1.列出安装hadoop流程步骤

a) 创建hadoop账号

b) 更改ip

c) 安装java 更改/etc/profile 配置环境变量

d) 修改host文件域名

e) 安装ssh 配置无密码登录

f) 解压hadoop

g) 配置hadoop  conf下面的配置文件

h) Hadoop namenode -format  格式化

i) Start 启动

2.列出hadoop集群启动中的所有进程和进程的作用

a) Namenode 管理集群  记录namenode文件信息

b) Secondname 可以做备份  对一定范围内的数据做快照

c) Datanode  存储数据

d) Jobtarcker 管理任务  分配任务

e) Tasktracker   执行任务

3.启动报nameNode错误 如何解决

a) 检查hdfs有没有启动成功

b) 检查输入文件是不是存在

4.写出下列执行命令

杀死一个job

Hadoop job -list  取得job id

Hadoop job kill job id

删除hdfs上的 /temp/aa 目录

Hadoop -daemon。Sh start datanode

加入一个新的节点或删除一个节点  刷新集群状态的命令

5.列出你所知道的调肚脐  说明其工作方法

a) Fifo schedular 默认的调肚脐  先进先出

b) Capacity schedular  计算能力调肚脐  选择占用内存小  优先级高的

c) Fair schedular 调肚脐  公平调肚脐  所有job 占用相同资源

6.列出开发map/reduce 元数据存储

a)

7.用你最熟悉的语言辨析一个map reduce 计算第四个原色的个数

a) Wordcount

8.你认为java streating pipe 开发map reduce 优缺点

a) Java 编写map reduce可以实现复杂的逻辑  如果需求简单  则显得繁琐

b) Hivesql 基本都是针对hive中表数据进行编写  对复杂的逻辑很难实现

9.Hive 有哪些保存元数据的方式  并有哪些特点

a) 内存数据库  derby

b) 本地[**MySQL**](http://lib.csdn.net/base/14) 常用

c) 远程mysql

10.简述hadoop怎么样实现二级缓存

11.简述hadoop实现join的及各种方式

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。

2.2 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。

2.3 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。

12.

13.Java 非递归 二分查找

1. public class BinarySearchClass

2. {

3.

4.     public static int binary\_search(int[] array, int value)

5.     {

6.         int beginIndex = 0;// 低位下标

7.         int endIndex = array.length - 1;// 高位下标

8.         int midIndex = -1;

9.         while (beginIndex <= endIndex) {

10.             midIndex = beginIndex + (endIndex - beginIndex) / 2;//防止溢出

11.             if (value == array[midIndex]) {

12.                 return midIndex;

13.             } else if (value < array[midIndex]) {

14.                 endIndex = midIndex - 1;

15.             } else {

16.                 beginIndex = midIndex + 1;

17.             }

18.         }

19.         return -1;

20.         //找到了，返回找到的数值的下标，没找到，返回-1

21.     }

22.

23.

24.     //start 提示：自动阅卷起始唯一标识，请勿删除或增加。

25.     public static void main(String[] args)

26.     {

27.         System.out.println("Start...");

28.         int[] myArray = new int[] { 1, 2, 3, 5, 6, 7, 8, 9 };

29.         System.out.println("查找数字8的下标：");

30.         System.out.println(binary\_search(myArray, 8));

31.     }

14.

15.Combiner 和partition的作用

16.combine分为map端和reduce端，作用是把同一个key的键值对合并在一起，可以自定义的。  
combine函数把一个map函数产生的<key,value>对（多个key,value）合并成一个新的<key2,value2>.将新的<key2,value2>作为输入到reduce函数中  
这个value2亦可称之为values，因为有多个。这个合并的目的是为了减少网络传输。  
  
partition是分割map每个节点的结果，按照key分别映射给不同的reduce，也是可以自定义的。这里其实可以理解归类。  
我们对于错综复杂的数据归类。比如在动物园里有牛羊鸡鸭鹅，他们都是混在一起的，但是到了晚上他们就各自牛回牛棚，羊回羊圈，鸡回鸡窝。partition的作用就是把这些数据归类。只不过在写程序的时候，mapreduce使用哈希HashPartitioner帮我们归类了。这个我们也可以自定义。  
  
shuffle就是map和reduce之间的过程，包含了两端的combine和partition。  
  
  
Map的结果，会通过partition分发到Reducer上，Reducer做完Reduce操作后，通过OutputFormat，进行输出  
shuffle阶段的主要函数是fetchOutputs(),这个函数的功能就是将map阶段的输出，copy到reduce 节点本地

17.俩个文件用linux命令完成下列工作

a) 俩个文件各自的ip数  和综述

b) 出现在b文件  没有出现在a文件的ip

c) 那个name的次数  和他对一个的ip

1.hive内部表和外部表区别

2.1、在导入数据到外部表，数据并没有移动到自己的数据仓库目录下，也就是说外部表中的数据并不是由它自己来管理的！而表则不一样；  
2、在删除表的时候，Hive将会把属于表的元数据和数据全部删掉；而删除外部表的时候，Hive仅仅删除外部表的元数据，数据是不会删除的！  
那么，应该如何选择使用哪种表呢？在大多数情况没有太多的区别，因此选择只是个人喜好的问题。但是作为一个经验，如果所有处理都需要由Hive完成，那么你应该创建表，否则使用外部表！

3.Hbase rowkey怎么创建表交好  列祖怎么创建比较好

HBase是一个分布式的、面向列的数据库，它和一般关系型数据库的最大区别是：HBase很适合于存储非结构化的数据，还有就是它基于列的而不是基于行的模式。

既然HBase是采用KeyValue的列存储，那Rowkey就是KeyValue的Key了，表示唯一一行。Rowkey也是一段二进制码流，最大长度为64KB，内容可以由使用的用户自定义。数据加载时，一般也是根据Rowkey的二进制序由小到大进行的。

HBase是根据Rowkey来进行检索的，系统通过找到某个Rowkey (或者某个 Rowkey 范围)所在的Region，然后将查询数据的请求路由到该Region获取数据。HBase的检索支持3种方式：

（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。

HBASE按单个Rowkey检索的效率是很高的，耗时在1毫秒以下，每秒钟可获取1000~2000条记录，不过非key列的查询很慢。

2 HBase的RowKey设计

2.1 设计原则

2.1.1 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。

[回到顶部](http://blog.csdn.net/qq_26562641/article/details/50233937#_labelTop)

2.1.2 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。

[回到顶部](http://blog.csdn.net/qq_26562641/article/details/50233937#_labelTop)

2.1.3 Rowkey唯一原则

必须在设计上保证其唯一性。

2.2 应用场景

基于Rowkey的上述3个原则，应对不同应用场景有不同的Rowkey设计建议。

[回到顶部](http://blog.csdn.net/qq_26562641/article/details/50233937#_labelTop)

2.2.1 针对事务数据Rowkey设计

事务数据是带时间属性的，建议将时间信息存入到Rowkey中，这有助于提示查询检索速度。对于事务数据建议缺省就按天为数据建表，这样设计的好处是多方面的。按天分表后，时间信息就可以去掉日期部分只保留小时分钟毫秒，这样4个字节即可搞定。加上散列字段2个字节一共6个字节即可组成唯一 Rowkey。如下图所示：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 事务数据Rowkey设计 | | | | | | |
| 第0字节 | 第1字节 | 第2字节 | 第3字节 | 第4字节 | 第5字节 | … |
| 散列字段 | | 时间字段(毫秒) | | | | 扩展字段 |
| 0~65535(0x0000~0xFFFF) | | 0~86399999(0x00000000~0x05265BFF) | | | |  |

这样的设计从操作系统内存管理层面无法节省开销，因为64位操作系统是必须8字节对齐。但是对于持久化存储中Rowkey部分可以节省25%的开销。也许有人要问为什么不将时间字段以主机字节序保存，这样它也可以作为散列字段了。这是因为时间范围内的数据还是尽量保证连续，相同时间范围内的数据查找的概率很大，对查询检索有好的效果，因此使用独立的散列字段效果更好，对于某些应用，我们可以考虑利用散列字段全部或者部分来存储某些数据的字段信息，只要保证相同散列值在同一时间（毫秒）唯一。

[回到顶部](http://blog.csdn.net/qq_26562641/article/details/50233937#_labelTop)

2.2.2 针对统计数据的Rowkey设计

统计数据也是带时间属性的，统计数据最小单位只会到分钟（到秒预统计就没意义了）。同时对于统计数据我们也缺省采用按天数据分表，这样设计的好处无需多说。按天分表后，时间信息只需要保留小时分钟，那么0~1400只需占用两个字节即可保存时间信息。由于统计数据某些维度数量非常庞大，因此需要4个字节作为序列字段，因此将散列字段同时作为序列字段使用也是6个字节组成唯一Rowkey。如下图所示：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 统计数据Rowkey设计 | | | | | | |
| 第0字节 | 第1字节 | 第2字节 | 第3字节 | 第4字节 | 第5字节 | … |
| 散列字段(序列字段） | | | | 时间字段(分钟) | | 扩展字段 |
| 0x00000000~0xFFFFFFFF) | | | | 0~1439(0x0000~0x059F) | |  |

同样这样的设计从操作系统内存管理层面无法节省开销，因为64位操作系统是必须8字节对齐。但是对于持久化存储中Rowkey部分可以节省25%的开销。预统计数据可能涉及到多次反复的重计算要求，需确保作废的数据能有效删除，同时不能影响散列的均衡效果，因此要特殊处理。

[回到顶部](http://blog.csdn.net/qq_26562641/article/details/50233937#_labelTop)

2.2.3 针对通用数据的Rowkey设计

通用数据采用自增序列作为唯一主键，用户可以选择按天建分表也可以选择单表模式。这种模式需要确保同时多个入库加载模块运行时散列字段（序列字段）的唯一性。可以考虑给不同的加载模块赋予唯一因子区别。设计结构如下图所示。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 通用数据Rowkey设计 | | | | |
| 第0字节 | 第1字节 | 第2字节 | 第3字节 | … |
| 散列字段(序列字段） | | | | 扩展字段（控制在12字节内） |
| 0x00000000~0xFFFFFFFF) | | | | 可由多个用户字段组成 |

[回到顶部](http://blog.csdn.net/qq_26562641/article/details/50233937#_labelTop)

2.2.4 支持多条件查询的RowKey设计

HBase按指定的条件获取一批记录时，使用的就是scan方法。 scan方法有以下特点：

（1）scan可以通过setCaching与setBatch方法提高速度（以空间换时间）；

（2）scan可以通过setStartRow与setEndRow来限定范围。范围越小，性能越高。

通过巧妙的RowKey设计使我们批量获取记录集合中的元素挨在一起（应该在同一个Region下），可以在遍历结果时获得很好的性能。

（3）scan可以通过setFilter方法添加过滤器，这也是分页、多条件查询的基础。

在满足长度、三列、唯一原则后，我们需要考虑如何通过巧妙设计RowKey以利用scan方法的范围功能，使得获取一批记录的查询速度能提高。下例就描述如何将多个列组合成一个RowKey，使用scan的range来达到较快查询速度。

