

实验七 基于 Yarn 部署 Spark

7.1 实验目的

- 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”

7.2 实验任务

- 完成 Spark 2.4.7 on Yarn 的单机伪分布式部署以及分布式部署
- 在两种部署方式下以不同的提交方式运行词频统计及 Pi 近似值计算两个示例程序

7.3 实验环境

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

7.4 实验步骤

7.4.1 单机伪分布式部署

(1) 准备工作

- 登录用户 dase-local

(2) 修改配置

- 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

在末尾添加：

```
1 export HADOOP_CONF_DIR=/home/dase-local/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>` 标签块中添加如下配置：

```
1 <!-- 是否进行物理内存限制比较 -->  
2 <property>
```

```

3   <name>yarn.nodemanager.pmem-check-enabled</name>
4   <value>false</value>
5 </property>
6 <!-- 是否进行虚拟内存限制比较 -->
7 <property>
8   <name>yarn.nodemanager.vmem-check-enabled</name>
9   <value>false</value>
10 </property>

```

(3) 启动服务

- 启动命令

```

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

```

(4) 查看 Yarn 服务信息

- 使用 jps 查看进程，验证是否成功启动服务

对比验证正常出现的进程，如图7.1所示。

```

2438 NameNode
29030 ResourceManager
29208 NodeManager
5593 Jps
3577 JobHistoryServer
2972 SecondaryNameNode
2686 DataNode
5439 HistoryServer

```

图 7.1 启动服务后存在的进程

- 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：
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.HistorySe`
`rver-1-*.`
`out`
- 访问 Yarn Web 界面
通过 `http://localhost:8088`, 如图7.2所示。

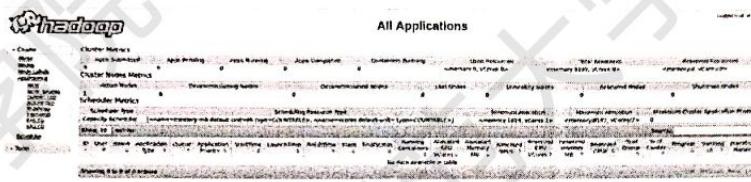


图 7.2 Yarn 界面

(5) 运行 Spark 应用程序

- 通过 Spark-Shell 运行应用程序
 - 准备输入文件(若已经准备过输入文件，则可以跳过)

```
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))
2 .reduceByKey(_ + _).collect
```

此处执行的是统计 RELEASE 文件中的单词数量，执行结果如图7.3所示。

```
res0: Array[(String, Int)] = Array((-Psparkr,1), (Build,1), (built,1), (-Pfume, 1), ((git,1), (-Mesos,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 7.3 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7.4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7.5所示，可以看到图片中输出的 Pi 的近似值。

```

16384 CoarseGrainedExecutorBackend
2438 NameNode
29030 ResourceManager
16264 ExecutorLauncher
29208 NodeManager
16521 Jps
3577 JobHistoryServer
2972 SecondaryNameNode
15965 SparkSubmit
2686 DataNode
5439 HistoryServer
16447 CoarseGrainedExecutorBackend

```

图 7.4 Client 提交方式运行过程中存在的进程

```

21/02/03 15:40:00 INFO scheduler.DAGScheduler: Job 0 finished: reduce at SparkPi
.scala:38, took 0.777026 s
Pi is roughly 3.13375566877834
21/02/03 15:40:00 INFO server.AbstractConnector: Stopped Spark@2c22a348{HTTP/1.1
,[http/1.1]}{0.0.0.0:4040}

```

图 7.5 Client 提交方式下示例程序运行结果

- Cluster 提交方式

因为单机伪分布式部署方式下只有 1 个 NodeManager，所以 ResourceManager 直接选择当前节点上的 NodeManager，由其启动 1 个 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）。

```

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

```

在运行过程中另起一个终端执行 jps 查看进程，如图7.6所示。在 Cluster 提交方式下，此时会存在一个 ApplicationMaster 进程，以及若干个 CoarseGrainedExecutorBackend 进程。

```

18885 CoarseGrainedExecutorBackend
2438 NameNode
29030 ResourceManager
18712 ApplicationMaster
29208 NodeManager
19001 Jps
3577 JobHistoryServer
18443 SparkSubmit
18971 CoarseGrainedExecutorBackend
2972 SecondaryNameNode
2686 DataNode
5439 HistoryServer

```

图 7.6 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，访问 <http://localhost: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7.7所示。

- 查看 Spark 应用程序日志

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://localhost:8088>，点击某一个应用程序的 ID，进入到如图7.8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://localhost:8088>，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7.9所示。

