

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

```

<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

```

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 main(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 {

```

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
71
72
73
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

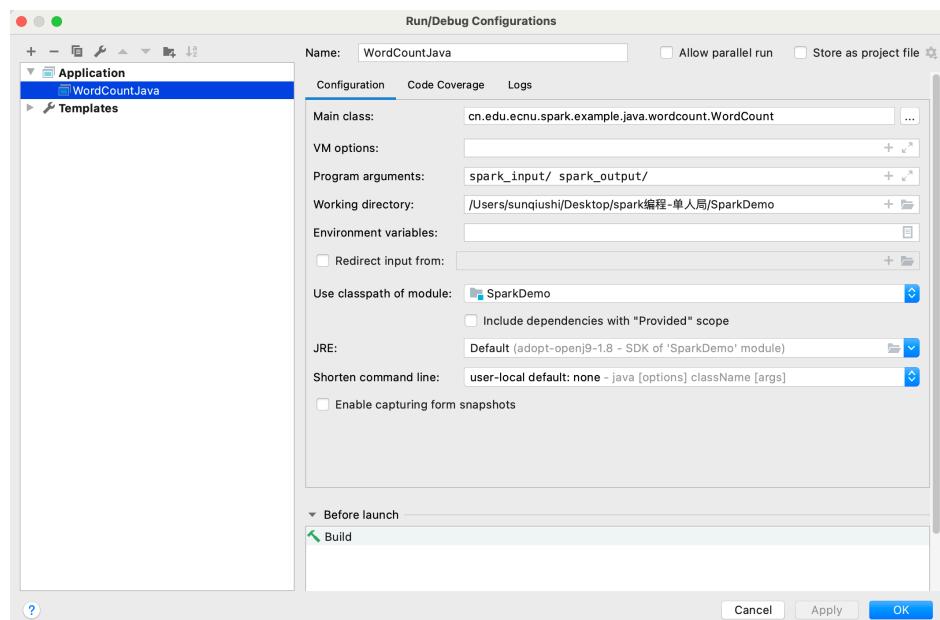


图 4: 配置运行环境

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

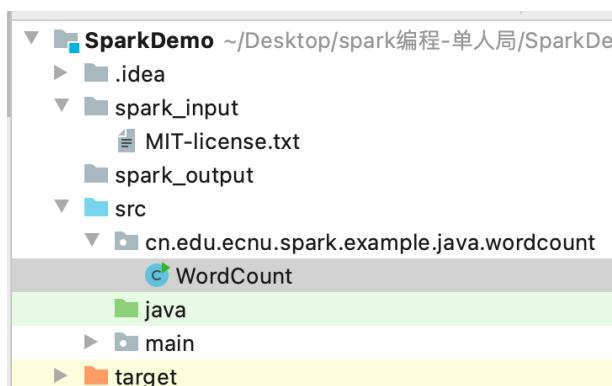


图 5: 目录结构，勿手动建立 output 路径

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

The screenshot shows the IntelliJ IDEA interface with the project 'SparkDemo' open. The 'spark_output/part-00000' file is selected in the left sidebar. The code in the editor is:

```

1 NONINFRINGEMENT,,1
2 (USE,,1)
3 (Software,,2)
4 (\cb3,,1)
5 (<copyright,,1)
6 (this,,2)
7 (limitation,,1)
8 (The,,1)
9 (PARTICULAR,,1)
10 (THE,,6)
11 (WARRANTY,,1)
12 (TORT,,1)
13 (\outl0\strokeWidth,,1)
14 (To,,1)
15 (granted,,1)
16 (MERCHANTABILITY,,1)
17 (\cf2,,1)
18 (\paperw11900\paperh16840\margl1440\margr1440\vieww11520\viewh8400\viewKind0,,1)
19 (distribute,,1)
20 (HOLDERS,,1)
21 (A,,1)

```

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。

