实验报告：WordCount实验

实验目的

本实验旨在通过MapReduce框架实现经典的WordCount任务，即统计给定文本中每个单词的出现次数。通过此实验，我们将深入了解MapReduce的工作原理，并掌握如何编写Mapper和Reducer函数以及运行MapReduce作业的方法。

实验环境

- 操作系统：Ubuntu 20.04 LTS
- Hadoop版本：Hadoop 3.4.0
- 编程语言：Python 3.10.1

MapReduce组成

> MapReduce 把任务分为 Map 阶段和 Reduce 阶段。开发人员使用存储在HDFS 中数据（可实现快速存储），编写 Hadoop 的 MapReduce 任务。由于 MapReduce工作原理的特性， Hadoop 能以并行的方式访问数据，从而实现快速访问数据

MapReduce体系结构主要由四个部分组成，分别是：Client、JobTracker、TaskTracker以及Task

1. Client 用户编写的MapReduce程序通过Client提交到JobTracker端 用户可通过Client提供的一些接口查看作业运行状态

2. JobTracker JobTracker是MapReduce作业的主管，负责作业的调度、监控和管理。它的主要作用包括：
    - 资源调度：JobTracker负责监控集群中各个节点的资源使用情况，并根据作业的需求进行资源调度，确保作业能够顺利执行。
    - 作业调度：JobTracker接收Client提交的作业，并将作业分解成一个个任务（Task），然后将这些任务分配给各个TaskTracker执行。
    - 任务监控：JobTracker负责跟踪任务的执行状态、资源使用情况等信息，并在任务失败或超时时进行处理，例如重新分配任务或标记任务为失败。
    - 故障恢复：JobTracker会周期性地备份作业执行的状态信息，以防止JobTracker自身发生故障时可以快速恢复。
3. TaskTracker TaskTracker是运行在各个数据节点上的守护进程，负责执行由JobTracker分配的任务。它主要有两个作用：
    - 执行Map和Reduce任务：TaskTracker接收JobTracker分配的任务，启动对应的Mapper或Reducer任务，并监控任务的执行情况。一旦任务完成，TaskTracker会将结果通知给JobTracker。
    - 向JobTracker汇报状态：TaskTracker会周期性地向JobTracker汇报自己的状态信息，包括可用资源、运行的任务等。这样，JobTracker可以及时了解到各个节点的负载情况，从而做出更好的调度决策。
4. Task 在MapReduce中，Task是指作业中的一个具体的执行单元，它可以是Map任务或Reduce任务。每个作业都由若干个Map任务和Reduce任务组成。Task的主要作用包括：
    - 执行Map或Reduce逻辑：Map任务负责处理输入数据并生成中间结果，而Reduce任务负责对中间结果进行合并和处理。Task根据作业的配置，执行相应的Map或Reduce逻辑。
    - 处理数据分片：Map任务负责处理输入数据的分片，将其转换为键值对。Reduce任务则负责合并和处理Map任务输出的中间结果。
    - 与TaskTracker通信：Task与TaskTracker之间通过心跳机制进行通信，以便TaskTracker可以监控任务的执行状态，并及时处理异常情况。

MapReduce工作过程

1. 输入阶段 输入数据通常存储在Hadoop分布式文件系统（HDFS）中，可以是一个或多个文件，每个文件可能被分成多个数据块，分布在不同的数据节点上。 在MapReduce作业开始时，输入阶段会从HDFS中并行读取数据，每个数据节点负责读取自己所存储的数据块，并将数据发送给相应的Mapper任务进行处理。
2. 映射阶段（Map Phase） 在映射阶段，每个Mapper任务会将**输入数据按照一定的逻辑拆分成<key, value>对**，并对每个<key, value>对执行一次映射操作。

3. 分区、排序和合并阶段： 在映射阶段输出结果后，MapReduce框架会对Mapper输出的中间结果进行分区和排序，以便将具有相同键的数据合并在一起。 这个过程可以让相同键的数据落到同一个Reducer任务上，减少数据的传输量和提高Reducer的处理效率。
4. 洗牌阶段（Shuffle Phase）： 洗牌阶段是MapReduce的核心步骤之一，**它负责将Mapper输出的中间结果按照键的哈希值范围进行重新分配，以便将具有相同键的数据发送到同一个Reducer任务上**。 这个过程需要进行大量的网络传输和数据交换，因为中间结果需要从各个Mapper节点传输到相应的Reducer节点。
5. 归约阶段（Reduce Phase）： 在归约阶段，每个Reducer任务会接收来自多个Mapper节点的中间结果，对相同键的数据进行归约操作，例如求和、计数等。 归约操作的结果会被写入到输出文件中，形成最终的输出结果。
6. 输出阶段： 最后，输出阶段将Reducer任务的输出结果写入到HDFS中指定的输出目录中，这样就完成了MapReduce作业的执行。 输出结果通常可以是一个或多个文件，可以是文本文件、序列化文件等形式，具体格式由用户指定。

实验步骤

1. 数据准备 我们准备了一个包含文本内容的文件input.txt，内容如下：

```
Hello world
Hello MapReduce
Hello Hadoop
MapReduce is powerful
```

