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：Spark 编程	学号：10185501402	上机实践日期：2021/4/15
上机实践编号：Lab8	组号：Group5	上机实践时间:

Part 1
实验目的

(1) 学习编写简单的基于 RDD API 的 Spark 程序。

(2) 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掌握在 IDEA 中调试 Spark 相关程序，以及在单机伪分布式、分布式模式下提交运行 Spark 相关程序的方法。

Part 2
实验任务

(1) 完成 WordCount 示例程序的编写。

(2) 在单机伪分布式和分布式模式下运行 WordCount 示例程序

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

(5) IDEA 版本：2020.2.3（Ultimate 版）

Part 4
实验过程

Section 1
编写 Spark 应用程序

新建一个 Maven 项目并且添加依赖

pom.xml (SparkDemo)

```
    <artifactId>SparkDemo</artifactId>
    <version>1.0-SNAPSHOT</version>

    <build>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-compiler-plugin</artifactId>
                <configuration>
                <source>8</source>
                <target>8</target>
    </configuration>
            </plugin>
        </plugins>
    </build>

    <dependencies>
        <dependency>
            <groupId>org.apache.spark</groupId>
                <artifactId>spark-core_2.11</artifactId>
            <version>2.4.7</version>
        </dependency>
    </dependencies>
</project>
```

图 1: 配置依赖项

编写 Spark 词频统计程序

pom.xml (SparkDemo) | WordCount.java

```
package cn.edu.ecnu.spark.example.java.wordcount;

import org.apache.spark.SparkConf;
import org.apache.spark.api.java.JavaPairRDD;
import org.apache.spark.api.java.JavaRDD;
import org.apache.spark.api.java.JavaSparkContext;
import org.apache.spark.api.java.function.*;
import scala.Tuple2;

import java.util.Arrays;
import java.util.Iterator;

public class WordCount {

    public static void run(String[] args) {
        /* 步骤1：通过SparkConf设置配置信息，并创建SparkContext */
        SparkConf conf = new SparkConf();
        conf.setAppName("WordCountJava");
        //conf.setMaster("local"); // 仅用于本地进行调试，如在集群中运行则删除本行
        JavaSparkContext sc = new JavaSparkContext(conf);

        /* 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 */
        // 读入文本数据，创建名为lines的RDD
        JavaRDD<String> lines = sc.textFile(args[0]);

        // 将lines中的每一个文本行按空格分割成单个单词
        JavaRDD<String> words =
                lines.flatMap(
                        new FlatMapFunction<String, String>() {
                            @Override
                            public Iterator<String> call(String line) throws Exception {
```

Event Log

图 2: 编写 WordCount 程序-1

需要注意，单机模式运行需要

```
conf.setMaster("local")
```

```
69
70          // 输出词频统计结果到文件
71          wordCounts.saveAsTextFile(args[1]);
72          /* 步骤3: 关闭SparkContext */
73          sc.stop();
74      }
75
76      public static void main(String[] args) {
77          run(args);
78      }
79  }
```

