# 一、Hadoop

## 1. 选择题

### 1.1 HDFS

1.下面哪个程序负责 HDFS 数据存储？

a)NameNode

b)Jobtracker

c)Datanode

d)secondaryNameNode

e)tasktracker

2.HDfS 中的 block 默认保存几份？

a)3份

b)2份

c)1份

d)不确定

4.      HDFS 默认 Block Size

a)32MB

b)64MB

c)128MB

5.      Client 端上传文件的时候下列哪项正确

a)数据经过 NameNode 传递给 DataNode

b)Client 端将文件切分为 Block，依次上传

c)Client 只上传数据到一台 DataNode，然后由 NameNode 负责 Block 复制工作

分析：

Client 向 NameNode 发起文件写入的请求。

NameNode 根据文件大小和文件块配置情况，返回给 Client 它所管理部分 DataNode 的信息。

Client 将文件划分为多个 Block，根据 DataNode 的地址信息，建立pipline，请求数据传输，先用管道打通n台dns,从客户端到dn1,dn1到dn2,dn2到dnn，客户端向管道中传输数据，数据流转到那台机器，就复制保存，效率高。如果没有管道，那么客户端会依次向三个dn上面上传数据，消耗大。

6.下面与 HDFS 类似的框架是？C

A NTFS

B FAT32

C GFS

D EXT3

### 1.2  集群管理

1.      下列哪项通常是集群的最主要瓶颈

a)CPU

b)网络

c)磁盘 IO

d)内存

解析：

由于大数据面临海量数据，读写数据都需要 io，然后还要冗余数据，hadoop 一般备 3 份数据，所以 IO就会打折扣。

2.关于SecondaryNameNode 哪项是正确的？

a)它是 NameNode 的热备

b)它对内存没有要求

c)它的目的是帮助 NameNode 合并编辑日志，减少 NameNode 启动时间

d)SecondaryNameNode 应与 NameNode 部署到一个节点

分析：NameNode如何把元数据保存到磁盘上的。这里有两个不同的文件：

1. fsimage - 它是在NameNode启动时对整个文件系统的快照

2. edit logs - 它是在NameNode启动后，对文件系统的改动序列

只有在NameNode重启时，edit logs才会合并到fsimage文件中，从而得到一个文件系统的最新快照。但是在产品集群中NameNode是很少重启的，这也意味着当NameNode运行了很长时间后，edit logs文件会变得很大。在这种情况下就会出现下面一些问题：

1. edit logs文件会变的很大，怎么去管理这个文件是一个挑战。

2. NameNode的重启会花费很长时间，因为在edit logs中有很多改动，要合并到fsimage文件上。

3. 如果NameNode挂掉了，那我们就丢失了很多改动因为此时的fsimage文件非常旧。

3.下列哪项可以作为集群的管理？

a)Puppet  b)Pdsh  c)ClouderaManager  d)Zookeeper

分析：

A：puppetpuppet 是一种 Linux、Unix、windows 平台的集中配置管理系统

B：pdsh 可以实现在在多台机器上执行相同的命令

详细参考：集群管理小工具介绍-pdsh

C：可以参考 Cloudera Manager 四大功能

首先这里给管理下一个定义：部署、配置、调试、监控，属于管理

4.配置机架感知的下面哪项正确

a)如果一个机架出问题，不会影响数据读写

b)写入数据的时候会写到不同机架的 DataNode 中

c)MapReduce 会根据机架获取离自己比较近的网络数据

5.下列哪个是 Hadoop 运行的模式

a)单机版 b)伪分布式 c)分布式

6.Cloudera 提供哪几种安装 CDH 的方法

a)Cloudera manager  b)Tarball  c)Yum  d)Rpm

## 2. 问答题

### 2.1 基本

1.      hdfs的体系结构

解答：

hdfs有namenode、secondraynamenode、datanode组成。

为n+1模式

namenode负责管理datanode和记录元数据

secondraynamenode负责合并日志

datanode负责存储数据

2.简要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。

流程：

1.创建hadoop用户

2.修改IP

3.安装JDK，并配置环境变量

4.修改host文件映射

5.安装SSH，配置无秘钥通信

6.上传解压hadoop安装包

7.配置conf文件夹下的hadoop-env.sh、core-site.xlmapre-site.xml、hdfs-site.xml

8.配置hadoop的环境变量

9.Hadoop namenode -format

10.start-all

3.请列出hadoop2.x的进程名称

解答：

1.namenode:管理集群，并记录datanode文件信息。

2.Secondname:可以做冷备，对一定范围内的数据做快照性备份。

3.Datanode：存储数据。

4. ResourceManager负责集群中所有资源的统一管理和分配，它接收来自各个节点（NodeManager）的资源汇报信息，并把这些信息按照一定的策略分配给各个应用程序（实际上是ApplicationManager）。

5. NodeManager是每一台机器框架的代理，是执行应用程序的容器，监控应用程序的资源使用情况 (CPU，内存，硬盘，网络 ) 并且向调度器（ResourceManager ）汇报。

5.Hadoop的核心配置是什么？

解答：

Hadoop的核心配置通过两个xml文件来完成：

1.hadoop-default.xml；

2.hadoop-site.xml。

这些文件都使用xml格式，因此每个xml中都有一些属性，包括名称和值，但是当下这些文件都已不复存在。

6.那当下又该如何配置？

解答：

Hadoop现在拥有3个配置文件：

1，core-site.xml；

2，hdfs-site.xml；

3，mapred-site.xml。

这些文件都保存在conf/子目录下。

7.“jps”命令的用处？

解答：

这个命令可以检查Namenode、Datanode等进程是否正常工作。

9.      请列出正常工作的 hadoop 集群中 hadoop 都需要启动哪些进程，他们的作用分别是什么？

解答：

10.启动 hadoop 报如下错误，该如何解决？

error org.apache.hadoop.hdfs.server.namenode.NameNode

org.apache.hadoop.hdfs.server.common.inconsistentFSStateExceptio

n Directory /tmp/hadoop-root/dfs/name is in an inconsistent

state storage direction does not exist or is not accessible?

11.  请写出以下执行命令

1)杀死一个 job?

