.1/bin/hdfs dfs -mkdir -p spark_input
2 ~/hadoop-2.10.1/bin/hdfs dfs -put ~/spark-2.4.7/RELEASE spark_input/
```

- 进入 Spark-Shell

(注: 如使用 localhost 无法正常启动, 可尝试将 localhost 改为 127.0.1.1)

```
1 ~/spark-2.4.7/bin/spark-shell --master spark://localhost:7077
```

- 在 *scala>* 后输入 Scala 代码

```
1 sc.textFile("spark_input/RELEASE").flatMap(_.split(" ")).map((_, 1))
2 .reduceByKey(_ + _).collect
```

此处执行的是统计 RELEASE 文件中的单词数量, 执行后应打印出如图5.7所示结果。

```
res0: Array[(String, Int)] = Array((-Psparkr,1), (Build,1), (built,1), (-Pflume, 1), (git,1), (-Mesos,1), (-Phadoop-provided,1), (1421la1),1), (-B,1), (Spark,1), (-Kubernetes,1), (-Pyarn,1), (revision,1), (-DzincPort=3038,1), (2.6.5,1), (flags:,1), (for,1), (-Pkafka-0-8,1), (2.4.7,1), (Hadoop,1))
```

图 5.7 示例程序运行结果

- 在 *scala>* 后输入 “:q” 来退出 Spark-Shell

• 通过提交 jar 包运行应用程序

(注: 如使用 localhost 无法正常启动, 可尝试将 localhost 改为 127.0.1.1。)

- Client 提交方式(默认)

该提交方式下 Driver 运行在客户端, 可以在客户端看到应用程序运行过程中的信息。

```
1 ~/spark-2.4.7/bin/spark-submit --deploy-mode client --master
2 spark://localhost:7077 --class org.apache.spark.examples.SparkPi
3 ~/spark-2.4.7/examples/jars/spark-examples_2.11-2.4.7.jar
```

在运行过程中另起一个终端执行 jps 查看进程, 如图5.8所示。此时会存在一个 CoarseGrainedExecutorBackend 进程, 负责创建及维护 Executor 对象。

运行结果如图5.9所示, 可以看到图片中输出的 Pi 的近似值。

- Cluster 提交方式

该提交方式下 Master 会随机选取一个 Worker 节点启动 Driver, 故在客户端看不到应用程序运行过程中的信息。

```
1 ~/spark-2.4.7/bin/spark-submit --deploy-mode cluster --master
```

```
21040 DataNode  
12420 SparkSubmit  
21319 SecondaryNameNode  
8808 HistoryServer  
12537 CoarseGrainedExecutorBackend  
8410 Master  
20794 NameNode  
8604 Worker  
12556 Jps
```

图 5.8 Client 提交方式运行过程中存在的进程

```
21/02/02 18:34:27 INFO scheduler.TaskSchedulerImpl: Removed TaskSet 0.0, whose tasks have all completed, from pool  
Pi is roughly 3.141475707378537  
21/02/02 18:34:27 INFO server.AbstractConnector: Stopped Spark@3c291aad{HTTP/1.1,[http/1.1]}{0.0.0.0:4040}  
21/02/02 18:34:27 INFO ui.SparkUI: Stopped Spark web UI at http://ecnu01:4040
```

图 5.9 Client 提交方式下示例程序运行结果

```
spark://localhost:7077 --class org.apache.spark.examples.SparkPi  
~/spark-2.4.7/examples/jars/spark-examples_2.11-2.4.7.jar
```

在运行过程中另起一个终端执行 jps 查看进程，如图5.10所示。在 Cluster 提交方式下，还可以看到一个 DriverWrapper 进程。

