

实验五 Spark 部署与编程 (分布式)

5.1 实验目的

- 学习 Spark 的分布式部署，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
- 掌握在分布式部署方式下运行 Spark 相关程序的方法
- 通过基于 Yarn 部署 Spark，深入理解 Yarn 的作用，体会“一个平台、多个框架”

5.2 实验任务

- 完成 Spark 的分布式部署和基于 Yarn 的分布式部署
- 在分布式部署和基于 Yarn 的分布式部署方式下，分别以 Client 和 Cluster 提交方式来运行示例程序
- 在分布式部署方式下运行 WordCount 示例程序

5.3 实验环境

- 操作系统: Ubuntu 18.04 或 Ubuntu 20.04
- JDK 版本: 1.8
- Spark 版本: 2.4.7
- Hadoop 版本: 2.10.1

5.4 实验步骤

注意: 本教程以名为 `dase-local` 或 `dase-dis` 的用户为例开展实验，读者在实验过程中需替换为自己的用户名（云主机上的默认用户名为 `ubuntu`）。

5.4.1 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，需要重新在 4 台机器之间实现免密钥登录
- 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
```

(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
   ~/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

```
1 ~/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5.1所示），在从节点节点上存在 Worker 进程（如图5.2所示）。


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

图 5.1 主节点存在的进程

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

图 5.2 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 5.3 所示。

(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

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 5.3 Spark Master 界面

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

– 在 `scala>` 后输入 Scala 代码

```
sc.textFile("spark_input/RELEASE").flatMap(_.split(" ")).map(_ ,  
1) 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 5.4 所示结果。

```
res0: Array[(String, Int)] = Array((-Psparkr,1), (Build,1), (built,1), (-Pflume,  
1), (git,1), (-Pmesos,1), (-Phadoop-provided,1), (14211a1,1), (-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 5.4 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 5.5 和图 5.6 所示。

运行结果如图 5.7 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


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

图 5.5 从节点执行 jps 的结果

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

图 5.6 客户端执行 jps 的结果

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
~/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 和图 5.9 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。

(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 5.10 所示。

```
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 5.7 示例程序运行结果

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

Spark Jobs (?)

User: dase-dis
Total Uptime: 12 s
Scheduling Mode: FIFO

▼ Event Timeline
Enable zooming

Executions

- Added
- Removed

Jobs

- Succeeded
- Failed
- Running

The image shows a Spark Event Timeline for a job. The timeline is divided into two sections: 'Executions' and 'Jobs'. The 'Executions' section shows 'Executor driver added' at 20:00:29.400 and 'Executor 1 added' at 20:00:30.600. The 'Jobs' section shows a job that is 'Succeeded' at 20:00:30.600. The timeline is zoomed in on the period from 20:00:29 to 20:00:30.

在客户端节点 (ecnu04) 执行以下命令:

ID: app-20210202195736-0007
 Name: Spark Pi
 User: dase-dis
 Cores: Unlimited (7 granted)
 Executor Limit: Unlimited (2 granted)
 Executor Memory: 1024.0 MB
 Submit Date: 2021/02/02 19:57:36
 State: FINISHED

▼ Executor Summary (2)

ExecutorID	Worker	Cores	Memory	State	Logs
1	worker-20210202191120-219.228.148.154-44845	3	1024	KILLED	stdout stderr
0	worker-20210202191120-219.228.148.67-42553	4	1024	KILLED	stdout stderr

▼ Removed Executors (2)

ExecutorID	Worker	Cores	Memory	State	Logs
1	worker-20210202191120-219.228.148.154-44845	3	1024	KILLED	stdout stderr
0	worker-20210202191120-219.228.148.67-42553	4	1024	KILLED	stdout stderr

图 5.11 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:

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 5.12 Spark History 界面

```

1 ~/hadoop-2.10.1/bin/hdfs dfs -cp ./input/pd.train ./spark_input
2 #pd.train数据文件之前已经上传至hdfs:///user/dase-dis/input
3 #spark_input目录之前也已创建完成

```

- 通过提交 jar 包运行应用程序
在客户端节点 (ecnu04) 执行以下命令：