2)删除 hdfs 上的/tmp/aaa 目录

3)加入一个新的存储节点和删除一个计算节点需要刷新集群状态命令？

解答：

hadoop job -list 记录job-id、hadoop job -kill job-id

hadoop fs -rmr /tmp/aaa

添加新节点：

hadoop -daemon.sh start datanode

hadoop -daemon.sh start nodemamager

移除一个节点：

hadoop mradmin -refreshnodes

hadoop dfsadmin -refreshnodes

12.请列出你所知道的 hadoop 调度器，并简要说明其工作方法？

解答：

1.FIFO schedular:默认，先进先出的原则

2.Capacity schedular:计算能力调度器，选择占用最小，优先级高的先执行，以此类推。

3.Fair schedular:公平调度，所有的job具有相同的资源。

13.  请列出在你以前工作中所使用过的开发 mapreduce 的语言？

解答：

14.  hadoop框架中怎么来优化

解答：

（1）  从应用程序角度进行优化。由于mapreduce是迭代逐行解析数据文件的，怎样在迭代的情况下，编写高效率的应用程序，是一种优化思路。

（2）  对Hadoop参数进行调优。当前hadoop系统有190多个配置参数，怎样调整这些参数，使hadoop作业运行尽可能的快，也是一种优化思路。

（3） 从系统实现角度进行优化。这种优化难度是最大的，它是从hadoop实现机制角度，发现当前Hadoop设计和实现上的缺点，然后进行源码级地修改。该方法虽难度大，但往往效果明显。

（4）linux内核参数调整

15. 使用自定义Writable

自带的Text很好用，但是字符串转换开销较大，故根据实际需要自定义Writable，注意作为Key时要实现WritableCompareable接口

避免output.collect(new Text( ),new Text())

提倡key.set( ) value.set( ) output.collect(key,value)

前者会产生大量的Text对象，使用完后Java垃圾回收器会花费大量的时间去收集这些对象

16. 使用StringBuilder

1.java中的String，StringBuilder，StringBuffer三者的区别**:1.** 首先说运行速度，或者说是执行速度，在这方面运行速度快慢为：StringBuilder > StringBuffer > String

String最慢的原因：　　String为字符串常量，而StringBuilder和StringBuffer均为字符串变量，即String对象一旦创建之后该对象是不可更改的，但后两者的对象是变量，是可以更改的。

2. 再来说线程安全

　　在线程安全上，StringBuilder是线程不安全的，而StringBuffer是线程安全的

如果一个StringBuffer对象在字符串缓冲区被多个线程使用时，StringBuffer中很多方法可以带有synchronized关键字，所以可以保证线程是安全的，但StringBuilder的方法则没有该关键字，所以不能保证线程安全，有可能会出现一些错误的操作。所以如果要进行的操作是多线程的，那么就要使用StringBuffer，但是在单线程的情况下，还是建议使用速度比较快的StringBuilder。

　　3. 总结一下

　　String：适用于少量的字符串操作的情况

　　StringBuilder：适用于单线程下在字符缓冲区进行大量操作的情况

　　StringBuffer：适用多线程下在字符缓冲区进行大量操作的情况

17. 使用DistributedCache加载文件

比如配置文件，词典，共享文件，避免使用static变量

18. 充分使用Combiner Parttitioner Comparator。

Combiner : 对map任务进行本地聚合

Parttitioner ： 合适的Parttitioner避免reduce端负载不均

Comparator ： 二次排序

比如求每天的最大气温，map结果为日期：气温，若气温是降序的，直接取列表首元素即可

19. 使用自定义InputFormat和OutputFormat

20. MR应避免

静态变量：不能用于计数，应使用Counter

大对象：Map List

递归：避免递归深度过大

超长正则表达式：消耗性能，要在map或reduce函数外编译正则表达式

不要创建本地文件：变向的把HDFS里面的数据转移到TaskTracker，占用网络带宽

不要大量创建目录和文件

不要大量使用System.out.println，而使用Logger

不要自定义过多的Counter，最好不要超过100个

不要配置过大内存，mapred.child.java.opts -Xmx2000m是用来设置mapreduce任务使用的最大heap量

21.关于map的数目

map数目过大[创建和初始化map的开销]，一般是由大量小文件造成的，或者dfs.block.size设置的太小，对于小文件可以archive文件或者Hadoop fs -merge合并成一个大文件.

map数目过少，造成单个map任务执行时间过长，频繁推测执行，且容易内存溢出，并行性优势不能体现出来。dfs.block.size一般为256M-512M

压缩的Text 文件是不能被分割的，所以尽量使用SequenceFile，可以切分

22.关于reduce的数目

reduce数目过大，产生大量的小文件，消耗大量不必要的资源，reduce数目过低呢，造成数据倾斜问题，且通常不能通过修改参数改变。

可选方案：mapred.reduce.tasks设为-1变成AutoReduce。

Key的分布，也在某种程度上决定了Reduce数目，所以要根据Key的特点设计相对应的Parttitioner 避免数据倾斜

23.Map-side相关参数优化

io.sort.mb（100MB）：通常k个map tasks会对应一个buffer，buffer主要用来缓存map部分计算结果，并做一些预排序提高map性能，若map输出结果较大，可以调高这个参数，减少map任务进行spill任务个数，降低 I/O的操作次数。若map任务的瓶颈在I/O的话，那么将会大大提高map性能。如何判断map任务的瓶颈？

io.sort.spill.percent(0.8)：spill操作就是当内存buffer超过一定阈值（这里通常是百分比）的时候，会将buffer中得数据写到Disk中。而不是等buffer满后在spill，否则会造成map的计算任务等待buffer的释放。一般来说，调整 io.sort.mb而不是这个参数。

io.sort.factor（10）：map任务会产生很多的spill文件，而map任务在正常退出之前会将这些spill文件合并成一个文件，即merger过程，缺省是一次合并10个参数，调大io.sort.factor，减少merge的次数，减少Disk I/O操作，提高map性能。

min.num.spill.for.combine：通常为了减少map和reduce数据传输量，我们会制定一个combiner，将map结果进行本地聚集。这里combiner可能在merger之前，也可能在其之后。那么什么时候在其之前呢？当spill个数至少为min.num.spill.for.combine指定的数目时同时程序指定了Combiner，Combiner会在其之前运行，减少写入到Disk的数据量，减少I/O次数。

24.压缩（时间换空间）

MR中的数据无论是中间数据还是输入输出结果都是巨大的，若不使用压缩不仅浪费磁盘空间且会消耗大量网络带宽。同样在spill，merge（reduce也对有一个merge）亦可以使用压缩。若想在cpu时间和压缩比之间寻找一个平衡，LzoCodec比较适合。通常MR任务的瓶颈不在CPU而在于I/O，所以大部分的MR任务都适合使用压缩。