例子：

我们在表中存储的是文件信息，每个文件有5个属性：文件id（long，全局唯一）、创建时间（long）、文件名（String）、分类名（String）、所有者（User）。

我们可以输入的查询条件：文件创建时间区间（比如从20120901到20120914期间创建的文件），文件名（“中国好声音”），分类（“综艺”），所有者（“浙江卫视”）。

假设当前我们一共有如下文件：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | CreateTime | Name | Category | UserID |
| 1 | 20120902 | 中国好声音第1期 | 综艺 | 1 |
| 2 | 20120904 | 中国好声音第2期 | 综艺 | 1 |
| 3 | 20120906 | 中国好声音外卡赛 | 综艺 | 1 |
| 4 | 20120908 | 中国好声音第3期 | 综艺 | 1 |
| 5 | 20120910 | 中国好声音第4期 | 综艺 | 1 |
| 6 | 20120912 | 中国好声音选手采访 | 综艺花絮 | 2 |
| 7 | 20120914 | 中国好声音第5期 | 综艺 | 1 |
| 8 | 20120916 | 中国好声音录制花絮 | 综艺花絮 | 2 |
| 9 | 20120918 | 张玮独家专访 | 花絮 | 3 |
| 10 | 20120920 | 加多宝凉茶广告 | 综艺广告 | 4 |

这里UserID应该对应另一张User表，暂不列出。我们只需知道UserID的含义：

1代表 浙江卫视； 2代表 好声音剧组； 3代表 XX微博； 4代表赞助商。调用查询接口的时候将上述5个条件同时输入find(20120901,20121001,”中国好声音”,”综艺”,”浙江卫视”)。此时我们应该得到记录应该有第1、2、3、4、5、7条。第6条由于不属于“浙江卫视”应该不被选中。我们在设计RowKey时可以这样做：采用 UserID + CreateTime + FileID组成RowKey，这样既能满足多条件查询，又能有很快的查询速度。

需要注意以下几点：

（1）每条记录的RowKey，每个字段都需要填充到相同长度。假如预期我们最多有10万量级的用户，则userID应该统一填充至6位，如000001，000002…

（2）结尾添加全局唯一的FileID的用意也是使每个文件对应的记录全局唯一。避免当UserID与CreateTime相同时的两个不同文件记录相互覆盖。

按照这种RowKey存储上述文件记录，在HBase表中是下面的结构：

rowKey（userID 6 + time 8 + fileID 6） name category ….

00000120120902000001

00000120120904000002

00000120120906000003

00000120120908000004

00000120120910000005

00000120120914000007

00000220120912000006

00000220120916000008

00000320120918000009

00000420120920000010

怎样用这张表？

在建立一个scan对象后，我们setStartRow(00000120120901)，setEndRow(00000120120914)。

这样，scan时只扫描userID=1的数据，且时间范围限定在这个指定的时间段内，满足了按用户以及按时间范围对结果的筛选。并且由于记录集中存储，性能很好。

然后使用 SingleColumnValueFilter（org.apache.hadoop.hbase.filter.SingleColumnValueFilter），共4个，分别约束name的上下限，与category的上下限。满足按同时按文件名以及分类名的前缀匹配。

4.

5.Mapreduce怎么处理数据倾斜

6.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。  
  
(1)设置一个hash份数N，用来对条数众多的key进行打散。  
  
(2)对有多条重复key的那份数据进行处理：从1到N将数字加在key后面作为新key，如果需要和另一份数据关联的话，则要重写比较类和分发类（方法如上篇《hadoop job解决大数据量关联的一种方法》）。如此实现多条key的平均分发。  
  
int iNum = iNum % iHashNum;  
  
String strKey = key + CTRLC + String.valueOf(iNum) + CTRLB + “B”;  
  
  
（3）上一步之后，key被平均分散到很多不同的reduce节点。如果需要和其他数据关联，为了保证每个reduce节点上都有关联的key，对另一份单一key的数据进行处理：循环的从1到N将数字加在key后面作为新key  
  
for(int i = 0; i < iHashNum; ++i){  
  
String strKey =key + CTRLC + String.valueOf(i) ;  
  
output.collect(new Text(strKey), new Text(strValues));}  
  
以此解决数据倾斜的问题，经试验大大减少了程序的运行时间。但此方法会成倍的增加其中一份数据的数据量，以增加shuffle数据量为代价，所以使用此方法时，要多次试验，取一个最佳的hash份数值。  
  
======================================  
  
用上述的方法虽然可以解决数据倾斜，但是当关联的数据量巨大时，如果成倍的增长某份数据，会导致reduce shuffle的数据量变的巨大，得不偿失，从而无法解决运行时间慢的问题。  
  
有一个新的办法可以解决 成倍增长数据 的缺陷：  
  
在两份数据中找共同点，比如两份数据里除了关联的字段以外，还有另外相同含义的字段，如果这个字段在所有log中的重复率比较小，则可以用这个字段作为计算hash的值，如果是数字，可以用来模hash的份数，如果是字符可以用hashcode来模hash的份数（当然数字为了避免落到同一个reduce上的数据过多，也可以用hashcode），这样如果这个字段的值分布足够平均的话，就可以解决上述的问题。-

7.Hadoop框架中怎么优化

Storm用于处理高速、大型数据流的分布式实时计算系统。为Hadoop添加了可靠的实时[数据处理](https://www.baidu.com/s?wd=%E6%95%B0%E6%8D%AE%E5%A4%84%E7%90%86&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1Y3ryn1mHm1mW9-nj9BrHT40AP8IA3qPjfsn1bkrjKxmLKz0ZNzUjdCIZwsrBtEXh9GuA7EQhF9pywdQhPEUiqkIyN1IA-EUBt3P1nkrHmkPjR3n1TdPHmYnHc)功能  
Spark采用了内存计算。从多[迭代](https://www.baidu.com/s?wd=%E8%BF%AD%E4%BB%A3&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1Y3ryn1mHm1mW9-nj9BrHT40AP8IA3qPjfsn1bkrjKxmLKz0ZNzUjdCIZwsrBtEXh9GuA7EQhF9pywdQhPEUiqkIyN1IA-EUBt3P1nkrHmkPjR3n1TdPHmYnHc)[批处理](https://www.baidu.com/s?wd=%E6%89%B9%E5%A4%84%E7%90%86&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1Y3ryn1mHm1mW9-nj9BrHT40AP8IA3qPjfsn1bkrjKxmLKz0ZNzUjdCIZwsrBtEXh9GuA7EQhF9pywdQhPEUiqkIyN1IA-EUBt3P1nkrHmkPjR3n1TdPHmYnHc)出发，允许将数据载入内存作反复查询，此外还融合数据仓库，流处理和图形计算等多种计算范式。Spark构建在[HDFS](https://www.baidu.com/s?wd=HDFS&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1Y3ryn1mHm1mW9-nj9BrHT40AP8IA3qPjfsn1bkrjKxmLKz0ZNzUjdCIZwsrBtEXh9GuA7EQhF9pywdQhPEUiqkIyN1IA-EUBt3P1nkrHmkPjR3n1TdPHmYnHc)上，能与Hadoop很好的结合。它的RDD是一个很大的特点。  
Hadoop当前大[数据管理](https://www.baidu.com/s?wd=%E6%95%B0%E6%8D%AE%E7%AE%A1%E7%90%86&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1Y3ryn1mHm1mW9-nj9BrHT40AP8IA3qPjfsn1bkrjKxmLKz0ZNzUjdCIZwsrBtEXh9GuA7EQhF9pywdQhPEUiqkIyN1IA-EUBt3P1nkrHmkPjR3n1TdPHmYnHc)标准之一，运用在当前很多商业应用系统。可以轻松地集成结构化、半结构化甚至非结构化数据集。

8.

9.Hbase内部是什么机制

在HBase 中无论是增加新行还是修改已有的行，其内部流程都是相同的。HBase 接到命令后存下变化信息，或者写入失败抛出异常。默认情况下，执行写入时会写到两个地方：预写式日志（write-ahead log，也称HLog）和MemStore（见图2-1）。HBase 的默认方式是把写入动作记录在这两个地方，以保证数据持久化。只有当这两个地方的变化信息都写入并确认后，才认为写动作完成。

MemStore 是内存里的写入缓冲区，HBase 中数据在永久写入硬盘之前在这里累积。当MemStore 填满后，其中的数据会刷写到硬盘，生成一个HFile。HFile 是HBase 使用的底层存储格式。HFile 对应于列族，一个列族可以有多个HFile，但一个HFile 不能存储多个列族的数据。在集群的每个节点上，每个列族有一个MemStore。

大型分布式系统中硬件故障很常见，HBase 也不例外。设想一下，如果MemStore还没有刷写，服务器就崩溃了，内存中没有写入硬盘的数据就会丢失。HBase 的应对办法是在写动作完成之前先写入WAL。HBase 集群中每台服务器维护一个WAL 来记录发生的变化。WAL 是底层文件系统上的一个文件。直到WAL 新记录成功写入后，写动作才被认为成功完成。这可以保证HBase 和支撑它的文件系统满足持久性。大多数情况下，HBase 使用Hadoop 分布式文件系统（HDFS）来作为底层文件系统。

如果HBase 服务器宕机，没有从MemStore 里刷写到HFile 的数据将可以通过回放WAL 来恢复。你不需要手工执行。Hbase 的内部机制中有恢复流程部分来处理。每台HBase 服务器有一个WAL，这台服务器上的所有表（和它们的列族）共享这个WAL。

你可能想到，写入时跳过WAL 应该会提升写性能。但我们不建议禁用WAL，除非你愿意在出问题时丢失数据。如果你想测试一下，如下代码可以禁用WAL：

注意：不写入WAL 会在RegionServer 故障时增加丢失数据的风险。关闭WAL，出现故障时HBase 可能无法恢复数据，没有刷写到硬盘的所有写入数据都会丢失。

10.