```
21040 DataNode  
15525 SparkSubmit  
15622 DriverWrapper  
21319 SecondaryNameNode  
15767 Jps  
15704 CoarseGrainedExecutorBackend  
8808 HistoryServer  
8410 Master  
20794 NameNode  
8604 Worker
```

图 5.10 Cluster 提交方式运行过程中存在的进程

(6) 查看 Spark 程序运行信息

- 实时查看应用运行情况

在应用运行过程中（如进入 Spark-Shell 之后），访问 <http://localhost:4040>。Web 界面如图5.11所示。

- 查看 Spark 应用程序日志

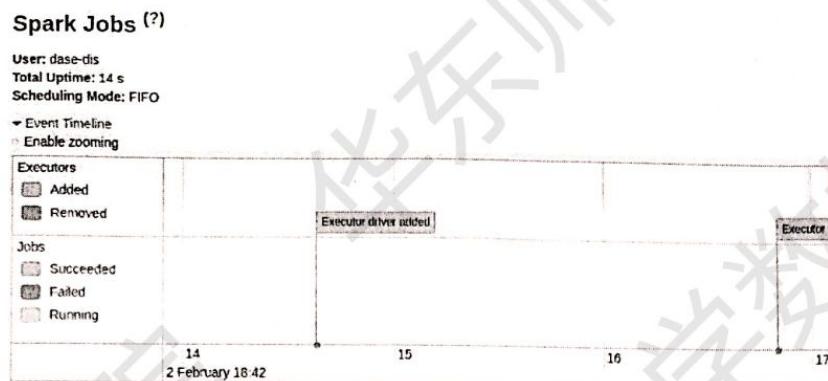


图 5.11 Spark Jobs 界面

在提交一个应用程序后，在`~/spark-2.4.7/work`下会出现应用程序运行日志文件夹。

访问 `http://localhost:8080`，点击 Completed Applications 下的对应应用程序的 Application ID，进入到如图5.12所示界面，点击 stdout 或者 stderr 可以查看对应日志信息。

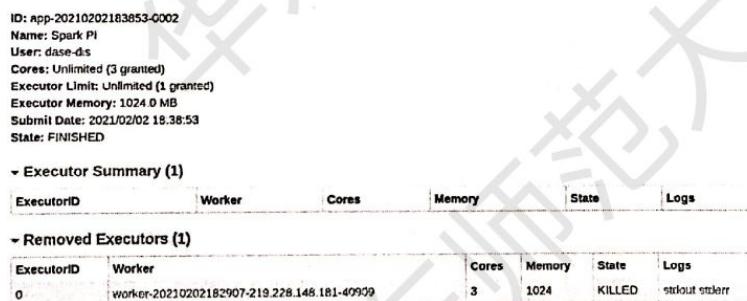


图 5.12 Spark 某个 Application 的界面

• 查看应用历史记录

在应用运行结束后，访问 `http://localhost:18080`。Web 界面如图5.13所示。

App ID	App Name	Started	Completed	Duration	Spark User	Last Updated	Event Log
app-20210202183853-0002	Spark PI	2021-02-02 18:38:53	2021-02-02 18:38:56	3 s	dase-dis	2021-02-02 18:38:56	stdout
app-20210202183474-0001	Spark PI	2021-02-02 18:34:24	2021-02-02 18:34:27	3 s	dase-dis	2021-02-02 18:34:27	stdout
app-20210202182931-0000	Spark Shell	2021-02-02 18:29:30	2021-02-02 18:33:07	3.6 min	dase-dis	2021-02-02 18:33:07	stdout

Show 1 to 3 of 3 entries
Show incomplete applications

图 5.13 Spark History 界面

(7) 停止服务

• 停止 Spark 服务

```
1 ~/spark-2.4.7/sbin/stop-all.sh      # 停止 Spark  
2 ~/spark-2.4.7/sbin/stop-history-server.sh # 停止日志服务器
```

- 停止 HDFS 服务