在应用运行结束后，访问 <http://localhost:18080> 可以查看应用历史记录。如图7.10所示。

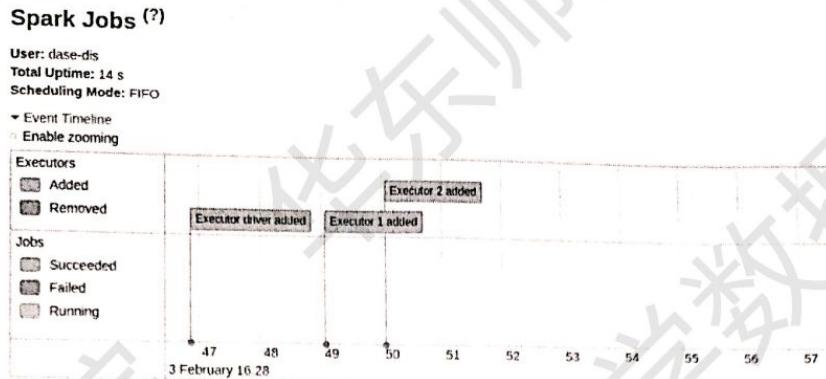


图 7.7 Spark Jobs 界面



图 7.8 Hadoop 监控下某个 Application 的界面

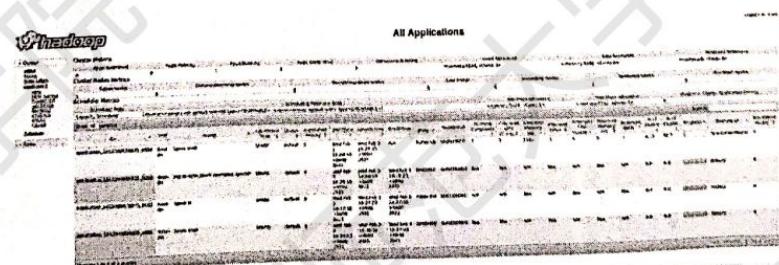


图 7.9 Hadoop Yarn 界面

| Event log directory: hdfs://ecnu01:9000/tmp/spark_history | | | | | | | |
|---|--|------------------------|------------------------|------|----------|------------------------|--------------------------|
| Last updated: 2021-02-03 16:41:53 | | | | | | | |
| Client local time zone: Asia/Shanghai | | | | | | | |
| Search: | | | | | | | |
| App ID | | | | | | | |
| App Name Started Completed Duration Spark User Last Updated Event Log | | | | | | | |
| application_1612340676975_0003 | | | | | | | |
| Spark PI | | 2021-02-03 16:28:16 | 2021-02-03 16:28:22 | 7 s | dase-dis | 2021-02-03 16:28:23 | Download |
| application_1612340676975_0002 | | | | | | | |
| Spark PI | | 2021-02-03 16:27:46 | 2021-02-03 16:27:59 | 13 s | dase-dis | 2021-02-03 16:27:59 | Download |
| application_1612340676975_0001 | | | | | | | |
| Spark shell | | 2021-02-03 16:26:51 | 2021-02-03 16:27:43 | 52 s | dase-dis | 2021-02-03 16:27:43 | Download |

Showing 1 to 3 of 3 entries
Show incomplete applications

图 7.10 Spark History 界面

(7) 停止服务

- 停止命令

```

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

```

7.4.2 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

(2) 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

```

(3) 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

```

(4) 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7.11所示，从节点节点上存在的进程应如图7.12所示)。

- 查看 ResourceManager、NodeManager、JobHistoryServer 进程日志

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

图 7.11 主节点存在的进程

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

图 7.12 从节点存在的进程

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：

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 7.13 所示。

(5) 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/
```

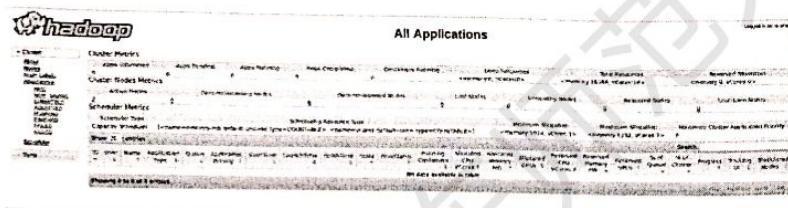


图 7.13 Yarn 界面

- 进入 Spark-Shell

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

- 在 `scala>` 后输入 Scala 代码.