```

1 ~/hadoop-2.10.1/bin/hdfs dfs -rm -r spark_output
2 ~/spark-2.4.7/bin/spark-submit \
3 --master spark://ecnu01:7077 \
4 --class cn.edu.ecnu.spark.example.java.wordcount.WordCount \
5 /home/dase-dis/spark-2.4.7/myApp/RddWordCountJava.jar
   hdfs://ecnu01:9000/user/dase-dis/spark_input
   hdfs://ecnu01:9000/user/dase-dis/spark_output

```

分布式环境下程序运行产生的文件如图5.13所示。

(8) 停止服务

- 停止 Spark (在主节点上执行)

Browse Directory

Permission	Owner	Group	Size	Replication	Block Size	Name
drwxr-xr-x	dase-dis	supergroup	0 B	3	128 MB	_SUCCESS
drwxr-xr-x	dase-dis	supergroup	130.85 MB	2	128 MB	part-00000
drwxr-xr-x	dase-dis	supergroup	130.67 MB	2	128 MB	part-00001
drwxr-xr-x	dase-dis	supergroup	130.5 MB	2	128 MB	part-00002
drwxr-xr-x	dase-dis	supergroup	130.67 MB	2	128 MB	part-00003
drwxr-xr-x	dase-dis	supergroup	130.59 MB	2	128 MB	part-00004
drwxr-xr-x	dase-dis	supergroup	130.54 MB	2	128 MB	part-00005
drwxr-xr-x	dase-dis	supergroup	130.23 MB	2	128 MB	part-00006
drwxr-xr-x	dase-dis	supergroup	130.73 MB	2	128 MB	part-00007
drwxr-xr-x	dase-dis	supergroup	130.77 MB	2	128 MB	part-00008
drwxr-xr-x	dase-dis	supergroup	130.67 MB	2	128 MB	part-00009
drwxr-xr-x	dase-dis	supergroup	130.57 MB	2	128 MB	part-00010
drwxr-xr-x	dase-dis	supergroup	130.52 MB	2	128 MB	part-00011
drwxr-xr-x	dase-dis	supergroup	130.86 MB	2	128 MB	part-00012
drwxr-xr-x	dase-dis	supergroup	130.6 MB	2	128 MB	part-00013
drwxr-xr-x	dase-dis	supergroup	130.51 MB	2	128 MB	part-00014
drwxr-xr-x	dase-dis	supergroup	130.6 MB	2	128 MB	part-00015
drwxr-xr-x	dase-dis	supergroup	130.09 MB	2	128 MB	part-00016
drwxr-xr-x	dase-dis	supergroup	130.59 MB	2	128 MB	part-00017

图 5.13 程序运行产生的文件

```
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
```

5.4.2 基于 Yarn 的分布式部署

(1) 修改配置

在主节点进行以下操作

- 修改 spark-env.sh 文件（文件路径：~/spark-2.4.7/conf/），使 Spark 能够读取 Yarn 配置

在文件末尾添加：

```
1 export HADOOP_CONF_DIR=/home/dase-dis/hadoop-2.10.1/etc/hadoop
2 # Hadoop 配置目录
```

- 修改 yarn-site.xml 文件（文件路径：~/hadoop-2.10.1/etc/hadoop/）

在 Hadoop 配置文件 yarn-site.xml 中的 <configuration> 标签块中添加如下代码

```
1 <property>
2   <name>yarn.nodemanager.pmem-check-enabled</name>
3   <value>false</value>
4 </property>
```

```

5 <property>
6   <name>yarn.nodemanager.vmem-check-enabled</name>
7   <value>false</value>
8 </property>
9 <property>
10   <name>yarn.log.server.url</name>
11   <value>http://ecnu01:19888/jobhistory/logs</value>
12 </property>

```

- 将修改后的文件同步到其他机器

```

1 scp ~/spark-2.4.7/conf/spark-env.sh
   dase-dis@ecnu02:~/spark-2.4.7/conf/spark-env.sh
2 scp ~/spark-2.4.7/conf/spark-env.sh
   dase-dis@ecnu03:~/spark-2.4.7/conf/spark-env.sh
3 scp ~/spark-2.4.7/conf/spark-env.sh
   dase-dis@ecnu04:~/spark-2.4.7/conf/spark-env.sh
4 scp ~/hadoop-2.10.1/etc/hadoop/yarn-site.xml
   dase-dis@ecnu02:~/hadoop-2.10.1/etc/hadoop/yarn-site.xml
5 scp ~/hadoop-2.10.1/etc/hadoop/yarn-site.xml
   dase-dis@ecnu03:~/hadoop-2.10.1/etc/hadoop/yarn-site.xml
6 scp ~/hadoop-2.10.1/etc/hadoop/yarn-site.xml
   dase-dis@ecnu04:~/hadoop-2.10.1/etc/hadoop/yarn-site.xml

```

(2) 启动服务

