华东师范大学数据学院上机实践报告

课程名称：分布式模型与编程	年级：2018	上机实践成绩:
指导教师：徐辰		姓名：孙秋实
上机实践名称：基于 Yarn 部署 Spark	学号：10185501402	上机实践日期：2021/4/29
上机实践编号：Lab9	组号：Group5	上机实践时间：

Part 1
实验目的

(1) 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”。

Part 2
实验任务

(1) 完成 Spark 2.4.7 on Yarn 的单机伪分布式部署以及分布式部署。并且两种部署方式下以不同的提交模式均能成功运行统计单词量以及 π 的近似值计算的示例程序

(2) 实验补充：在 Yarn 上同时运行 Spark 任务和 MapReduce 任务

Part 3
使用环境

(1) 操作系统：Ubuntu 18.04

(2) JDK 版本：1.8

(3) Hadoop 版本：2.10.1

(4) Spark 版本：2.4.7

Part 4
实验过程

Section 1
单机伪分布式部署

准备工作
登录用户 dase-local，修改 *spark-env.sh* 文件，使 Spark 能读取 Yarn 配置
在文件末尾添加下列信息，如图所示

```
HADOOP_HOME=/home/dase-local/softwares/hadoop-2.10.1
export SPARK_DIST_CLASSPATH=$($HADOOP_HOME/bin/hadoop classpath)
export LD_LIBRARY_PATH=$HADOOP_HOME/lib/native:$LD_LIBRARY_PATH

export SPARK_MASTER_HOST=localhost
export SPARK_MASTER_PORT=7077

export HADOOP_CONF_DIR=/home/dase-local/softwares/hadoop-2.10.1/etc/hadoop
-- INSERT --                                                79,57         Bot
```

图 1: 修改 *spark-env.sh* 文件

随后，修改 Hadoop 配置文件 *yran-site.xml* 文件，在 $< configuration >$ 标签块中添加如下代码（这里第一个代码块结尾的 name 出现了 typo 错误，后面修正了）

```
        <property>
                <name>yarn.node.manager.pmem-check-enabled</nmae>
                <value>false</value>
        </property>
        <property>
                <name>yarn.node.manager.vmem-check-enabled</name>
                <value>false</value>
        </property>
-- Site specific YARN configuration properties -->
```

图 2: 修改 *yran-site.xml* 文件

启动 Spark on Yarn

```
dase-local@10-24-21-18:~/spark-2.4.7$ cd sbin
dase-local@10-24-21-18:~/spark-2.4.7/sbin$ start-history-server.sh
starting org.apache.spark.deploy.history.HistoryServer, logging to /home/dase-lo
cal/spark-2.4.7/logs/spark-dase-local-org.apache.spark.deploy.history.HistorySer
ver-1-10-24-21-18.out
dase-local@10-24-21-18:~/spark-2.4.7/sbin$ jps
7472 HistoryServer
5760 NodeManager
7508 Jps
6341 NameNode
5574 ResourceManager
6119 JobHistoryServer
6552 DataNode
7293 SecondaryNameNode
dase-local@10-24-21-18:~/spark-2.4.7/sbin$
```

图 3: 启动 Spark on Yarn

随后如上图所示，使用 jps 命令检查各进程是否都正常启动

访问 Yarn Web

如果上述操作都正常执行，我们就可以通过 Web UI 来查看 Yarn 状态了

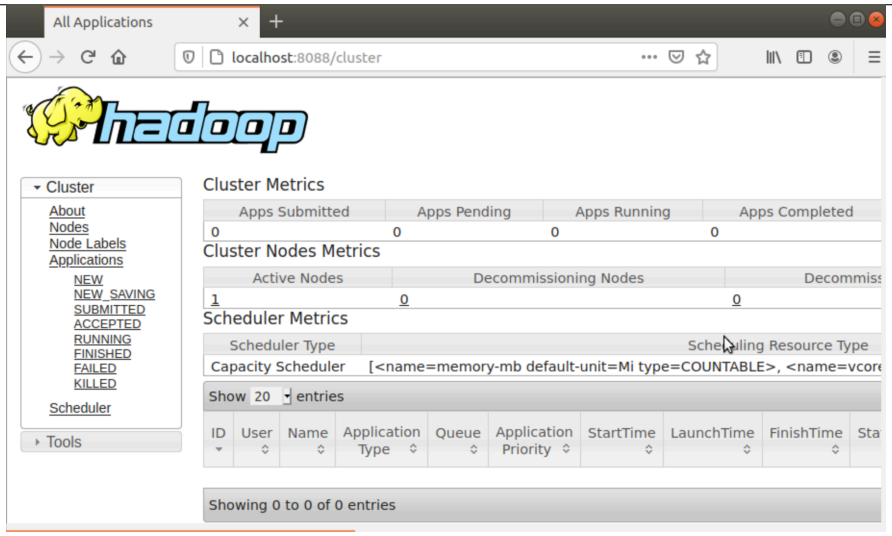

图 4: 访问 Yarn Web

可以看到当前有 Active Nodes

运行 Spark 程序

我们先通过 Spark Shell 来运行应用程序

下一步是先准备输入文件，如果之前已经准备过的话可以跳过

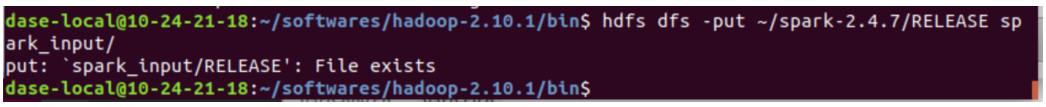

图 5: 准备输入

使用命令

```
spark-shell --master yarn
```

启动 Spark Shell

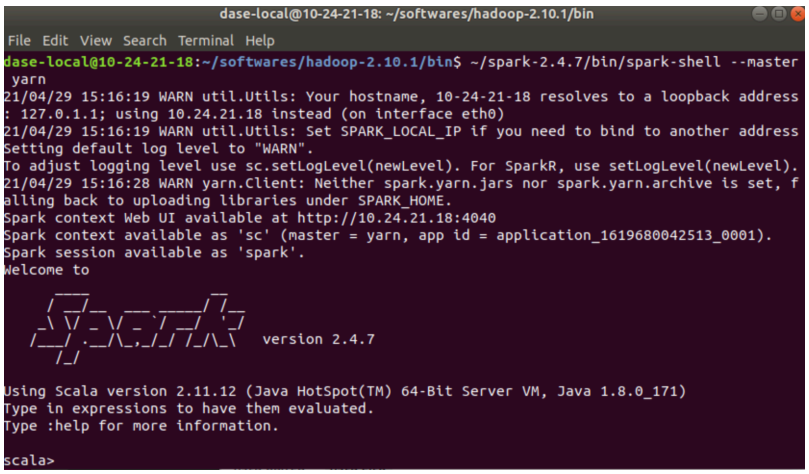

图 6: Spark Shell

使用以下命令

```
sc.textFile("hdfs://localhost:9000/user/dase-local/spark_input/RELEASE")
.flatMap(_.split(" "))
.map((_,1))
.reduceByKey(_ + _).collect
```

统计 RELEASE 文件的单词个数，打印结果如下所示

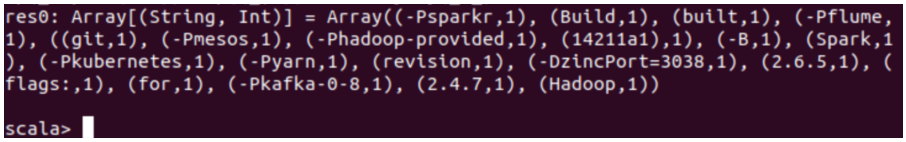

图 7: RELEASE 文件单词个数统计

接着，我们通过**提交 jar 包的方式运行应用程序**
首先是 Client 模式下提交，这个模式下 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

