实验二：Hadoop并行编程

1. 实验目的

* 学习Hadoop分布式文件系统、MapReduce编程模型，理解MapReduce的思想。
* 学习使用Hadoop基本命令，使用MapReduce编写简单的并⾏行程序。
* 编写WordCount程序，对样例数据进行词频统计。

1. 任务与要求

* 了解分布式文件系统原理，以及分布式文件系统与本地文件系统的区别。
* 掌握Hadoop分布式文件系统-ls、-copyFromLocal、-copyToLocal等指令的操作。
* 使用MapReduce编程模型编写词频统计程序并运行。
* 设计并实现一个简单的分布式文件系统。

1. 实验原理

3.1． MapReduce理论简介

3.1.1． MapReduce编程模型

MapReduce采用”分而治之"的思想，把对大规模数据集的操作，分发给一个主节点管理下的各个分节点共同完成，然后通过整合各个节点的中间结果，得到最终结果。简单地说，MapReduce就是"任务的分解与结果的汇总”。

在Hadoop中，用于执行MapReduce任务的机器角色有两个: 一个是JobTracker;另一个是TaskTracker, JobTracker是用于调度工作的，TaskTracker是 用于执行工作的。一个Hadoop集群中只有一台JobTracker。

在分布式计算中，MapReduce框架负责处理了并行编程中分布式存储、工作调度、负载均衡、容错均衡、容错处理以及网络通信等复杂问题，把处理过程高度抽象为两个函数: map和reduce，map负责把任务分解成多个任务，reduce负责把分解后多任务处理的结果汇总起来。

需要注意的是，用MapReduce来处理的数据集(或任务)必须具备这样的特点：**待处理的数据集可以分解成许多小的数据集，而且每一个小数据集都可以完全并行地进行处理**。

3.2．MapReduce的执行过程

MapReduce运行在⼤规模集群之上,要完成⼀个并行计算,还需要任务调度、本地计算、Shuffle(洗牌)过程等一系列环节共同支撑计算的过程。

3.2.1 任务调度与执行

MapReduce任务由一个JobTracker和多个TaskTracker两类节点控制完成。JobTracker主要负责调度和管理TaskTracker,它通常情况下运行在Master节点上。JobTracker 将Mappers和Reducers分配给空闲的TaskTracker后，由TaskTracker负责这些任务的并行执行。TaskTracker必须运行在DataNode上，所以DataNode既是数据的存储节点也是计算节点。JobTracker 也负责监控任务的运行状况如果某个TaskTracker 发生故障,JobTracker就会将其负责的任务分配给其他空闲的TaskTracker重新执行。MapReduce 框架的这种设计很适合于集群上任务的调度和执行，当然JobTracker的故障将引起整个任务失败，在Hadoop以后的发行版本中或许会通过运行多个JobTracker解决这个问题。

3.2.2 本地计算

把计算节点和数据节点置于同一台计算机上,MapReduce框架尽最大的努力保证在那些存储了数据的节点上执行计算任务。这种方式有效地减少了数据在网络中的传输降低了任务对网络带宽的需求避免了使网络带宽成为瓶颈，所以“本地计算”可以说是节约带宽最有效的方式，业界称之为“移动计算比移动数据更经济”。也正是因为如此,split通常情况下应该小于或等于HDFS数据块的大小(默认情况下64MB),从而保证split 不会跨越两台计算机存储便于“本地计算”。

3.2.3 Shuffle过程

MapReduce会将Mapper的输出结果按照key值分成R份(R是预先定义的Reducers的个数),划分时常使用哈希函数，如Hash(key) mod R。这样可以保证某一范围内的 key一定由某个Reducer来处理从而简化Reduce的过程。

3.2.4 合并Mapper输出

正如之前所说，带宽资源非常宝贵，所以MapReduce允许在Shuffle之前先对结果进行合并(Combine过程),即将中间结果中有相同key值的多组<key, value>对合并成一对。Combine 过程和Reduce过程类似，很多情况下可以直接使用reduce函数，但Combine过程是Mapper的一部分，在map函数之后执行。Combine过程通常情况下可以有效地减少中间结果的数量，从而减少数据传输过程中的网络流量。值得注意的是,Hadoop并不保证其会对一个Mapper输出执行多少次Combine过程，也就是说，开发人员必须保证不论Combine过程执行多少次，得到的结果都是一样的。