放在HDFS上` hadoop fs -put input.txt /test/`

2. 编写Mapper函数 编写Python脚本mapper.py作为Mapper函数，其代码如下：

```
import sys

# 从标准输入读取数据
for line in sys.stdin:
    # 拆分每行为单词
    words = line.strip().split()
    # 输出每个单词及其出现次数，使用制表符分隔
    for word in words:
        print(f"{word}\t1")
```

3. 编写Reducer函数 编写Python脚本reducer.py作为Reducer函数，其代码如下：

```
import sys

current_word = None
current_count = 0

# 从标准输入读取Mapper函数输出的中间结果
for line in sys.stdin:
    # 解析Mapper函数输出的单词和次数
    word, count = line.strip().split('\t')
    count = int(count)

    # 如果单词与当前处理的单词不同，则输出当前单词的统计结果，并更新当前单词和计数
    if word != current_word:
        if current_word:
            print(f"{current_word}\t{current_count}")
        current_word = word
        current_count = 0

    # 更新当前单词的出现次数
    current_count += count

# 输出最后一个单词的统计结果
if current_word:
    print(f"{current_word}\t{current_count}")
```

4. 运行MapReduce作业 使用Hadoop Streaming来运行WordCount作业，执行以下命令：

```
# 使用此命令找到Streaming JAR文件
find / -name "hadoop-streaming*.jar" 2>/dev/null
```

```
oslab@oslab-virtual-machine:~/Desktop$ find / -name "hadoop-streaming*.jar" 2>/d
ev/null
/usr/local/hadoop/share/hadoop/tools/sources/hadoop-streaming-3.4.0-test-sources
.jar
/usr/local/hadoop/share/hadoop/tools/sources/hadoop-streaming-3.4.0-sources.jar
/usr/local/hadoop/share/hadoop/tools/lib/hadoop-streaming-3.4.0.jar
```

以下这行命令使用了Hadoop Streaming接口来运行MapReduce作业

- hadoop：这是Hadoop的命令行工具，用于执行各种Hadoop相关的操作。
- jar：这个命令指示Hadoop执行一个Java JAR文件，即hadoop-streaming.jar，它包含了Hadoop Streaming所需的代码和依赖项。

- hadoop-streaming.jar：这是Hadoop Streaming JAR文件的路径。该文件包含了Hadoop Streaming的代码和依赖项，用于运行MapReduce作业。
- -mapper 'python mapper.py'：这个参数指定了Mapper函数的执行方式。python mapper.py告诉Hadoop使用Python解释器来执行名为mapper.py的脚本，这个脚本实现了Mapper函数的逻辑。
- -reducer 'python reducer.py'：这个参数指定了Reducer函数的执行方式。同样，python reducer.py告诉Hadoop使用Python解释器来执行名为reducer.py的脚本，这个脚本实现了Reducer函数的逻辑。
- -input input.txt：这个参数指定了输入数据的路径。在这个例子中，输入数据是一个名为input.txt的文本文件，它位于Hadoop文件系统中。
- -output wordcount_output：这个参数指定了输出结果的路径。在这个例子中，输出结果将被写入到一个名为wordcount_output的目录中，它位于Hadoop文件系统中

```
hadoop jar /usr/local/hadoop/share/hadoop/tools/lib/hadoop-streaming-3.4.0.jar \
  -mapper 'python3 ./mapper.py' \
  -reducer 'python3 ./reducer.py' \
  -input /test/input.txt \
  -output /test/wordcount_output
```

```
                Reduce output records=6
                Spilled Records=18
                Shuffled Maps =1
                Failed Shuffles=0
                Merged Map outputs=1
                GC time elapsed (ms)=57
                Total committed heap usage (bytes)=419430400
        Shuffle Errors
                BAD_ID=0
                CONNECTION=0
                IO_ERROR=0
                WRONG_LENGTH=0
                WRONG_MAP=0
                WRONG_REDUCE=0
        File Input Format Counters
                Bytes Read=63
        File Output Format Counters
                Bytes Written=53
2024-06-08 19:18:08,683 INFO streaming.StreamJob: Output directory: /test/wordco
unt_output
oslab@oslab-virtual-machine:~/Desktop$ hdfs dfs -ls /test
2024-06-08 19:18:51,379 WARN util.NativeCodeLoader: Unable to load native-hadoop
 library for your platform... using builtin-java classes where applicable
Found 4 items
-rw-r--r--   1 oslab supergroup         63 2024-06-08 19:05 /test/input.txt
-rw-r--r--   1 oslab supergroup      34625 2024-06-08 18:09 /test/large.txt
-rw-r--r--   1 oslab supergroup          0 2024-06-08 17:14 /test/test.txt
drwxr-xr-x   - oslab supergroup          0 2024-06-08 19:18 /test/wordcount_outp
ut
```

- 查看结果 作业执行完成后，我们可以在输出目录wordcount_output中找到结果文件part-00000，其中包含了每个单词及其出现次数的统计结果。

实验结果

下面是作业执行后得到的统计结果

实验总结

通过本次实验，我成功地使用MapReduce框架实现了WordCount任务，并得到了每个单词的出现次数统计结果。深入了解了MapReduce的工作原理，掌握了编写Mapper和Reducer函数的方法，并学会了如何在Hadoop环境下运行MapReduce作业。这些知识和技能对于进一步学习和应用大数据处理技术具有重要意义。