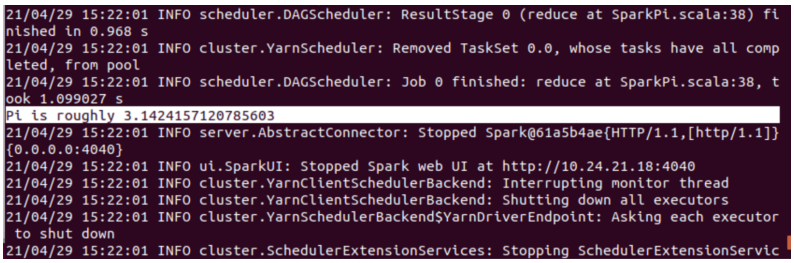

图 8: Client 模式提交

其中我们可以看到 π 的计算结果。
在运行期间，我们另启一个 Terminal，使用 jps 命令查看运行过程出现的进程：

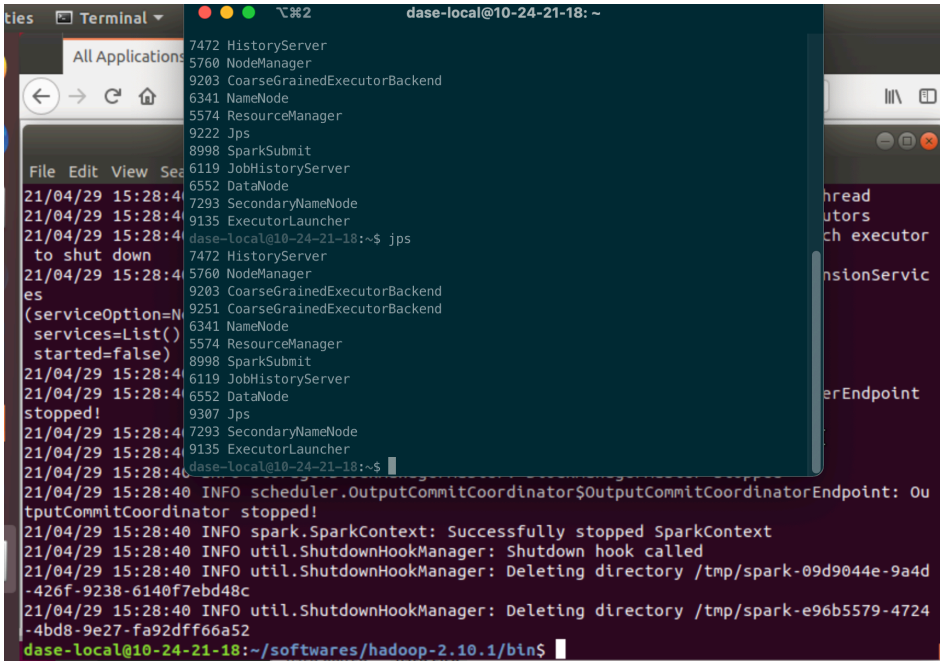

图 9: Client 模式提交

tips：这里出现了两个 CoarseGrainedExecutorBackend 进程，Executor 负责计算任务，即执行 task，而

Executor 对象的创建及维护是由 CoarseGrainedExecutorBackend 负责的，我们需要知道这些 CoarseGrainedExecutorBackend 进程在 Spark 运行期是互相独立的进程。一个 CoarseGrainedExecutorBackend 进程有且仅有一个 executor 对象，它负责将 Task 包装成 taskRunner，并从线程池中抽取出一个空闲线程运行 Task，这样，每个 CoarseGrainedExecutorBackend 能并行运行 Task 的数据就取决于分配给它的 CPU 的个数。

其次是在 Cluster 模式下提交，这个模式下因为 ResourceManager 直接选择在当前节点上的 NodeManager，由其启动 1 个 ApplicationMaster。Driver 运行在 ApplicationMaster 中，客户端看不到运行过程中的信息

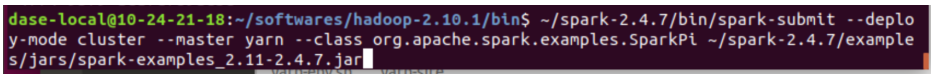

图 10: Cluster 模式提交

随后另启一个 Terminal，使用 jps 命令查看运行过程出现的进程：

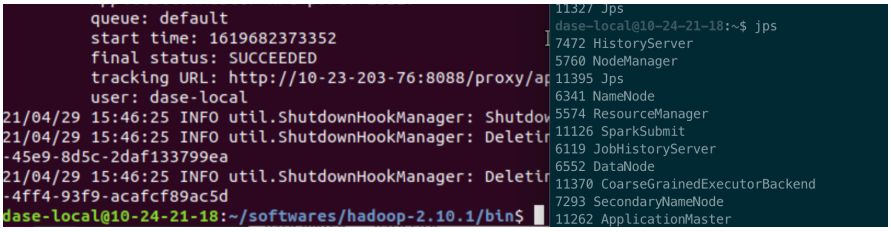

图 11: Cluster 模式提交 Cont'd

可以看到，和 Client 模式提交最明显的区别，就是在 Cluster 模式下没有 ExecutorLauncher 这个进程，在 Cluster 模式下，对应的进程是一个 ApplicationMaster