11.我们在开发分布式计算jiob的是不是可以去掉reduce

由于MapReduce计算输入和输出都是基于HDFS文件，所以大多数公司的做法是把mysql或sqlserver的数据导入到HDFS，计算完后再导出到常规的数据库中，这是MapReduce不够灵活的地方之一。 MapReduce优势在于提供了比较简单的分布式计算编程模型，使开发此类程序变得非常简单，像之前的MPI编程就相当复杂。

狭隘的来讲，MapReduce是把计算任务给规范化了，它可以等同于小和尚中Worker的业务逻辑部分。 MapReduce把业务逻辑给拆分成2个大部分，Map和Reduce，可以先在Map部分把任务计算一半后，扔给Reduce部分继续后面的计算。 当然在Map部分把计算任务全做完也是可以的。 关于Mapreduce实现细节部分不多解释，有兴趣的同学可以查相关资料或看下楼主之前的C#模拟实现的博客【[探索C#之微型MapReduce](http://www.cnblogs.com/mushroom/p/4521221.html)】。

如果把小明产品经理的需求放到Hadoop来做，其处理流程大致如下：

1. 把100G数据导入到HDFS

2. 按照Mapreduce的接口编写处理逻辑，分Map、Reduce两部分。

3. 把程序包提交到Mapreduce平台上，存储在HDFS里。

4. 平台中有个叫Jobtracker进程的角色进行分发任务。 这个类似小和尚的Master负载调度管理。

5. 如果有5台机器进行计算的话，就会提前运行5个叫TaskTracker的slave进程。 这类似小和尚worker的分离版，平台把程序和业务逻辑进行分离了， 简单来说就是在机器上运行个独立进程，它能动态加载、执行jar或dll的业务逻辑代码。

6. Jobtracker把任务分发到TaskTracker后，TaskTracker把开始动态加载jar包，创建个独立进程执行Map部分，然后把结果写入到HDFS上。

7. 如果有Reduce部分，TaskTracker会创建个独立进程把Map输出的HDFS文件，通过RPC方式远程拉取到本地，拉取成功后，Reduce开始计算后续任务。

8. Reduce再把结果写入到HDFS中

9. 从HDFS中把结果导出。

这样一看好像是把简单的计算任务给

12.

13.Hdfs数据压缩算法

14.1、在HDFS之上将数据压缩好后，再存储到HDFS  
2、在HDFS内部支持数据压缩，这里又可以分为几种方法：  
    2.1、压缩工作在DataNode上完成，这里又分两种方法：  
           2.1.1、数据接收完后，再压缩  
                     这个方法对HDFS的改动最小，但效果最低，只需要在block文件close后，调用压缩工具，将block文件压缩一下，然后再打开block文件时解压一下即可，几行代码就可以搞定  
           2.1.2、边接收数据边压缩，使用第三方提供的压缩库  
                     效率和复杂度折中方法，Hook住系统的write和read操作，在数据写入磁盘之前，先压缩一下，但write和read对外的接口行为不变，比如：原始大小为100KB的数据，压缩后大小为10KB，当写入100KB后，仍对调用者返回100KB，而不是10KB  
    2.2、压缩工作交给DFSClient做，DataNode只接收和存储  
           这个方法效果最高，压缩分散地推给了HDFS客户端，但DataNode需要知道什么时候一个block块接收完成了。  
推荐最终实现采用2.2这个方法，该方法需要修改的HDFS代码量也不大，但效果最高。

15.Mapreduce调度模式

MapReduce是hadoop提供一个可进行分布式计算的框架或者平台，显然这个平台是多用户的，每个合法的用户可以向这个平台提交作业，那么这就带来一个问题，就是作业调度。

      任何调度策略都考虑自己平台调度需要权衡的几个维度，例如操作系统中的进程调度，他需要考虑的维度就是资源（CPU）的最大利用率（吞吐）和实时性，操作系统对实时性的要求很高，所以操作系统往往采用基于优先级的、可抢占式的调度策略，并且赋予IO密集型（相对于计算密集型）的进程较高的优先级，扯的有点远。

      回到hadoop平台，其实MapReduce的作业调度并没有很高的实时性的要求，本着最大吞吐的原则去设计的，所以MapReduce默认采用的调度策略是FIFO（基于优先级队列实现的FIFO，不是纯粹的FIFO，这样每次h），这种策略显然不是可抢占式的调度，所以带来的问题就是高优先级的任务会被先前已经在运行并且还要运行很久的低优先级的作业给堵塞住。

16.

17.Hive底层和数据库交互原理

Hive和Hbase有各自不同的特征：hive是高延迟、结构化和面向分析的，hbase是低延迟、非结构化和面向编程的。Hive数据仓库在hadoop上是高延迟的。Hive集成Hbase就是为了使用hbase的一些特性。如下是hive和hbase的集成架构：

图1 hive和hbase架构图

        Hive集成HBase可以有效利用HBase数据库的存储特性，如行更新和列索引等。在集成的过程中注意维持HBase jar包的一致性。Hive集成HBase需要在Hive表和HBase表之间建立映射关系，也就是Hive表的列(columns)和列类型(column types)与HBase表的列族(column families)及列限定词(column qualifiers)建立关联。每一个在Hive表中的域都存在于HBase中，而在Hive表中不需要包含所有HBase中的列。HBase中的RowKey对应到Hive中为选择一个域使用:key来对应，列族(cf:)映射到Hive中的其它所有域，列为(cf:cq)。例如下图2为Hive表映射到HBase表：

图2 Hive表映射HBase表

18.

19.Hbase过滤器实现原则

到现在为止你了解到HBase拥有灵活的逻辑模式和简单的物理模型，它们允许应用系统的计算工作更接近硬盘和网络，并

在这个层次进行优化。设计有效的模式是使用HBase的一个方面，你已经掌握了一堆概念用来做到这点。你可以设计行键

以使访问的数据在硬盘上也存放在一起，所以读写操作时可以节省硬盘寻道时间。在读取数据时，你经常需要基于某种标

准进行操作，你可以进一步优化数据访问。过滤器就是在这种情况下使用的一种强大的功能。  
我们还没有谈到使用过滤器的真实使用场景；一般来说调整表设计就可以优化访问模式。但是有时你已经把表设计调整得

尽可能好了，为不同访问模式优化得尽可能好了。当你仍然需要减少返回客户端的数据时，这就是考虑使用过滤器的时候

了。有时侯过滤器也被称为下推判断器（push-down predicates），支持你把数据过滤标准从客户端下推到服务器（如图

4.16）。这些过滤逻辑在读操作时使用，对返回客户端的数据有影响。这样通过减少网络传输的数据来节省网络IO。但是

数据仍然需要从硬盘读进RegionServer，过滤器在RegionServer里发挥作用。因为你有可能在HBase表里存储大量数据，

网络IO的节省是有重要意义的，并且先读出全部数据送到客户端再过滤出有用的数据，这种做法开销很大。  
图 4.16 在客户端完成数据过滤：从RegionServer把数据读取到客户端，在客户端使用过滤器逻辑处理数据；或者在服务

器端完成数据过滤：把过滤逻辑下推到RegionServer，因此减少了在网络上传输到客户端的数据量。实质上过滤器节省了

网络IO的开销，有时甚至是硬盘IO的开销。  
HBase提供了一个API，你可以用来实现定制过滤器。多个过滤器也可以捆绑在一起使用。可以在读过程最开始的地方，基

于行健进行过滤处理。此后，也可以基于HFile读出的KeyValues进行过滤处理。过滤器必须实现HBase Jar包中的Filter

接口，或者继承扩展一个实现了该接口的抽象类。我们推荐继承扩展FilterBase抽象类，这样你不需要写样板代码。继承

扩展其他诸如CompareFilter类也是一个选择，同样可以正常工作。当读取一行时该接口有下面的方法在多个地方可以调

用（顺序如图4.17）。它们总是按照下面描述的顺序来执行：  
1 这个方法第一个被调用，基于行健执行过滤：  
boolean filterRowKey(byte[] buffer, int offset, int length)  
基于这里的逻辑，如果行被过滤掉了（不出现在发送结果集合里）返回true，否则如果发送给客户端则返回false。  
2 如果该行没有在上一步被过滤掉，接着调用这个方法处理当前行的每个KeyValue对象：  
ReturnCode filterKeyValue(KeyValue v)  
这个方法返回一个ReturnCode，这是在Filter接口中定义的一个枚举（enum）类型。返回的ReturnCode判断某个KeyValue

对象发生了什么。  
3 在第2步过滤KeyValues对象后，接着是这个方法：  
void filterRow(List kvs)  
这个方法被传入成功通过过滤的KeyValue对象列表。倘若这个方法访问到这个列表，此时你可以在列表里的元素上执行任

何转换或运算。  
4 如果你选择过滤掉某些行，此时这个方法再一次提供了这么做的机会：  
boolean filterRow()  
过滤掉考虑中的行，则返回true。  
5 你可以在过滤器里构建逻辑来提早停止一次扫描。你可以把该逻辑放进这个方法：  
boolean filterAllRemaining()  
当你扫描很多行，在行健、列标识符或单元值里查找指定东西时，一旦找到目标，你就不再关心剩下的行了。此时这个方

法用起来很方便。这是过滤流程中最后调用的方法。  
图 4.17 过滤流程的各个步骤。扫描器对象扫描范围里的每行都会执行这个流程。  
   
另一个有用的方法是reset()。它会重置过滤器，在被应用到整行后由服务器调用。  
   
注意 这个API很强大，但是我们不觉得它是在应用系统里大量使用的。许多情况下，如果模式设计改变了，使用过滤器的

20.

21.Reduce后输出的量有多大

22.找到离存数据最近的一台机器运行和这个数据相关的map任务，reduce是按照你整理出的key有多少个来决定的。一个机器很难说，处理的快的处理多一点，保持所有机器使用平衡。  
  
上面你都自己写了20个map，和文件大小个数有关，和数据条数无关。  
  
要看你选择的输入格式是什么，默认是行偏移量，然后由你编写map函数，指定key和value是什么。相同的key整合起来传给reduce，由reduce进行下一步处理，最后输出到指定的地方。

23.Mapreduce掌握情况 和 hive hsql掌握情况

Hive 是基于Hadoop 构建的一套数据仓库分析系统，它提供了丰富的SQL查询方式来分析存储在Hadoop 分布式文件系统中的数据，可以将结构

化的数据文件映射为一张数据库表，并提供完整的SQL查询功能，可以将SQL语句转换为MapReduce任务进行运行，通过自己的SQL 去查询分析需

要的内容，这套SQL 简称Hive SQL，使不熟悉mapreduce 的用户很方便的利用SQL 语言查询，汇总，分析数据。而mapreduce开发人员可以把

己写的mapper 和reducer 作为插件来支持Hive 做更复杂的数据分析。

     它与关系型数据库的SQL 略有不同，但支持了绝大多数的语句如DDL、DML 以及常见的聚合函数、连接查询、条件查询。HIVE不适合用于联机

online)事务处理，也不提供实时查询功能。它最适合应用在基于大量不可变数据的批处理作业。  
  
    HIVE的特点：可伸缩（在Hadoop的集群上动态的添加设备），可扩展，容错，输入格式的松散耦合。

     Hive 的官方文档中对查询语言有了很详细的描述，请参考：http://wiki.apache.org/hadoop/Hive/LanguageManual ，本文的内容大部分翻译自该页面，期间加入了一些在使用过程中需要注意到的事项。

1.  DDL 操作

DDL

•建表

•删除表

•修改表结构

•创建／删除视图

•创建数据库

•显示命令

建表：

CREATE [EXTERNAL] TABLE [IF NOT EXISTS] table\_name   
  [(col\_name data\_type [COMMENT col\_comment], ...)]   
  [COMMENT table\_comment]   
  [PARTITIONED BY (col\_name data\_type [COMMENT col\_comment], ...)]   
  [CLUSTERED BY (col\_name, col\_name, ...)   
  [SORTED BY (col\_name [ASC|DESC], ...)] INTO num\_buckets BUCKETS]   
  [ROW FORMAT row\_format]   
  [STORED AS file\_format]   
  [LOCATION hdfs\_path]

•CREATE TABLE 创建一个指定名字的表。如果相同名字的表已经存在，则抛出异常；用户可以用 IF NOT EXIST 选项来忽略这个异常

•EXTERNAL 关键字可以让用户创建一个外部表，在建表的同时指定一个指向实际数据的路径（LOCATION）

•LIKE 允许用户复制现有的表结构，但是不复制数据

•COMMENT可以为表与字段增加描述

•ROW FORMAT

    DELIMITED [FIELDS TERMINATED BY char] [COLLECTION ITEMS TERMINATED BY char]

        [MAP KEYS TERMINATED BY char] [LINES TERMINATED BY char]

   | SERDE serde\_name [WITH SERDEPROPERTIES (property\_name=property\_value, property\_name=property\_value, ...)]