25. reduce-side相关参数优化

reduce：copy->sort->reduce，也称shuffle

mapred.reduce.parellel.copies（5）：任一个map任务可能包含一个或者多个reduce所需要数据，故一个map任务完成后，相应的reduce就会立即启动线程下载自己所需要的数据。调大这个参数比较适合map任务比较多且完成时间比较短的Job。

mapred.reduce.copy.backoff：reduce端从map端下载数据也有可能由于网络故障，map端机器故障而失败。那么reduce下载线程肯定不会无限等待，当等待时间超过mapred.reduce.copy.backoff时，便放弃，尝试从其他地方下载。需注意：在网络情况比较差的环境，我们需要调大这个参数，避免reduce下载线程被误判为失败。

io.sort.factor：recude将map结果下载到本地时，亦需要merge，如果reduce的瓶颈在于I/O，可尝试调高增加merge的并发吞吐，提高reduce性能、

mapred.job.shuffle.input.buffer.percent（0.7）：reduce从map下载的数据不会立刻就写到Disk中，而是先缓存在内存中，mapred.job.shuffle.input.buffer.percent指定内存的多少比例用于缓存数据，内存大小可通过mapred.child.java.opts来设置。和map类似，buffer不是等到写满才往磁盘中写，也是到达阈值就写，阈值由mapred.job,shuffle.merge.percent来指定。若Reduce下载速度很快，容易内存溢出，适当增大这个参数对增加reduce性能有些帮助。

mapred.job.reduce.input.buffer.percent (0)：当Reduce下载map数据完成之后，就会开始真正的reduce的计算，reduce的计算必然也是要消耗内存的，那么在读物reduce所需要的数据时，同样需要内存作为buffer，这个参数是决定多少的内存百分比作为buffer。默认为0，也就是说reduce全部从磁盘读数据。若redcue计算任务消耗内存很小，那么可以设置这个参数大于0，使一部分内存用来缓存数据。

26.  从应用程序角度进行优化

解答：

（1） 避免不必要的reduce任务

如果mapreduce程序中reduce是不必要的，那么我们可以在map中处理数据, Reducer设置为0。这样避免了多余的reduce任务。

（2） 为job添加一个Combiner

为job添加一个combiner可以大大减少shuffle阶段从map task拷贝给远程reduce task的数据量。一般而言，combiner与reducer相同。

（3） 根据处理数据特征使用最适合和简洁的Writable类型

Text对象使用起来很方便，但它在由数值转换到文本或是由UTF8字符串转换到文本时都是低效的，且会消耗大量的CPU时间。当处理那些非文本的数据时，可以使用二进制的Writable类型，如IntWritable， FloatWritable等。二进制writable好处：避免文件转换的消耗；使map task中间结果占用更少的空间。

（4） 重用Writable类型

很多MapReduce用户常犯的一个错误是，在一个map/reduce方法中为每个输出都创建Writable对象。例如，你的Wordcout mapper方法可能这样写：

public void map(...) {

  …

  for (String word : words) {

    output.collect(new Text(word), new IntWritable(1));

  }

}

这样会导致程序分配出成千上万个短周期的对象。Java垃圾收集器就要为此做很多的工作。更有效的写法是：

class MyMapper … {

  Text wordText = new Text();

  IntWritable one = new IntWritable(1);

  public void map(...) {

    for (String word: words) {

      wordText.set(word);

      output.collect(wordText, one);

    }

  }

}

（5） 使用StringBuffer而不是String

当需要对字符串进行操作时，使用StringBuffer而不是String，String是read-only的，如果对它进行修改，会产生临时对象，而StringBuffer是可修改的，不会产生临时对象。

27.  datanode在什么情况下不会备份

解答：

当分备份数为1时。

28.  combiner出现在那个过程

解答：

出现在map阶段的map方法后。

29.  3个datanode中有一个datanode出现错误会怎样？

解答：

这个datanode的数据会在其他的datanode上重新做备份。

30.  描述一下hadoop中，有哪些地方使用了缓存机制，作用分别是什么？

解答：在执行MapReduce时，可能Mapper之间需要共享一些信息，如果信息量不大，可以将其从HDFS中加载到内存中，这就是Hadoop分布式缓存机制。

31.  如何确定hadoop集群的健康状态

解答：

32.  hadoop 的 namenode 宕机,怎么解决

解答：

先分析宕机后的损失，宕机后直接导致client无法访问，内存中的元数据丢失，但是硬盘中的元数据应该还存在，如果只是节点挂了，重启即可，如果是机器挂了，重启机器后看节点是否能重启，不能重启就要找到原因修复了。但是最终的解决方案应该是在设计集群的初期就考虑到这个问题，做namenode的HA。

33.  一个datanode 宕机,怎么一个流程恢复

解答：

Datanode宕机了后，如果是短暂的宕机，可以实现写好脚本监控，将它启动起来。如果是长时间宕机了，那么datanode上的数据应该已经被备份到其他机器了，那这台datanode就是一台新的datanode了，删除他的所有数据文件和状态文件，重新启动。

### 2.2  Hadoop原理

1.      请简述 hadoop 怎么样实现二级排序？

解答：

在Reduce阶段，先对Key排序，再对Value排序

最常用的方法是将Value放到Key中，实现一个组合Key，然后自定义Key排序规则（为Key实现一个WritableComparable）。

2.      如何使用MapReduce实现两个表join，可以考虑一下几种情况：（1）一个表大，一个表小（可放到内存中）；（2）两个表都是大表？

解答：

第一种情况比较简单，只需将小表放到DistributedCache中即可；

第二种情况常用的方法有：map-side join（要求输入数据有序，通常用户Hbase中的数据表连接），reduce-side join，semi join（半连接）

3.      MapReduce中排序发生在哪几个阶段？这些排序是否可以避免？为什么？

解答：

一个MapReduce作业由Map阶段和Reduce阶段两部分组成，这两阶段会对数据排序，从这个意义上说，MapReduce框架本质就是一个Distributed Sort。在Map阶段， Map Task会在本地磁盘输出一个按照key排序（采用的是快速排序）的文件（中间可能产生多个文件，但最终会合并成一个），在Reduce阶段，每个Reduce Task会对收到的数据排序，这样，数据便按照Key分成了若干组，之后以组为单位交给reduce（）处理。很多人的误解在Map阶段，如果不使用Combiner便不会排序，这是错误的，不管你用不用Combiner，Map Task均会对产生的数据排序（如果没有Reduce Task，则不会排序， 实际上Map阶段的排序就是为了减轻Reduce端排序负载）。由于这些排序是MapReduce自动完成的，用户无法控制，因此，在hadoop 1.x中无法避免，也不可以关闭，但hadoop2.x是可以关闭的。

4.      请简述 mapreduce 中，combiner，partition 作用？

解答：

combiner是reduce的实现，在map端运行计算任务，减少map端的输出数据。

作用就是优化。

但是combiner的使用场景是mapreduce的map输出结果和reduce输入输出一样。

partition的默认实现是hashpartition，是map端将数据按照reduce个数取余，进行分区，不同的reduce来copy自己的数据。

partition的作用是将数据分到不同的reduce进行计算，加快计算效果。

1、combiner最基本是实现本地key的聚合，对map输出的key排序，value进行迭代。如下所示：

　　map: (K1, V1) → list(K2, V2)

　　combine: (K2, list(V2)) → list(K2, V2)

　　reduce: (K2, list(V2)) → list(K3, V3)

2、combiner还具有类似本地的reduce功能.