```

localhost: starting org.apache.spark.deploy.worker.Worker, logging to /home/dase-local/spark-2.4.7/logs/spark-dase-local-org.apache.spark.deploy.worker.Worker-1-10-24-21-197.out
dase-local@10-24-21-197:~$ jps
5077 Jps
4663 SecondaryNameNode
4408 DataNode
4840 Master
5019 Worker
4204 NameNode
dase-local@10-24-21-197:~$ ~/spark-2.4.7/sbin/start-history-server.sh
starting org.apache.spark.deploy.history.HistoryServer, logging to /home/dase-local/spark-2.4.7/logs/spark-dase-local-org.apache.spark.deploy.history.HistoryServer-1-10-24-21-197.out
dase-local@10-24-21-197:~$ jps
5175 Jps
4663 SecondaryNameNode
4408 DataNode
4840 Master
5019 Worker
4204 NameNode
5117 HistoryServer
dase-local@10-24-21-197:~$

```

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

```

sunqlush1@sunqlush1deMacBook-Pro:~/Desktop/spark-program$ scp /Users/sunqlush1/Desktop/spark-program/WordCount.jar dase-local@10.24.21.197:myApp/

```

图 8: 复制 wordcount.jar

接下来我们开始执行 WordCount 任务

```
dase-local@10-24-21-197: ~
File Edit View Search Terminal Help
... 39 more
21/04/21 14:32:56 INFO spark.SparkContext: Invoking stop() from shutdown hook
21/04/21 14:32:56 INFO server.AbstractConnector: Stopped Spark@ideb2c43[HTTP/1.1
,[http://1.1]{0.0.0.0:4040}
21/04/21 14:32:56 INFO util.SparkUI: Stopped Spark web UI at http://10.24.21.197:4040
21/04/21 14:32:56 INFO cluster.StandaloneSchedulerBackend: Shutting down all executors
21/04/21 14:32:56 INFO cluster.CoarseGrainedSchedulerBackend$DriverEndpoint: Asking each executor to shut down
21/04/21 14:32:56 INFO spark.MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint stopped!
21/04/21 14:32:56 INFO memory.MemoryStore: MemoryStore cleared
21/04/21 14:32:56 INFO storage.BlockManager: BlockManager stopped
21/04/21 14:32:56 INFO storage.BlockManagerMaster: BlockManagerMaster stopped
21/04/21 14:32:56 INFO scheduler.OutputCommitCoordinator$OutputCommitCoordinatorEndpoint: OutputCommitCoordinator stopped!
21/04/21 14:32:56 INFO spark.SparkContext: Successfully stopped SparkContext
21/04/21 14:32:56 INFO util.ShutdownHookManager: Shutdown hook called
21/04/21 14:32:56 INFO util.ShutdownHookManager: Deleting directory /tmp/spark-8721774-45c1-4c1a-abbd4-cebf23dac58
21/04/21 14:32:56 INFO util.ShutdownHookManager: Deleting directory /tmp/spark-c2748f7d-fa06-4f85-b3d2-4e888c99e621
dase-local@10-24-21-197:~$
```

图 9: 执行 WordCount 任务

使用 Web UI 查看信息

Worker ID	Address	State	Cores	Memory
worker-20210421142205-10.24.21.197-29483	10.24.21.197:29483	ALIVE	4 (0 Used)	6.6 GB (0.0 B Used)

Application ID	Name	Cores	Memory per Executor	Submitted Time	User	State	Duration
app-20210421143253-0000	WordCountJava	4	1024 MB	2021/04/21 14:32:53	dase-local	FINISHED	3 s

图 10: 通过 Web UI 查看 WordCount 任务