         用户在建表的时候可以自定义 SerDe 或者使用自带的 SerDe。如果没有指定 ROW FORMAT 或者 ROW FORMAT DELIMITED，将会使用自带的 SerDe。在建表的时候，用户还需要为表指定列，用户在指定表的列的同时也会指定自定义的 SerDe，Hive 通过 SerDe 确定表的具体的列的数据。

•STORED AS

            SEQUENCEFILE

            | TEXTFILE

            | RCFILE

            | INPUTFORMAT input\_format\_classname OUTPUTFORMAT             output\_format\_classname

       如果文件数据是纯文本，可以使用 STORED AS TEXTFILE。如果数据需要压缩，使用 STORED AS SEQUENCE 。

创建简单表：

hive> CREATE TABLE pokes (foo INT, bar STRING);

创建外部表：

CREATE EXTERNAL TABLE page\_view(viewTime INT, userid BIGINT,

     page\_url STRING, referrer\_url STRING,

     ip STRING COMMENT 'IP Address of the User',

     country STRING COMMENT 'country of origination')

 COMMENT 'This is the staging page view table'

 ROW FORMAT DELIMITED FIELDS TERMINATED BY '\054'

 STORED AS TEXTFILE

 LOCATION '<hdfs\_location>';

建分区表

CREATE TABLE par\_table(viewTime INT, userid BIGINT,

     page\_url STRING, referrer\_url STRING,

     ip STRING COMMENT 'IP Address of the User')

 COMMENT 'This is the page view table'

 PARTITIONED BY(date STRING, pos STRING)

ROW FORMAT DELIMITED ‘\t’

   FIELDS TERMINATED BY '\n'

STORED AS SEQUENCEFILE;

建Bucket表

CREATE TABLE par\_table(viewTime INT, userid BIGINT,

     page\_url STRING, referrer\_url STRING,

     ip STRING COMMENT 'IP Address of the User')

 COMMENT 'This is the page view table'

 PARTITIONED BY(date STRING, pos STRING)

 CLUSTERED BY(userid) SORTED BY(viewTime) INTO 32 BUCKETS

 ROW FORMAT DELIMITED ‘\t’

   FIELDS TERMINATED BY '\n'

STORED AS SEQUENCEFILE;

创建表并创建索引字段ds

hive> CREATE TABLE invites (foo INT, bar STRING) PARTITIONED BY (ds STRING);

复制一个空表

CREATE TABLE empty\_key\_value\_store

LIKE key\_value\_store;

例子

create table  user\_info (user\_id int, cid string, ckid string, username string)

row format delimited

fields terminated by '\t'

 lines terminated by '\n';

导入数据表的数据格式是：字段之间是tab键分割，行之间是断行。

及要我们的文件内容格式：

100636  100890  c5c86f4cddc15eb7        yyyvybtvt  
100612  100865  97cc70d411c18b6f        gyvcycy  
100078  100087  ecd6026a15ffddf5        qa000100

显示所有表：

hive> SHOW TABLES;

按正条件（正则表达式）显示表，

hive> SHOW TABLES '.\*s';

修改表结构

•增加分区、删除分区

•重命名表

•修改列的名字、类型、位置、注释

•增加/更新列

•增加表的元数据信息

表添加一列 ：

hive> ALTER TABLE pokes ADD COLUMNS (new\_col INT);

添加一列并增加列字段注释

hive> ALTER TABLE invites ADD COLUMNS (new\_col2 INT COMMENT 'a comment');

更改表名：

hive> ALTER TABLE events RENAME TO 3koobecaf;

删除列：

hive> DROP TABLE pokes;

增加、删除分区

•增加

ALTER TABLE table\_name ADD [IF NOT EXISTS] partition\_spec [ LOCATION 'location1' ] partition\_spec [ LOCATION 'location2' ] ...

      partition\_spec:

  : PARTITION (partition\_col = partition\_col\_value, partition\_col = partiton\_col\_value, ...)

•删除

ALTER TABLE table\_name DROP partition\_spec, partition\_spec,...

重命名表

•ALTER TABLE table\_name RENAME TO new\_table\_name

修改列的名字、类型、位置、注释：

•ALTER TABLE table\_name CHANGE [COLUMN] col\_old\_name col\_new\_name column\_type [COMMENT col\_comment] [FIRST|AFTER column\_name]

•这个命令可以允许改变列名、数据类型、注释、列位置或者它们的任意组合

表添加一列 ：

hive> ALTER TABLE pokes ADD COLUMNS (new\_col INT);

添加一列并增加列字段注释

hive> ALTER TABLE invites ADD COLUMNS (new\_col2 INT COMMENT 'a comment');

增加/更新列

•ALTER TABLE table\_name ADD|REPLACE COLUMNS (col\_name data\_type [COMMENT col\_comment], ...)

• ADD是代表新增一字段，字段位置在所有列后面(partition列前)

     REPLACE则是表示替换表中所有字段。

增加表的元数据信息

•ALTER TABLE table\_name SET TBLPROPERTIES table\_properties table\_properties:

         :[property\_name = property\_value…..]

•用户可以用这个命令向表中增加metadata

改变表文件格式与组织

•ALTER TABLE table\_name SET FILEFORMAT file\_format

•ALTER TABLE table\_name CLUSTERED BY(userid) SORTED BY(viewTime) INTO num\_buckets BUCKETS

•这个命令修改了表的物理存储属性

创建／删除视图

•CREATE VIEW [IF NOT EXISTS] view\_name [ (column\_name [COMMENT column\_comment], ...) ][COMMENT view\_comment][TBLPROPERTIES (property\_name = property\_value, ...)] AS SELECT

•增加视图

•如果没有提供表名，视图列的名字将由定义的SELECT表达式自动生成

•如果修改基本表的属性，视图中不会体现，无效查询将会失败

•视图是只读的，不能用LOAD/INSERT/ALTER

•DROP VIEW view\_name

•删除视图

创建数据库

•CREATE DATABASE name

显示命令

•show tables;

•show databases;

•show partitions ;

•show functions

•describe extended table\_name dot col\_name

2.  DML 操作:元数据存储

     hive不支持用insert语句一条一条的进行插入操作，也不支持update操作。数据是以load的方式加载到建立好的表中。数据一旦导入就不可以修改。

DML包括：INSERT[插入](http://baike.baidu.com/view/226806.htm)、UPDATE[更新](http://baike.baidu.com/view/23174.htm)、DELETE[删除](http://baike.baidu.com/view/14787.htm)

•向数据表内加载文件

•将查询结果插入到Hive表中

•0.8新特性 insert into

向数据表内加载文件

•LOAD DATA [LOCAL] INPATH 'filepath' [OVERWRITE] INTO TABLE tablename [PARTITION (partcol1=val1, partcol2=val2 ...)]

•Load 操作只是单纯的复制/移动操作，将数据文件移动到 Hive 表对应的位置。

•filepath

•相对路径，例如：project/data1

•绝对路径，例如： /user/hive/project/data1

•包含模式的完整 URI，例如：hdfs://namenode:9000/user/hive/project/data1

例如：

hive> LOAD DATA LOCAL INPATH './examples/files/kv1.txt' OVERWRITE INTO TABLE pokes;

加载本地数据，同时给定分区信息

•加载的目标可以是一个表或者分区。如果表包含分区，必须指定每一个分区的分区名

•filepath 可以引用一个文件（这种情况下，Hive 会将文件移动到表所对应的目录中）或者是一个目录（在这种情况下，Hive 会将目录中的所有文件移动至表所对应的目录中）

LOCAL关键字

•指定了LOCAL，即本地

•load 命令会去查找本地文件系统中的 filepath。如果发现是相对路径，则路径会被解释为相对于当前用户的当前路径。用户也可以为本地文件指定一个完整的 URI，比如：file:///user/hive/project/data1.

•load 命令会将 filepath 中的文件复制到目标文件系统中。目标文件系统由表的位置属性决定。被复制的数据文件移动到表的数据对应的位置

例如：加载本地数据，同时给定分区信息：

hive> LOAD DATA LOCAL INPATH './examples/files/kv2.txt' OVERWRITE INTO TABLE invites PARTITION (ds='2008-08-15');

• 没有指定LOCAL

         如果 filepath 指向的是一个完整的 URI，hive 会直接使用这个 URI。 否则

•如果没有指定 schema 或者 authority，Hive 会使用在 hadoop 配置文件中定义的 schema 和 authority，fs.default.name 指定了 Namenode 的 URI

•如果路径不是绝对的，Hive 相对于 /user/ 进行解释。 Hive 会将 filepath 中指定的文件内容移动到 table （或者 partition）所指定的路径中

加载DFS数据 ，同时给定分区信息：

hive> LOAD DATA INPATH '/user/myname/kv2.txt' OVERWRITE INTO TABLE invites PARTITION (ds='2008-08-15');  
The above command will load data from an HDFS file/directory to the table. Note that loading data from HDFS will result in moving the file/directory. As a result, the operation is almost instantaneous.

OVERWRITE

•指定了OVERWRITE

•目标表（或者分区）中的内容（如果有）会被删除，然后再将 filepath 指向的文件/目录中的内容添加到表/分区中。

•如果目标表（分区）已经有一个文件，并且文件名和 filepath 中的文件名冲突，那么现有的文件会被新文件所替代。

将查询结果插入Hive表

•将查询结果插入Hive表

•将查询结果写入HDFS文件系统

•基本模式

     INSERT OVERWRITE TABLE tablename1 [PARTITION (partcol1=val1, partcol2=val2 ...)] select\_statement1 FROM from\_statement

•多插入模式

 FROM from\_statement

INSERT OVERWRITE TABLE tablename1 [PARTITION (partcol1=val1, partcol2=val2 ...)] select\_statement1

[INSERT OVERWRITE TABLE tablename2 [PARTITION ...] select\_statement2] ...

