分布式系统 Homework 2

张言健 16300200020

目录

1 介绍短文

Hadoop 为 MapReduce 提供了不同的 API，可以方便我们使用不同的编程语言来使用 MapReduce 框架，而不是只局限于 Java。这里要介绍的就是 Hadoop streaming API。Hadoop streaming 使用 Unix 的 standard streams 作为我们 mapreduce 程序和 MapReduce 框架之间的接口。所以可以用任何语言来编写 MapReduce 程序，只要该语言可以往 standard input/output 上进行读写。

streamming 是天然适用于文字处理的（text processing），当然，也仅适用纯文本的处理，对于需要对象和序列化的场景，hadoop streaming 无能为力。它力图使我们能够快捷的通过各种脚本语言，快速的处理大量的文本文件。以下是 steaming 的一些特点：

同时，Hadoop Streaming 还提供了足够多的参数控制，使得使用者直接通过 streaming 参数，而不需要使用 java 语言修改。很多 mapreduce 的高阶功能，都可以通过 steaming 参数的调整来完成。

而 Mapper 和 reducer 的前后都要进行标准输入和标准输出的转化，涉及数据拷贝和解析，带来了一定的开销。所以 Hadoop Streaming 比较适合做一些简单的任务，比如用 Python 写只有一两百行的脚本。如果项目比较复杂，或者需要进行比较细致的优化，则适宜用 JAVA 来处理。

2 实验报告

2.1 实验设置

操作系统	Ubuntu 18.04
JDK	java 1.8.0.201
Hadoop	3.2.0

2.2 基于 Hadoop streaming 的单词计数程序

2.2.1 测试文件和程序代码

测试文件 word.txt

```
张三 李四
黄二 小牛
张三 小牛
李四 黄一
```

测试程序 map.py

```
import sys

```

```
for line in sys.stdin:
    ss = line.strip().split(' ')
    for word in ss:
        print('\t'.join([word.strip(),"1"]))
```

测试程序 reduce.py

```
import sys

cur_word = None
sum=0

for line in sys.stdin:
    word,cnt = line.strip().split('\t')

    if cur_word == None:
        cur_word = word
    if cur_word!=word:
        print('\t'.join([cur_word,str(sum)]))
        cur_word = word
        sum=0
    sum+=int(cnt)
```

2.2.2 测试文件运行结果

输出 map 运行结果

```
$ cat word.txt | python3 map.py

张三    1
李四    1
黄二    1
小牛    1
张三    1
小牛    1
李四    1
黄一    1
```

使用 sort 模拟排序过程，输出 reduce 运行结果

```
$ cat word.txt | python3 map.py | sort -k 1 | python3 reduce.py

```

```
黄二    1
黄一    1
李四    2
小牛    2
```

2.2.3 使用 hadoop-streaming

在当前目录下创建 run.sh 脚本

```
HADOOP_CMD="/usr/local/hadoop/bin/hadoop"
STREAM_JAR_PATH="/usr/local/hadoop/share/hadoop/tools/lib/hadoop-streaming
    -3.2.0.jar"
INPUT_FILE_PATH_1="input" #设置输入文件
OUTPUT_PATH="output" #设置输出文件
$HADOOP_CMD fs -rmr -skipTrash $OUTPUT_PATH #删除原有输出文件
#使用streaming框架
$HADOOP_CMD jar $STREAM_JAR_PATH \
-input $INPUT_FILE_PATH_1 \
-output $OUTPUT_PATH \
-mapper "python3 map.py" \
-reducer "python3 reduce.py" \
-file ./map.py \
-file ./reduce.py
```

将文件传输到 hdfs 中

```
start-dfs.sh
start-yarn.sh
mapred --daemon start historyserver
hadoop fs -put word.txt input  #将word.txt传输到hdfs的input中
```

运行 run.sh 脚本，所得结果如图1，显示成功结束

```
bash run.sh
```

将结果导出，输出到屏幕，结果如图2，与未使用 hadoop 的结果一致。

2.3 项目细节

BUG 1. 找不到或无法加载主类 org.apache.hadoop.mapreduce.v2.app.MRAppMaster，根据所附参考链接，删除 hdfs/data 下的 current 文件夹，再重启 HDFS 即可

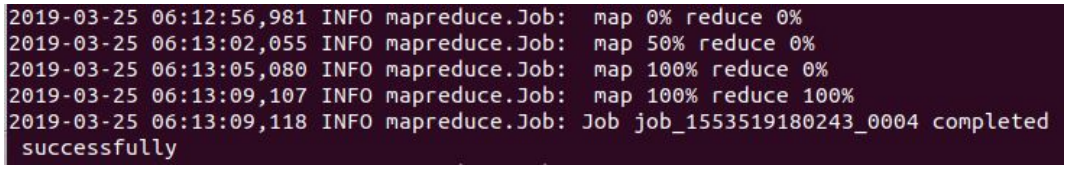

图 1: 运行结束

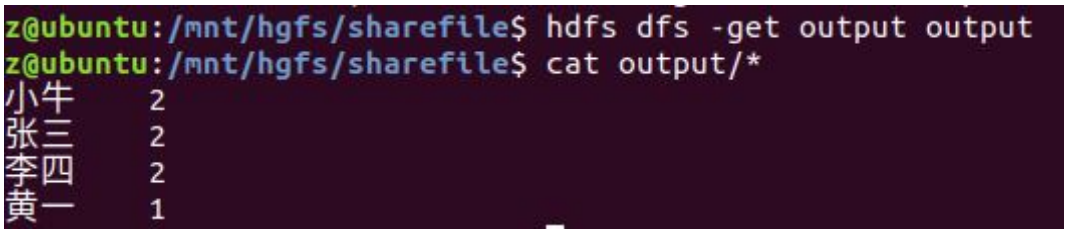

图 2: 输出结果

BUG 2. MapReduce job hangs, waiting for AM container to be allocated，我依据所附参考链接对 yarn-site.xml 进行了修改

BUG 3. Healthy Status 中发现 can not open /tmp/nm-local-dir/, 我使用 chmod 777 tmp，让其变得可访问

3 参考资料

Hadoop Streaming 研究短文:

1. Hadoop Streaming 详解
2. Hadoop streaming 详细介绍

基于 Hadoop streaming 实现单词计数程序:

1. 使用 hadoop-streaming 初体验 mapreduce
2. Linux 上传本地文件到 HDFS
3. hadoop 找不到或无法加载主类 org.apache.hadoop.mapreduce.v2.app.MRAppMaster
4. MapReduce job hangs, waiting for AM container to be allocated