图 3: 编写 WordCount 程序-2

Section 2
调试 Spark 应用程序

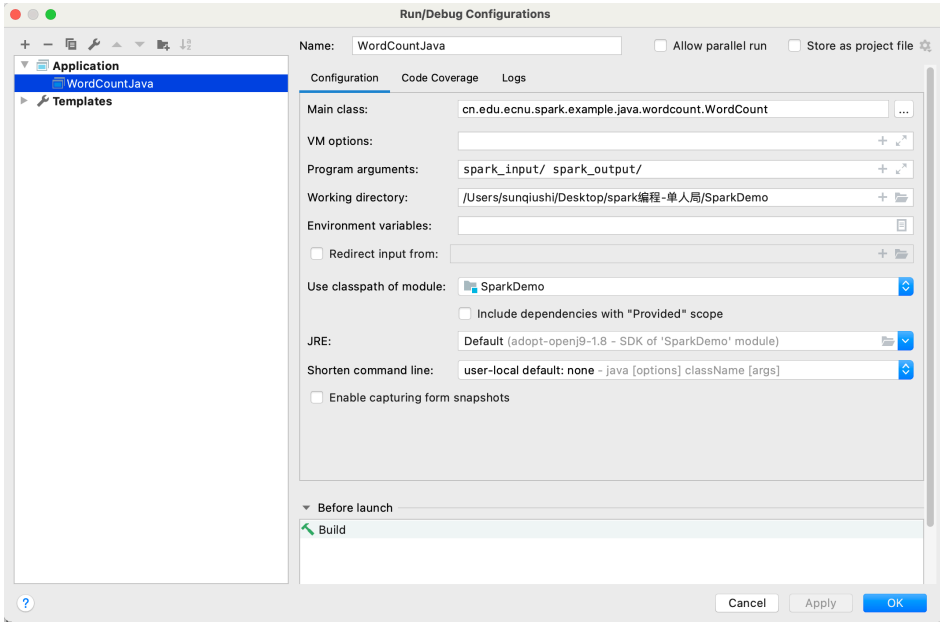

图 4: 配置运行环境

tips: 这里要注意，不要自己手动给输出路径建立目录！否则会报错

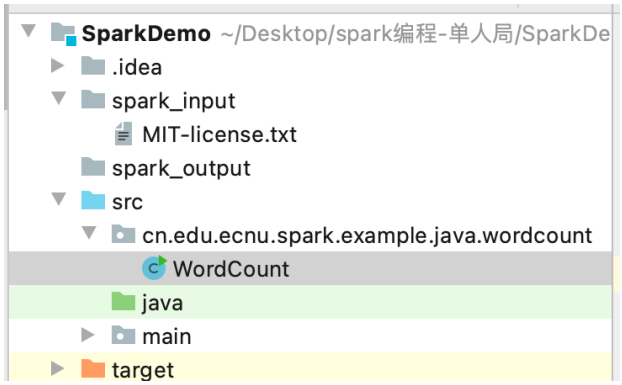

图 5: 目录结构，勿手动建立 output 路径

Section 3
运行 Spark 应用程序

因为本第没有 pd.train 文件，我找了一个常见的 MIT-License 文档来作为词频统计的输入

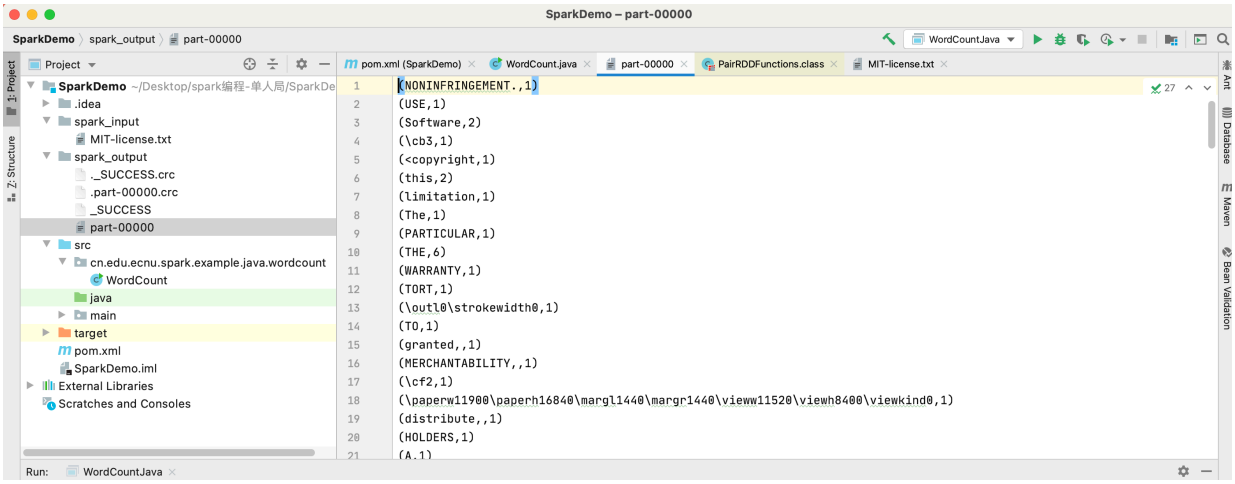

图 6: WordCount 输出结果

可以看到统计了 MIT-License 中的词频

Section 4
单机伪分布式部署

与之前的实验一样，我们在实验开始之前先启动 hdfs 和 spark 服务，还有任务历史服务器。

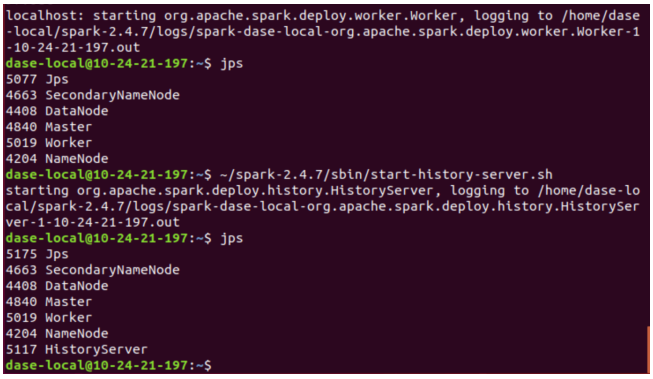

图 7: 启动相关服务

随后在根目录或者是 spark 目录下，建立 myApp 文件夹，再通过 scp 命令将本地的 jar 包复制到云主机的 dase-local 之中