•自动分区模式

 INSERT OVERWRITE TABLE tablename PARTITION (partcol1[=val1], partcol2[=val2] ...) select\_statement FROM from\_statement

将查询结果写入HDFS文件系统

•INSERT OVERWRITE [LOCAL] DIRECTORY directory1 SELECT ... FROM ...

        FROM from\_statement

        INSERT OVERWRITE [LOCAL] DIRECTORY directory1 select\_statement1

     [INSERT OVERWRITE [LOCAL] DIRECTORY directory2 select\_statement2]

•

•数据写入文件系统时进行文本序列化，且每列用^A 来区分，\n换行

INSERT INTO

•INSERT INTO  TABLE tablename1 [PARTITION (partcol1=val1, partcol2=val2 ...)] select\_statement1 FROM from\_statement

3.  DQL 操作:数据查询SQL

SQL操作

•基本的Select 操作

•基于Partition的查询

•Join

3.1 基本的Select 操作

SELECT [ALL | DISTINCT] select\_expr, select\_expr, ...

FROM table\_reference

[WHERE where\_condition]

[GROUP BY col\_list [HAVING condition]]

[   CLUSTER BY col\_list

  | [DISTRIBUTE BY col\_list] [SORT BY| ORDER BY col\_list]

]

[LIMIT number]

•使用ALL和DISTINCT选项区分对重复记录的处理。默认是ALL，表示查询所有记录。DISTINCT表示去掉重复的记录

•

•Where 条件

•类似我们传统SQL的where 条件

•目前支持 AND,OR ,0.9版本支持between

•IN, NOT IN

•不支持EXIST ,NOT EXIST

ORDER BY与SORT BY的不同

•ORDER BY 全局排序，只有一个Reduce任务

•SORT BY 只在本机做排序

Limit

•Limit 可以限制查询的记录数

SELECT \* FROM t1 LIMIT 5

•实现Top k 查询

•下面的查询语句查询销售记录最大的 5 个销售代表。

SET mapred.reduce.tasks = 1   
  SELECT \* FROM test SORT BY amount DESC LIMIT 5

•REGEX Column Specification

SELECT 语句可以使用正则表达式做列选择，下面的语句查询除了 ds 和 hr 之外的所有列：

SELECT `(ds|hr)?+.+` FROM test

例如

按先件查询

hive> SELECT a.foo FROM invites a WHERE a.ds='<DATE>';

将查询数据输出至目录：

hive> INSERT OVERWRITE DIRECTORY '/tmp/hdfs\_out' SELECT a.\* FROM invites a WHERE a.ds='<DATE>';

将查询结果输出至本地目录：

hive> INSERT OVERWRITE LOCAL DIRECTORY '/tmp/local\_out' SELECT a.\* FROM pokes a;

选择所有列到本地目录 ：

hive> INSERT OVERWRITE TABLE events SELECT a.\* FROM profiles a;  
hive> INSERT OVERWRITE TABLE events SELECT a.\* FROM profiles a WHERE a.key < 100;  
hive> INSERT OVERWRITE LOCAL DIRECTORY '/tmp/reg\_3' SELECT a.\* FROM events a;  
hive> INSERT OVERWRITE DIRECTORY '/tmp/reg\_4' select a.invites, a.pokes FROM profiles a;  
hive> INSERT OVERWRITE DIRECTORY '/tmp/reg\_5' SELECT COUNT(1) FROM invites a WHERE a.ds='<DATE>';  
hive> INSERT OVERWRITE DIRECTORY '/tmp/reg\_5' SELECT a.foo, a.bar FROM invites a;  
hive> INSERT OVERWRITE LOCAL DIRECTORY '/tmp/sum' SELECT SUM(a.pc) FROM pc1 a;

将一个表的统计结果插入另一个表中：

hive> FROM invites a INSERT OVERWRITE TABLE events SELECT a.bar, count(1) WHERE a.foo > 0 GROUP BY a.bar;  
hive> INSERT OVERWRITE TABLE events SELECT a.bar, count(1) FROM invites a WHERE a.foo > 0 GROUP BY a.bar;  
JOIN  
hive> FROM pokes t1 JOIN invites t2 ON (t1.bar = t2.bar) INSERT OVERWRITE TABLE events SELECT t1.bar, t1.foo, t2.foo;

将多表数据插入到同一表中：

FROM src  
INSERT OVERWRITE TABLE dest1 SELECT src.\* WHERE src.key < 100  
INSERT OVERWRITE TABLE dest2 SELECT src.key, src.value WHERE src.key >= 100 and src.key < 200  
INSERT OVERWRITE TABLE dest3 PARTITION(ds='2008-04-08', hr='12') SELECT src.key WHERE src.key >= 200 and src.key < 300  
INSERT OVERWRITE LOCAL DIRECTORY '/tmp/dest4.out' SELECT src.value WHERE src.key >= 300;

将文件流直接插入文件：

hive> FROM invites a INSERT OVERWRITE TABLE events SELECT TRANSFORM(a.foo, a.bar) AS (oof, rab) USING '/bin/cat' WHERE a.ds > '2008-08-09';  
This streams the data in the map phase through the script /bin/cat (like hadoop streaming). Similarly - streaming can be used on the reduce side (please see the Hive Tutorial or examples)

3.2 基于Partition的查询

•一般 SELECT 查询会扫描整个表，使用 PARTITIONED BY 子句建表，查询就可以利用分区剪枝（input pruning）的特性

•Hive 当前的实现是，只有分区断言出现在离 FROM 子句最近的那个WHERE 子句中，才会启用分区剪枝

3.3 Join

Syntax

join\_table:   
   table\_reference JOIN table\_factor [join\_condition]   
  | table\_reference {LEFT|RIGHT|FULL} [OUTER] JOIN table\_reference join\_condition   
  | table\_reference LEFT SEMI JOIN table\_reference join\_condition   
  
table\_reference:   
    table\_factor   
  | join\_table   
  
table\_factor:   
    tbl\_name [alias]   
  | table\_subquery alias   
  | ( table\_references )   
  
join\_condition:   
    ON equality\_expression ( AND equality\_expression )\*   
  
equality\_expression:   
    expression = expression

•Hive 只支持等值连接（equality joins）、外连接（outer joins）和（left semi joins）。Hive 不支持所有非等值的连接，因为非等值连接非常难转化到 map/reduce 任务

•LEFT，RIGHT和FULL OUTER关键字用于处理join中空记录的情况

•LEFT SEMI JOIN 是 IN/EXISTS 子查询的一种更高效的实现

•join 时，每次 map/reduce 任务的逻辑是这样的：reducer 会缓存 join 序列中除了最后一个表的所有表的记录，再通过最后一个表将结果序列化到文件系统

•实践中，应该把最大的那个表写在最后

join 查询时，需要注意几个关键点

•只支持等值join

•SELECT a.\* FROM a JOIN b ON (a.id = b.id)

•SELECT a.\* FROM a JOIN b   
    ON (a.id = b.id AND a.department = b.department)

•可以 join 多于 2 个表，例如

  SELECT a.val, b.val, c.val FROM a JOIN b   
    ON (a.key = b.key1) JOIN c ON (c.key = b.key2)

•如果join中多个表的 join key 是同一个，则 join 会被转化为单个 map/reduce 任务

LEFT，RIGHT和FULL OUTER

•例子

•SELECT a.val, b.val FROM a LEFT OUTER JOIN b ON (a.key=b.key)

•如果你想限制 join 的输出，应该在 WHERE 子句中写过滤条件——或是在 join 子句中写

•

•容易混淆的问题是表分区的情况

• SELECT c.val, d.val FROM c LEFT OUTER JOIN d ON (c.key=d.key)   
  WHERE a.ds='2010-07-07' AND b.ds='2010-07-07‘

•如果 d 表中找不到对应 c 表的记录，d 表的所有列都会列出 NULL，包括 ds 列。也就是说，join 会过滤 d 表中不能找到匹配 c 表 join key 的所有记录。这样的话，LEFT OUTER 就使得查询结果与 WHERE 子句无关

•解决办法

•SELECT c.val, d.val FROM c LEFT OUTER JOIN d   
  ON (c.key=d.key AND d.ds='2009-07-07' AND c.ds='2009-07-07')

LEFT SEMI JOIN

•LEFT SEMI JOIN 的限制是， JOIN 子句中右边的表只能在 ON 子句中设置过滤条件，在 WHERE 子句、SELECT 子句或其他地方过滤都不行

•

•SELECT a.key, a.value   
  FROM a   
  WHERE a.key in   
   (SELECT b.key   
    FROM B);

       可以被重写为：

      SELECT a.key, a.val   
   FROM a LEFT SEMI JOIN b on (a.key = b.key)

UNION ALL

•用来合并多个select的查询结果，需要保证select中字段须一致

•select\_statement UNION ALL select\_statement UNION ALL select\_statement ...

4.  从SQL到HiveQL应转变的习惯

1、Hive不支持等值连接

•SQL中对两表内联可以写成：

•select \* from dual a,dual b where a.key = b.key;

•Hive中应为

•select \* from dual a join dual b on a.key = b.key;

而不是传统的格式：

SELECT t1.a1 as c1, t2.b1 as c2FROM t1, t2 WHERE t1.a2 = t2.b2

2、分号字符

•分号是SQL语句结束标记，在HiveQL中也是，但是在HiveQL中，对分号的识别没有那么智慧，例如：

•select concat(key,concat(';',key)) from dual;

•但HiveQL在解析语句时提示：

        FAILED: Parse Error: line 0:-1 mismatched input '<EOF>' expecting ) in function specification

•解决的办法是，使用分号的八进制的ASCII码进行转义，那么上述语句应写成：

•select concat(key,concat('\073',key)) from dual;

3、IS [NOT] NULL

•SQL中null代表空值, 值得警惕的是, 在HiveQL中String类型的字段若是空(empty)字符串, 即长度为0, 那么对它进行IS NULL的判断结果是False.