　　例如hadoop自带的wordcount的例子和找出value的最大值的程序，combiner和reduce完全一致。如下所示：

　　map: (K1, V1) → list(K2, V2)

　　combine: (K2, list(V2)) → list(K3, V3)

　　reduce: (K3, list(V3)) → list(K4, V4)

　　3、如果不用combiner，那么，所有的结果都是reduce完成，效率会相对低下。使用combiner，先完成的map会在本地聚合，提升速度。

　　4、对于hadoop自带的wordcount的例子，value就是一个叠加的数字，所以map一结束就可以进行reduce的value叠加，而不必要等到所有的map结束再去进行reduce的value叠加。

　　combiner使用的合适，可以在满足业务的情况下提升job的速度，如果不合适，则将导致输出的结果不正确。

5.      解释―hadoop‖和―hadoop 生态系统‖两个概念

解答：

6.      说明 Hadoop 2.0 的基本构成

解答：

分别说明hdfs，yarn，mapreduce

7.      相比于 HDFS1.0,HDFS 2.0 最主要的改进在哪几方面？

解答：

8.      试使用―步骤 1，步骤 2，步骤 3…..‖说明 YARN 中运行应用程序的基本流程

解答：

首先由Driver向resourceManager提交请求运行一个Job命令，resourceManager同意之后，返回给一个jobid和存储切片，jar文件的位置，然后客户端根据返回的信息将切片和jar文件上传到hdfs的指定位置中，都是临时文件，Driver申请一个容器，运行进程。默认值是1.5G，resourceMAnager同意之后，生成容器任务，放进资源队列，资源队列形式不确定，可能是公平资源竞争，也可能是资源调度优先级算法，（操作原理内容，不详述），当任务开始执行的时候，创建任务容器，MapReduceAppManager，以下简称Manager，Manager加载任务信息，Driver发送一个启动命令，运行程序，Manager接受命令，加载配置信息，并且向ResourceManager申请运行yarnchild，yarnchild领到资源之后，根据Manager加载的任务信息下载jar包和切片 配置文件等，之后MapTask开始运行，进行数据处理，当MapTask运行结束后，向Manager申请新的容器，运行新的yarnchild（Reduce Task），reduceTask开始运行，拉取数据，当所有的任务执行完毕之后，Manager向ResourceManager注销本次的job，并推出程序。

9.      MapReduce 2.0‖与YARN‖是否等同，尝试解释说明

解答：

10.  MapReduce 2.0 中，MRAppMaster 主要作用是什么，MRAppMaster如何实现任务容错的？

解答：

它主要作用在于管理作业的生命周期：

11作业的管理：作业的创建，初始化以及启动等

2向RM申请资源和再分配资源

3Container的启动与释放

4监控作业运行状态

5作业恢复

11.  hdfs 原理，以及各个模块的职责

解答：

12.  mr 的工作原理

解答：

Map—combiner—partition—sort—copy—sort—grouping—reduce

13.  map 方法是如何调用 reduce 方法的

解答：

14.  shell 如何判断文件是否存在，如果不存在该如何处理？

解答：

15.  fsimage 和 edit 的区别？

解答：

16.  hadoop1 和 hadoop2 的区别？

解答：

17.  hdfs 中的 block 默认备份几份？

解答：

18.  哪个程序通常与 nn 在一个节点启动？并做分析

解答：

19.  列举几个配置文件优化？

解答：

20.  datanode 首次加入 cluster 的时候，如果 log 报告不兼容文件版本，那该如何处理？

解答：

21.  用mapreduce怎么处理数据倾斜问题？

解答：

数据倾斜：map /reduce程序执行时，reduce节点大部分执行完毕，但是有一个或者几个reduce节点运行很慢，导致整个程序的处理时间很长，这是因为某一个key的条数比其他key多很多（有时是百倍或者千倍之多），这条key所在的reduce节点所处理的数据量比其他节点就大很多，从而导致某几个节点迟迟运行不完，此称之为数据倾斜。

用hadoop程序进行数据关联时，常碰到数据倾斜的情况，这里提供一种解决方法。

自己实现partition类，用key和value相加取hash值：

方式1：

源代码：

public int getPartition(K key, V value,

                          int numReduceTasks) {

    return (key.hashCode() & Integer.MAX\_VALUE) % numReduceTasks;

  }

修改后

public int getPartition(K key, V value,

                          int numReduceTasks) {

    return ((（key).hashCode()+value.hashCode()） & Integer.MAX\_VALUE) % numReduceTasks;

  }

方式2：

public class HashPartitioner<K, V> extends Partitioner<K, V> {

private int aa= 0;

  /\*\* Use {@link Object#hashCode()} to partition. \*/

  public int getPartition(K key, V value,

                          int numReduceTasks) {

    return (key.hashCode()+(aa++) & Integer.MAX\_VALUE) % numReduceTasks;

  }

22.  谈谈数据倾斜，如何发生的，并给出优化方案

解答：

23.  mapreduce 基本执行过程

解答：

24.  谈谈 hadoop1 和 hadoop2 的区别

解答：

25.  hadoop中Combiner的作用?

解答：

combiner是reduce的实现，在map端运行计算任务，减少map端的输出数据。

作用就是优化。

但是combiner的使用场景是mapreduce的map和reduce输入输出一样。

26.  Mapreduce 的 map 数量 和 reduce 数量 怎么确定 ,怎么配置

解答：

map的数量有数据块决定，reduce数量随便配置。

27.  在hadoop中文件的压缩带来了两大好处：

解答：

（1）它减少了存储文件所需的空间；

（2）加快了数据在网络上或者从磁盘上或到磁盘上的传输速度；

28. mapreduce的调度模式

### 2.3  Hadoop使用

1.hdfs写流程

流程：

1.client链接namenode存数据

2.namenode记录一条数据位置信息（元数据），告诉client存哪。

3.client用hdfs的api将数据块（默认是64M）存储到datanode上。

4.datanode将数据水平备份。并且备份完将反馈client。

5.client通知namenode存储块完毕。

6.namenode将元数据同步到内存中。

7.另一块循环上面的过程。

2.hdfs读流程

流程：

1.client链接namenode，查看元数据，找到数据的存储位置。

2.client通过hdfs的api并发读取数据。

3.关闭连接。

3.举一个简单的例子说明mapreduce是怎么来运行的 ?