图 8: 复制 wordcount.jar

接下来我们开始执行 WordCount 任务

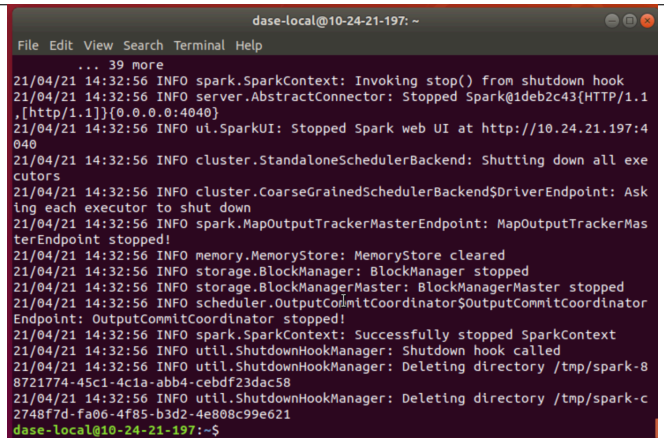

图 9: 执行 WordCount 任务

使用 Web UI 查看信息

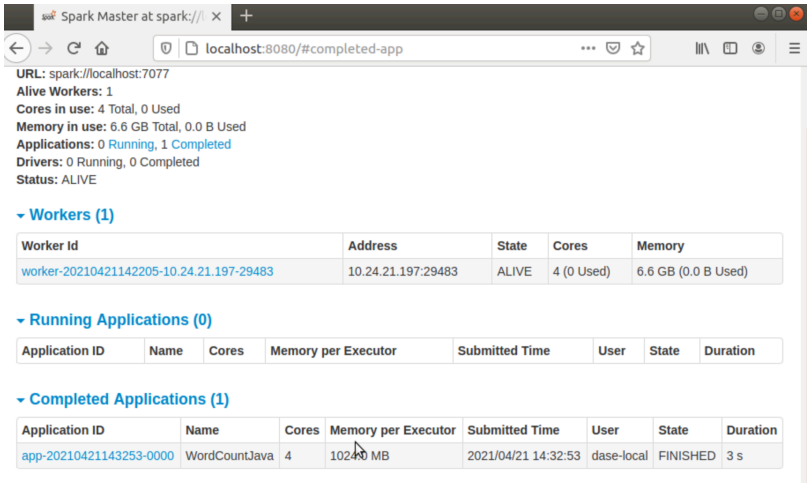

图 10: 通过 Web UI 查看 WordCount 任务

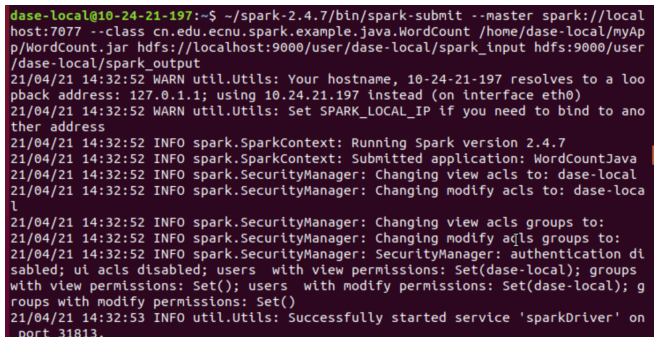

图 11: 执行 WordCount 任务 Cont'd

执行完后，我们通过 Web UI 来查看这个任务的输出

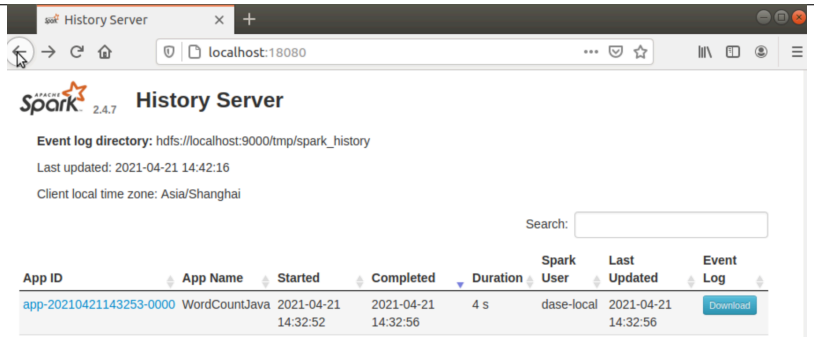

图 12: 查看任务输出

访问 HDFS 的文件信息页面，点击 user，查看文件输出

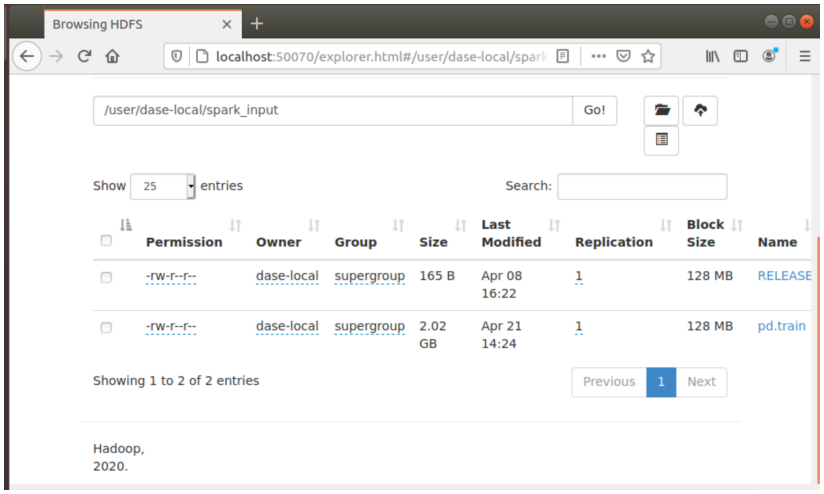

图 13: 查看任务输出

接着，检查 WordCount 任务的输出

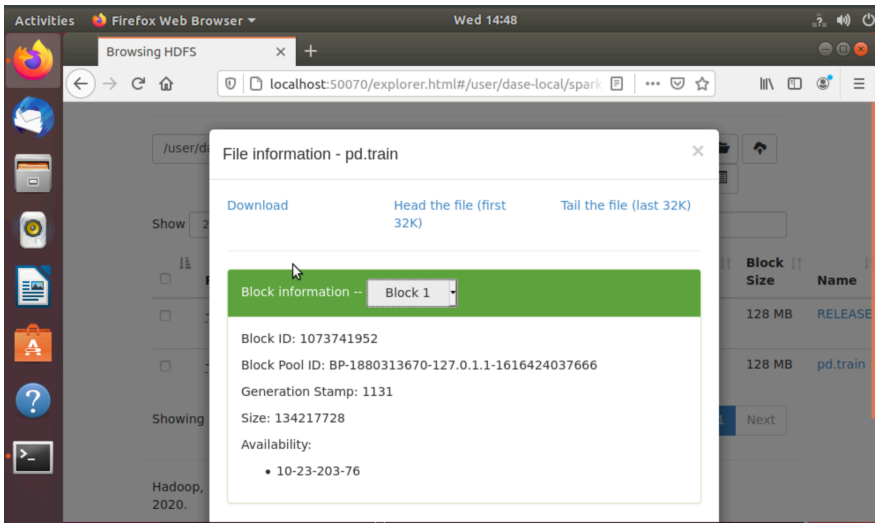

图 14: 检查 WordCount 任务的输出

最后，依次关闭 HDFS、Spark 和 HistoryServer

