《大规模分布式系统》实验报告

——Hadoop Streaming

姓名：刘佰川 专业：计算机科学与技术（数据科学方向）学号：16307130214

0. 实验环境

VMware+Ubuntu18.04+Hadoop2.9.1+Java1.8.0_201

1. 实验要求

I. 在网上查找资料，撰写短文介绍 Hadoop streaming

II. 基于 Hadoop streaming 机制 用 Python 或 Java 编写一个 MapReduce 的单词计数程序：对多个文本串进行单词词频的统计

2. 实验过程

I. Hadoop Streaming 简介

Hadoop 本身是用 Java 开发的，程序也需要用 Java 编写，但是通过 Hadoop Streaming，我们可以使用任意语言来编写程序，其是 Hadoop 的一个工具，它帮助用户创建和运行一类特殊的 map/reduce 作业，这些特殊的 map/reduce 作业是由一些可执行文件或脚本文件充当 mapper 或者 reducer。简单的说，hadoop-streaming 是一个框架，可以让任何语言编写的 map-reduce 程序都能在 hadoop 上运行，只要遵循标准的输入 stdin 和输出 stdout 即可。

Streaming 的工作原理：如果一个可执行文件被用于 mapper，则在 mapper 初始化时，每一个 mapper 任务会把这个可执行文件作为一个单独的进程启动。Mapper 任务运行时，它把输入切分成行并把每一行提供给可执行文件进程的标准输入。同时，mapper 收集可执行文件进程标准输出的内容，并把收到的每一行内容转化成 key/value 对，作为 mapper 的输出。默认情况下，一行中第一个 tab 之前的部分作为 key，之后的（不包括 tab）作为 value。如果没有 tab，整行作为 key 值，value 值为 null。如果一个可执行文件被用于 reducer，每个 reducer 任务会把这个可执行文件作为一个单独的进程启动。Reducer 任务运行时，它把输入切分成行并把每一行提供给可执行文件进程的标准输入。同时，

reducer 收集可执行文件进程标准输出的内容，并把每一行内容转化成 key/value 对，作为 reducer 的输出。

Hadoop Streaming 的使用方式：$HADOOP_HOME/bin/hadoop jar $HADOOP_HOME/.../hadoop-streaming.jar [genericOptions] [streamingOptions]。在这行命令中，有先后顺序，一定要保证[genericOptions]写在[streamingOptions]之前，否则 hadoop streaming 命令将失效。下面介绍解释一些常用的 genericOptions 和 streamingOptions。

常用的 genericOptions 如下：

-D property=value 指定额外的配置信息变量

-files file1,file2,... 指定需要拷贝到集群节点的文件，格式以逗号分隔，通常为我们自己编写的 mapper 和 reducer 脚本或可执行程序，因为你的 mapper 和 reducer 通常要由集群中不同的节点来执行，而很可能你的脚本或可执行程序仅仅存在于你提交任务时所用的那个节点，因此遇到这种情况便需要将它们分发出去，-files 其后的参数用不用引号括起来都可以。

常用的 streamingOptions 如下：

-file filename 指定需要拷贝到集群节点的文件，与-files 的功能类似，只不过如果使用-file 的话，就需要一个文件一个文件地去上传，比方说如果我要将我的 mapper.py，reducer.py 上传到集群上去运行，那就得需要两个-file 参数。而在实际使用-file 时，hadoop 似乎并不希望我们使用-file 参数，比如如下这条 warning。“18/03/26 20:17:40 WARN streaming.StreamJob: -file option is deprecated, please use generic option -files instead.”

-input myInputDirs 指定给 mapreduce 任务输入的文件位置，通常为 hdfs 上的文件路径，多个文件或目录用逗号隔开

-output myOutputDir 指定给 mapreduce 任务输出的目录，通常为 hdfs 上的文件路径。

-mapper executable or JavaClassName 用于 mapper 的可执行程序或 java 类，如果是脚本文件，应该以命令行完整调用的格式作为可执行程序参数并且须加引号，比如-mapper "python mapper.py" \

-reducer executable or JavaClassName 用于 reducer 的可执行程序或 java 类，

要求同上

-partitionerJavaClassName 自定义的 partitionerjava 类

-combiner streamingCommandor JavaClassName 自定义的 combiner 类或命令

下面解释一些常见的命令操作。

使用自定义的方法切分行来形成 Key/Value 对：“-jobconf stream.map.output.field.separator=.”指定“.”作为 map 输出内容的分隔符，并且从在第四个 “.”之前的部分作为 key，之后的部分作为 value（不包括这第四个 “.”）。如果一行中的“.”少于四个，则整行的内容作为 key，value 设为空的 Text 对象（就像这样创建了一个 Text：new Text("")）。