```
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7.14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 7.14 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7.15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7.16所示, 可以看到图片中输出的 Pi 的近似值。

- Cluster 提交方式

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

图 7.15 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 7.16 Client 提交方式示例程序运行结果

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）。

```
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7.17所示，另外的从节点存在的进程如图7.18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 7.17 启动 ApplicationMaster 的从节点存在的进程

(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，访问 <http://ecnu01:8088>，点击对应名称的应用记录的

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

图 7.18 未启动 ApplicationMaster 的从节点存在的进程

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 7.19 所示。

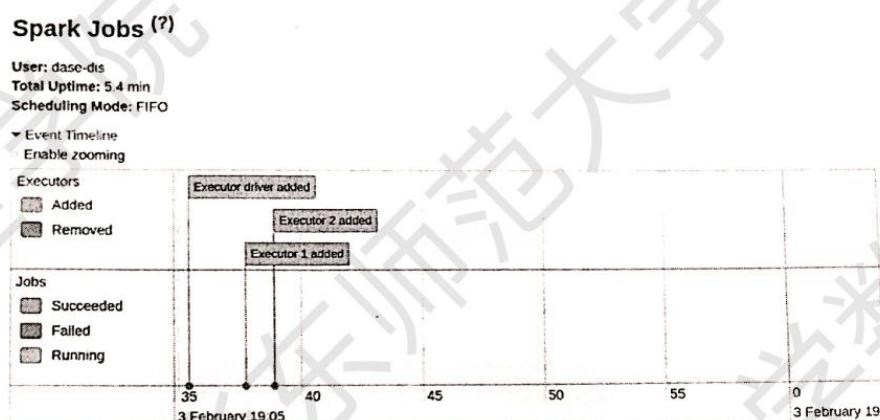


图 7.19 Spark Jobs 界面

- 查看 Spark 应用程序日志

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 7.20 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 7.21 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 7.22 所示。

(7) 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
```

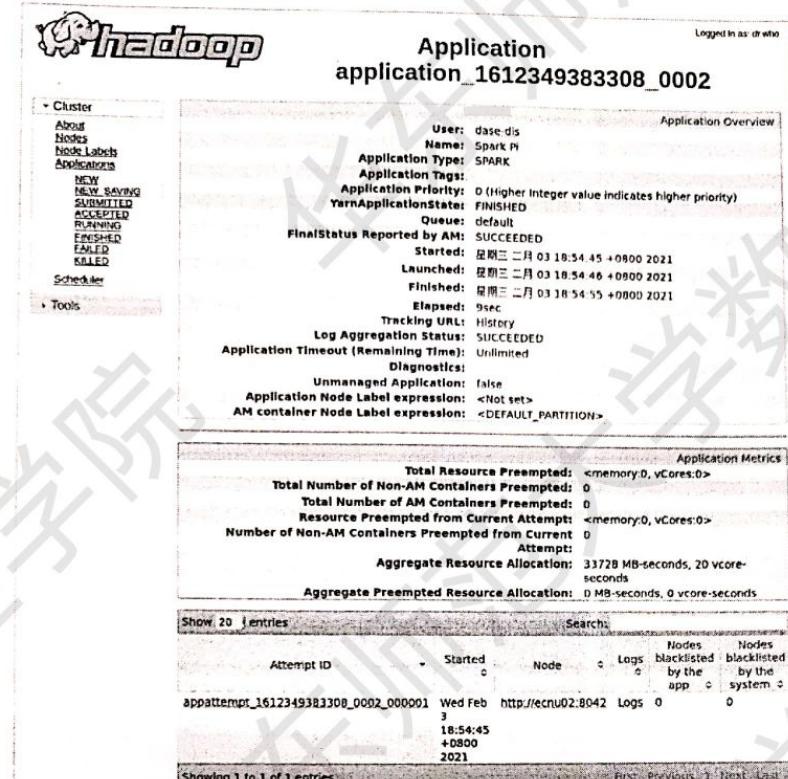


图 7.20 Hadoop 监控下某个 Application 的界面

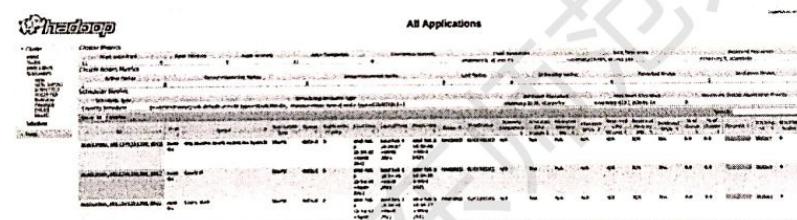


图 7.21 Hadoop Yarn History 界面

| 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 7.22 Spark History 界面

7.5 思考题

- 1 为什么会在 Hadoop 的 Web UI 中看到关于 Spark 应用程序的记录？
- 2 对于 Client 提交方式，为什么在 Spark on Yarn 部署时会出现 ExecutorLauncher 进程，而在实验五中非 on Yarn 的 Spark 部署时不会出现 ExecutorLauncher 进程？
- 3 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 }
```