3.2.5 读取中间过程

在完成Combine和Shuffle的过程后,Mapper的输出结果被直接写到本地磁盘。然后，通知JobTracker中间结果文件的位置再由JobTracker告知Reducer到哪个DataNode.上去取中间结果。注意所有的Mapper产生的中间结果均按其key值用同一个哈希函数划分成R份,R个Reducer各自负责一段key值区间。每个Reducer需要向多个Mapper节点取得落在其负责的key值区间内的中间结果然后执行reduce函数,形成一个最终的结果文件。  需要说明的是,Mapper的输出结果被直接写到本地磁盘而非HDFS,因为Mapper输出的是中间数据，当任务完成之后就可以直接删除了，如果存储在HDFS上,HDFS的备份机制会造成性能的损失。

3.2.6 任务管道

R个Reducer会产生R个结果，很多情况下这R个结果并不是所需要的最终结果，而是会将这R个结果作为另一个计算任务的输入，并开始另一个MapReduce任务。

1. 实验步骤

**4.1 任务1. 掌握Hadoop DFS常用指令**

在实验一中，我们通过Linux指令对本地文件系统进行操作，如使用ls查看文件/目录信息、使用cp进行文件复制、使用cat查看文件内容。在分布式文件系统中，也有一套相似的指令，接下来我们需要掌握一些基本的指令。

首先查看Hadoop DFS支持哪些指令

|  |
| --- |
| 2018211009@thumm01:~$ hadoop fs  Usage: hadoop fs [generic options]  [-cat [-ignoreCrc] <src> ...]  **[-copyFromLocal [-f] [-p] [-l] [-d] [-t <thread count>] <localsrc> ... <dst>]**  **[-copyToLocal [-f] [-p] [-ignoreCrc] [-crc] <src> ... <localdst>]**  [-cp [-f] [-p | -p[topax]] [-d] <src> ... <dst>]  [-head <file>]  **[-help [cmd ...]]**  **[-ls [-C] [-d] [-h] [-q] [-R] [-t] [-S] [-r] [-u] [-e] [<path> ...]]**  **[-mkdir [-p] <path> ...]**  [-moveFromLocal <localsrc> ... <dst>]  [-moveToLocal <src> <localdst>]  [-mv <src> ... <dst>]  [-rm [-f] [-r|-R] [-skipTrash] [-safely] <src> ...]  [-rmdir [--ignore-fail-on-non-empty] <dir> ...]  ...... |

上面是DFS中常用的指令，这些指令中有一些我们在本地文件系统中也用过，如**ls、cp、mv、rm、mkdir、cat、head**，还有一些指令是DFS特有的，例如**copyFromLocal、copyToLocal、copyFromLocal、copyToLocal，**主要用于DFS与本地文件系统的数据交换。

首先使用ls指令查看DFS中根目录下文件/文件夹的信息

|  |
| --- |
| 2018211009@thumm01:~$ hadoop fs -ls /  Found 2 items  drwxr-xr-x - root supergroup 0 2019-10-18 16:06 /dsjxtjc  -rw-r--r-- 3 root supergroup 13588590 2019-10-18 14:50 /wc\_dataset.txt |

可以看到，现在DFS根目录下一共有两项，其中wc\_dataset.txt是实验一中的数据集，dsjxtjc是一个文件夹，在这个文件夹下面有每位同学的文件夹，例如某位同学的学号是201921xxxx，那么他/她对应的文件夹为/dsjxtjc/201921xxxx/。为了保证实验过程中不同用户之间不会产生干扰，每位同学只能在自己的文件夹下进行操作。

首先查看自己的文件下的内容:

|  |
| --- |
| 2018211009@thumm01:~$ hadoop fs -ls /dsjxtjc/2018211009  2018211009@thumm01:~$ |

接下来在本地创建一个test.txt文件

|  |
| --- |
| 2018211009@thumm01:~$ touch test.txt  2018211009@thumm01:~$ echo "Hello Hadoop" > test.txt  2018211009@thumm01:~$ cat test.txt  Hello Hadoop |