4、Hive不支持将数据插入现有的表或分区中，

仅支持覆盖重写整个表，示例如下：

1. INSERT OVERWRITE TABLE t1

2. SELECT \* FROM t2;

4、hive不支持INSERT INTO, UPDATE, DELETE操作

    这样的话，就不要很复杂的锁机制来读写数据。  
     INSERT INTO syntax is only available starting in version 0.8。INSERT INTO就是在表或分区中追加数据。

5、hive支持嵌入mapreduce程序，来处理复杂的逻辑

如：

1. FROM (

2. MAP doctext USING '[**Python**](http://lib.csdn.net/base/11) wc\_mapper.py' AS (word, cnt)

3. FROM docs

4. CLUSTER BY word

5. ) a

6. REDUCE word, cnt USING 'python wc\_reduce.py';

--doctext: 是输入

--word, cnt: 是map程序的输出

--CLUSTER BY: 将wordhash后，又作为reduce程序的输入

并且map程序、reduce程序可以单独使用，如：

1. FROM (

2. FROM session\_table

3. SELECT sessionid, tstamp, data

4. DISTRIBUTE BY sessionid SORT BY tstamp

5. ) a

6. REDUCE sessionid, tstamp, data USING 'session\_reducer.sh';

--DISTRIBUTE BY: 用于给reduce程序分配行数据

6、hive支持将转换后的数据直接写入不同的表，还能写入分区、hdfs和本地目录。

这样能免除多次扫描输入表的开销。

1. FROM t1

2.

3. INSERT OVERWRITE TABLE t2

4. SELECT t3.c2, count(1)

5. FROM t3

6. WHERE t3.c1 <= 20

7. GROUP BY t3.c2

8.

9. INSERT OVERWRITE DIRECTORY '/output\_dir'

10. SELECT t3.c2, avg(t3.c1)

11. FROM t3

12. WHERE t3.c1 > 20 AND t3.c1 <= 30

13. GROUP BY t3.c2

14.

15. INSERT OVERWRITE LOCAL DIRECTORY '/home/dir'

16. SELECT t3.c2, sum(t3.c1)

17. FROM t3

18. WHERE t3.c1 > 30

19. GROUP BY t3.c2;

5.  实际示例

创建一个表

CREATE TABLE u\_data (  
userid INT,  
movieid INT,  
rating INT,  
unixtime STRING)  
ROW FORMAT DELIMITED  
FIELDS TERMINATED BY '/t'  
STORED AS TEXTFILE;

下载示例数据文件，并解压缩  
wget <http://www.grouplens.org/system/files/ml-data.tar__0.gz>  
tar xvzf ml-data.tar\_\_0.gz

加载数据到表中:

LOAD DATA LOCAL INPATH 'ml-data/u.data'  
OVERWRITE INTO TABLE u\_data;

统计数据总量:

SELECT COUNT(1) FROM u\_data;

现在做一些复杂的数据分析:

创建一个 weekday\_mapper.py: 文件，作为数据按周进行分割   
import sys  
import datetime

for line in sys.stdin:  
line = line.strip()  
userid, movieid, rating, unixtime = line.split('/t')

生成数据的周信息

weekday = datetime.datetime.fromtimestamp(float(unixtime)).isoweekday()  
print '/t'.join([userid, movieid, rating, str(weekday)])

使用映射脚本

//创建表，按分割符分割行中的字段值  
CREATE TABLE u\_data\_new (  
userid INT,  
movieid INT,  
rating INT,  
weekday INT)  
ROW FORMAT DELIMITED  
FIELDS TERMINATED BY '/t';  
//将python文件加载到系统  
add FILE weekday\_mapper.py;

将数据按周进行分割

INSERT OVERWRITE TABLE u\_data\_new  
SELECT  
TRANSFORM (userid, movieid, rating, unixtime)  
USING 'python weekday\_mapper.py'  
AS (userid, movieid, rating, weekday)  
FROM u\_data;

SELECT weekday, COUNT(1)  
FROM u\_data\_new  
GROUP BY weekday;

处理Apache Weblog 数据

将WEB日志先用正则表达式进行组合，再按需要的条件进行组合输入到表中  
add jar ../build/contrib/hive\_contrib.jar;

CREATE TABLE apachelog (  
host STRING,  
identity STRING,  
user STRING,  
time STRING,  
request STRING,  
status STRING,  
size STRING,  
referer STRING,  
agent STRING)  
ROW FORMAT SERDE 'org.apache.hadoop.hive.contrib.serde2.RegexSerDe'  
WITH SERDEPROPERTIES (  
"input.regex" = "([^ ]\*) ([^ ]\*) ([^ ]\*) (-|//[[^//]]\*//]) ([^ /"]\*|/"[^/"]\*/") (-|[0-9]\*) (-|[0-9]\*)(?: ([^ /"]\*|/"[^/"]\*/") ([^ /"]\*|/"[^/"]\*/"))?",  
"output.format.string" = "%1$s %2$s %3$s %4$s %5$s %6$s %7$s %8$s %9$s"  
)  
STORED AS TEXTFILE;

24.

25.Datanode在什么情况下不备份

26.在配置文件中datanode的数量设置为1时

27.Combiner出现在那个过程

28.Hdfs体系结构

我们首先介绍HDFS的体系结构，HDFS采用了主从（Master/Slave）结构模型，一个HDFS集群是由一个NameNode和若干个DataNode组成的。其中NameNode作为主服务器，管理文件系统的命名空间和客户端对文件的访问操作；集群中的DataNode管理存储的数据。HDFS允许用户以文件的形式存储数据。从内部来看，文件被分成若干个数据块，而且这若干个数据块存放在一组DataNode上。NameNode执行文件系统的命名空间操作，比如打开、关闭、重命名文件或目录等，它也负责数据块到具体DataNode的映射。DataNode负责处理文件系统客户端的文件读写请求，并在NameNode的统一调度下进行数据块的创建、删除和复制工作。图1-3给出了HDFS的体系结构。

NameNode和DataNode都被设计成可以在普通商用计算机上运行。这些计算机通常运行的是GNU/Linux操作系统。HDFS采用Java语言开发，因此任何支持Java的机器都可以部署NameNode和DataNode。一个典型的部署场景是集群中的一台机器运行一个NameNode实例，其他机器分别运行一个DataNode实例。当然，并不排除一台机器运行多个DataNode实例的情况。集群中单一的NameNode的设计则大大简化了系统的架构。NameNode是所有HDFS元数据的管理者，用户数据永远不会经过NameNode。

|  |
| --- |
|  |
| （点击查看大图）图1-3　HDFS的体系结构图 |

29.

30.Flush体系结构

31.什么是列队    linkedList 队列

32.List和set去表

33.数据库三大范式

34.第一范式，又称1NF，它指的是在一个应用中的数据都可以组织成由行和列的表格形式，且表格的任意一个行列交叉点即单元格，都不可再划分为行和列的形式，实际上任意一张表格都满足1NF； 第二范式，又称2NF，它指的是在满足1NF的基础上，一张数据表中的任何非主键字段都全部依赖于主键字段，没有任何非主键字段只依赖于主键字段的一部分。即，可以由主键字段来唯一的确定一条记录。比如学号+课程号的联合主键，可以唯一的确定某个成绩是哪个学员的哪门课的成绩，缺少学号或者缺少课程号，都不能确定成绩的意义。 第三范式，又称3NF，它是指在满足2NF的基础上，数据表的任何非主键字段之间都不产生函数依赖，即非主键字段之间没有依赖关系，全部只依赖于主键字段。例如将学员姓名和所属班级名称放在同一张表中是不科学的，因为学员依赖于班级，可将学员信息和班级信息单独存放，以满足3NF。

35.三个datanode 当有一个出现错误会怎么样

36.Sgoop导入数据到mysql中 如何让数据不重复导入？ 存在数据问题sgoop会怎么样

37.

38.

39.Hadop基础知识和问题分析能力

3.1 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

3.2 请描述 https://issues.apache.org/jira/browse/HDFS-2379 说的是什么问题，最终解

决的思路是什么？

4、MapReduce 开发能力

请参照 wordcount 实现一个自己的 map reduce，需求为：

    a 输入文件格式：

       xxx,xxx,xxx,xxx,xxx,xxx,xxx

    b 输出文件格式：

       xxx,20

       xxx,30

       xxx.40

    c 功能：根据命令行参数统计输入文件中指定关键字出现的次数，并展示出来

       例如：hadoop jar xxxxx.jar keywordcount xxx,xxx,xxx,xxx(四个关键字）

5、MapReduce 优化

请根据第五题中的程序, 提出如何优化 MR 程序运行速度的思路

6、Linux 操作系统知识考察

请列举曾经修改过的/etc 下的配置文件，并说明修改要解决的问题？

7、Java 开发能力

7.1 写代码实现 1G 大小的文本文件，行分隔符为\x01\x02,统计一下该文件中的总行数，

要求注意边界情况的处理

7.2 请描述一下在开发中如何对上面的程序进行性能分析，对性能进行优化的过程。

五、来自\*\*\*\*\*提供的 hadoop 面试题 21 道：

1、设计一套系统，使之能够从不断增加的不同的数据源中，提取指定格式的数据。

要求：

编者 QQ：1040195253 16

1)、运行结果要能大致得知提取效果，并可据此持续改进提取方法；

2)、由于数据来源的差异性，请给出可弹性配置的程序框架；

3)、数据来源可能有 Mysql,sqlserver 等；

4)、该系统具备持续挖掘的能力，即，可重复提取更多信息

2. 经典的一道题：

现有 1 亿个整数均匀分布，如果要得到前 1K 个最大的数，求最优的算法。

（先不考虑内存的限制，也不考虑读写外存，时间复杂度最少的算法即为最优算法）

我先说下我的想法:分块，比如分 1W 块，每块 1W 个，然后分别找出每块最大值，从这最

大的 1W 个值中找最大 1K 个，

那么其他的 9K 个最大值所在的块即可扔掉，从剩下的最大的 1K 个值所在的块中找前 1K

个即可。那么原问题的规模就缩小到了 1/10。

问题：

（1）这种分块方法的最优时间复杂度。

（2）如何分块达到最优。比如也可分 10W 块，每块 1000 个数。则问题规模可降到原来

1/100。但事实上复杂度并没降低。

 (3)还有没更好更优的方法解决这个问题。

3. MapReduce 大致流程？

4. combiner, partition 作用？

5.用 mapreduce 实现 sql 语句 select count(x) from a group by b？

6. 用 mapreduce 如何实现两张表连接，有哪些方法？

7.知道 MapReduce 大致流程，map, shuffle, reduce

编者 QQ：1040195253 17

8.知道 combiner, partition 作用，设置 compression

9.搭建 hadoop 集群，master/slave 都运行那些服务

10.HDFS，replica 如何定位

11.版本 0.20.2->0.20.203->0.20.205, 0.21, 0.23, 1.0. 1

新旧 API 有什么不同

12.Hadoop 参数调优，cluster level: JVM, map/reduce slots, job level: reducer

#,memory, use combiner? use compression?

13.pig latin, Hive 语法有什么不同

14.描述 HBase, zookeeper 搭建过程

15.hadoop 运行的原理?

16.mapreduce 的原理?

17.HDFS 存储的机制?

18.举一个简单的例子说明 mapreduce 是怎么来运行的 ?

19.使用 mapreduce 来实现下面实例

      实例 :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。

20.hadoop 中 Combiner 的作用?

21.如何确认 Hadoop 集群的健康状况。

六、来自\*\*\*\*提供的 hadoop 面试题 9 道：

1.使用的 hadoop 版本都是什么？

QQ942609288????，???????QQ??