解答：

    Word count例子接口

============================

一个MapReduce作业（job）通常会把输入的数据集切分为若干独立的数据块，由map任务（task）以完全并行的方式处理它们。框架会对map的输出先进行排序，然后把结果输入给reduce任务。通常作业的输入和输出都会被存储在文件系统中。整个框架负责任务的调度和监控，以及重新执行已经失败的任务。

　　通常，MapReduce框架和分布式文件系统是运行在一组相同的节点上的，也就是说，计算节点和存储节点通常在一起。这种配置允许框架在那些已经存好数据的节点上高效地调度任务，这可以使整个集群的网络带宽被非常高效地利用。

　　MapReduce框架由一个单独的master JobTracker和每个集群节点一个slave TaskTracker共同组成。master负责调度构成一个作业的所有任务，这些任务分布在不同的slave上，master监控它们的执行，重新执行已经失败的任务。而slave仅负责执行由master指派的任务

4.      用mapreduce来实现下面需求？现在有10个文件夹,每个文件夹都有1000000个url.现在让你找出top1000000url。

5.      yarn流程

解答：

1)      用户向YARN 中提交应用程序， 其中包括ApplicationMaster 程序、启动ApplicationMaster 的命令、用户程序等。

2)      ResourceManager 为该应用程序分配第一个Container， 并与对应的NodeManager 通信，要求它在这个Container 中启动应用程序的ApplicationMaster。

3)      ApplicationMaster 首先向ResourceManager 注册， 这样用户可以直接通过ResourceManage 查看应用程序的运行状态，然后它将为各个任务申请资源，并监控它的运行状态，直到运行结束，即重复步骤4~7。

4)      ApplicationMaster 采用轮询的方式通过RPC 协议向ResourceManager 申请和领取资源。

5)      一旦ApplicationMaster 申请到资源后，便与对应的NodeManager 通信，要求它启动任务。

6)      NodeManager 为任务设置好运行环境（包括环境变量、JAR 包、二进制程序等）后，将任务启动命令写到一个脚本中，并通过运行该脚本启动任务。

7)      各个任务通过某个RPC 协议向ApplicationMaster 汇报自己的状态和进度，以让ApplicationMaster 随时掌握各个任务的运行状态，从而可以在任务失败时重新启动任务。在应用程序运行过程中，用户可随时通过RPC 向ApplicationMaster 查询应用程序的当前运行状态。

8)      应用程序运行完成后，ApplicationMaster 向ResourceManager 注销并关闭自己。

## 3. 编程题

1.      当前日志采样格式为，请用你最熟悉的语言编写一个 mapreduce，并计算第四列每个元素出现的个数。

1.  a,b,c,d

2.  b,b,f,e

3.  a,a,c,f

2.      给定 a、b 两个文件，各存放 50 亿个 url，每个 url 各占 64 字节，内存限制是 4G，让你找出 a、b 文件共同的 url？

解答：

方案 1：将大文件分成能够被内存加载的小文件。

可以估计每个文件安的大小为 50G×64=320G，远远大于内存限制的 4G。所以不可能将其完全加载到内存中处理。考虑采取分而治之的方法。

遍历文件 a，对每个 url 求取 ，然后根据所取得的值将 url 分别存储到 1000 个小文件（记为 ）中。这样每个小文件的大约为 300M。

遍历文件 b，采取和 a 相同的方式将 url 分别存储到 1000 各小文件（记为 ）。这样处理后，所有可能相同的 url 都在对应的小文件（ ）中，不对应的小文件不可能有相同的 url。然后我们只要求出 1000 对小文件中相同的 url 即可。

求每对小文件中相同的 url 时，可以把其中一个小文件的 url 存储到 hash\_set 中。然后遍历另一个小文件的每个 url，看其是否在刚才构建的 hash\_set 中，如果是，那么就是共同的 url，存到文件里面就可以了。

方案 2：内存映射成 BIT 最小存储单元。

如果允许有一定的错误率，可以使用 Bloom filter，4G 内存大概可以表示 340 亿 bit。将其中一个文件中的url使用Bloom filter映射为这340亿bit，然后挨个读取另外一个文件的url，检查是否与Bloom filter，如果是，那么该 url 应该是共同的 url（注意会有一定的错误率）。

3.      有 10 个文件，每个文件 1G，每个文件的每一行存放的都是用户的 query，每个文件的 query 都可能重复。要求你按照query 的频度排序。

解答：

方案 1：

顺序读取 10 个文件，按照 hash(query)%10 的结果将 query 写入到另外 10 个文件（记为 ）中。这样新生成的文件每个的大小大约也 1G（假设 hash 函数是随机的）。

找一台内存在 2G 左右的机器，依次对 用 hash\_map(query, query\_count)来统计每个 query 出现的次数。利用快速/堆/归并排序按照出现次数进行排序。将排序好的 query 和对应的 query\_cout 输出到文件中。这样得到了 10 个排好序的文件（记为 ）。对 这 10 个文件进行归并排序（内排序与外排序相结合）。

方案 2：

一般 query 的总量是有限的，只是重复的次数比较多而已，可能对于所有的 query，一次性就可以加入到内存了。这样，我们就可以采用 trie 树/hash\_map 等直接来统计每个 query 出现的次数，然后按出现

次数做快速/堆/归并排序就可以了。

方案 3：

与方案 1 类似，但在做完 hash，分成多个文件后，可以交给多个文件来处理，采用分布式的架构来处理（比如 MapReduce），最后再进行合并。

//一般在大文件中找出出现频率高的，先把大文件映射成小文件，模 1000，在小文件中找到高频的

4.      有一个 1G 大小的一个文件，里面每一行是一个词，词的大小不超过 16 字节，内存限制大小是 1M。返回频数最高的 100 个词。

解答：

方案 1：顺序读文件中，对于每个词 x，取 ，然后按照该值存到 5000 个小文件（记为 ）中。这样每个文件大概是 200k 左右。如果其中的有的文件超过了 1M 大小，还可以按照类似的方法继续往下分，知道分解得到的小文件的大小都不超过 1M。 对每个小文件，统计每个文件中出现的词以及相应的频率（可以采用 trie 树/hash\_map 等），并取出出现频率最大的 100 个词（可以用含 100 个结 点的最小堆），并把 100词及相应的频率存入文件，这样又得到了 5000 个文件。下一步就是把这 5000 个文件进行归并（类似与归并排序）的过程了。