```
1 ~/hadoop-2.10.1/sbin/stop-dfs.sh
```

5.4.3 分布式部署

(1) 准备工作

- 准备 4 台机器，其中包括 1 个主节点（主机名：ecnu01）、2 个从节点（主机名：ecnu02 和 ecnu03）、1 个客户端（主机名：ecnu04）
- 4 台机器均已创建用户 dase-dis
- 4 台机器之间实现免密钥登录
- 登录 dase-dis 用户
- 检查 IP 与主机名映射是否正确，若 IP 地址发生改变，参考第??节进行修改
- 在客户端上修改.bashrc 文件

说明：若终端配置的 TERM 环境变量为 xterm-256color，则会导致基于 Scala-2.11 构建的 Spark-2.4.7 中的 Spark-Shell 出现错误。对于该问题，需要将 TERM 环境变量设置为 xterm-color。而基于 Scala-2.12 构建的 Spark-2.4.7 不存在此问题，若采用 Scala-2.12 构建的 Spark-2.4.7，则无需修改 TERM 环境变量的配置。

```
1 vi ~/.bashrc
```

添加如下内容：

```
1 export TERM=xterm-color
```

使.bashrc 配置生效：

```
1 source ~/.bashrc
```

- 在主节点上启动已部署的 HDFS 分布式服务

```
1 ~/hadoop-2.10.1/sbin/start-dfs.sh
```

- 在主节点上将之前下载的 spark-2.4.7-bin-without-hadoop.tgz 拷贝至当前用户目录下并解压

```
1 scp dase-local@localhost:~/spark-2.4.7-bin-without-hadoop.tgz ~/
```

```

2 tar -zxvf spark-2.4.7-bin-without-hadoop.tgz
3 mv ~/spark-2.4.7-bin-without-hadoop ~/spark-2.4.7

```

(2) 修改配置

以下操作在主节点执行：

- 修改 spark-env.sh 文件（文件路径：~/spark-2.4.7/conf/）

```
1 mv ~/spark-2.4.7/conf/spark-env.sh.template ~/spark-2.4.7/conf/spark-env.sh
```

以文本编辑器打开该文件，在 spark-env.sh 文件末尾添加以下内容：

```

1 # 由于我们下载的是 Hadoop Free 版本的 Spark，所以需要配置 Hadoop 的路径
2 HADOOP_HOME=/home/dase-dis/hadoop-2.10.1
3 export SPARK_DIST_CLASSPATH=$(HADOOP_HOME/bin/hadoop classpath)
4 export LD_LIBRARY_PATH=$HADOOP_HOME/lib/native:$LD_LIBRARY_PATH
5
6 export SPARK_MASTER_HOST=ecnu01 # 主节点主机名
7 export SPARK_MASTER_PORT=7077 # 端口号

```

- 修改 slaves 文件（文件路径：~/spark-2.4.7/conf/）

```
1 mv ~/spark-2.4.7/conf/slaves.template ~/spark-2.4.7/conf/slaves
```

以文本编辑器打开该文件，将文件修改为如下内容（注意注释 localhost 一行）：

```

1 # localhost
2 ecnu02
3 ecnu03

```

- 修改 spark-defaults.conf 文件（文件路径：~/spark-2.4.7/conf/）

```

1 mv ~/spark-2.4.7/conf/spark-defaults.conf.template
2 ~/spark-2.4.7/conf/spark-defaults.conf

```

以文本编辑器打开 spark-defaults.conf 文件，在末尾添加以下内容：

```

1 spark.eventLog.enabled=true
2 spark.eventLog.dir=hdfs://ecnu01:9000/tmp/spark_history
3 spark.history.fs.logDirectory=hdfs://ecnu01:9000/tmp/spark_history

```

- 修改 spark-config.sh 文件（文件路径：~/spark-2.4.7/sbin/）

在文件末尾添加以下内容：

```
1 export JAVA_HOME=/usr/local/jdk1.8
```

- 将配置好的 Spark 拷贝到其它节点

```

1 scp -r ~/spark-2.4.7 dase-dis@ecnu02:~
2 scp -r ~/spark-2.4.7 dase-dis@ecnu03:~
3 scp -r ~/spark-2.4.7 dase-dis@ecnu04:~

```

- 在 HDFS 中建立目录 /tmp/spark_history