```

1 ~/hadoop-2.10.1/sbin/start-yarn.sh           # 启动 Yarn
2 ~/hadoop-2.10.1/sbin/mr-jobhistory-daemon.sh start historyserver # 启动 Yarn
   历史服务器
3 ~/hadoop-2.10.1/sbin/start-dfs.sh           # 启动 HDFS
4 ~/spark-2.4.7/sbin/start-history-server.sh  # 启动 Spark 应用日志服务器

```

(3) 查看 Yarn 服务信息

- 使用 jps 查看进程，验证是否成功启动服务

正常启动后主节点上存在的进程应如图5.14所示，从节点节点上存在的进程应如图5.15所示)。

- 查看 ResourceManager、NodeManager、JobHistoryServer 进程日志

本实验 Yarn 相关的服务日志记录在 `~/hadoop-2.10.1/logs/`，前缀为 `yarn`，后缀为 `.out` 的文件中。Spark 的 JobHistoryServer 服务日志记录在 `~/spark-2.4.7/logs/` 路径下。

- ResourceManager 进程日志：

```
dase-dis@ecnu01:~$ jps
14132 Jps
10698 JobHistoryServer
13357 SecondaryNameNode
10238 ResourceManager
14047 HistoryServer
12975 NameNode
```

图 5.14 主节点存在的进程

```
dase-dis@ecnu02:~$ jps
4898 NodeManager
11464 Jps
9151 DataNode
```

图 5.15 从节点存在的进程

默认位置: `~/hadoop-2.10.1/logs/yarn-*-resourcemanager-*.log`

– NodeManager 进程日志:

默认位置: `~/hadoop-2.10.1/logs/yarn-*-nodemanager-*.log`

– JobHistoryServer 进程日志:

默认位置: `/spark-2.4.7/logs/spark-*-org.apache.spark.deploy.history.HistoryServer-1-*.out`

• 访问 Yarn Web 界面

通过 `http://ecnu01:8088`, 如图5.16所示。



图 5.16 Yarn 界面

(4) 运行 Spark 应用程序

• 通过 Spark-Shell 运行应用程序

– 准备输入文件(之前操作过可以跳过)


```
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 yarn
```

– 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), (-Pmesos,1), (-Phadoop-provided,1), (14211a1,1), (-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。

```
1 ~/spark-2.4.7/bin/spark-submit \
2 --deploy-mode client \
3 --master yarn \
4 --class org.apache.spark.examples.SparkPi \
5 ~/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 5.18 所示。此时会存在一个 `ExecutorLauncher` 进程, 以及若干个 `CoarseGrainedExecutorBackend` 进程。

运行结果如图 5.19 所示, 可以看到图片中输出的 π 的近似值。

– Cluster 提交方式

此方式下 `ResourceManager` 会随机选取一个 `NodeManager` 所在节点在其上启动 `ApplicationMaster`, `Driver` 运行在 `ApplicationMaster` 中, 故在客户端看不到应用程序运行过程中的信息 (当然, `Yarn` 的信息是可以看到的)。

```
dase-dis@ecnu03:~$ jps
671998 CoarseGrainedExecutorBackend
672013 Jps
662931 NodeManager
662739 DataNode
671857 ExecutorLauncher
```

图 5.18 Client 提交方式从节点存在的进程

```
21/02/03 18:54:54 INFO scheduler.DAGScheduler: Job 0 finished: reduce at SparkPi
.scala:38, took 0.667958 s
Pi is roughly 3.1348956744783725
21/02/03 18:54:54 INFO server.AbstractConnector: Stopped Spark@33c2bd{HTTP/1.1,[
http/1.1]}{0.0.0.0:4040}
```