方案2:

1.       将文件逐行读写到另一个文件中，并将每行单词全变成小写

2.       十六次循环执行，将每行单词按照a-z写到不同文件里

3.       最后相同的单词都写在了通一个文件里

4.       再将文件读写到各自另一个文件里，内容是“单词 个数”

5.       定义一个treemap，大小是100，依次插入大的，移除小的

6.       最后得出结果

5.      海量日志数据，提取出某日访问百度次数最多的那个 IP。

解答：

1.       先根据日期在日志文件中提取出ip，根据ip哈希进行分写N个文件。

2.       采用mapreduce的word cont

方案 1：首先是这一天，并且是访问百度的日志中的 IP 取出来，逐个写入到一个大文件中。注意到 IP是 32 位的，最多有 个 IP。同样可以采用映射的方法，比如模 1000，把整个大文件映射为 1000 个小文件，再找出每个小文中出现频率最大的 IP（可以采用 hash\_map 进行频率统计，然后再找出频率最大的几个）及相应的频率。然后再在这 1000 个最大的 IP 中，找出那个频率最大的 IP，即为所求。

6.      在 2.5 亿个整数中找出不重复的整数，内存不足以容纳这 2.5 亿个整数。

解答：

方案 1：采用 2-Bitmap（每个数分配 2bit，00 表示不存在，01 表示出现一次，10 表示多次，11 无意义）进 行，共需内存 内存，还可以接受。然后扫描这 2.5 亿个整数，查看 Bitmap 中相对应位，如果是00 变 01，01 变 10，10 保持不变。所描完事后，查看 bitmap，把对应位是 01 的整数输出即可。

方案 2：也可采用上题类似的方法，进行划分小文件的方法。然后在小文件中找出不重复的整数，并排序。然后再进行归并，注意去除重复的元素。

方案3：

1.       将2.5亿个整数重写到一个文件里，内个整数占一行。

2.       进行对半排序重写到新的文件里，这样最后2.5亿个整数在文件里便是有序的了

3.       读取文本，将不重复的写到一个新的文件里即可。

7.      海量数据分布在 100 台电脑中，想个办法高校统计出这批数据的 TOP10。

解答：

方案 1：（方法不正确，取出来得不一定是top10）

s 在每台电脑上求出 TOP10，可以采用包含 10 个元素的堆完成（TOP10 小，用最大堆，TOP10 大，用最小堆）。比如求 TOP10 大，我们首先取前 10 个元素调整成最小堆，如果发现，然后扫描后面的数据，并与堆顶元素比较，如果比堆顶元素大，那么用该元素替换堆顶，然后再调整为最小堆。最后堆中的元 素就是 TOP10 大。

求出每台电脑上的 TOP10 后，然后把这 100 台电脑上的 TOP10 组合起来，共 1000 个数据，再利用上面类似的方法求出 TOP10 就可以了。

8.      怎么在海量数据中找出重复次数最多的一个？

解答：

方案 1：（同上，方法错误）

先做 hash，然后求模映射为小文件，求出每个小文件中重复次数最多的一个，并记录重复次数。然后找出上一步求出的数据中重复次数最多的一个就是所求（具体参考前面的题）。

正确的方法，先排除

9.      上千万或上亿数据（有重复），统计其中出现次数最多的前 N 个数据。

解答：

10.  1000 万字符串，其中有些是重复的，需要把重复的全部去掉，保留没有重复的字符串。请怎么设计和实现？

解答：

11.  一个文本文件，大约有一万行，每行一个词，要求统计出其中最频繁出现的前 10 个词，请给出思想，给出时间复杂度分析。

解答：

方案 1：这题是考虑时间效率。用 trie 树统计每个词出现的次数，时间复杂度是 O(n\*le)（le 表示单词的平准长 度）。然后是找出出现最频繁的前 10 个词，可以用堆来实现，前面的题中已经讲到了，时间复杂度是 O(n\*lg10)。所以总的时间复杂度，是 O(n\*le)与 O(n\*lg10)中较大的哪一个。

12.  一个文本文件，找出前 10 个经常出现的词，但这次文件比较长，说是上亿行或十亿行，总之无法一次读入内存，问最优解。

解答：

13.  100w 个数中找出最大的 100 个数。

解答：

# 二、Hbase

## 1. 选择题

### 1.1 基础

1.HBase 来源于哪篇博文？ C

A TheGoogle File System

B MapReduce

C BigTable

D Chubby

2.下面对 HBase 的描述哪些是正确的？ B、C、D

A 不是开源的

B 是面向列的

C 是分布式的

D 是一种 NoSQL 数据库

3.      HBase 依靠（）存储底层数据 A

A HDFS

B Hadoop

C Memory

D MapReduce

4.HBase 依赖（）提供消息通信机制 A

A Zookeeper

B Chubby

C RPC

D Socket

5.HBase 依赖（）提供强大的计算能力 D

A Zookeeper

B Chubby

C RPC

D MapReduce

6.MapReduce 与 HBase 的关系，哪些描述是正确的？ B、C

A 两者不可或缺，MapReduce 是 HBase 可以正常运行的保证

B 两者不是强关联关系，没有 MapReduce，HBase 可以正常运行

C MapReduce 可以直接访问 HBase

D 它们之间没有任何关系

7.下面哪些选项正确描述了HBase 的特性？ A、B、C、D

A 高可靠性

B 高性能

C 面向列

D 可伸缩

8.下面哪些概念是 HBase 框架中使用的？A、C

A HDFS

B GridFS

C Zookeeper

D EXT3

### 1.2  Hbase核心

1.HFile 数据格式中的 Data 字段用于（）。A

A 存储实际的 KeyValue 数据

B 存储数据的起点

C 指定字段的长度

D 存储数据块的起点

7.HFile 数据格式中的 MetaIndex 字段用于（）。D

A Meta 块的长度

B Meta 块的结束点

C Meta 块数据内容

D Meta 块的起始点

8.HFile 数据格式中的 Magic 字段用于（）。A

A 存储随机数，防止数据损坏

B 存储数据的起点

C 存储数据块的起点

D 指定字段的长度

9. HFile 数据格式中的 KeyValue 数据格式，下列选项描述正确的是（）。A、D

A 是 byte[]数组

B 没有固定的结构

C 数据的大小是定长的

D 有固定的结构

10.  HFile 数据格式中的 KeyValue 数据格式中 Value 部分是（）。C

A 拥有复杂结构的字符串

B 字符串

C 二进制数据

D 压缩数据

### 1.3  HBase 高级应用介绍

1.HBase 中的批量加载底层使用（）实现。A

A MapReduce

B Hive

C Coprocessor

D BloomFilter

分析：在进行数据传输中，批量加载数据到HBase集群有多种方式，比如通过HBase API进行批量写入数据、使用Sqoop工具批量导数到HBase集群、使用MapReduce批量导入等。这些方式，在导入数据的过程中，如果数据量过大，可能耗时会比较严重或者占用HBase集群资源较多（如磁盘IO、HBase Handler数等）。