将本地文件传输至DFS中

|  |
| --- |
| 2018211009@thumm01:~$ hadoop fs -copyFromLocal ./test.txt /dsjxtjc/2018211009/  2018211009@thumm01:~$ hadoop fs -cat /dsjxtjc/2018211009/test.txt  Hello Hadoop |

可以看到文件以及传输到DFS上。

copyFromLocal/copyToLocal 用于本地文件系统与DFS之间文件的复制，moveFromLocal/moveToLocal 用于本地文件系统与DFS之间文件的移动，这些指令的详细用法可以使用-help指令查看，例如我们想了解copyFromLocal的用法：

|  |
| --- |
| 2018211009@thumm01:~$ hadoop fs -help copyFromLocal  **-copyFromLocal** [-f] [-p] [-l] [-d] [-t <thread count>] **<localsrc>** ... **<dst>** :  Copy files from the local file system into fs. Copying fails if the file already  exists, unless the -f flag is given.  Flags:  -p Preserves access and modification times, ownership and the mode.  -f Overwrites the destination if it already exists.  -t <thread count> Number of threads to be used, default is 1.  -l Allow DataNode to lazily persist the file to disk. Forces  replication factor of 1. This flag will result in reduced  durability. Use with care.  -d Skip creation of temporary file(<dst>.\_COPYING\_). |

可以看到该指令有两个必填参数，第一个参数是本地路径，第二个参数是DFS路径。

**4.2 任务2. 使用MapReduce模型进行词频统计**

接下来我们要做的是使用MapReduce计算模型进行词频统计。

首先创建一个文件夹input, 文件夹下存放着一些文本文件：

|  |
| --- |
| 2018211009@thumm01:~$ mkdir wc\_input  2018211009@thumm01:~$ cd wc\_input  2018211009@thumm01:~/wc\_input$ echo "Hello World" > test1.txt  2018211009@thumm01:~/wc\_input$ echo "Hello Hadoop" > test2.txt  2018211009@thumm01:~/wc\_input$ cat test1.txt test2.txt  Hello World  Hello Hadoop |

将input文件传到DFS中

|  |
| --- |
| 2018211009@thumm01:~/input$ hadoop fs -copyFromLocal -d ~/wc\_input /dsjxtjc/2018211009/  2018211009@thumm01:~/wc\_input$ hadoop fs -ls /dsjxtjc/2018211009/wc\_input  Found 2 items  -rw-r--r-- 3 2018211009 dsjxtjc 12 2019-10-18 23:30 /dsjxtjc/2018211009/wc\_input/test1.txt  -rw-r--r-- 3 2018211009 dsjxtjc 13 2019-10-18 23:29 /dsjxtjc/2018211009/wc\_input/test2.txt |

在主目录下创建word\_count文件夹，并在其中创建WordCount.java文件

|  |
| --- |
| 2018211009@thumm01:~$ mkdir -p word\_count  2018211009@thumm01:~$ cd word\_count  2018211009@thumm01:~/word\_count$ vim WordCount.java |

WordCount.java的内容如下：

|  |
| --- |
| import java.io.IOException;  import java.util.StringTokenizer;  import org.apache.hadoop.conf.Configuration;  import org.apache.hadoop.fs.Path;  import org.apache.hadoop.io.IntWritable;  import org.apache.hadoop.io.Text;  import org.apache.hadoop.mapreduce.Job;  import org.apache.hadoop.mapreduce.Mapper;  import org.apache.hadoop.mapreduce.Reducer;  import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;  import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;  public class WordCount {  public static class TokenizerMapper extends Mapper<Object, Text, Text, IntWritable>{  private final static IntWritable one = new IntWritable(1);  private Text word = new Text();  public void map(Object key, Text value, Context context  ) throws IOException, InterruptedException {  StringTokenizer itr = new StringTokenizer(value.toString());  while (itr.hasMoreTokens()) {  word.set(itr.nextToken());  context.write(word, one);  }  }  }  public static class IntSumReducer extends Reducer<Text,IntWritable,Text,IntWritable> {  private IntWritable result = new IntWritable();  public void reduce(Text key, Iterable<IntWritable> values,  Context context  ) throws IOException, InterruptedException {  int sum = 0;  for (IntWritable val : values) {  sum += val.get();  }  result.set(sum);  context.write(key, result);  }  }  public static void main(String[] args) throws Exception {  Configuration conf = new Configuration();  Job job = Job.getInstance(conf, "word count");  job.setNumReduceTasks(3); //@设置reducer的数量  job.setJarByClass(WordCount.class);  job.setMapperClass(TokenizerMapper.class);  job.setCombinerClass(IntSumReducer.class);  job.setReducerClass(IntSumReducer.class);  job.setOutputKeyClass(Text.class);  job.setOutputValueClass(IntWritable.class);  FileInputFormat.addInputPath(job, new Path(args[0]));  FileOutputFormat.setOutputPath(job, new Path(args[1]));  System.exit(job.waitForCompletion(true) ? 0 : 1);  }  } |