图 5.19 Client 提交方式示例程序运行结果

```
1 ~/spark-2.4.7/bin/spark-submit \
2 --deploy-mode cluster \
3 --master yarn \
4 --class org.apache.spark.examples.SparkPi \
5 ~/spark-2.4.7/examples/jars/spark-examples_2.11-2.4.7.jar
```

运行过程中分别在主节点、从节点与客户端另起一个终端执行 `jps` 查看进程。启动 `ApplicationMaster` 的从节点存在的进程如图 5.20 所示，另外的从节点存在的进程如图 5.21 所示。在 `Cluster` 提交方式下，此时总共会存在一个 `ApplicationMaster` 进程，以及若干个 `CoarseGrainedExecutorBackend` 进程。

```
dase-dis@ecnu02:~$ jps
7714 DataNode
8322 NodeManager
23941 Jps
23705 ApplicationMaster
23902 CoarseGrainedExecutorBackend
```

图 5.20 启动 `ApplicationMaster` 的从节点存在的进程

(5) 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 `http://ecnu01:8088`，点击对应名称的应用记录的 `Tracking UI` 列中的 `ApplicationMaster`，跳转至 Spark Web 界面，可以查看应用程序的运行情况。Web 界面如图 5.22 所示。

- 查看 Spark 应用程序日志

```
dase-dis@ecnu03:~$ jps
666986 Jps
666943 CoarseGrainedExecutorBackend
662931 NodeManager
662739 DataNode
```

图 5.21 未启动 ApplicationMaster 的从节点存在的进程

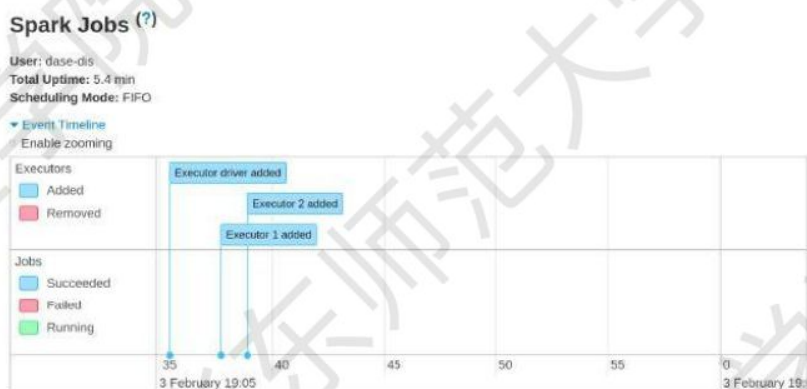


图 5.22 Spark Jobs 界面

在提交一个应用程序后，在 `~/hadoop-2.10.1/logs/userlogs` 下会出现应用程序运行日志文件夹。

访问 `http://ecnu01:8088`，点击某一个应用程序的 ID，进入到如图 5.23 所示界面，点击 logs 后即可查看对应的 stderr 或者 stdout 日志信息。

- 查看应用历史记录

访问 `http://ecnu01:18080`，可以看到本次启动 Yarn 后提交的所有应用程序的相关信息。如图 5.24 所示

在应用运行结束后，访问 `http://ecnu01:18080` 可以查看应用历史记录。如图 5.25 所示。

(6) 停止服务

- 停止命令（在 Yarn 主节点执行）

```
1 ~/spark-2.4.7/sbin/stop-history-server.sh
2 ~/hadoop-2.10.1/sbin/stop-yarn.sh
3 ~/hadoop-2.10.1/sbin/mr-jobhistory-daemon.sh stop historyserver
4 ~/hadoop-2.10.1/sbin/stop-dfs.sh
```




图 5.23 Hadoop 监控下某个 Application 的界面



图 5.24 Hadoop Yarn History 界面

Event log directory: hdfs://ecnu01:9000/tmp/spark_history

Last updated: 2021-02-03 19:17:08

Client local time zone: Asia/Shanghai

Search:

App ID	App Name	Started	Completed	Duration	Spark User	Last Updated	Event Log
application_1612349383308_0003	Spark Pi	2021-02-03 18:59:39	2021-02-03 18:59:47	8 s	dase-dis	2021-02-03 18:59:48	Download
application_1612349383308_0002	Spark Pi	2021-02-03 18:54:26	2021-02-03 18:54:54	28 s	dase-dis	2021-02-03 18:54:55	Download
application_1612349383308_0001	Spark shell	2021-02-03 18:53:29	2021-02-03 18:54:17	48 s	dase-dis	2021-02-03 18:54:17	Download