```
~/hadoop-2.10.1/bin/hdfs dfs -mkdir -p /tmp/spark_history
```

(3) 启动服务

- 启动命令 (在主节点上执行)

```

1 ~/spark-2.4.7/sbin/start-all.sh      # 启动 Spark
2 ~/spark-2.4.7/sbin/start-history-server.sh # 启动应用日志服务器

```

(4) 查看 Spark 服务信息

- 使用 jps 查看进程，验证是否成功启动服务

在分布式部署方式下，主节点充当 Master 角色，从节点充当 Worker，故在主节点上存在 Master 进程（如图5.14所示），在从节点节点上存在 Worker 进程（如图5.15所示）。

```
dase-dis@ecnu01:~$ jps
7204 HistoryServer
29174 NameNode
29575 SecondaryNameNode
6954 Master
14543 Jps
```

图 5.14 主节点存在的进程

```
dase-dis@ecnu02:~$ jps
9618 DataNode
29933 Worker
11407 Jps
```

图 5.15 从节点存在的进程

- 查看 Master、Worker、HistoryServer 进程日志

本实验的进程日志记录在各节点 `~/spark-2.4.7/logs/` 路径下，后缀为.out 的文件中。主节点的日志中会出现 Master 进程启动的记录信息，从节点的日志中会出现 Worker 进程启动的记录信息。

- Master 进程日志: `spark-*org.apache.spark.deploy.master.Master-1-*.`.out
- Worker 进程日志: `spark-*org.apache.spark.deploy.worker.Worker-1-*.`.out
- HistoryServer 进程日志: `spark-*org.apache.spark.deploy.history.HistoryServer-1-*.`.out

- 访问 Spark Web 界面

通过 `http://ecnu01:8080`，可看到 Master 和 Worker，如图5.16所示。

The screenshot shows the Spark Master UI at `http://ecnu01:8080`. It displays the following information:

- Cluster Statistics:**
 - URL: `spark://ecnu01:7077`
 - Alive Workers: 2
 - Cores in use: 8 Total, 0 Used
 - Memory in use: 21.3 GB Total, 0.0 B Used
 - Applications: 0 Running, 0 Completed
 - Drivers: 0 Running, 0 Completed
 - Status: ALIVE
- Workers (2):**

Worker Id	Address	State	Cores	Memory
worker-20210202191120-219.228.148.154-44845	219.228.148.154:44845	ALIVE	4 (0 Used)	14.6 GB (0.0 B Used)
worker-20210202191120-219.228.148.67-42553	219.228.148.67:42553	ALIVE	4 (0 Used)	6.7 GB (0.0 B Used)
- Running Applications (0):**

Application ID	Name	Cores	Memory per Executor	Submitted Time	User	State	Duration
- Completed Applications (0):**

Application ID	Name	Cores	Memory per Executor	Submitted Time	User	State	Duration

图 5.16 Spark Master 界面

(5) 运行 Spark 应用程序

以下操作在客户端执行：

- 通过 Spark-Shell 运行应用程序

- 准备输入文件

```
1 ~/hadoop-2.10.1/bin/hdfs dfs -mkdir -p spark_input
2 ~/hadoop-2.10.1/bin/hdfs dfs -put ~/spark-2.4.7/RELEASE spark_input/
```

- 进入 Spark-Shell

```
1 ~/spark-2.4.7/bin/spark-shell --master spark://ecnu01:7077
```

- 在 `scala>` 后输入 Scala 代码

```
1 sc.textFile("spark_input/RELEASE").flatMap(_.split(" ")).map((_, 1)).reduceByKey(_ + _).collect
```

此处执行的是统计 `RELEASE` 文件中的单词数量，执行后应打印出如图5.17所

示结果。