查看 Spark 程序信息

我们通过 Spark 的 Web UI 来查看 Spark 程序信息

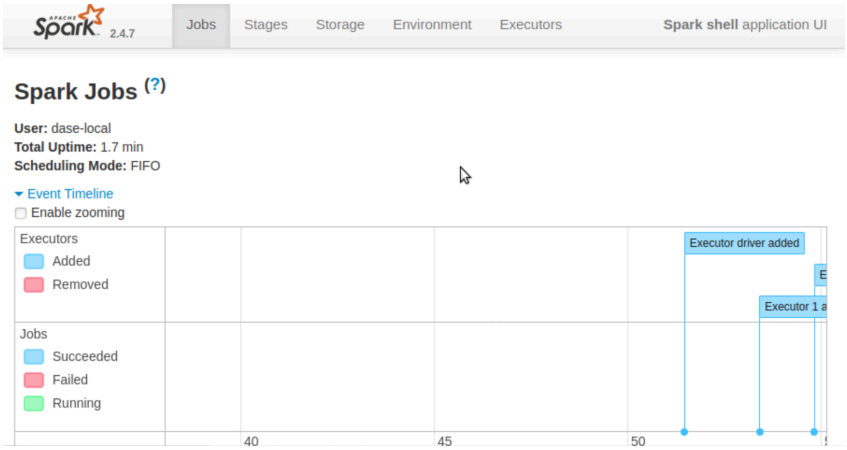

图 12: Spark Web 界面查看

点击一个应用程序的 ID，我们可以通过界面来查看它的运行状况

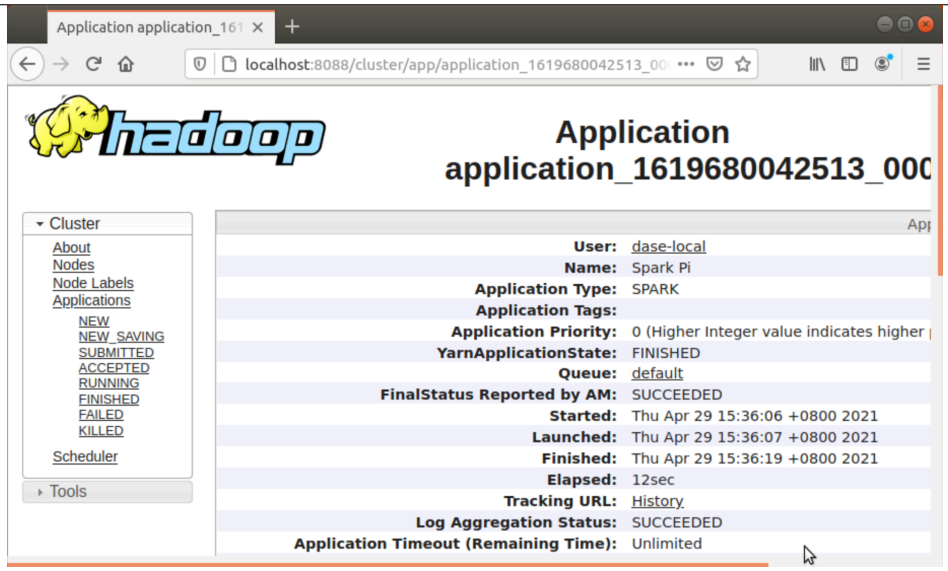

图 13: Hadoop 监控 Spark Pi Application 的界面

查看应用历史记录

我们通过 Web UI 来查看本次启动 Yarn 后提交的所有应用程序相关信息

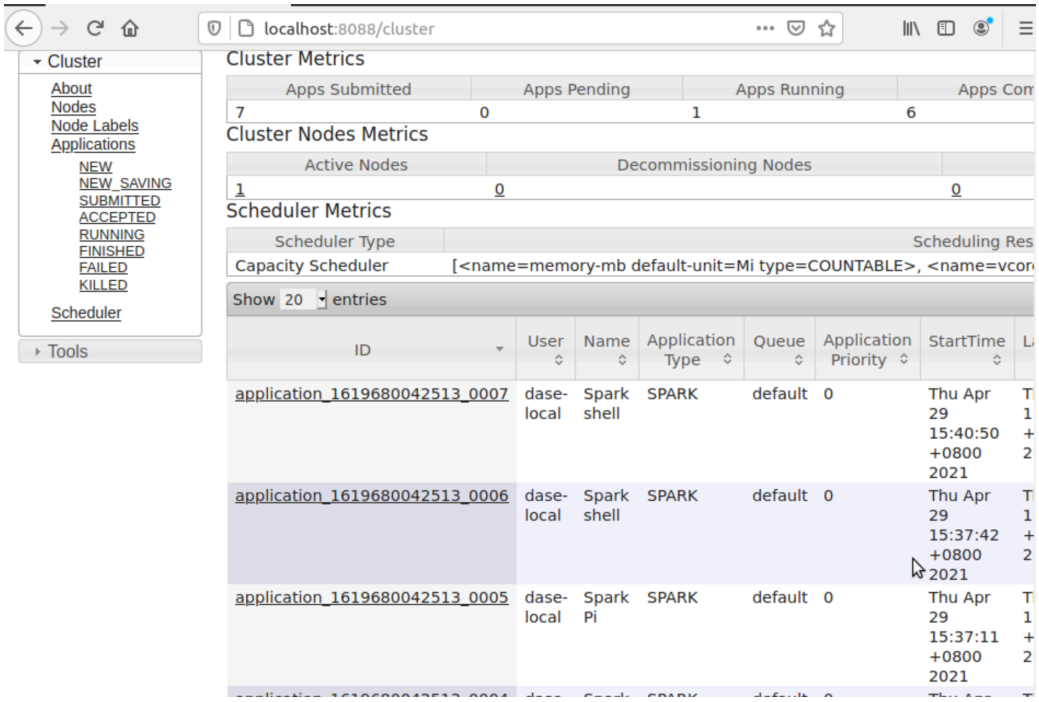

图 14: Hadoop Yarn 界面

最后停止服务，结束单机伪分布式下的实验。

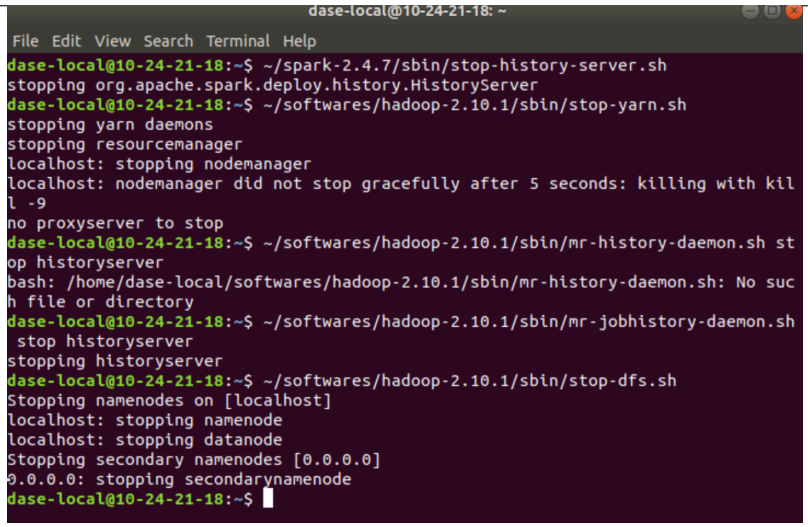

图 15: 停止服务

Section 2
分布式部署

接下来开始 Spark on Yarn 的分布式部署

主节点修改 *spark-env.sh* 文件，slaves 文件，*spark-defaults.conf* 文件和 *spark-config.sh* 文件后将这些内容全部拷贝到其他节点，随后启动服务