```
dase-local@10-24-21-197:~$ ~/spark-2.4.7/bin/spark-submit --master spark://localhost:7077 -class cn.edu.ecnu.spark.example.java.WordCount /home/dase-local/myApp/WordCount.jar hdfs://localhost:9000/user/dase-local/spark_input hdfs:9000/user/dase-local/spark_output
21/04/21 14:32:52 WARN util.Utils: Your hostname, 10-24-21-197 resolves to a loopback address: 127.0.1.1; using 10.24.21.197 instead (on interface eth0)
21/04/21 14:32:52 WARN util.Utils: Set SPARK_LOCAL_IP if you need to bind to another address
21/04/21 14:32:52 INFO spark.SparkContext: Running Spark version 2.4.7
21/04/21 14:32:52 INFO spark.SparkContext: Submitted application: WordCountJava
21/04/21 14:32:52 INFO spark.SecurityManager: Changing view acls to: dase-local
21/04/21 14:32:52 INFO spark.SecurityManager: Changing modify acls to: dase-local
21/04/21 14:32:52 INFO spark.SecurityManager: Changing view acls groups to:
21/04/21 14:32:52 INFO spark.SecurityManager: Changing modify acls groups to:
21/04/21 14:32:52 INFO spark.SecurityManager: SecurityManager: authentication disabled; users with view permissions: Set(dase-local); groups with view permissions: Set(); users with modify permissions: Set(dase-local); groups with modify permissions: Set()
21/04/21 14:32:53 INFO util.Utils: Successfully started service 'sparkDriver' on port 31813.
```

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

The screenshot shows the Spark History Server interface at `localhost:18080`. It displays a table of application logs. The table has columns: App ID, App Name, Started, Completed, Duration, Spark User, Last Updated, and Event Log. A single row is shown for the application `app-20210421143253-0000`, which is `WordCountJava`. The application started at `2021-04-21 14:32:52` and completed at `2021-04-21 14:32:56`, with a duration of `4 s`. The spark user is `dase-local`, and it was last updated at `2021-04-21 14:32:56`. There is a `Download` button next to the last update timestamp.

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

The screenshot shows the HDFS browser interface at `localhost:50070/explorer.html#/user/dase-local/spark`. It displays a list of files in the directory `/user/dase-local/spark_input`. There are two files listed: `RELEASE` and `pd.train`. Both files have a size of `128 MB` and were modified on `Apr 08 16:22` and `Apr 21 14:24` respectively. The `pd.train` file is selected, and its details are shown in a modal window. The modal shows the file's name is `pd.train`, it is a `text/plain` file, and it is located in the `spark_input` directory. It also provides download links for the first 32K and the last 32K of the file.

图 13: 查看任务输出

接着，检查 WordCount 任务的输出

The screenshot shows the HDFS browser interface at `localhost:50070/explorer.html#/user/dase-local/spark`. It displays a modal window titled "File information - pd.train". The modal shows detailed information about the `pd.train` file, including its block information. The modal lists the following details:

- Block ID: `1073741952`
- Block Pool ID: `BP-1880313670-127.0.1.1-1616424037666`
- Generation Stamp: `1131`
- Size: `134217728`
- Availability: `10-23-203-76`

图 14: 检查 WordCount 任务的输出

最后，依次关闭 HDFS、Spark 和 HistoryServer

```
dase-local@10-24-21-197:~/softwares/hadoop-2.10.1/sbin/stop-dfs.sh
Stopping namenodes on [localhost]
localhost: stopping namenode
localhost: stopping datanode
Stopping secondary namenodes [0.0.0.0]
0.0.0.0: stopping secondarynamenode
dase-local@10-24-21-197:~/spark-2.4.7/sbin/stop-all.sh
localhost: stopping org.apache.spark.deploy.worker.Worker
stopping org.apache.spark.deploy.Master
dase-local@10-24-21-197:~/spark-2.4.7/sbin/stop-history-server.sh
stopping org.apache.spark.deploy.history.HistoryServer
dase-local@10-24-21-197:~
```

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
```