```
res0: Array[(String, Int)] = Array((-Psparkr,1), (Build,1), (built,1), (-Pflume,1), ((git,1), (-Mesos,1), (-Phadoop-provided,1), (1421la1),1), (-B,1), (Spark,1), (-Pkubernetes,1), (-Pyarn,1), (revision,1), (-DzincPort=3038,1), (2.6.5,1), (flags:,1), (for,1), (-Pkafka-0-8,1), (2.4.7,1), (Hadoop,1))
```

图 5.17 示例程序运行结果

- 在 `scala>` 后输入 “`:q`” 来退出 Spark-Shell

- 通过提交 jar 包运行应用程序

- Client 提交方式（默认）

该提交方式下 Driver 运行在客户端，可以在客户端看到应用程序运行过程中的信息。

```
~/spark-2.4.7/bin/spark-submit --deploy-mode client --master
spark://ecnu01:7077 --class org.apache.spark.examples.SparkPi
~/spark-2.4.7/examples/jars/spark-examples_2.11-2.4.7.jar
```

在运行过程中另起一个终端执行 `jps` 查看进程。此时各 Worker 节点上会存在一个 `CoarseGrainedExecutorBackend` 进程，负责创建及维护 Executor 对象。如图5.18和图5.19所示。

```
dase-dis@ecnu02:~$ jps
9618 DataNode
11430 Jps
11399 CoarseGrainedExecutorBackend
29933 Worker
```

图 5.18 从节点执行 `jps` 的结果

```
dase-dis@ecnu04:~$ jps
19095 Jps
19004 SparkSubmit
```

图 5.19 客户端执行 `jps` 的结果

运行结果如图5.20所示，可以看到图片中第一行为 Pi 的近似计算结果。

- Cluster 提交方式

该提交方式下 Master 会随机选取一个 Worker 节点启动 Driver，故在客户端看不到应用程序运行过程中的信息。

```
~/spark-2.4.7/bin/spark-submit --deploy-mode cluster --master
spark://ecnu01:7077 --class org.apache.spark.examples.SparkPi
```

```
Pi is roughly 3.140875704378522
21/02/02 19:44:46 INFO server.AbstractConnector: Stopped Spark@588ab592{HTTP/1.1}
,[http/1.1]}{0.0.0.0:4040}
21/02/02 19:44:46 INFO ui.SparkUI: Stopped Spark web UI at http://ecnu04:4040
21/02/02 19:44:46 INFO cluster.StandaloneSchedulerBackend: Shutting down all executors
```

图 5.20 示例程序运行结果

```
~/spark-2.4.7/examples/jars/spark-examples_2.11-2.4.7.jar
```

在运行过程中另起一个终端执行 jps 查看进程。如图5.21和图5.22所示，在 Cluster 提交方式下，可以在其中一个 Worker 节点中看到 DriverWrapper 进程。

```
dase-dis@ecnu02:~$ jps
9618 DataNode
31956 Jps
31752 DriverWrapper
31900 CoarseGrainedExecutorBackend
29933 Worker
```

图 5.21 从节点执行 jps 的结果