查看 Spark 服务信息

启动服务后，使用 jps 命令在主从节点上分别查看进程状态，分布式下主节点充当 Master，从节点充当 Worker，二者可以在下面的进程状态截图看到

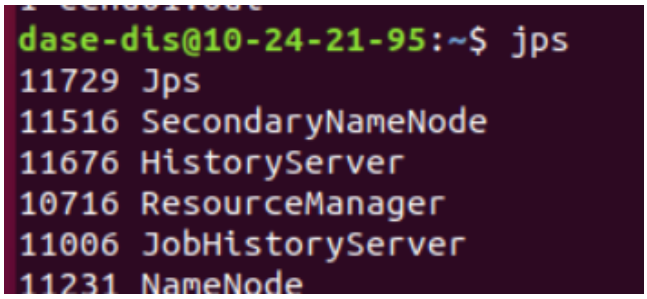

图 16: 分布式部署-主节点

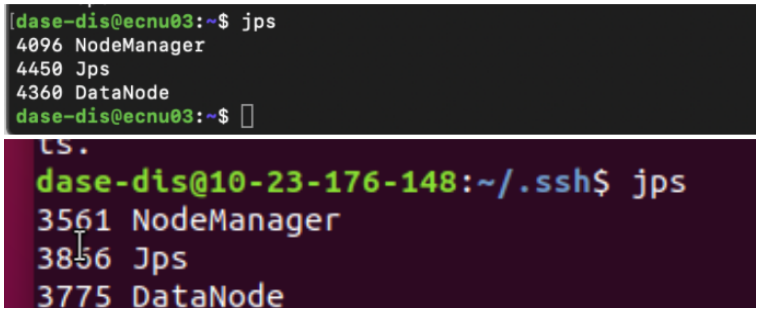

图 17: 分布式部署-从节点

查看 Spark 进程日志

可以在 /*spark*-2.4.7/*logs* 路径下查看进程日志

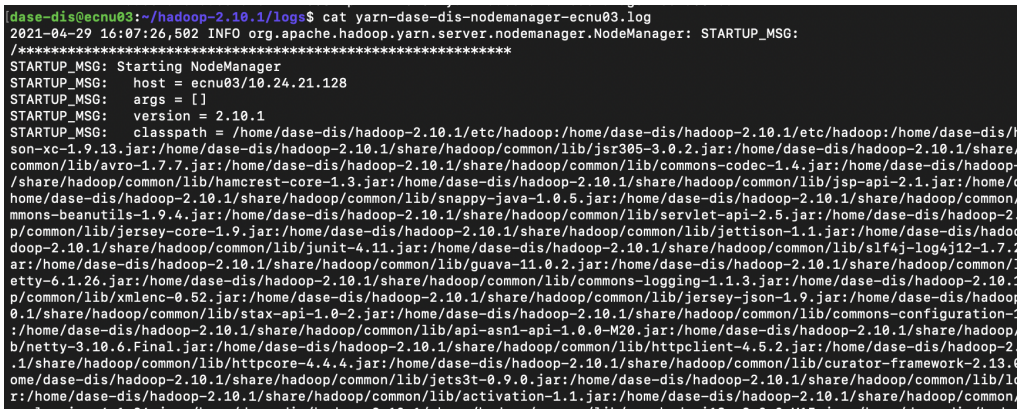

图 18: 查看进程日志: NodeManager

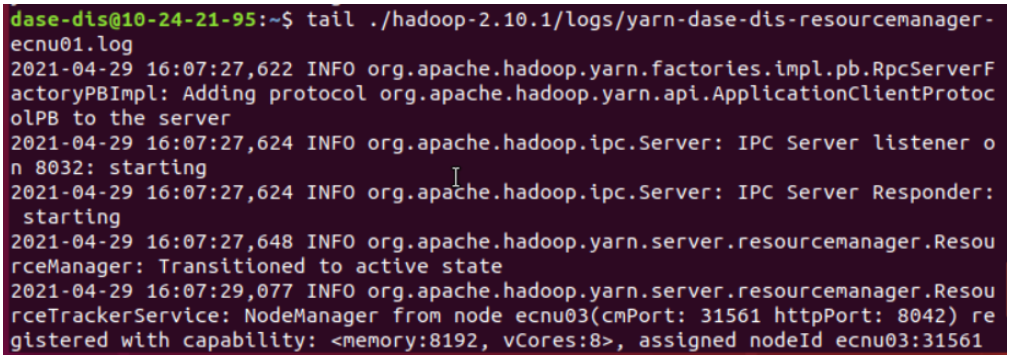

图 19: 查看进程日志 Cont'd: ResourceManager

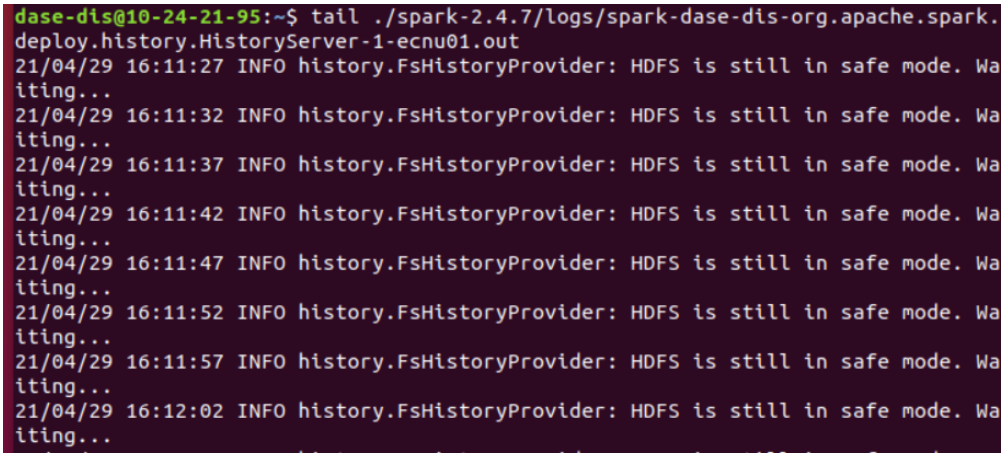

图 20: 查看进程日志 Cont'd: HistoryServer

运行 Spark 程序

在这一步很容易出现与 Safe mode 相关的报错，导致一个从节点无法被启动，我们查阅资料用以下方法解决

Safe mode can happen in 2 ways.

- One is we can forcefully switch namenode to safemode
- Another is namenode automatically enters into the safemode because of some issues.

我们目前遇到的情况显然是第二种，我把搜集的有用的信息放在这里做参考

- If the storage of namenode is full. Then namenode will go to safemode
- If the namenode lacks physical memory, it will enter into safemode
- If the cluster storage is full, namenode will enter into safemode