2.HBase 性能优化包含下面的哪些选项？A、B、C、D

A 读优化

B 写优化

C 配置优化

D JVM 优化

3.Rowkey 设计的原则，下列哪些选项的描述是正确的？A、B、C

A 尽量保证越短越好

B 可以使用汉字

C 可以使用字符串

D 本身是无序的

分析：可以用中文，读取的时候把binary 转成String展现就行了

4.HBase 构建二级索引的实现方式有哪些？ A、B

A MapReduce

B Coprocessor

C BloomFilter

D Filter

分析：常见的二级索引方案

HBase的一级索引就是rowkey，我们只能通过rowkey进行检索。如果我们相对hbase里面列族的列列进行一些组合查询，就需要采用HBase的二级索引方案来进行多条件的查询。

1. MapReduce方案

2. ITHBASE（Indexed-Transanctional HBase）方案

3. IHBASE（Index HBase）方案

4. Hbase Coprocessor(协处理器)方案

5. Solr+hbase方案

6. CCIndex（complementalclustering index）方案

5.关于 HBase 二级索引的描述，哪些是正确的？A、B

A 核心是倒排表

B 二级索引概念是对应 Rowkey 这个“一级”索引

C 二级索引使用平衡二叉树

D 二级索引使用 LSM 结构

6.下列关于 Bloom Filter 的描述正确的是？A、C

A 是一个很长的二进制向量和一系列随机映射函数

B 没有误算率

C 有一定的误算率

D 可以在 Bloom Filter 中删除元素

### 1.4  HBase 安装、部署、启动

1.HBase 官方版本可以安装在什么操作系统上？A、B、C

A CentOS

B Ubuntu

C RedHat

D Windows

2. HBase 虚拟分布式模式需要（）个节点？A

A 1

B 2

C 3

D 最少 3 个

3.HBase 分布式模式最好需要（）个节点？C

A 1

B 2

C 3

D 最少

4.下列哪些选项是安装 HBase 前所必须安装的？A、B

A 操作系统

B JDK

C ShellScript

D JavaCode

5.解压.tar.gz 结尾的 HBase 压缩包使用的 Linux 命令是？A

A tar-zxvf

B tar -zx

C tar -s

D tar -nf

## 2. 问答题

### 2.1  Hbase基础

1.      介绍一下 hbase 过滤器

解答：

2.      hbase 集群安装注意事项

解答：

3.      hbase的rowkey怎么创建好？列族怎么创建比较好？

解答：

hbase存储时，数据按照Row key的字典序(byte order)排序存储。设计key时，要充分排序存储这个特性，将经常一起读取的行存储放到一起。(位置相关性)

一个列族在数据底层是一个文件，所以将经常一起查询的列放到一个列族中，列族尽量少，减少文件的寻址时间。

因为hbase是列式数据库，列非表schema的一部分，所以在设计初期只需要考虑rowkey 和 columnFamily即可，rowkey有位置相关性，所以如果数据是练习查询的，最好对同类数据加一个前缀，而每个columnFamily实际上在底层是一个文件，那么文件越小，查询越快，所以讲经常一起查询的列设计到一个列簇，但是列簇不宜过多。

 Rowkey长度原则

Rowkey是一个二进制码流，Rowkey的长度被很多开发者建议说设计在10~100个字节，不过建议是越短越好，不要超过16个字节。

原因如下：

（1）数据的持久化文件HFile中是按照KeyValue存储的，如果Rowkey过长比如100个字节，1000万列数据光Rowkey就要占用100\*1000万=10亿个字节，将近1G数据，这会极大影响HFile的存储效率；

（2）MemStore将缓存部分数据到内存，如果Rowkey字段过长内存的有效利用率会降低，系统将无法缓存更多的数据，这会降低检索效率。因此Rowkey的字节长度越短越好。

（3）目前操作系统是都是64位系统，内存8字节对齐。控制在16个字节，8字节的整数倍利用操作系统的最佳特性。

Rowkey散列原则

如果Rowkey是按时间戳的方式递增，不要将时间放在二进制码的前面，建议将Rowkey的高位作为散列字段，由程序循环生成，低位放时间字段，这样将提高数据均衡分布在每个Regionserver实现负载均衡的几率。如果没有散列字段，首字段直接是时间信息将产生所有新数据都在一个 RegionServer上堆积的热点现象，这样在做数据检索的时候负载将会集中在个别RegionServer，降低查询效率。

Rowkey唯一原则

必须在设计上保证其唯一性。

4.      简述Hbase性能优化的思路

解答：

1、在库表设计的时候，尽量考虑rowkey和columnfamily的特性

2、进行hbase集群的调优

5.      简述Hbase filter的实现原理是什么？结合实际项目经验，写出几个使用filter的场景。

解答：

hbase的filter是通过scan设置的，所以是基于scan的查询结果进行过滤。

1.在进行订单开发的时候，我们使用rowkeyfilter过滤出某个用户的所有订单

2.在进行云笔记开发时，我们使用rowkey过滤器进行redis数据的恢复。

6.      ROWKEY的后缀匹配怎么实现？列如ROWKEY是yyyyMMDD-UserID形式，如UserID为条件查询数据，怎么实现。

解答：

7.      HBase的检索支持3种方式：

解答：

（1） 通过单个Rowkey访问，即按照某个Rowkey键值进行get操作，这样获取唯一一条记录；

（2） 通过Rowkey的range进行scan，即通过设置startRowKey和endRowKey，在这个范围内进行扫描。这样可以按指定的条件获取一批记录；

（3） 全表扫描，即直接扫描整张表中所有行记录。

8.      简述HBase的瓶颈

解答：

HBase的瓶颈就是硬盘传输速度。HBase的操作，它可以往数据里面insert,也可以update一些数据，但update的实际上也是insert，只是插入一个新的时间戳的一行。Delete数据，也是insert，只是insert一行带有delete标记的一行。Hbase的所有操作都是追加插入操作。Hbase是一种日志集数据库。它的存储方式，像是日志文件一样。它是批量大量的往硬盘中写，通常都是以文件形式的读写。这个读写速度，就取决于硬盘与机器之间的传输有多快。而Oracle的瓶颈是硬盘寻道时间。它经常的操作时随机读写。要update一个数据，先要在硬盘中找到这个block，然后把它读入内存，在内存中的缓存中修改，过段时间再回写回去。由于你寻找的block不同，这就存在一个随机的读。硬盘的寻道时间主要由转速来决定的。而寻道时间，技术基本没有改变，这就形成了寻道时间瓶颈。