Showing 1 to 4 of 4 entries

[Show incomplete applications](#)

图 5.25 Spark History 界面

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 在基于 Yarn 的分布式部署中，为什么会在 Hadoop 的 Web UI 中看到关于 Spark 应用程序的记录？
- 3 对于 Client 提交方式，为什么在 Spark on Yarn 部署时会出现 ExecutorLauncher 进程，而在非 on Yarn 的 Spark 部署时不会出现 ExecutorLauncher 进程？
- 4 上述 Spark 程序中仅包含一个 Spark 应用，然而一个 Spark 程序中允许包含多个 Spark 应用。附录A是一个包含 2 个应用的 Spark 程序，请将该 Spark 程序提交到 Yarn 上运行，并结合 Yarn 的 Web UI 的主界面，观察在 Yarn 上启动了多少个 Spark 集群（一个 Spark 集群包含一个 Driver 和若干个 CoarseGrainedExecutorBackend），以及两个 Spark 应用是否是并行执行的？

附录

A 包含两个应用的 Spark 示例程序

```
1 import org.apache.spark.SparkConf;
2 import org.apache.spark.api.java.JavaPairRDD;
3 import org.apache.spark.api.java.JavaRDD;
4 import org.apache.spark.api.java.JavaSparkContext;
5 import org.apache.spark.api.java.function.*;
6 import scala.Tuple2;
7
8 import java.util.Arrays;
9 import java.util.Iterator;
10
11 public class DemoWith2SparkContext {
12
13     public static void run(String[] args) {
14         /* 步骤1: 通过SparkConf设置配置信息, 并创建SparkContext */
15         SparkConf conf = new SparkConf();
16         conf.setAppName("DemoWith2SparkContext");
17         conf.setMaster("local"); // 仅用于本地进行调试, 如在集群中运行则删除本行
18         JavaSparkContext sc = new JavaSparkContext(conf);
19
20         /* 步骤2: 按应用逻辑使用操作算子编写DAG, 其中包括RDD的创建、转换和行动等 */
21         // 读入文本数据, 创建名为lines的RDD
22         JavaRDD<String> lines = sc.textFile(args[0]);
23
24         // 将lines中的每一个文本行按空格分割成单个单词
25         JavaRDD<String> words =
26             lines.flatMap(
27                 new FlatMapFunction<String, String>() {
28                     @Override
29                     public Iterator<String> call(String line) throws Exception {
30                         return Arrays.asList(line.split(" ")).iterator();
31                     }
32                 });
33         // 将每个单词的频数设置为1, 即将每个单词映射为[单词, 1]
34         JavaPairRDD<String, Integer> pairs =
35             words.mapToPair(
36                 new PairFunction<String, String, Integer>() {
```



```
37         @Override
38         public Tuple2<String, Integer> call(String word) throws Exception {
39             return new Tuple2<String, Integer>(word, 1);
40         }
41     });
42     // 按单词聚合，并对相同单词的频数使用sum进行累计
43     JavaPairRDD<String, Integer> wordCounts =
44         pairs.groupByKey()
45             .mapToPair(
46                 new PairFunction<Tuple2<String, Iterable<Integer>>, String, Integer>() {
47                     @Override
48                     public Tuple2<String, Integer> call(Tuple2<String, Iterable<Integer>> t)
49                         throws Exception {
50                         Integer sum = Integer.valueOf(0);
51                         for (Integer i : t._2) {
52                             sum += i;
53                         }
54                         return new Tuple2<String, Integer>(t._1, sum);
55                     }
56                 });
57     // 合并机制
58     /*JavaPairRDD<String, Integer> wordCounts =
59     pairs.reduceByKey(
60         new Function2<Integer, Integer, Integer>() {
61             @Override
62             public Integer call(Integer t1, Integer t2) throws Exception {
63                 return t1 + t2;
64             }
65         });*/
66
67     // 输出词频统计结果
68     wordCounts.foreach(t -> System.out.println(t._1 + " " + t._2));
69
70     /* 步骤3: 关闭SparkContext */
71     sc.stop();
72 }
73
74 public static void main(String[] args) {
75     run(args);
76     run(args);
77 }
78 }
```