链接：Why-Hadoop-Cluster-Runs-In-Safe-Mode

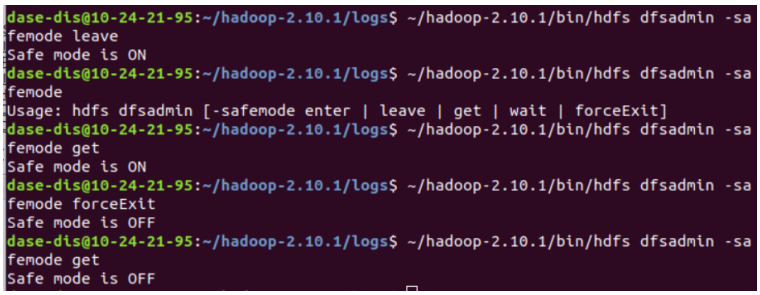

图 21: 关于 SafeMode

检查发现 Data Node 节点均正常，因此怀疑是因为内存的原因，因此，我们限制 child 进程的堆内存，修改如下：

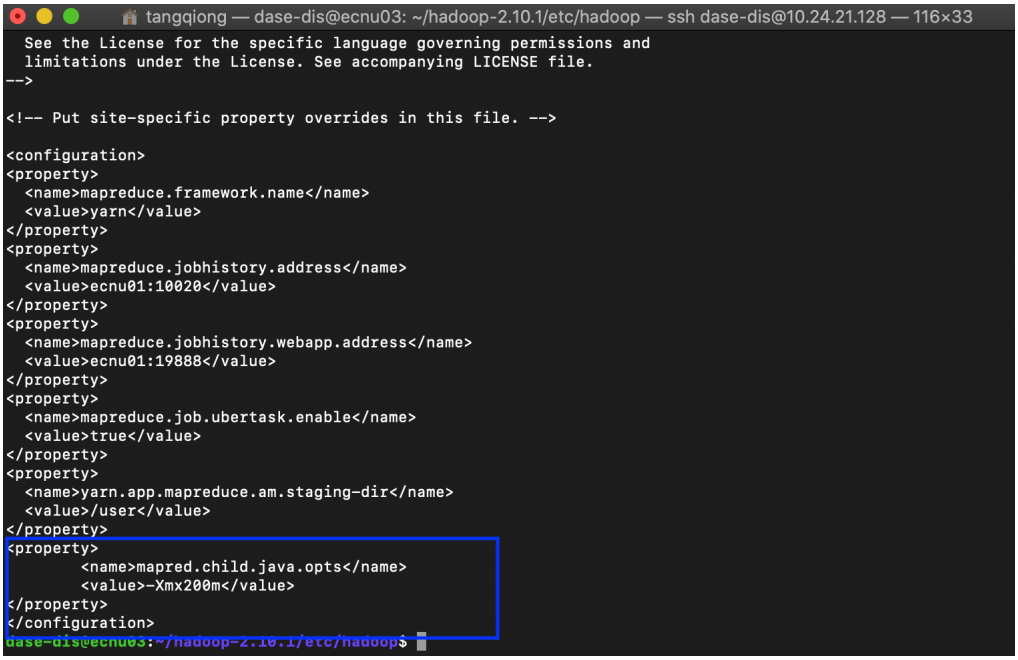

图 22: 调整 Child 进程堆内存大小

修改后重启 hadoop，则不会自动进入 safe mode，一切顺利运行
我们现在先在客户端进入 Spark Shell

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

运行一个对 RELEASE 文件的词频统计

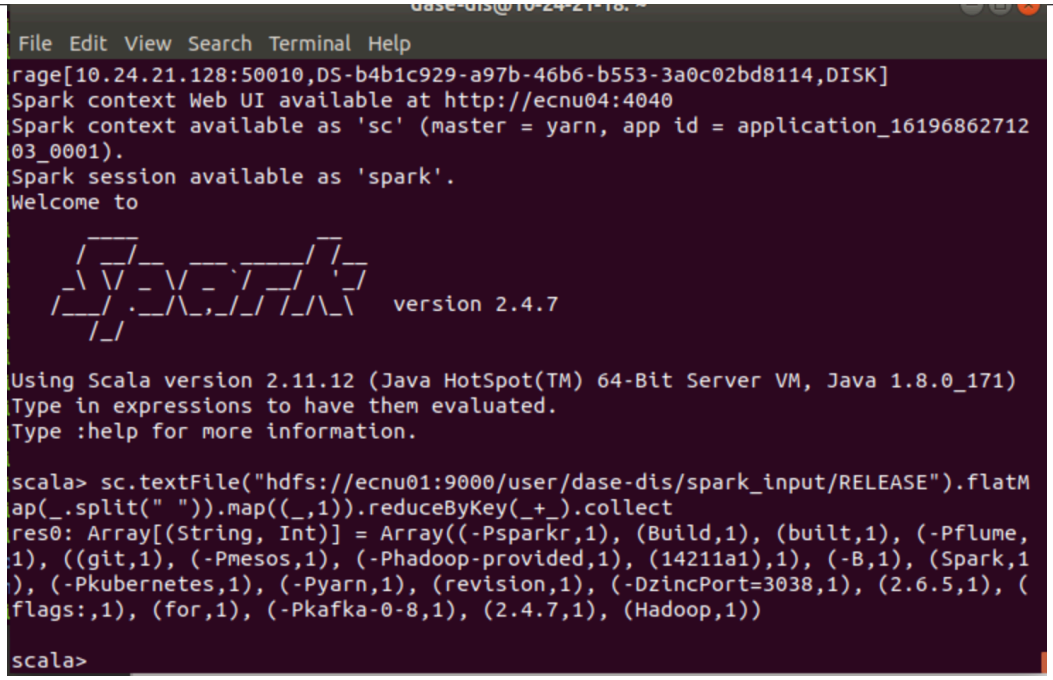

图 23: 分布式-Spark Shell 提交词频统计

随后通过提交 jar 包运行程序
先是在 Client 模式下提交

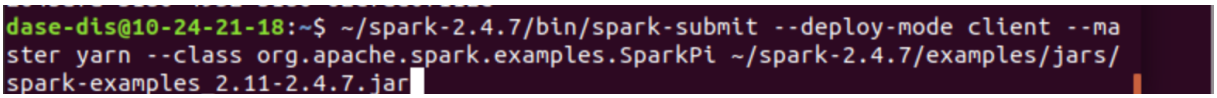

图 24: Client 模式下提交

图 25: 从节点查看进程状态

接着在 Cluster 模式下提交

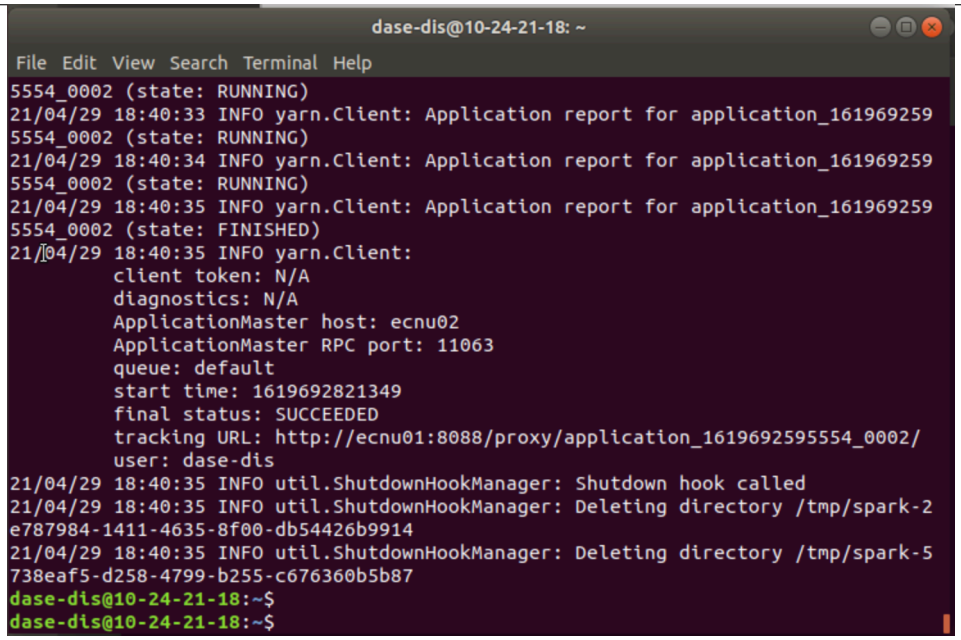

图 26: 客户端 Cluster 模式下提交

检查从节点进程状态，Cluster 模式提交下，此时共存在一个 ApplicationMaster 进程，同时有若干个 CoarseGrainedExecutorBackedn 进程