QQ942609288????，???????QQ??

编者 QQ：1040195253 18

2.mpareduce 原理是什么？

3.mapreduce 作业，不使用 reduce 来输出，用什么能代替 reduce 的功能

4.hive 如何调优？

5.hive 如何权限控制？

6.hbase 写数据的原理是什么？

7.hive 能像关系数据库那样，建多个库吗？

8.hbase 宕机如何处理？

9.假设公司要建一个数据中心，你会如何规划？

七、hadoop 选择判断题 33 道：

单项选择题

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

3. 下列哪个程序通常与 NameNode 在一个节点启动？

a)SecondaryNameNode b)DataNode c)TaskTracker d)Jobtracker

4. Hadoop 作者

a)Martin Fowler b)Kent Beck c)Doug cutting

5. HDFS 默认 Block Size

a)32MB  b)64MB c)128MB

6. 下列哪项通常是集群的最主要瓶颈

QQ942609288????，???????QQ??

QQ942609288????，???????QQ??

编者 QQ：1040195253 19

a)CPU   b)网络 c)磁盘  d)内存

7. 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

多选题：

8. 下列哪项可以作为集群的管理工具

a)Puppet b)Pdsh c)Cloudera Manager d)d)Zookeeper

9. 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

10. 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

11. 下列哪个是 Hadoop 运行的模式

a)单机版 b)伪分布式 c)分布式

12. Cloudera 提供哪几种安装 CDH 的方法

a)Cloudera manager b)Tar ball c)Yum d)Rpm

判断题：

13. Ganglia 不仅可以进行监控，也可以进行告警。（ ）

14. Block Size 是不可以修改的。（ ）

15. Nagios 不可以监控 Hadoop 集群，因为它不提供 Hadoop 支持。（ ）

16. 如果 NameNode 意外终止，SecondaryNameNode 会接替它使集群继续工作。（ ）

17. Cloudera CDH 是需要付费使用的。（ ）

18. Hadoop 是 Java 开发的，所以 MapReduce 只支持 Java 语言编写。（ ）

19. Hadoop 支持数据的随机读写。（ ）

20. NameNode 负责管理 metadata，client 端每次读写请求，它都会从磁盘中读取或则

会写

入 metadata 信息并反馈 client 端。（ ）

21. NameNode 本地磁盘保存了 Block 的位置信息。（ ）

22. DataNode 通过长连接与 NameNode 保持通信。（ ）

23. Hadoop 自身具有严格的权限管理和安全措施保障集群正常运行。（ ）

24. Slave 节点要存储数据，所以它的磁盘越大越好。（ ）

25. hadoop dfsadmin –report 命令用于检测 HDFS 损坏块。（ ）

26. Hadoop 默认调度器策略为 FIFO（ ）

27. 集群内每个节点都应该配 RAID，这样避免单磁盘损坏，影响整个节点运行。（ ）

28. 因为 HDFS 有多个副本，所以 NameNode 是不存在单点问题的。（ ）

29. 每个 map 槽就是一个线程。（ ）

30. Mapreduce 的 input split 就是一个 block。（ ）

31. NameNode 的 Web UI 端口是 50030，它通过 jetty 启动的 Web 服务。（ ）

32. Hadoop 环境变量中的 HADOOP\_HEAPSIZE 用于设置所有 Hadoop 守护线程的内

存。它默

认是 200 GB。（ ）

33. DataNode 首次加入 cluster 的时候，如果 log 中报告不兼容文件版本，那需要

NameNode

执行“Hadoop namenode -format”操作格式化磁盘。（ ）

八、mr 和 hive 实现手机流量统计面试题 6 道：

1.hive 实现统计的查询语句是什么？

2.生产环境中为什么建议使用外部表？

3.hadoop mapreduce 创建类 DataWritable 的作用是什么？

4.为什么创建类 DataWritable？

5.如何实现统计手机流量？

6.对比 hive 与 mapreduce 统计手机流量的区别？

九、来自 aboutyun 的面试题 1 道：

最近去面试，出了个这样的题目，大家有兴趣也试试。

用 Hadoop 分析海量日志文件，每行日志记录了如下数据：

TableName(表名)，Time(时间)，User(用户)，TimeSpan(时间开销)。

要求：

编写 MapReduce 程序算出高峰时间段（如上午 10 点）哪张表被访问的最频繁，以及

十、来自 aboutyun 的面试题 6 道：

前段时间接到阿里巴巴面试云计算,拿出来给我们共享下

1、hadoop 运转的原理?

2、mapreduce 的原理?

3、HDFS 存储的机制?

4、举一个简略的比方阐明 mapreduce 是怎么来运转的 ?

5、面试的人给你出一些疑问,让你用 mapreduce 来完成？

      比方 :如今有 10 个文件夹 ,每个文件夹都有 1000000 个 url.如今让你找出

top1000000url。

6、hadoop 中 Combiner 的效果?

论坛某网友的回复：

1.hadoop 即是 mapreduce 的进程，服务器上的一个目录节点加上多个数据节点，将程序

传递到各个节点，再节点上进行计算。

2.mapreduce 即是将数据存储到不一样的节点上，用 map 方法对应办理，在各个节点上

进行计算，最后由 reduce 进行合并。

3.java 程序和 namenode 合作，把数据存放在不一样的数据节点上

4.怎么运转用图来表明最好了。图无法画。谷歌下

5.不思考歪斜，功能，运用 2 个 job，第一个 job 直接用 filesystem 读取 10 个文件夹作为

map 输入，url 做 key，reduce 计算个 url 的 sum，

下一个 job map 顶用 url 作 key，运用-sum 作二次排序，reduce 中取 top10000000

QQ942609288????，???????QQ??

QQ942609288????，???????QQ??

编者 QQ：1040195253 23

第二种方法，建 hive 表 A，挂分区 channel，每个文件夹是一个分区.

select x.url,x.c from(select url,count(1) as c from A  where channel ='' group by

url)x order by x.c desc limie 1000000;

6 combiner 也是一个 reduce，它可以削减 map 到 reudce 的数据传输，进步 shuff 速度。

牢记平均值不要用。需求输入=map 的输出，输出=reduce 的输入。

十一、小萝卜(hadoop 月薪 13k)的笔试和面试题 11 道:

一、笔试

1、java 基础类：

1）继承：写的一段代码，让写出结果；

2）引用对象和值对象；

 Java 基础类记不太清了，有很多都是基础。

2、linux 基础：

1）find 用法

2）给出一个文本：比如 http://aaa.com

http://bbb.com

http://bbb.com

http://bbb.com

http://ccc.com

http://ccc.com

         让写 shell 统计，最后输出结果：aaa 1

         Ccc 2

         Bbb 3

         要求结果还要排序

     还有别的，也是比较基础的

3、数据库类：oracle 查询语句

二、面试

讲项目经验：问的很细，给纸，笔，让画公司 hadoop 的项目架构,最后还让自己说几条业

务数据，然后经过平台后，出来成什么样子。

java 方面：io 输入输出流里有哪些常用的类，还有 webService,线程相关的知识

linux：问到 jps 命令，kill 命令，问 awk,sed 是干什么用的、还有 hadoop 的一些常用命

令

hadoop：讲 hadoop1 中 map,shuffle,reduce 的过程，其中问到了 map 端和 reduce 端

溢写的细节（幸好我之前有研究过）

项目部署：问了项目是怎么部署，代码怎么管理

Hive 也问了一些，外部表，还有就是 hive 的物理模型跟传统数据库的不同

三、某互联网公司的面试：

   问到分析人行为的算法：我当时想到我们做的反洗钱项目中，有用到。我就给举例：我

们是怎么筛选出可疑的洗钱行为的。

十二、闪客、找自己、大数等提供的面试题 26 道:

\*\*\*\*信 Hadoop 面试笔试题（共 14题，还有一题记不住了）

1、hadoop 集群搭建过程，写出步骤。

2、hadoop 集群运行过程中启动那些线程，各自的作用是什么？

3、/tmp/hadoop-root/dfs/name    the path is not exists or is not accessable.

    NameNode main 中报错，该怎么解决。（大意这样 一个什么异常）

4、工作中编写 mapreduce 用到的语言，编写一个 mapreduce 程序。

5、hadoop 命令

   1）杀死一个 job 任务   （杀死 50030 端口的进程即可）

   2）删除/tmp/aaa 文件目录

   3）hadoop 集群添加或删除节点时，刷新集群状态的命令

6、日志的固定格式：

    a,b,c,d

    a,a,f,e

    b,b,d,f

   使用一种语言编写 mapreduce 任务，统计每一列最后字母的个数。

7、hadoop 的调度器有哪些，工作原理。

8、mapreduce 的 join 方法有哪些？

9、Hive 元数据保存的方法有哪些，各有什么特点？

10、java 实现非递归二分法算法。

11、mapreduce 中 Combiner 和 Partition 的作用。

12、用 linux 实现下列要求：

  ip             username

  a.txt

  210.121.123.12 zhangsan

  34.23.56.78    lisi

编者 QQ：1040195253 26

  11.56.56.72    wanger

  .....

  b.txt

  58.23.53.132   liuqi

  34.23.56.78    liba

  .....

  a.txt,b.txt 中至少 100 万行。

  1）a.txt,b.txt 中各自的 ip 个数，ip 的总个数。

  2）a.txt 中存在的 ip 而 b.txt 中不存在的 ip。

  3）每个 username 出现的总个数，每个 username 对应的 ip 个数。

13、大意是 hadoop 中 java、streaming、pipe 处理数据各有特点。

14、如何实现 mapreduce 的二次排序。

大数遇到的面试题：

15、面试官上来就问 hadoop 的调度机制；

16、机架感知；

17、MR 数据倾斜原因和解决方案；

18、集群 HA。

@找自己 提供的面试题：

19、如果让你设计，你觉得一个分布式文件系统应该如何设计，考虑哪方面内容；

每天百亿数据入 hbase，如何保证数据的存储正确和在规定的时间里全部录入完毕，

不残留数据。

20、对于 hive，你写过哪些 UDF 函数，作用是什么

21、hdfs 的数据压缩算法

22、mapreduce 的调度模式

23、hive 底层与数据库交互原理

24、hbase 过滤器实现原则

25、对于 mahout，如何进行推荐、分类、聚类的代码二次开发分别实现那些借口

26、请问下，直接将时间戳作为行健，在写入单个 region 时候会发生热点问题，为什么呢？

十三、飞哥(hadoop 月薪 13k)提供的面试题 17 道：

1、hdfs 原理，以及各个模块的职责

2、mr 的工作原理

3、map 方法是如何调用 reduce 方法的

4、shell 如何判断文件是否存在，如果不存在该如何处理？

5、fsimage 和 edit 的区别？

6、hadoop1 和 hadoop2 的区别？

笔试：

1、hdfs 中的 block 默认报错几份？

2、哪个程序通常与 nn 在一个节点启动？并做分析

3、列举几个配置文件优化？

4、写出你对 zookeeper 的理解

5、datanode 首次加入 cluster 的时候，如果 log 报告不兼容文件版本，那需要 namenode

执行格式化操作，这样处理的原因是？

6、谈谈数据倾斜，如何发生的，并给出优化方案

7、介绍一下 hbase 过滤器

8、mapreduce 基本执行过程

9、谈谈 hadoop1 和 hadoop2 的区别

10、hbase 集群安装注意事项

11、记录包含值域 F 和值域 G，要分别统计相同 G 值的记录中不同的 F 值的数目，简单编

写过程。

十四、飞哥(hadoop 月薪 13k)提供的面试题 3 道：

1、算法题：有 2 个桶，容量分别为 3 升和 5 升，如何得到 4 升的水，假设水无限使用，写

出步骤。

2、java 笔试题：忘记拍照了，很多很基础的 se 知识。后面还有很多 sql 相关的题,常用的

查询 sql 编写，答题时间一个小时。

3、Oracle 数据库中有一个表字段 name，name varchar2(10),如何在不改变表数据的情况

下将此字段长度改为 varchar2（2）？

十五、海量数据处理算法面试题 10 道：

第一部分：十道海量数据处理面试题

1、海量日志数据，提取出某日访问百度次数最多的那个 IP。

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 2^32 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

QQ942609288????，???????QQ??