编译Java程序并打包为jar包

|  |
| --- |
| 2018211009@thumm01:~/word\_count$ hadoop com.sun.tools.javac.Main WordCount.java  2018211009@thumm01:~/word\_count$ jar cf wc.jar WordCount\*.class |

运行jar包

|  |
| --- |
| 2018211009@thumm01:~/word\_count$ hadoop jar wc.jar WordCount /dsjxtjc/2018211009/wc\_input /dsjxtjc/2018211009/wc\_output  ...... //省略部分信息  2019-10-19 00:40:27,639 INFO impl.YarnClientImpl: Submitted application application\_1571414388168\_0010  2019-10-19 00:40:27,699 INFO mapreduce.Job: The url to track the job: http://thumm01:8088/proxy/application\_1571414388168\_0010/  2019-10-19 00:40:27,700 INFO mapreduce.Job: Running job: job\_1571414388168\_0010  2019-10-19 00:40:34,898 INFO mapreduce.Job: Job job\_1571414388168\_0010 running in uber mode : false  2019-10-19 00:40:34,901 INFO mapreduce.Job: map 0% reduce 0%  2019-10-19 00:40:39,996 INFO mapreduce.Job: map 100% reduce 0%  2019-10-19 00:40:47,053 INFO mapreduce.Job: map 100% reduce 100%  **2019-10-19 00:40:47,067 INFO mapreduce.Job: Job job\_1571414388168\_0010 completed successfully**  2019-10-19 00:40:47,213 INFO mapreduce.Job: Counters: 55  ...... //省略部分信息  File Output Format Counters  Bytes Written=25 |

经过map和reduce环节后，显示Job XXX completed successfully, 表示运行成功~

接下来查看结果，将结果从DFS复制到本地并查看:

|  |
| --- |
| 2018211009@thumm01:~/word\_count$ hadoop fs -copyToLocal /dsjxtjc/2018211009/wc\_output ./  2018211009@thumm01:~/word\_count$ ls wc\_output/  part-r-00000 \_SUCCESS  2018211009@thumm01:~/word\_count$ cat wc\_output/part-r-00000  Hadoop 1  Hello 2  World 1 |

词频统计结束。

**4.3 任务3. 设计并实现自己的分布式文件系统**

* 借鉴DFS和MapReduce的思想，用你熟悉的编程语言，自己写一个简单的DFS+MapReduce框架
* 利用该框架实现 **均值和方差统计** 功能，数据可以自己采集或随机生成，无固定格式要求，数据文件大小需要大于1G，小于5G
* 推荐通过控制台输入命令参数的方式与程序进行交互，比如 your\_dfs\_command -copyFromLocal <local\_path> <dfs\_path>
* 在集群`thumm02-thumm05`机器中，请将数据块和程序中间结果放在`/home/dsjxtjc/学号`目录下
* 详细阐述你的设计，包括整体设计思想，系统框架，数据分割方案，任务分配和整合方案等细节，体现你对DFS和MapReduce思想的理解
* （加分项）有一定的任务错误处理机制（比如某台节点宕机或者出现数据块丢失）
* 具体实现可以参考附件DFS.zip，可以在其基础上添加要求功能，也可以自己重新设计，编程语言不限。

1. 作业提交要求

* 将以上作业所用到的数据、命令、步骤截图等写入实验报告，然后连同所有代码文件一同打包成压缩文件，上交至网络学堂。
* 迟交作业一周以内，以50%比例计分；一周以上不再计分。另一经发现抄袭情况，零分处理。
* 助教联系方式：严浩鹏（yhp18@mails.tsinghua.edu.cn）