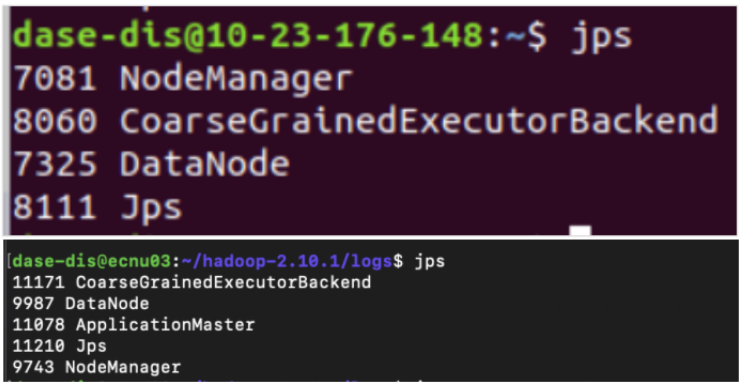

图 27: Cluster 模式下提交-从节点进程状态

查看 Spark 程序运行信息

我们通过 Web UI 分别查看

- Yarn History
- Spark Jobs
- Spark History

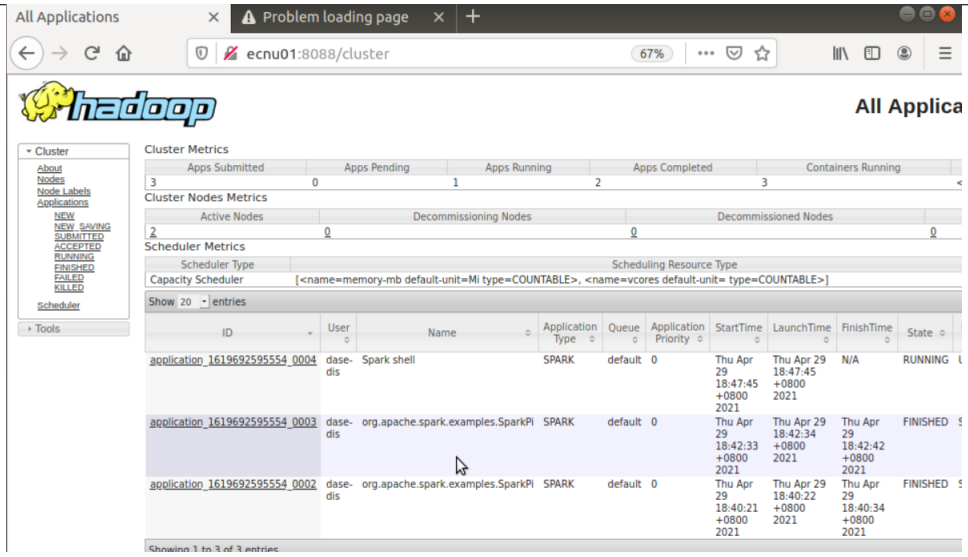

图 28: Web UI 查看 Yarn History

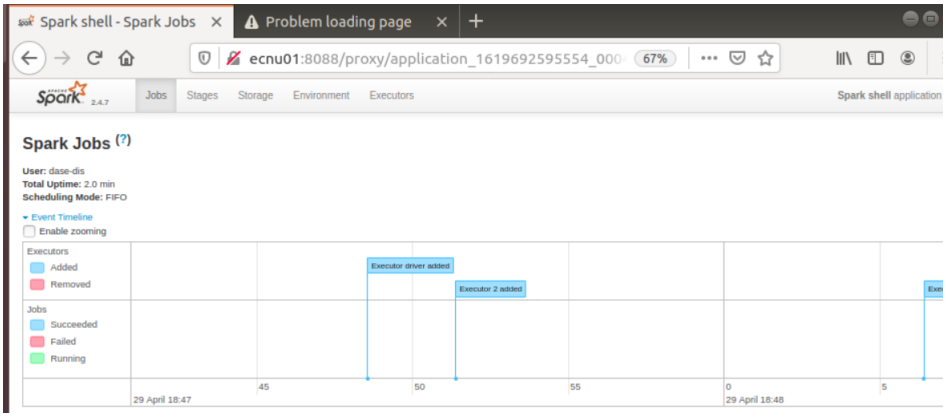

图 29: Web UI 查看 Spark Jobs

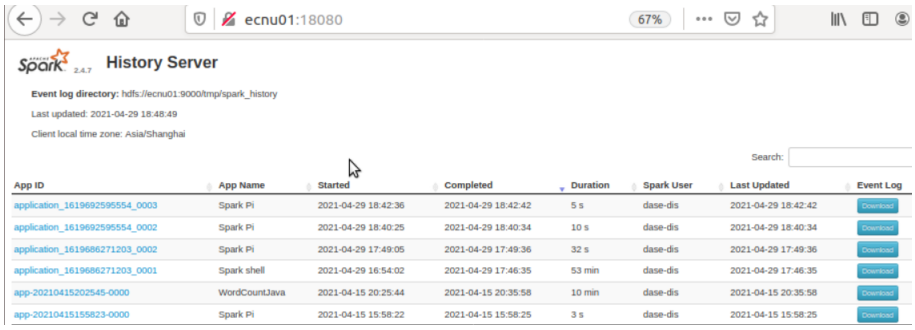

图 30: Web UI 查看 Spark History

Part 5

思考题

Section 1

为什么会在 Hadoop 的 Web UI 中看到关于 Spark 应用程序的记录?

由于 Spark 应用程序是在 Hadoop 上运行的，Hadoop 提供了对应用的管理，使得用户可以获得在 Hadoop 的 RM 上运行的应用的信息。根据 Hadoop 的文档，当一个 RM 被启动运行时，除了 About 页以外的 Web 请求会被自动重定向到该 RM 上，而 Hadoop 提供了一系列 API 用于在启动的 RM 上获取应用的相关信息。因此会在 Hadoop 的 Web UI 中看到关于 Spark 应用程序的记录。

Section 2

对于 Client 提交模式，为什么在 Spark on Yarn 部署时会出现 ExecutorLauncher 进程，而在第七章非 on Yarn 的 Spark 部署时不会出现 ExecutorLauncher 进程?

对于 Client 模式下 Spark，如果不是在 Yarn 上部署，提交应用后由 SparkSubmit 来充当 Driver，不需要 ApplicationMaster。如果是在 Yarn 上部署，则需要启动 ApplicationMaster，因此启动 ExecutorLauncher 来启动 AM。实际上此时调用了 ApplicationMaster 的主方法启动 ApplicationMaster。

Spark 的进程状态基于不同的框架的而不同。在使用 Yarn 框架时，需要满足 Yarn 框架的要求，因此才启动了 ExecutorLauncher，而非 on Yarn 的 Spark 部署时不会出现这个 ExecutorLauncher 进程。

参考资料：Spark on Yarn 源码分析

Part 6

实验补充：在 Yarn 上同时运行 Spark 任务和 MapReduce 任务

Yarn 是一个资源调度平台，负责为运算程序提供服务器运算资源，而 MapReduce 等则当于运行于其上的应用程序。

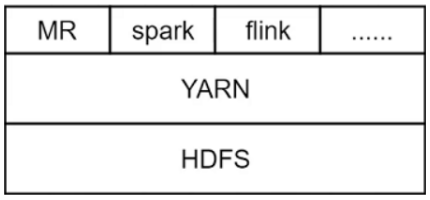

图 31: Yarn Structure

使用 Yarn 是为了对集群资源进行整合，让我们资源得到最大化利用，同一套硬件资源集群上可以运行多个任务，但我们的实验中目前只涉及了在 Yarn 上部署 Spark 任务。我们根据周一理论课上老师的提示，现在尝试一下如何**在分布式模式下基于 Yarn 同时使用 Spark 和 MapReduce**

也就是说，我们希望能获得客户端同时提交一个 MapReduce 任务和一个 Spark 任务，从节点同时执行二者，如下图所示（图片来自课件）

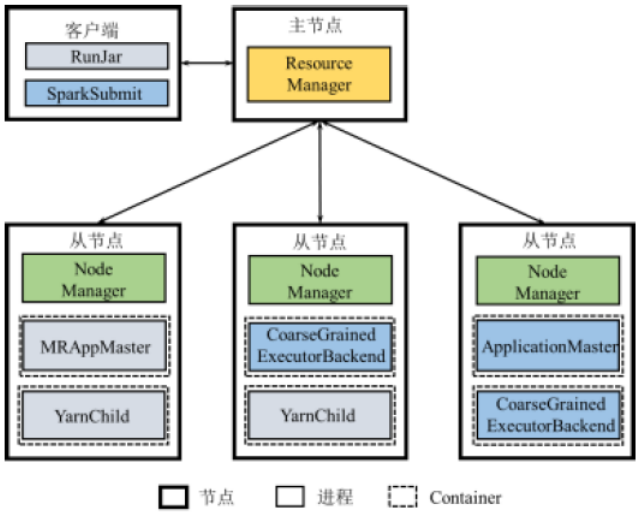

图 32: 基于 Yarn 运行 MapReduce 和 Spark

我们试图同时启动 MR 和 spark 进程。首先，为了保证 spark 在 MR 执行过程中一直处于执行状态，我们在客户端开启一个终端，启动 spark shell。启动后，在从节点使用 jps 查看进程，如下：

```
dase-dis@ecnu03:~$ jps
17970 DataNode
18483 ExecutorLauncher
18265 NodeManager
18526 CoarseGrainedExecutorBackend
19518 Jps
```

图 33: 从节点 ecnu03 查看

```
dase-dis@10-23-176-148:~$ jps
8964 DataNode
9095 CoarseGrainedExecutorBackend
9752 RunJar
7081 NodeManager
9854 Jps
```

图 34: 从节点 ecnu04 查看

查看 Web UI 界面的提示如下，可以看到同时显示两个应用，其中 shell 对应的应用正常运行，处于 Running 状态，而 MR 的应用一直处于 Accepted 状态，即目前还没有运行起来。

ID	User	Name	Application Type	Queue	Application Priority	StartTime	LaunchTime	FinishTime	State	FinalStatus	Runn Conta
plication_1619692595554_0009	dase-dis	grep-search	MAPREDUCE	default	0	Thu Apr 29 19:18:25 +0800 2021	N/A	N/A	ACCEPTED	UNDEFINED	0
plication_1619692595554_0008	dase-dis	Spark shell	SPARK	default	0	Thu Apr 29 19:00:39 +0800 2021	Thu Apr 29 19:00:39 +0800 2021	N/A	RUNNING	UNDEFINED	3