```
dase-dis@ecnu03:~$ jps
463157 Worker
461087 DataNode
469389 CoarseGrainedExecutorBackend
469416 Jps
```

图 5.22 从节点执行 jps 的结果

(6) 查看 Spark 程序运行信息

- 实时查看应用运行情况

在应用运行过程中，访问 Driver 所在节点的 4040 端口。如果是在客户端通过 Client 提交方式提交的应用程序，则访问 <http://ecnu04:4040>；如果是在客户端通过 Cluster 提交方式提交的应用程序，则需要先找到 DriverWrapper 进程所在的节点，然后访问 DriverWrapper 进程所在的节点的 4040 端口。此外，如果是在客户端通过 Shell 提交的应用程序，则访问 <http://ecnu04:4040>，因为 Shell 使用的是 Client 提交方式。Web 界面如图5.23所示。

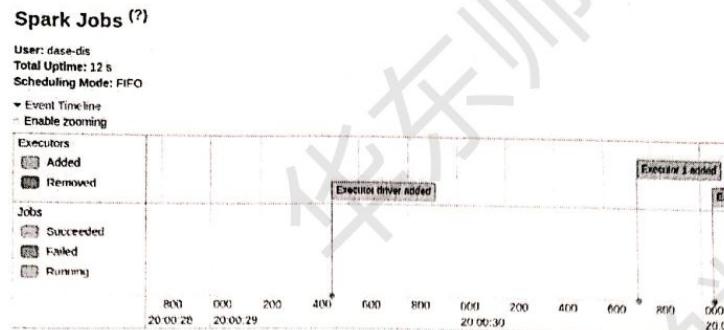


图 5.23 Spark Jobs 界面

- 查看 Spark 应用程序日志

在提交一个应用程序后，在`~/spark-2.4.7/work`下会出现应用程序运行日志文件夹。

访问`http://ecnu01:8080`，点击Completed Applications下的对应应用程序的Application ID，进入到如图5.24所示界面，点击stdout或者stderr可以查看对应日志信息。

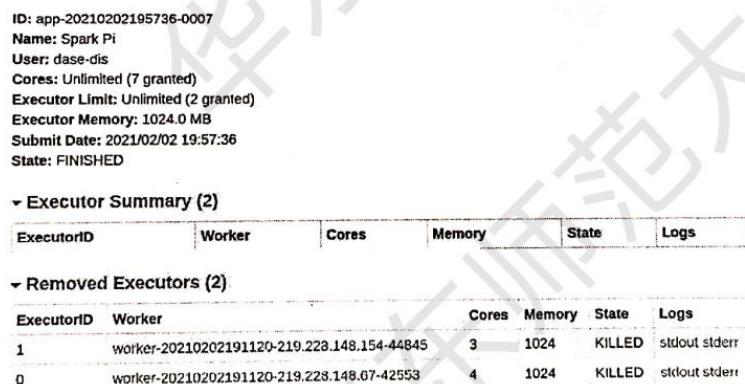


图 5.24 Spark 某个 Application 的界面

Event log directory: hdfs://ecnu01:9000/tmp/spark_history							
Last updated: 2021-02-02 20:09:35							
Client local time zone: Asia/Shanghai							
Search: <input type="text"/>							
App ID	App Name	Started	Completed	Duration	Spark User	Last Updated	Event Log
app-20210202195736-0002	Spark Pi	2021-02-02 19:57:35	2021-02-02 19:57:38	3 s	dase-dis	2021-02-02 19:57:38	Download
app-20210202194348-0001	Spark Pi	2021-02-02 19:43:47	2021-02-02 19:43:50	3 s	dase-dis	2021-02-02 19:43:50	Download
app-20210202192827-0000	Spark shell	2021-02-02 19:28:26	2021-02-02 19:43:11	15 min	dase-dis	2021-02-02 19:43:11	Download

Showing 1 to 8 of 8 entries
Show incomplete applications

图 5.25 Spark History 界面

- 查看应用历史记录

在应用运行结束后，访问 <http://ecnu01:18080>。Web 界面如图5.25所示。

(7) 停止服务

- 停止 Spark（在主节点上执行）

```
1 ./spark-2.4.7/sbin/stop-all.sh      # 停止 Spark  
2 ./spark-2.4.7/sbin/stop-history-server.sh # 停止日志服务器
```

- 停止 HDFS 服务（在主节点上执行）

```
1 ./hadoop-2.10.1/sbin/stop-dfs.sh
```

5.5 思考题

- 1 对于分布式部署的 Spark 集群，如果在集群启动过程中某个从节点上的 Worker 进程没有启动成功，那么应该在哪个节点上查看哪个路径下的什么日志？
- 2 既然采用 Cluster 提交方式在客户端看不到程序运行过程中的信息，那么这些运行过程中的信息如何查看？（提示：结合 HistoryServer 的 Web UI 进行阐述）