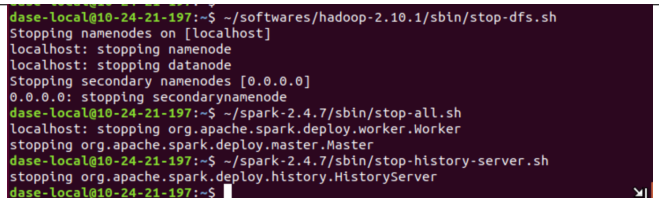

图 15: 结束单机伪分布式实验

Section 5
分布式部署

接下来，我们在分布式环境下运行这个任务，因为已经做过多次分布式实验，这里就不再赘述修改 hosts/免密登陆等操作

主节点启动服务，随后客户端 ecnu04 将需要使用的数据先上传到 HDFS 之中，随后即可提交程序

Remark: 实验手册 Page96 中在客户端准备数据时，如果之前没有在 HDFS 里存过 pd.train 数据集的话，应该先使用 put 命令进行数据上传

随后在客户端使用

```
~/spark-2.4.7/bin/spark-submit \
--master spark://ecnu01:7077 \
-- class cn.edu.ecnu.spark.example.java.wordcount.WordCount \
/home/dase-dis/spark-2.4.7/myApp/WordCount.jar

hdfs://ecnu01:9000/user/dase-dis/spark_input
hdfs://ecnu01:9000/user/dase-dis/spark_output
```