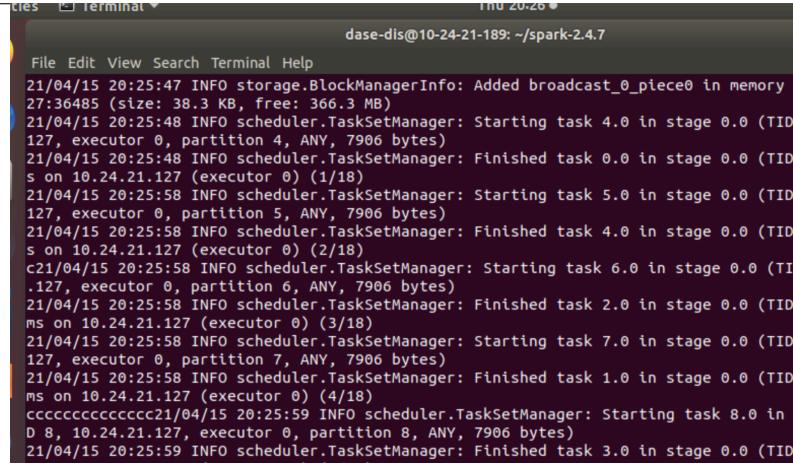
和单机模式一样！务必不要自己创建输出目录！建议在执行前先尝试删除 spark_output 文件夹

```
dase-dis@10-24-21-189: ~/spark-2.4.7
File Edit View Search Terminal Help
(MapPartitionsRDD[3] at mapToPair at WordCount.java:37) (first 15 tasks are for partitions Vector(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14))
21/04/15 20:25:46 INFO scheduler.TaskSchedulerImpl: Adding task set 0.0 with 18 tasks
21/04/15 20:25:46 INFO cluster.CoarseGrainedSchedulerBackend$DriverEndpoint: Registered executor NettyRpcEndpointRef(spark-client://Executor) (10.24.21.127:36282) with ID 0
21/04/15 20:25:46 INFO scheduler.TaskSetManager: Starting task 0.0 in stage 0.0 (TID 0, 10.24.21.127, executor 0, partition 0, ANY, 7905 bytes)
21/04/15 20:25:46 INFO scheduler.TaskSetManager: Starting task 1.0 in stage 0.0 (TID 1, 10.24.21.127, executor 0, partition 1, ANY, 7906 bytes)
21/04/15 20:25:46 INFO scheduler.TaskSetManager: Starting task 2.0 in stage 0.0 (TID 2, 10.24.21.127, executor 0, partition 2, ANY, 7906 bytes)
21/04/15 20:25:46 INFO scheduler.TaskSetManager: Starting task 3.0 in stage 0.0 (TID 3, 10.24.21.127, executor 0, partition 3, ANY, 7906 bytes)
21/04/15 20:25:46 INFO storage.BlockManagerMasterEndpoint: Registering block manager 10.24.21.127:36485 with 366.3 MB RAM, BlockManagerId(0, 10.24.21.127, 36485, None)
21/04/15 20:25:47 INFO storage.BlockManagerInfo: Added broadcast_1_piece0 in memory on 10.24.21.127:36485 (size: 3.2 KB, free: 366.3 MB)
21/04/15 20:25:47 INFO storage.BlockManagerInfo: Added broadcast_0_piece0 in memory on 10.24.21.127:36485 (size: 38.0 KB, free: 366.3 MB)
21/04/15 20:25:48 INFO scheduler.TaskSetManager: Starting task 4.0 in stage 0.0 (TID 4, 10.24.21.127, executor 0, partition 4, ANY, 7906 bytes)
21/04/15 20:25:48 INFO scheduler.TaskSetManager: Finished task 0.0 in stage 0.0 (TID 0) in 1654 ms on 10.24.21.127 (executor 0) (1/18)
```