图 35: 通过 WebUI 查看 MapReduce 程序和 Spark 程序运行

我们在客户端也可以看到，Spark Shell 启动的时候，MapReduce 的任务一直卡着没有运行

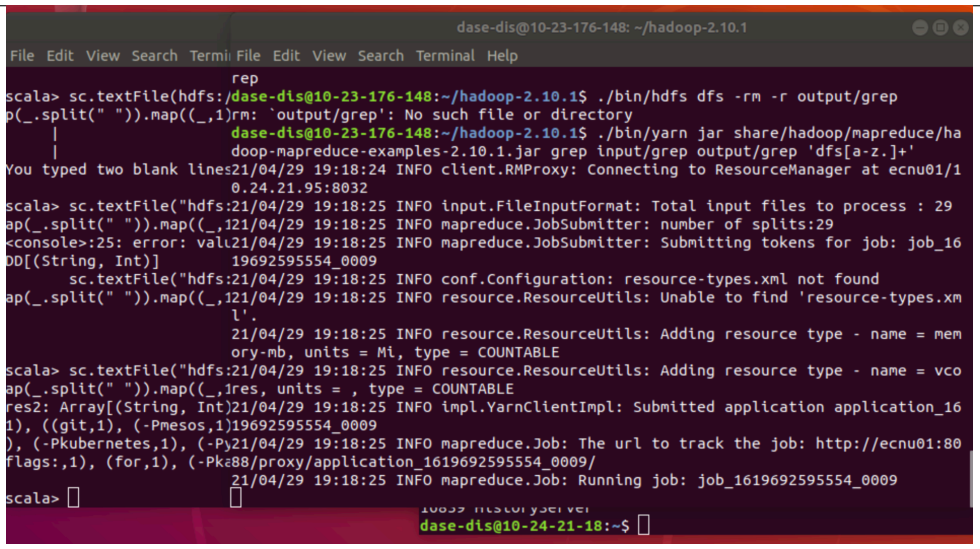

图 36: 客户端查看两个程序的运行情况

这个时候，在客户端退出 Spark Shell，MapReduce 任务才会开始正常运行

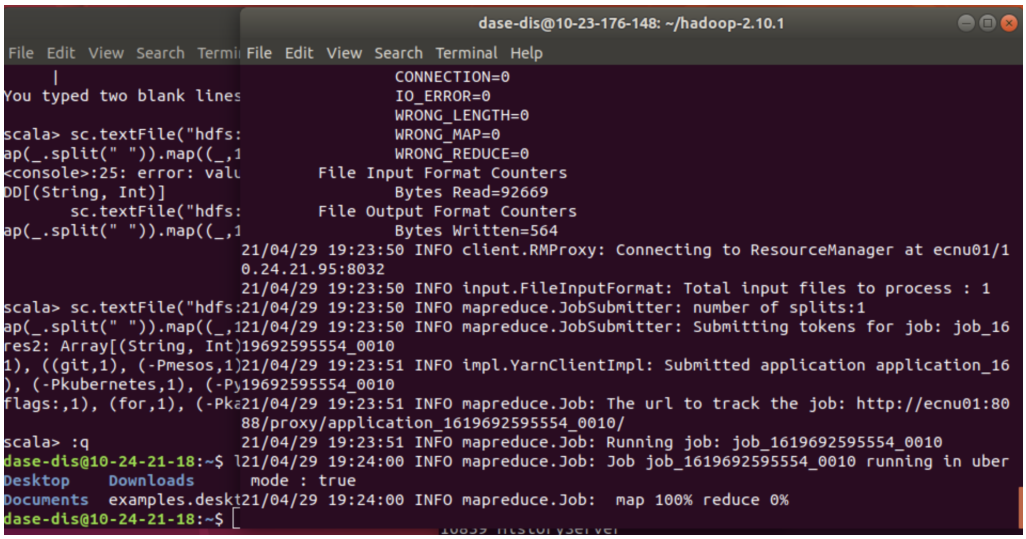

图 37: MapReduce 任务开始正常运行

与此同时，我们关掉 Spark Shell 后在一个从节点上可以查看到 MapReduce 的任务开始运行了，如下所示

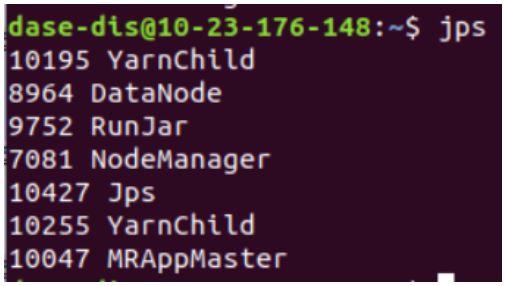

图 38: 关闭 Spark Shell 后从节点 ecnu04 查看

```
[dase-dis@ecnu03:~/hadoop-2.10.1/logs$ jps
11906 DataNode
14678 YarnChild
14663 YarnChild
14649 YarnChild
14666 YarnChild
14973 Jps
9743 NodeManager
```

图 39: 关闭 Spark Shell 后从节点 ecnu03 查看

退出一个 Spark 后才能让 MapReduce 执行，这显然不符合我们的需求，于是我们设法排查错误。

打开 WebUI，发现 Spark Shell 被分配到了 5120mb 的内存，而 MapReduce 的任务没有被分配到任何内存，我们觉得应该是在内存的时候上出现了问题导致一个任务被卡死无法执行

User	Name	Application Type	Queue	Application Priority	StartTime	LaunchTime	FinishTime	State	FinalStatus	Running Containers	Allocated CPU VCores	Allocated Memory MB	Alloc GPU
dase-dis	grep-search	MAPREDUCE	default	0	Thu Apr 29 19:38:03 +0800 2021	N/A	N/A	ACCEPTED	UNDEFINED	0	0	0	-1
dase-dis	Spark shell	SPARK	default	0	Thu Apr 29 19:37:56 +0800	Thu Apr 29 19:38:19 +0800 2021	N/A	RUNNING	UNDEFINED	3	3	5120	-1

图 40: WebUI 查看内存使用

随后在 Web UI 中查看错误诊断，我们能发现错误的原因 *Queue's AM resource limit exceeded.* 原因是达到了队列 AM 可用资源上限，即队列的 AM 已使用资源和 AM 新申请资源之和超出了队列的 AM 资源上限。

为了解决这个报错，我们查阅了阿里云论坛的的 Flink on YARN（下）：常见问题与排查思路

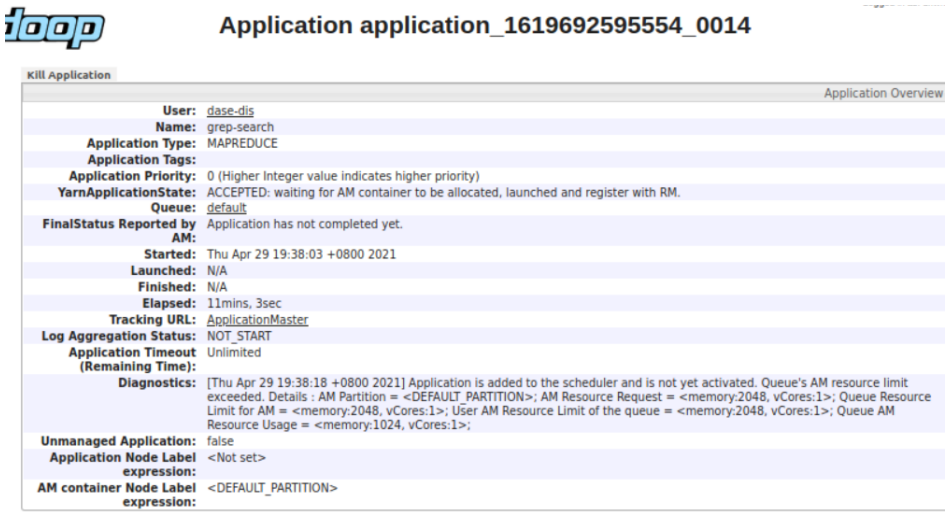

图 41: 在 Web UI 中查看错误诊断

我们需要调整队列 AM 可用资源百分比的配置项：yarn.scheduler.capacity..maximum-am-resource-percent，才能排除这个错误。