和单机模式一样！务必不要自己创建输出目录！建议在执行前先尝试删除 spark_output 文件夹

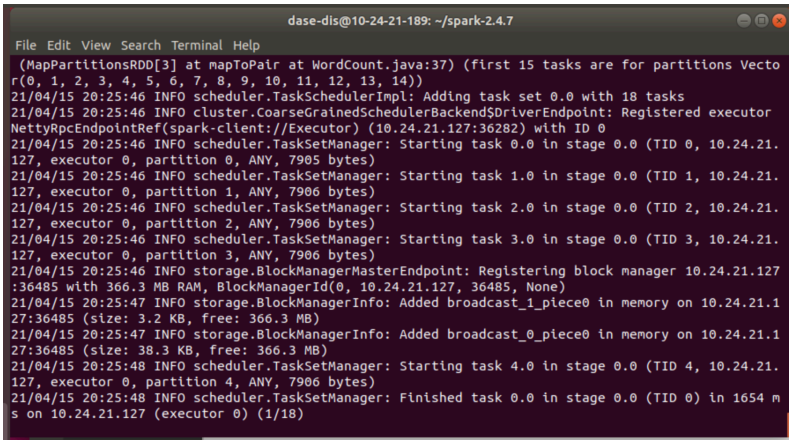

图 16: 分布式环境提交任务

接下来可以查看任务的执行过程

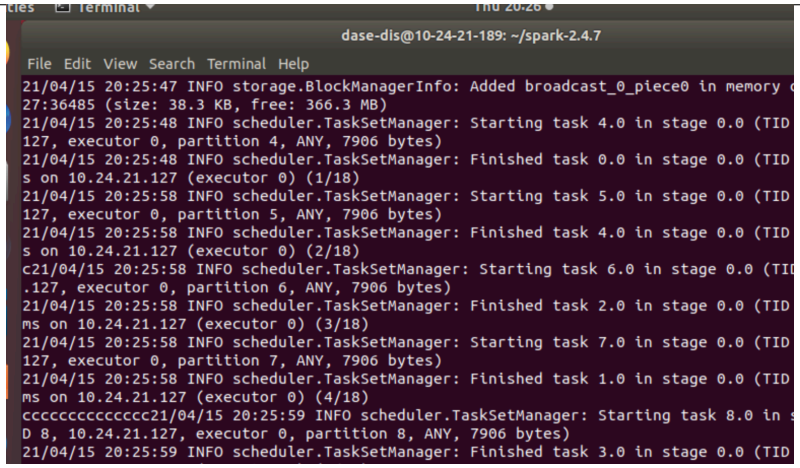

图 17: 分布式环境任务执行

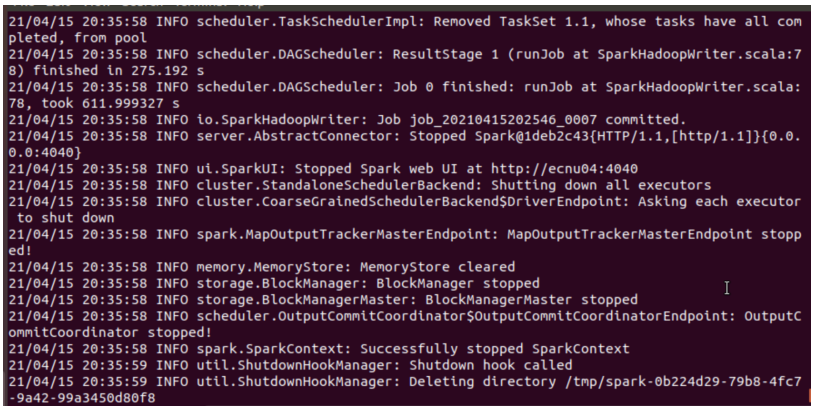

图 18: 分布式环境任务执行 Cont'd

通过 Web UI 来查看任务的输出

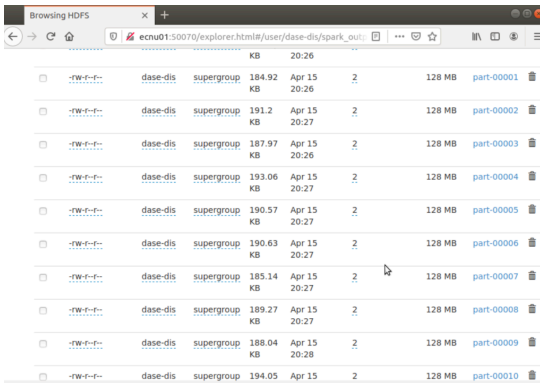

图 19: 任务运行输出

Part 5

思考题

Section 1

第 8.4.3 节中编写的 WordCount 应用程序由多少个 Stage 组成？请结合 Spark 的 Web UI 进行说明。

如下所示，WordCount 应用程序由两个 Stage 组成

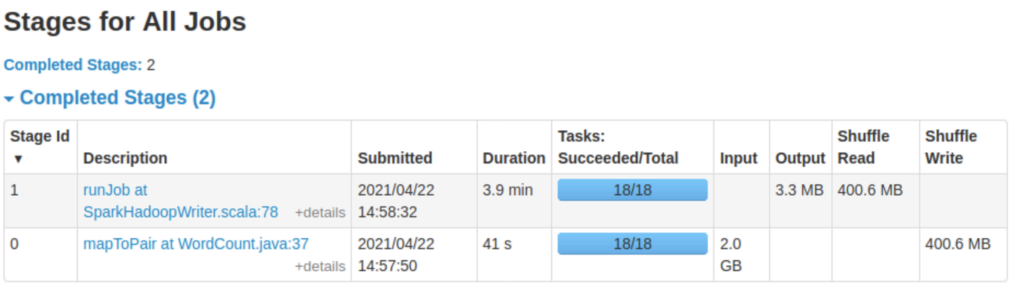