# 三、Zookeeper

## 1. 选择题

1.      下面与 Zookeeper 类似的框架是？D

A Protobuf

B Java

C Kafka

D Chubby

## 2. 问答题

1.      写出你对 zookeeper 的理解

解答：

# 五、Hive

## 1. 选择题

## 2. 问答题

1.      hive 有哪些方式保存元数据，各有哪些特点？

解答：

1、内存数据库derby，安装小，但是数据存在内存，不稳定

2、mysql数据库，数据存储模式可以自己设置，持久化好，查看方便。

2.      hive内部表和外部表的区别

解答：

内部表：加载数据到hive所在的hdfs目录，删除时，元数据和数据文件都删除

外部表：不加载数据到hive所在的hdfs目录，删除时，只删除表结构。

3.      生产环境中为什么建议使用外部表？

解答：

1、因为外部表不会加载数据到hive，减少数据传输、数据还能共享。

2、hive不会修改数据，所以无需担心数据的损坏

3、删除表时，只删除表结构、不删除数据。

4.      你们数据库怎么导入hive 的,有没有出现问题

解答：

在导入hive的时候，如果数据库中有blob或者text字段，会报错。有个参数limit

5.      简述Hive中的虚拟列作用是什么，使用它的注意事项

解答：

Hive提供了三个虚拟列：

INPUT\_\_FILE\_\_NAME

BLOCK\_\_OFFSET\_\_INSIDE\_\_FILE

ROW\_\_OFFSET\_\_INSIDE\_\_BLOCK

但ROW\_\_OFFSET\_\_INSIDE\_\_BLOCK默认是不可用的，需要设置hive.exec.rowoffset为true才可以。可以用来排查有问题的

6.      hive partition分区

解答：

分区表，动态分区

7.      insert into 和 override write区别？

解答：

insert into：将某一张表中的数据写到另一张表中

override write：覆盖之前的内容。

8.      假如一个分区的数据主部错误怎么通过hivesql删除hdfs

解答：

alter table ptable drop partition (daytime='20140911',city='bj');

元数据，数据文件都删除，但目录daytime= 20140911还在

9.      Hive里面用什么代替in查询

解答：

提示：Hive中的left semi join替换sql中的in操作

# 六、Flume

## 1. 选择题

## 2. 问答题

1.      flume管道内存，flume宕机了数据丢失怎么解决

解答：

2、Flume的channel分为很多种，可以将数据写入到文件

3、防止非首个agent宕机的方法数可以做集群或者主备

4.      flume配置方式，flume集群（问的很详细）

解答：

Flume的配置围绕着source、channel、sink叙述，flume的集群是做在agent上的，而非机器上。

5.      flume不采集Nginx日志，通过Logger4j采集日志，优缺点是什么？

解答：

优点：Nginx的日志格式是固定的，但是缺少sessionid，通过logger4j采集的日志是带有sessionid的，而session可以通过redis共享，保证了集群日志中的同一session落到不同的tomcat时，sessionId还是一样的，而且logger4j的方式比较稳定，不会宕机。

缺点：不够灵活，logger4j的方式和项目结合过于紧密，而flume的方式比较灵活，拔插式比较好，不会影响项目性能。

6.      flume和kafka采集日志区别，采集日志时中间停了，怎么记录之前的日志。

解答：

Flume采集日志是通过流的方式直接将日志收集到存储层，而kafka试讲日志缓存在kafka集群，待后期可以采集到存储层。

Flume采集中间停了，可以采用文件的方式记录之前的日志，而kafka是采用offset的方式记录之前的日志。

# 七、Sqoop

## 1. 选择题

## 2. 问答题

1.      命令：

sqoop import --connect jdbc:mysql://192.168.56.204:3306/sqoop --username hive --password hive --table jobinfo --target-dir /sqoop/test7 --inline-lob-limit 16777216 --fields-terminated-by '\t' -m 2

 sqoop create-hive-table --connect jdbc:mysql://192.168.56.204:3306/sqoop --table jobinfo --username hive --password hive --hive-table sqtest --fields-terminated-by "\t" --lines-terminated-by "\n";

2.      sqoop在导入数据到mysql中，如何让数据不重复导入？如果存在数据问题sqoop如何处理？

解答：

Sqoop是一个用来将Hadoop和关系型数据库中的数据相互转移的工具，可以将一个关系型数据库（例如 ： MySQL ,Oracle ,Postgres等）中的数据导进到Hadoop的HDFS中，也可以将HDFS的数据导进到关系型数据库中。

首先需以下要准备：

第一：hadoop的NameNode节点下lib文件夹中要有相应数据库驱动的jar包和sqoop的jar包。

第二：预先在相应的数据库创建Table，注：在HDFS的某个目录上的数据格式要和相应的表中的字段数量一致。

3.10其他

3.10.1 Redis

1.      Redis,传统数据库,hbase,hive 每个之间的区别

解答：

redis：分布式缓存，强调缓存，内存中数据

传统数据库：注重关系

hbase：列式数据库，无法做关系数据库的主外键，用于存储海量数据，底层基于hdfs

hive：数据仓库工具，底层是mapreduce。不是数据库，不能用来做用户的交互存储

# 八、数据库

1.      反向索引

解答：

倒排索引(Inverted index)

适用范围：搜索引擎，关键字查询

基本原理及要点：为何叫倒排索引？一种索引方法，被用来存储在全文搜索下某个单词在一个文档或者一组文档中的存储位置的映射。

以英文为例，下面是要被索引的文本：

T0 = “it is what it is”

T1 = “what is it”

T2 = “it is a banana”

我们就能得到下面的反向文件索引：

“a”: {2}

“banana”: {2}

“is”: {0, 1, 2}

“it”: {0, 1, 2}

“what”: {0, 1}

检索的条件”what”,”is”和”it”将对应集合的交集。

正向索引开发出来用来存储每个文档的单词的列表。正向索引的查询往往满足每个文档有序 频繁的全文查询和每个单词在校验文档中的验证这样的查询。在正向索引中，文档占据了中心的位置，每个文档指向了一个它所包含的索引项的序列。也就是说文档指向了它包含的那些单词，而反向索引则是单词指向了包含它的文档，很容易看到这个反向的关系。