图 16: 分布式环境提交任务

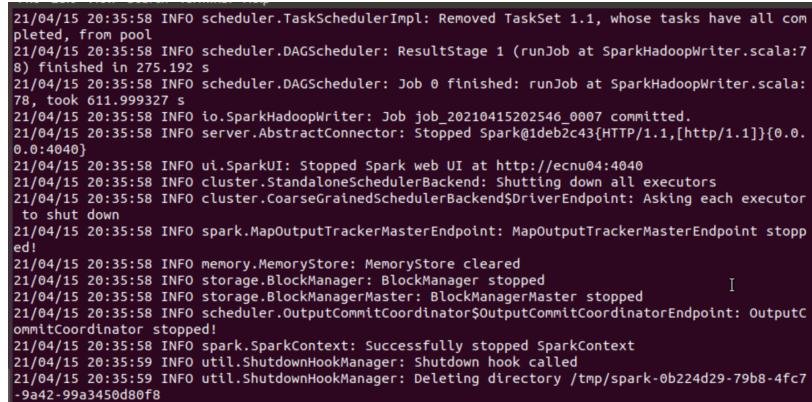
接下来可以查看任务的执行过程

华东师范大学数据科学与工程学院学生实验报告



```
dase-dis@10-24-21-189: ~/spark-2.4.7
File Edit View Search Terminal Help
21/04/15 20:25:47 INFO storage.BlockManagerInfo: Added broadcast_0_piece0 in memory on 27:36485 (size: 38.3 KB, free: 366.3 MB)
21/04/15 20:25:48 INFO scheduler.TaskSetManager: Starting task 4.0 in stage 0.0 (TID 127, executor 0, partition 4, ANY, 7906 bytes)
21/04/15 20:25:48 INFO scheduler.TaskSetManager: Finished task 0.0 in stage 0.0 (TID 0 on 10.24.21.127 (executor 0) (1/18)
21/04/15 20:25:58 INFO scheduler.TaskSetManager: Starting task 5.0 in stage 0.0 (TID 127, executor 0, partition 5, ANY, 7906 bytes)
21/04/15 20:25:58 INFO scheduler.TaskSetManager: Finished task 4.0 in stage 0.0 (TID 0 on 10.24.21.127 (executor 0) (2/18)
21/04/15 20:25:58 INFO scheduler.TaskSetManager: Starting task 6.0 in stage 0.0 (TID 127, executor 0, partition 6, ANY, 7906 bytes)
21/04/15 20:25:58 INFO scheduler.TaskSetManager: Finished task 2.0 in stage 0.0 (TID 0 on 10.24.21.127 (executor 0) (3/18)
21/04/15 20:25:58 INFO scheduler.TaskSetManager: Starting task 7.0 in stage 0.0 (TID 127, executor 0, partition 7, ANY, 7906 bytes)
21/04/15 20:25:58 INFO scheduler.TaskSetManager: Finished task 1.0 in stage 0.0 (TID 0 on 10.24.21.127 (executor 0) (4/18)
ccccccccccccc21/04/15 20:25:59 INFO scheduler.TaskSetManager: Starting task 8.0 in stage 0.0 (TID 8, 10.24.21.127, executor 0, partition 8, ANY, 7906 bytes)
21/04/15 20:25:59 INFO scheduler.TaskSetManager: Finished task 3.0 in stage 0.0 (TID 0 on 10.24.21.127 (executor 0) (5/18)
```

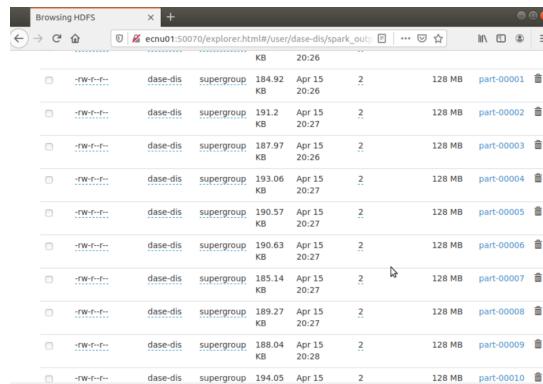
图 17: 分布式环境任务执行



```
21/04/15 20:35:58 INFO scheduler.TaskSchedulerImpl: Removed TaskSet 1.1, whose tasks have all completed, from pool
21/04/15 20:35:58 INFO scheduler.DAGScheduler: ResultStage 1 (runJob at SparkHadoopWriter.scala:78) finished in 275.192 s
21/04/15 20:35:58 INFO scheduler.DAGScheduler: Job 0 finished: runJob at SparkHadoopWriter.scala:78, took 611.999327 s
21/04/15 20:35:58 INFO io.SparkHadoopWriter: Job job_20210415202546_0007 committed.
21/04/15 20:35:58 INFO server.AbstractConnector: Stopped Spark@idebZc43{[HTTP/1.1]}{0.0.0.0:4040}
21/04/15 20:35:58 INFO ui.SparkUI: Stopped Spark web UI at http://ecnu04:4040
21/04/15 20:35:58 INFO cluster.StandaloneSchedulerBackend: Shutting down all executors
21/04/15 20:35:58 INFO cluster.CoarseGrainedSchedulerBackend$DriverEndpoint: Asking each executor to shut down
21/04/15 20:35:58 INFO spark.MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint stopped!
21/04/15 20:35:58 INFO memory.MemoryStore: MemoryStore cleared
21/04/15 20:35:58 INFO storage.BlockManager: BlockManager stopped
21/04/15 20:35:58 INFO storage.BlockManagerMaster: BlockManagerMaster stopped
21/04/15 20:35:58 INFO scheduler.OutputCommitCoordinator$OutputCommitCoordinatorEndpoint: OutputCommitCoordinator stopped!
21/04/15 20:35:58 INFO spark.SparkContext: Successfully stopped SparkContext
21/04/15 20:35:59 INFO util.ShutdownHookManager: Shutdown hook called
21/04/15 20:35:59 INFO util.ShutdownHookManager: Deleting directory /tmp/spark-0b224d29-79b8-4fc7-9a42-99a3450d80f8
```

