**学习日志**

姓名：李奥 学号： 1812998 日期： 6.24

**学习内容安排：**

**了解MapReduce以及Hadoop分布式文件系统（HDFS）和MapReduce应用开发**

**学习反馈：**

1、检验数据是否损坏最常见的措施是：在数据第一次引入系统时计算校验和并在数据通过一个不可靠通道进行传输时再次计算校验和，这样就能发现数据是否被损坏。HDFS会对写入的所有数据计算校验和，并在读取数据时验证校验和。

2、客户端写数据：正在写数据的客户端将数据及其校验和发送到由一系列datanode组成的管线，管线中最后一个datanode负责验证校验和。如果检测到错误，客户端会收到一个ChecksumException异常。（datanode负责在收到数据后存储该数据并验证校验和，它在收到客户端的数据或者复制其他datanode的数据时执行这个操作）

客户端从datanode读数据：验证校验和，将它们与datanode中存储的校验和进行比较。每个datanode中持久保存有一个用于验证校验和的校验日志，它知道每个数据块的最后一次验证时间，客户端成功验证一个数据块后，会告诉datanode，由此更新日志。

每个datanode也会在后台中运行一个DataBlockScanner，从而定期验证存储在这个datanode上的所有数据块。

若datanode读取数据时发现数据损坏，首先向namenode报告，再抛出ChecksumException异常，namenode将这个数据块标记为损坏，之后它安排这个数据块的一个复本复制到另一个datanode，如此一来，数据的复本因子回到期望水平，已损坏的复本会被删除。

3、在mapreduce中使用压缩

对最终输出进行压缩：

FileOutputFormat.setCompressOutput(job, true);

FileOutputFormat.setOutputCompressorClass(job, GzipCodec.class);

对map任务输出进行压缩：

.

// vv MaxTemperatureWithMapOutputCompression

Configuration conf = new Configuration();

conf.setBoolean(Job.MAP\_OUTPUT\_COMPRESS, true);

conf.setClass(Job.MAP\_OUTPUT\_COMPRESS\_CODEC, GzipCodec.class,

CompressionCodec.class);

Job job = new Job(conf);

// ^^ MaxTemperatureWithMapOutputCompression

4、序列化在分布式数据处理的两大领域经常出现：进程间通信和永久存储。  
通常情况下，序列化格式需要满足：

紧凑（紧凑格式能充分利用网络带宽）

快速（分布式系统需要尽量减少序列化和反序列化的性能开销）

可扩展（能够满足协议变化的需求，在控制客户端和服务器的过程中，可以直接引进相应的协议）

支持互操作（可以支持不同语言写的客户端和服务器交互）

5、1、MapReduce 的输入输出模型  
MapReduce 中，reduce函数的输入类型必须与map 函数的输出类型一致，例如  
map: (k1,v1)->list(k2,v2)  
reduce:(k2,list(v2))->list(k3,v3)

**MapReduce 中的常用设置：**

1、输入数据类型由输入格式(InputFormat)设置  
map 输出的key 类型通过setMapOutputKeyClass 设置

2、MapReduce 输入格式：  
MapReduce 处理的输入文件一般存储在HDFS上,这些输入文件的格式多种多样，比如基于行的日志文件、二进制文件等等，而且有些文件非常之大。因此，MapReduce 定义了一些接口来读取这些文件数据。

InputFormat 接口  
InputFormat 能够从一个job中得到一个split 集合，再为这个split集合配上一个合适的RecordReader来读取每个split 中的数据。

3、MapReduce 输出格式：  
Hadoop 默认情况下只有一个Reduce，即输出只有一个文件。输出文件的个数有Reduce的个数一致。

OutputFormat 接口：  
OutputFormat 主要用于描述数据的格式，它能够将用户提供的key/value 对写入到特定格式的文件中。

4、Combiner ：  
Hadoop 框架使用Mapper 将数据处理成一个<key,value> 键值对，再在网络空间中对其进行整理(shuffile)，然后使用Reducer 处理数据并进行最终输出。

在MapReduce 框架中，Combiner 是为了避免map任务和Reduce 任务之间的无效数据传输而设置的。Hadoop 允许用户针对map任务的输出指定一个合并函数，减少传输到Reducer中的数据量。通过削减mapper 的输出数量，来减少网络带宽和Reducer 上面的负载。

可以将Combiner操作看作是在每个单独的节点上先做一次Reducer操作，其输入和输出的参数和Reducer 是一样的。

5、Partitioner:  
在进行MapReduce 计算时，有时候需要将最终的输出数据分到不同的文件中，比如按照省份、性别等等来进行划分。如果想要得到多个文件，就意味着有同等数量的Reducer 任务在运行。Reducer任务的数据来自于Mapper 任务，也就是说Mapper 任务要划分数据，将不同的数据分配给给不同的Reducer 运行，Mapper 任务划分数据的过程称为Partition,负责划分数据的类为Partitioner。  
MapReduce 默认的Partitioner 是HashPartitioner。一般Patitioner 先计算key 的散列值，然后对Reducer 个数进行取模运算。这种方式能随机地将整个key 空间均匀地分配给每个Reducer,同时也能确保不同Mapper 产生的key 能被分配到同一个Reducer。

6、自定义RecordReader：  
RecordReader 表示以怎样的方式从分片中读取一条记录，每读取一条记录都要调用一次RecordReader 类，系统默认的RocordReader 是 LineRecorderReader,它是TextInputFormat 对应的RecordReader。SequenceFileInputFormat 对应的RecordReader 是SequenceFileRecordReader。

LineRecordReader 以每行的偏移量作为读入Map 的Key,每行读入的内容作为Map 的Value。  
有时候我们需要自定义RecordReader以我们需要的方式去读取记录，比如以文件名或者行号去读取记录。

自定义RecordReader 的实现步骤：  
1、继承抽象类RecordReader，实现RecordReader 的一个实例。  
2、实现自定义InputFormat 类，并重写其中的CreateRecordReader()方法，返回值是自定义的RecordReader实例。  
3、配置job.setInputFormatClass() 为自定义的InputFormat 实例。

未掌握知识：

学习心得：简单了解了MapReduce以及Hadoop分布式文件系统（HDFS）和MapReduce应用开发