一个实用的 Partitioner 类（二次排序，-partitioner org.apache.hadoop.mapred.lib.KeyFieldBasedPartitioner 选项），一个实用的 Partitioner 类（二次排序，-partitioner org.apache.hadoop.mapred.lib. KeyFieldBasedPartitioner 选项）。对 key 切分了主键和副键，主键用于切分块，主键和副键的组合用于排序。

II. 编写 MapReduce 的单词计数程序

输入 mapper 的文本每一行是一行文本。

```
hello world
hello hadoop hadoop
nihao world
hello mapreduce
~
```

即此行文本为 key，而 value 的值为空。

利用 python 编写 mapper 和 reducer 函数。

```
#!/usr/bin/python
import sys

for line in sys.stdin:
    line = line.strip()
    words = line.split(' ')
    for word in words:
        print('%s\t%s'%(word,1))
~
~
```

```
#!/usr/bin/python
import sys

current_count = 0
current_word = None

for line in sys.stdin:
    line = line.strip()
    word,count = line.split('\t',1)
    count = int(count)
    if current_word == word:
        current_count += count
    else:
        if current_word:
            print("%s\t%s" % (current_word,current_count))

        current_count = count
        current_word = word
~
~
~
```

先进行本地调试

```
junjin@ubuntu:~$ cat /home/junjin/input.txt | ./mapper.py | sort | ./reducer.py

hadoop  2
hello   3
mapreduce       1
nihao   1
```

现在开始尝试用 hadoop streaming 执行程序。

创建一个新的文件夹用于存放输入文件，假设叫 input。

```
junjin@ubuntu:/usr/local/hadoop/bin$ ./hadoop fs -mkdir input
```

将本地文件放到 input 文件夹下。

```
junjin@ubuntu:/usr/local/hadoop/bin$ ./hadoop fs -copyFromLocal /home/junjin/in
put.txt input/
```

运行

```
junjin@ubuntu:/usr/local/hadoop/bin$ ./hadoop jar /usr/local/hadoop/share/hadoo
p/tools/lib/hadoop-streaming-2.9.1.jar -mapper 'python mapper.py' -file /home/j
unjin/mapper.py -reducer 'python reducer.py' -file /home/junjin/reducer.py -inp
ut input/* -output outputs3
```

打开输出文件夹查看运行结果

```
junjin@ubuntu:/usr/local/hadoop/bin$ ./hadoop fs -ls outputs3
Found 2 items
-rw-r--r--   1 junjin supergroup          0 2019-03-18 07:45 outputs3/_SUCCESS
-rw-r--r--   1 junjin supergroup         37 2019-03-18 07:45 outputs3/part-0000
0
junjin@ubuntu:/usr/local/hadoop/bin$ ./hadoop fs -cat outputs3/part-00000
hadoop  2
hello   3
mapreduce       1
nihao   1
```

成功执行完这个任务之后，你用 output 参数在 HDFS 上指定的输出文件夹里就会多出几个文件，一个空白文件_SUCCESS，表明 job 运行成功，这个文件可以让其他程序只要查看一下 HDFS 就能判断这次 job 是否成功运行，从而

进行相关处理。part-00000, ... part-xxxxx 文件，有多少个 reducer 后面的数字就会有多大，对应每个 reducer 的输出结果。

实验完成，运行成功。

3. 实验问题

在运行 hadoop 时，一开始直接输入 hadoop 被提示命令行不存在，因为 hadoop 是一个文件，故后输入./hadoop 才成功运行。

在 hadoop 中一开始不会查看 HDFS 下的文件，查询后知道要用./Hadoop fs 和一些 linux 系统中文件操作的命令一起运行进行 HDFS 下的文件操作。

前几次因为文件输入有问题导致运行不成功，但是 hadoop 却在 HDFS 创建了此次运行的 output 文件。而 reducer 输出时，为了避免覆盖文件，此时的输出文件是不能存在的。所以在后面运行的时候会报错，此时需要删除之前的 output 文件夹或者重新创建一个新的 output 文件。

4. 参考资料

http://www.uml.org.cn/sjjm/201512111.asp

https://blog.csdn.net/m0_37637511/article/details/80329552

https://zhuanlan.zhihu.com/p/34903460

https://hadoop.apache.org/docs/r1.2.1/streaming.html