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



			KB	20:26		
□	-rw-r--r--	dase-dis	supergroup	184.92 Apr 15 20:26	2	128 MB part-00001
□	-rw-r--r--	dase-dis	supergroup	191.2 KB Apr 15 20:27	2	128 MB part-00002
□	-rw-r--r--	dase-dis	supergroup	187.97 KB Apr 15 20:26	2	128 MB part-00003
□	-rw-r--r--	dase-dis	supergroup	193.06 KB Apr 15 20:27	2	128 MB part-00004
□	-rw-r--r--	dase-dis	supergroup	190.57 KB Apr 15 20:27	2	128 MB part-00005
□	-rw-r--r--	dase-dis	supergroup	190.63 KB Apr 15 20:27	2	128 MB part-00006
□	-rw-r--r--	dase-dis	supergroup	185.14 KB Apr 15 20:27	2	128 MB part-00007
□	-rw-r--r--	dase-dis	supergroup	189.27 KB Apr 15 20:27	2	128 MB part-00008
□	-rw-r--r--	dase-dis	supergroup	188.04 KB Apr 15 20:28	2	128 MB part-00009
□	-rw-r--r--	dase-dis	supergroup	194.05 KB Apr 15 20:28	2	128 MB part-00010

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

Stages for All Jobs

[Completed Stages: 2](#)

[▼ Completed Stages \(2\)](#)

Stage Id	Description	Submitted	Duration	Tasks: Succeeded/Total	Input	Output	Shuffle Read	Shuffle Write
1	runJob at SparkHadoopWriter.scala:78 +details	2021/04/22 14:58:32	3.9 min	18/18		3.3 MB	400.6 MB	
0	mapToPair at WordCount.java:37 +details	2021/04/22 14:57:50	41 s	18/18	2.0 GB			400.6 MB

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

Details for Stage 0 (Attempt 0)

Total Time Across All Tasks: 2.6 min
Locality Level Summary: Any: 18
Input Size / Records: 2.0 GB / 12655585
Shuffle Write: 400.6 MB / 389987949

[▼ DAG Visualization](#)

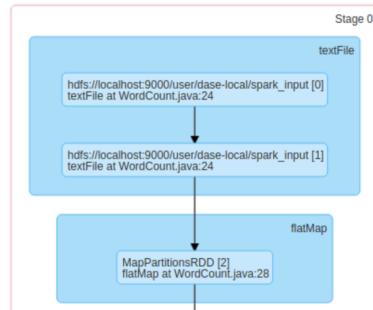


图 21: 思考题 2-1

Details for Stage 1 (Attempt 0)

Total Time Across All Tasks: 15 min
Locality Level Summary: Node local: 18
Output: 3.3 MB / 247204
Shuffle Read: 400.6 MB / 389987949
Shuffle Spill (Memory): 1387.8 MB
Shuffle Spill (Disk): 155.4 MB