我们查阅了 大数据之 Yarn—Capacity 调度器概念以及配置 来定位到需要修改的配置文件

主要是修改 *yarn.scheduler.capacity..maximum-am-resource-percent* 来设置有多少资源可以用来运行 App master，即控制当前激活状态的应用，这个值默认只有 10%，显然无法满足我们的需求。我们将其修改，提高到 50%。

```
<property>
        <name>yarn.scheduler.capacity..maximum-am-resource-percent</name>
        <value>50</value>
</property>
```

图 42: 修改配置

完成上述修改配置之后，重启所有机器

同时提交 MapReduce 任务和 Spark 任务，我们可以看到，Spark Shell 在正常运行的情况下，MapReduce 任务也同样在运行，没有出现之前 Spark 任务占用大部分资源导致 MapReduce 任务被卡住的情况。

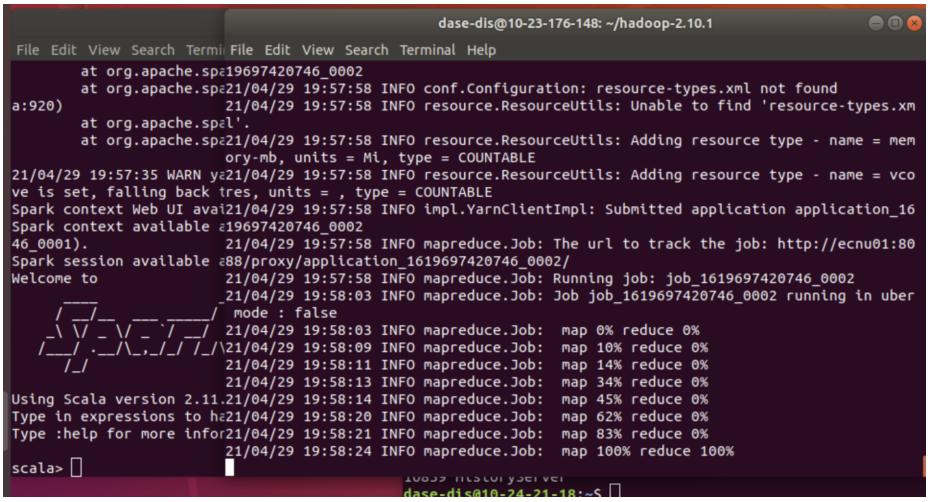

图 43: 客户端查看二者运行情况

随后我们在两个从节点机器中分别使用 jps 命令查看进程，可以发现 ecnu04 的从节点在执行 MapReduce 任务，ecnu03 的从节点既有 MRAppMaster 又有 CoarseGrainedExecutorBackend 这个任务，也就是说这个节点目前既执行 MapReduce 任务，又在执行 Spark 任务。这样的话，实现在 Yarn 上同时部署 Spark 任务和 MapReduce 任务成功了。

同时，我们也可以也可以看出，对于不同任务量的任务，Yarn 分配的进程数量是不同的。对于 Spark Shell 而言，其没有运行计算任务，只启动了一个 CoarseGrainedExecutorBackend。而 MapReduce 的应用程序需要进行计算，所以启动的 Yarn child 进程数量多于前者。

```
dase-dis@10-23-176-148:~$ jps
15346 YarnChild
15331 YarnChild
15333 YarnChild
15511 Jps
14520 DataNode
15337 YarnChild
15321 YarnChild
15324 YarnChild
15325 YarnChild
14814 NodeManager
15118 RunJar
15327 YarnChild
```

图 44: 从节点 ecnu04

```
[dase-dis@ecnu03:~/hadoop-2.10.1/logs$ jps
18784 YarnChild
18800 YarnChild
17970 DataNode
18483 ExecutorLauncher
18885 Jps
18791 YarnChild
18265 NodeManager
18652 MRAppMaster
18526 CoarseGrainedExecutorBackend
```

图 45: 从节点 ecnu03

到此为止，我们成功实现了在 Yarn 上同时运行 Spark 任务和 MapReduce 任务。

Part 7
实验总结

最后，回顾一下两种提交方式的不同
这部分参考了 Medium 上一篇介绍 Spark on Yarn 的文章：Running Spark Jobs on YARN

- Yarn-cluster 模式：在 Yarn-cluster 模式下，driver 运行在 Appliaction Master 上，Appliaction Master 进程同时负责驱动 Application 和从 Yarn 中申请资源，该进程运行在 Yarn container 内，所以启动 Application Master 的 client 可以立即关闭而不必持续到 Application 的生命周期

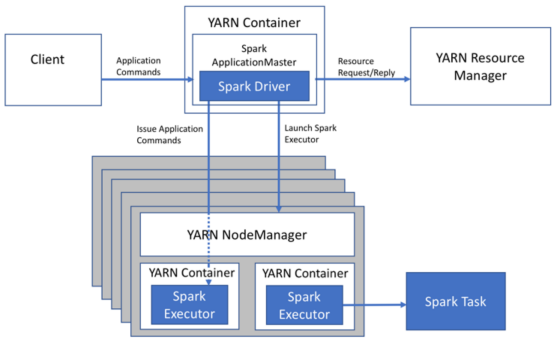

图 46: Yarn Cluster

- Yarn-client 模式：在 Yarn-Client 模式下，Application Master 仅仅从 Yarn 中申请资源给 Executor，之后 client 会跟 container 通信进行作业的调度

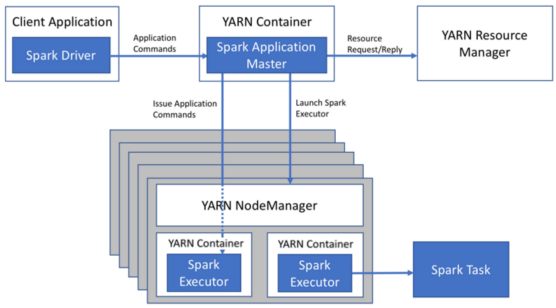

图 47: Yarn Client

两种提交方式的对比

	YARN Cluster	YARN Client	Spark Standalone
Driver runs in:	Application Master	Client	Client
Who requests resources?	Application Master	Application Master	Client
Who starts executor processes?	YARN NodeManager	YARN NodeManager	Spark Slave
Persistent services	YARN ResourceManager and NodeManagers	YARN ResourceManager and NodeManagers	Spark Master and Workers
Supports Spark Shell?	No	Yes	Yes

图 48: 提交方式的对比

总结：本次实验收获颇多，在集中式和分布式两种模式下基于 Yarn 部署 Spark 应用程序，最后我们依据老师上课时给出的提问，自己补充了在 Yarn 上同时部署 Spark 和 MapReduce 的实验，深刻体会了 Yarn “一个平台、多个框架”的特点。

Part 7
相关参考

- Medium 上一篇介绍 Spark on Yarn 的文章：Running Spark Jobs on YARN
- StackOverflow 上解决 Safe Mode 问题的回答：Why-Hadoop-Cluster-Runs-In-Safe-Mode
- 研究 ExecutorLauncher 的参考：Spark on Yarn 源码分析
- Yarn 调度器：大数据之 Yarn—Capacity 调度器概念以及配置
- 阿里云论坛上解决 AM 资源上限问题的回答：Flink on YARN（下）：常见问题与排查思路（虽然是针对 Flink 的问题，但是解决方案是一样的）