**学习心得**

18183023 谭智元 6.25

**今天学习的内容：**

1.MapReduce类型与格式：

**输入格式：**

1）输入分片与记录：

a）JobClient通过指定的输入文件的格式来生成数据分片InputSpilit

抽象类：FileInputFormat

1、FileInputFormat是所有使用文件作为数据源的InputFormat实现的基类

2、FileInputFormat输入数据格式的分片大小由数据块大小决定

b）一个分片不是数据本身，而是可分片数据的引用

c）InputFormat接口负责生成分片

2）文件输入：

抽象类：CombineFileInputFormat

a）可以使用CombineFileInputFormat来合并小文件

b）因为CombineFileInputFormat是一个抽象类，使用的时候需要创建一个CombineFileInputFormat的实体类，并且实现getRecordReader（）的方法

3、避免文件分割的方法：

a）数据块大小尽可能大，这样使文件的大小小于数据块的大小，就不用进行分片

b）继承FileInputFormat，并且重载isSplitable（）方法

3）文本输入

类名：TextInputFormat

a）TextInputFormat是默认的InputFormat，每一行数据就是一条记录

b）TextInputFormat的key是LongWritable类型的，存储该行在整个文件的偏移量，value是每行的数据内容，Text类型

c）输入分片与HDFS数据块关系：TextInputFormat每一条记录就是一行，很有可能某一行跨数据块存放

类名：KeyValueTextInputFormat

a）KeyValueTextInputFormat类可以通过设置key为行号的方式来知道记录的行号，并且可以通过key.value.separator.in.input设置key与value的分割符

类名：NLineInputFormat

a）可以设置每个mapper处理的行数，可以通过mapred.line.input.format.linespermap属性设置

4）二进制输入

类：SequenceFileInputFormat

SequenceFileAsTextInputFormat

SequenceFileAsBinaryInputFormat

2.MapReduce特性：

**1）计数器：**

计数器是手机作业统计信息的有效手段之一，用于质量控制或应用级统计，计数器还可以辅助诊断系统故障

a)内置计数器：

Hadoop为每个作业维护若干内置计数器，如处理的字节数，和记录数

b) 任务计数器：

任务计数器由其关联任务维护，并定期发送给application master

c) 作业计数器：

作业计数器由application master维护，无需再网络间传输数据，这些计数器都是做业级别的统计，值不会随着任务运行而改变。如：启动的map数

d) 用户定义的Java计数器：

计数器的值可在 mapper 或 reducer 中增加，计数器由一个Java枚举类型来定义，以便对有关计数器分组，枚举类型的名称即为组的名称，枚举类型的字段就是计数器的名称

e) 动态计数器：

Java枚举类型字段在编译阶段必须指定，因而无法使用枚举类型动态新建计数器

f) 获取计数器：

Java API 还支持在作业运行期间就能够获取计数器的值

**2）排序：**

a) 部分排序：

默认情况下，MapReduce根据输入记录的键对数据进行排序

b) 按键进行排序：

键的排列顺序有RawComparator控制的

c) 全排序：

最简单方法是使用一个分区，但是在处理大型问件事效率极低，因为每一台机器必须处理所有输入文件，丧失了MapReduce提供的并行架构优势

替代方案：创建一系列拍还需的文件，串联这些文件，生成一个全局排序文件，使用一个Partitioner描述输出的。

d) 辅助排序：

MapReduce在记录到达Reducer之前按键对记录排序，但键所对应的值没有排序

**3）连接：**

连接操作的具体实现技术取决于数据集的规模及分区方式

a) map端连接：

两个大规模数据之间的map端连接，会在数据到达map函数之前就执行连接操作。

为达到目的，map的输入数据必须先分区，并且以特定方式排序，各个输入数据集被划分成相同数量的分区，并且均按相同键（连接键）排序，同一键的所有记录均会放在同一分区中。

b) reduce端连接：

reduce端连接并不要求输入数据集符合特定结构，因而更为常用，但是两个数据集需要经过shuffle过程，所以效率更低，基本思路是mapper为各个记录标记源，并且使用连接键作为map输出键，使键相同的记录放在同一个reducer中，以下记录可以帮助实现reduce端连接：

i)多输入：数据集的输入源往往有多个，可以使用MultipleInputs类

ii) 辅助排序：为了更好的执行连接操作，一个源的数据排列在另一个源数据前是非常重要的

**4）边数据分布：**

a)利用JobConf配置作业

b)分布式缓存：

在任务运行过程中及时的将文件和存档复制到任务节点以供节点使用：

-files 选项指定分发的文件，文件内包含以逗号隔开的URI列表

-archives选项想自己的任务中复制存档文件（JAR，ZIP，tar等）

-libjars 选项会把JAR文件添加到mapper和reducer任务的类路径中

c) 分布式缓存API：

可以通过GenericOptionsParser间接使用分部署缓存，大多数情况使用job中的相关方法：

job.addCacheFile(uri);

job.addCacheArchive(uri);

job.setCacheFiles(files);

job.setCacheArchives(archives);

job.addFileToClassPath(file);

job.addArchiveToClassPath(archive);

job.createSymlink();

3.插曲——虚拟机断网：

在尝试启动Hadoop时，提示无法连接，经检查发现是网络出现问题，ping命令无法正确使用。

出现问题的原因：经查阅后发现，可能是经过长时间的挂起虚拟机的网络服务中断且无法重启

解决方法：尝试使用systemctl restart network命令重启网络服务，却发现报错。检查日志后发现错误：Failed to start LSB: Bring up/down。经查阅并尝试后，使用systemctl stop NetworkManager && systemctl disable NetworkManager命令禁用NetworkManager即可重新连接网络

**反思：**

1. 关于上述内容有许多java代码示例需要认真研究

2.明天尝试Hadoop集群的搭建