图 20: 思考题 1

Section 2

对于第 8.4.3 节中编写的 WordCount 应用程序，根据实际运行情况，结合 Spark 的 Web UI 说明运行过程中 Shuffle 的数据量有多大。

如下图所示，Shuffle 的数据量为 400.6MB

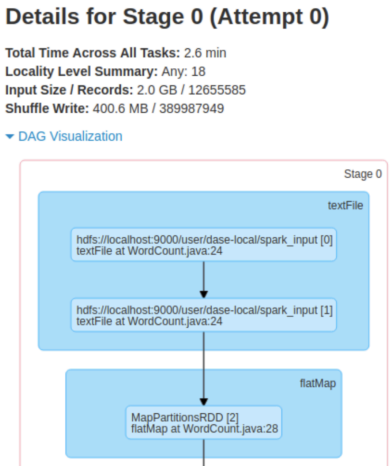

图 21: 思考题 2-1

图 22: 思考题 2-2

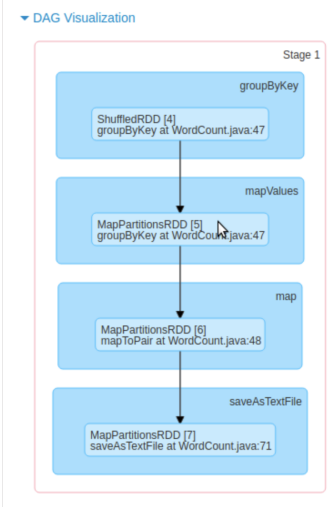

图 23: 思考题 2-3

Part 6

思考与探索

Remark1: 不论是在单机模式执行还是分布式模式执行，都不要自己在工作路径下建立 spark_output 这个文件夹，在分布式环境部署应该先做一下检查，这个地方一开始没反应过来，导致我作为客户端提交后始终显示路径已经存在，后来才一下子反应过来主节点那个地方需要清空，分布式环境下多次实验一定要注意这个。不过排查起来也算方便，看客户端抛出的异常描述就可以八九不离十的把问题搞清楚。

