Hadoop基础知识面试题整理

一、 问答题

1、简单描述如何安装配置一个apache开源版hadoop，只描述即可，无需列出完整步骤，能列出步骤更好。

1) 安装JDK并配置环境变量（/etc/profile）
2) 关闭防火墙
3) 配置hosts文件，方便hadoop通过主机名访问（/etc/hosts）
4) 设置ssh免密码登录
5) 解压缩hadoop安装包，并配置环境变量
6) 修改配置文件（$HADOOP_HOME/conf）
hadoop-env.sh core-site.xml hdfs-site.xml mapred-site.xml
7) 格式化hdfs文件系统（hadoop namenode -format）
8) 启动hadoop（$HADOOP_HOME/bin/start-all.sh）
9) 使用jps查看进程

2、请列出正常工作的hadoop集群中hadoop都分别需要启动那些进程，他们的作用分别是什么，尽可能写的全面些。

1) NameNode: HDFS的守护进程，负责记录文件是如何分割成数据块，以及这些数据块分别被存储到那些数据节点上，它的主要功能是对内存及IO进行集中管理
2) Secondary
NameNode：辅助后台程序，与NameNode进行通信，以便定期保存HDFS元数据的快照。
3) DataNode：负责把HDFS数据块读写到本地的文件系统。
4) JobTracker：负责分配task，并监控所有运行的task。
5) TaskTracker：负责执行具体的task，并与JobTracker进行交互。

3、请列出你所知道的hadoop调度器，并简要说明其工作方法。

比较流行的三种调度器有：默认调度器FIFO，计算能力调度器Capacity Scheduler，公平调度器Fair Scheduler
1) 默认调度器FIFO
hadoop中默认的调度器，采用先进先出的原则
2) 计算能力调度器Capacity Scheduler
选择占用资源小，优先级高的先执行
3) 公平调度器Fair Scheduler
同一队列中的作业公平共享队列中所有资源

4、Hive有那些方式保存元数据的，各有那些特点。

1) 内存数据库derby，较小，不常用
2) 本地mysql，较常用
3) 远程mysql，不常用

5、请简述hadoop怎样实现二级排序。

在Hadoop中，默认情况下是按照key进行排序，如果要按照value进行排序怎么办？
有两种方法进行二次排序，分别为：buffer and in memory sort和 value-to-key conversion。

buffer and in memory sort
主要思想是：在reduce()函数中，将某个key对应的所有value保存下来，然后进行排序。
这种方法最大的缺点是：可能会造成out of memory。

value-to-key conversion
主要思想是：将key和部分value拼接成一个组合key（实现WritableComparable接口或者调setSortComparatorClass函数），这样reduce获取的结果便是先按key排序，后按value排序的结果，需要注意的是，用户需要自己实现Paritioner，以便只按照key进行数据划分。Hadoop显式的支持二次排序，在Configuration类中有个setGroupingComparatorClass()方法，可用于设置排序group的key值。《Hadoop&Spark解决二次排序问题(Hadoop篇)》

6、简述hadoop实现Join的几种方法。

(1)、reduce side join
reduce side join是一种最简单的join方式，其主要思想如下：
在map阶段，map函数同时读取两个文件File1和File2，为了区分两种来源的key/value数据对，对每条数据打一个标签（tag）,比如：tag=0表示来自文件File1，tag=2表示来自文件File2。即：map阶段的主要任务是对不同文件中的数据打标签。
在reduce阶段，reduce函数获取key相同的来自File1和File2文件的value list， 然后对于同一个key，对File1和File2中的数据进行join（笛卡尔乘积）。即：reduce阶段进行实际的连接操作。

(2)、map side join
之所以存在reduce side join，是因为在map阶段不能获取所有需要的join字段，即：同一个key对应的字段可能位于不同map中。Reduce side
join是非常低效的，因为shuffle阶段要进行大量的数据传输。
Map side join是针对以下场景进行的优化：两个待连接表中，有一个表非常大，而另一个表非常小，以至于小表可以直接存放到内存中。这样，我们可以将小表复制多份，让每个map
task内存中存在一份（比如存放到hash
table中），然后只扫描大表：对于大表中的每一条记录key/value，在hash
table中查找是否有相同的key的记录，如果有，则连接后输出即可。
为了支持文件的复制，Hadoop提供了一个类DistributedCache，使用该类的方法如下：
（1）用户使用静态方法DistributedCache.addCacheFile()指定要复制的文件，它的参数是文件的URI（如果是HDFS上的文件，可以这样：hdfs://namenode:9000/home/XXX/file，其中9000是自己配置的NameNode端口号）。JobTracker在作业启动之前会获取这个URI列表，并将相应的文件拷贝到各个TaskTracker的本地磁盘上。（2）用户使用DistributedCache.getLocalCacheFiles()方法获取文件目录，并使用标准的文件读写API读取相应的文件。

(3)、SemiJoin
SemiJoin，也叫半连接，是从分布式数据库中借鉴过来的方法。它的产生动机是：对于reduce side join，跨机器的数据传输量非常大，这成了join操作的一个瓶颈，如果能够在map端过滤掉不会参加join操作的数据，则可以大大节省网络IO。
实现方法很简单：选取一个小表，假设是File1，将其参与join的key抽取出来，保存到文件File3中，File3文件一般很小，可以放到内存中。在map阶段，使用DistributedCache将File3复制到各个TaskTracker上，然后将File2中不在File3中的key对应的记录过滤掉，剩下的reduce阶段的工作与reduce side join相同。

(4)、reduce side join + BloomFilter
在某些情况下，SemiJoin抽取出来的小表的key集合在内存中仍然存放不下，这时候可以使用BloomFiler以节省空间。
BloomFilter最常见的作用是：判断某个元素是否在一个集合里面。它最重要的两个方法是：add()和contains()。最大的特点是不会存在false
negative，即：如果contains()返回false，则该元素一定不在集合中，但会存在一定的true negative，即：如果contains()返回true，则该元素可能在集合中。
因而可将小表中的key保存到BloomFilter中，在map阶段过滤大表，可能有一些不在小表中的记录没有过滤掉（但是在小表中的记录一定不会过滤掉），这没关系，只不过增加了少量的网络IO而已。

7、请简述MapReduce中combiner、partition的作用

(1)、combiner
有时一个map可能会产生大量的输出，combiner的作用是在map端对输出先做一次合并，以减少网络传输到reducer的数量。
注意：mapper的输出为combiner的输入，reducer的输入为combiner的输出。

(2)、partition
把map任务输出的中间结果按照key的范围划分成R份(R是预先定义的reduce任务的个数)，划分时通常使用hash函数，如：hash(key) mod R
这样可以保证一段范围内的key，一定会由一个reduce任务来处理。