图 22: 思考题 2-2

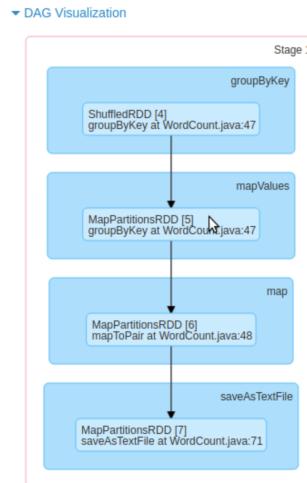


图 23: 思考题 2-3

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。

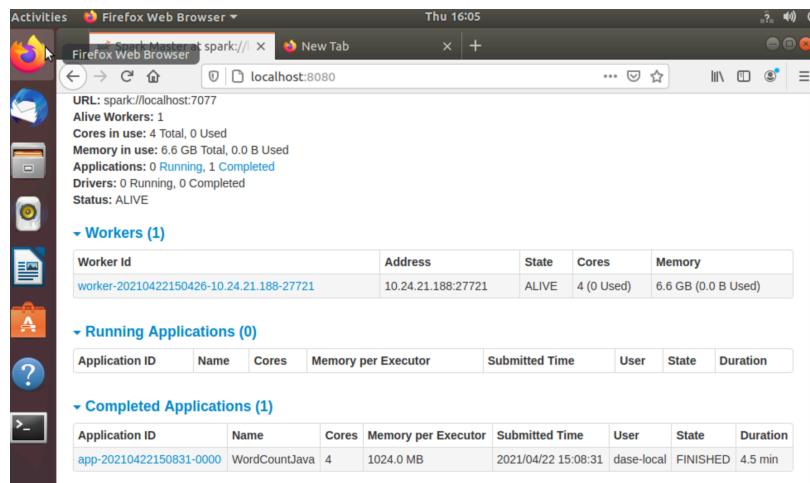


图 24: 单机伪分布式部署：时间开销

```

File Edit View Search Terminal Help
dase-dis@10-23-138-4:~
cale:78) finished in 213.201 s
21/04/22 16:28:22 INFO scheduler.DAGScheduler: Job 0 finished: runJob at SparkHadoopWriter.
scale:78, took: 235.637170 s
21/04/22 16:28:22 INFO io.SparkHadoopWriter: Job job_20210422162427_0007 committed.
21/04/22 16:28:22 INFO server.AbstractConnector: Stopped Spark@1deb2c43[HTTP/1.1,[http/1.1]
]{0.0.0.0:4040}
21/04/22 16:28:22 INFO ui.SparkUI: Stopped Spark web UI at http://ecnu04:4940
21/04/22 16:28:22 INFO cluster.StandaloneSchedulerBackend: Shutting down all executors
21/04/22 16:28:22 INFO cluster.CoarseGrainedSchedulerBackendDriverEndpoint: Asking each executor to shut down
21/04/22 16:28:22 INFO spark.MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint stopped!
21/04/22 16:28:22 INFO memory.MemoryStore: MemoryStore cleared
21/04/22 16:28:22 INFO storage.BlockManager: BlockManager stopped
21/04/22 16:28:22 INFO storage.BlockManagerMaster: BlockManagerMaster stopped
21/04/22 16:28:22 INFO scheduler.OutputCommitCoordinator$OutputCommitCoordinatorEndpoint: OutputCommitCoordinator stopped!
21/04/22 16:28:22 INFO spark.SparkContext: Successfully stopped SparkContext
21/04/22 16:28:23 INFO util.ShutdownHookManager: Shutdown hook called
21/04/22 16:28:23 INFO util.ShutdownHookManager: Deleting directory /tmp/spark-79000c14-46f
lb-4711-bb35-70771135b635
21/04/22 16:28:23 INFO util.ShutdownHookManager: Deleting directory /tmp/spark-8d48b606-62c
0-4dd6-9bc9-451d3911c8a6
dase-dis@10-23-138-4:~$
```

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。](#)）

```

dase-dis@10-23-190-97:~$ ./hadoop-2.10.1/bin/dfsadmin -safemode -leave
Usage: hdfs dfsadmin [-safemode enter | leave | get | wait | forceExit]
dase-dis@10-23-190-97:~$ ./hadoop-2.10.1/bin/dfsadmin -safemode leave
Safe mode is OFF
dase-dis@10-23-190-97:~$
```

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](#)