Remark2: 对比一下同样对 pd.train 数据集进行 WordCount，分布式模式部署和单机伪分布式部署在运行时间上的差异。

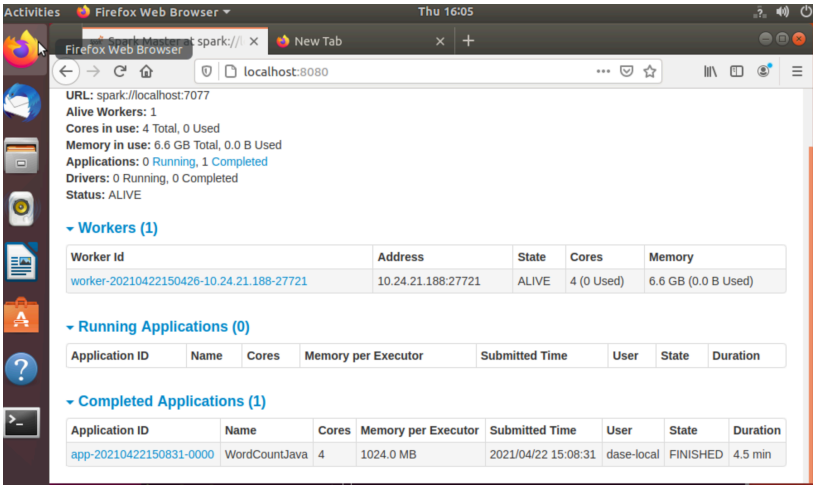

图 24: 单机伪分布式部署：时间开销

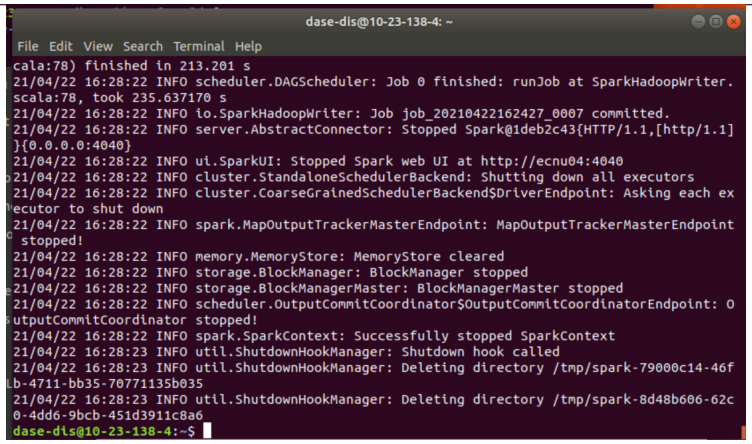

图 25: 分布式部署：时间开销

我们可以看到，单机伪分布式下执行完对 pd.train 数据集的 WordCount 花了 4.5min，在分布式模式下部署花费了 4min 不到一些，速度上有提升但并是压倒性的优势，这可能是从节点个数比较少 + 程序还没有优化好导致的。

Remark3:

实验手册 Page96 中在客户端准备数据时，如果之前没有在 HDFS 里存过 pd.train 数据集的话，应该先使用

```
~/softwares/hadoop-2.10.1/bin/hdfs dfs -put .....
```

这个命令上传文件，而非直接使用-cp 命令

Remark4:

从节点提交任务时，可能会出现一个错误即主节点启动了 Safe Mode 导致无法继续执行，执行如下命令即可（这个错误没复现，但是还挺好解决的，但是随着我们做接下来的实验，发现这个坑是和从节点内存分配有关系。）

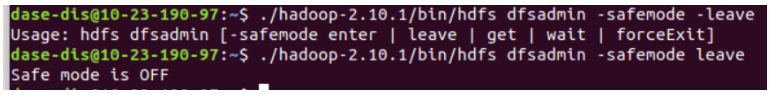

图 26: 主节点解除 Safe Mode

Part 7

参考资料

(1) Spark 编程参考：Spark 中文指南 (入门篇)-Spark 编程模型

(2) MapReduce 与 Spark 用于大数据分析之比较