QQ942609288????，???????QQ??

编者 QQ：1040195253 29

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。

或者如下阐述（雪域之鹰）：

算法思想：分而治之+Hash

（1）.IP 地址最多有 2^32=4G 种取值情况，所以不能完全加载到内存中处理；

（2）.可以考虑采用“分而治之”的思想，按照 IP 地址的 Hash(IP)%1024 值，把海量 IP

日志分别存储到 1024 个小文件中。这样，每个小文件最多包含 4MB 个 IP 地址；

（3）.对于每一个小文件，可以构建一个 IP 为 key，出现次数为 value 的 Hash map，同

时记录当前出现次数最多的那个 IP 地址；

（4）.可以得到 1024 个小文件中的出现次数最多的 IP，再依据常规的排序算法得到总体上

出现次数最多的 IP；

2、搜索引擎会通过日志文件把用户每次检索使用的所有检索串都记录下来，每个查询串的

长度为 1-255 字节。

    假设目前有一千万个记录（这些查询串的重复度比较高，虽然总数是 1 千万，但如果

除去重复后，不超过 3 百万个。一个查询串的重复度越高，说明查询它的用户越多，也就

是越热门。），请你统计最热门的 10 个查询串，要求使用的内存不能超过 1G。

典型的 Top K 算法，还是在这篇文章里头有所阐述，详情请参见：十一、从头到尾彻底解

析 Hash 表算法。

文中，给出的最终算法是：

第一步、先对这批海量数据预处理，在 O（N）的时间内用 Hash 表完成统计（之前写成了

排序，特此订正。July、2011.04.27）；

第二步、借助堆这个数据结构，找出 Top K，时间复杂度为 N‘logK。

即，借助堆结构，我们可以在 log 量级的时间内查找和调整/移动。因此，维护一个 K(该题

目中是 10)大小的小根堆，然后遍历 300 万的 Query，分别 和根元素进行对比所以，我们

最终的时间复杂度是：O（N） + N’\*O（logK），（N 为 1000 万，N’为 300 万）。ok，

更多，详情，请参考原文。

或者：采用 trie 树，关键字域存该查询串出现的次数，没有出现为 0。最后用 10 个元素的

最小推来对出现频率进行排序。

3、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。

方案：顺序读文件中，对于每个词 x，取 hash(x)%5000，然后按照该值存到 5000 个小文

件（记为 x0,x1,…x4999）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，直到分解得到

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 个词及相应

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。

4、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。

还是典型的 TOP K 算法，解决方案如下：

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。

5、 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？

方案 1：可以估计每个文件安的大小为 5G×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 hash(url)%1000，然后根据所取得的值将 url 分别存储到 1000

个小文件（记为 a0,a1,…,a999）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 小文件（记为 b0,b1,…,b999）。

QQ942609288????，???????QQ??

QQ942609288????，???????QQ??

编者 QQ：1040195253 32

这样处理后，所有可能相同的 url 都在对应的小 文件（a0vsb0,a1vsb1,…,a999vsb999）

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：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）。

Bloom filter 日后会在本 BLOG 内详细阐述。

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

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 2^32 \* 2 bit=1 GB 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：也可采用与第 1 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。

7、腾讯面试题：给 40 亿个不重复的 unsigned int 的整数，没排过序的，然后再给一个数，

如何快速判断这个数是否在那 40 亿个数当中？

与上第 6 题类似，我的第一反应时快速排序+二分查找。以下是其它更好的方法：

方案 1：oo，申请 512M 的内存，一个 bit 位代表一个 unsigned int 值。读入 40 亿个数，

设置相应的 bit 位，读入要查询的数，查看相应 bit 位是否为 1，为 1 表示存在，为 0 表示

不存在。

dizengrong：

方案 2：这个问题在《编程珠玑》里有很好的描述，大家可以参考下面的思路，探讨一下：

又因为 2^32 为 40 亿多，所以给定一个数可能在，也可能不在其中；

这里我们把 40 亿个数中的每一个用 32 位的二进制来表示

假设这 40 亿个数开始放在一个文件中。

然后将这 40 亿个数分成两类:

1.最高位为 0

2.最高位为 1

并将这两类分别写入到两个文件中，其中一个文件中数的个数<=20 亿，而另一个>=20 亿

（这相当于折半了）；

与要查找的数的最高位比较并接着进入相应的文件再查找

再然后把这个文件为又分成两类:

1.次最高位为 0

2.次最高位为 1

并将这两类分别写入到两个文件中，其中一个文件中数的个数<=10 亿，而另一个>=10 亿

（这相当于折半了）；

与要查找的数的次最高位比较并接着进入相应的文件再查找。

…….

以此类推，就可以找到了,而且时间复杂度为 O(logn)，方案 2 完。

附：这里，再简单介绍下，位图方法：

使用位图法判断整形数组是否存在重复

判断集合中存在重复是常见编程任务之一，当集合中数据量比较大时我们通常希望少进行几

次扫描，这时双重循环法就不可取了。

位图法比较适合于这种情况，它的做法是按照集合中最大元素 max 创建一个长度为 max+1

的新数组，然后再次扫描原数组，遇到几就给新数组的第几位置上 1，如遇到 5 就给新数

组的第六个元素置 1，这样下次再遇到 5 想置位时发现新数组的第六个元素已经是 1 了，这

说明这次的数据肯定和以前的数据存在着重复。这 种给新数组初始化时置零其后置一的做

法类似于位图的处理方法故称位图法。它的运算次数最坏的情况为 2N。如果已知数组的最

大值即能事先给新数组定长的话效 率还能提高一倍。

欢迎，有更好的思路，或方法，共同交流。

8、怎么在海量数据中找出重复次数最多的一个？

方案 1：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）。

9、上千万或上亿数据（有重复），统计其中出现次数最多的钱 N 个数据。

方案 1：上千万或上亿的数据，现在的机器的内存应该能存下。所以考虑采用 hash\_map/

搜索二叉树/红黑树等来进行统计次数。然后就是取出前 N 个出现次数最多的数据了，可以

用第 2 题提到的堆机制完成。

10、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。

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。

附、100w 个数中找出最大的 100 个数。

方案 1：在前面的题中，我们已经提到了，用一个含 100 个元素的最小堆完成。复杂度为

O(100w\*lg100)。

方案 2：采用快速排序的思想，每次分割之后只考虑比轴大的一部分，知道比轴大的一部分

在比 100 多的时候，采用传统排序算法排序，取前 100 个。复杂度为 O(100w\*100)。

方案 3：采用局部淘汰法。选取前 100 个元素，并排序，记为序列 L。然后一次扫描剩余的

元素 x，与排好序的 100 个元素中最小的元素比，如果比这个最小的 要大，那么把这个最

小的元素删除，并把 x 利用插入排序的思想，插入到序列 L 中。依次循环，知道扫描了所有

的元素。复杂度为 O(100w\*100)。

致谢：http://www.cnblogs.com/youwang/。

第二部分：十个海量数据处理方法大总结

ok，看了上面这么多的面试题，是否有点头晕。是的，需要一个总结。接下来，本文将简

单总结下一些处理海量数据问题的常见方法，而日后，本 BLOG 内会具体阐述这些方法。

下面的方法全部来自 http://hi.baidu.com/yanxionglu/blog/博客，对海量数据的处理方

法进行了一个一般性的总结，当然这些方法可能并不能完全覆盖所有的问题，但是这样的一

些方法也基本可以处理绝大多数遇到的问题。下面的一些问题基本直接来源于公司的面试笔

试题目，方法不一定最优，如果你有更好的处理方法，欢迎讨论。

一、Bloom filter

适用范围：可以用来实现数据字典，进行数据的判重，或者集合求交集

基本原理及要点：

对于原理来说很简单，位数组+k 个独立 hash 函数。将 hash 函数对应的值的位数组置 1，

查找时如果发现所有 hash 函数对应位都是 1 说明存在，很明显这个过程并不保证查找的结

果是 100%正确的。同时也不 支持删除一个已经插入的关键字，因为该关键字对应的位会

牵动到其他的关键字。所以一个简单的改进就是 counting Bloom filter，用一个 counter

数组代替位数组，就可以支持删除了。

还有一个比较重要的问题，如何根据输入元素个数 n，确定位数组 m 的大小及 hash 函数 个

数。当 hash 函数个数 k=(ln2)\*(m/n)时错误率最小。在错误率不大于 E 的情况下，m 至少

要等于 n\*lg(1/E)才能表示任意 n 个元素的集 合。但 m 还应该更大些，因为还要保证 bit

数组里至少一半为 0，则 m 应该>=nlg(1/E)\*lge 大概就是 nlg(1/E)1.44 倍(lg 表示以 2 为

底的对数)。

举个例子我们假设错误率为 0.01，则此时 m 应大概是 n 的 13 倍。这样 k 大概是 8 个。

注意这里 m 与 n 的单位不同，m 是 bit 为单位，而 n 则是以元素个数为单位(准确的说是不

同元素的个数)。通常单个元素的长度都是有很多 bit 的。所以使用 bloom filter 内存上通

常都是节省的。

扩展：

Bloom filter 将集合中的元素映射到位数组中，用 k（k 为哈希函数个数）个映射位是否全

1 表示元素在不在这个集合中。Counting bloom filter（CBF）将位数组中的每一位扩展为

一个 counter，从而支持了元素的删除操作。Spectral Bloom Filter（SBF）将其与集合元

素的出现次数关联。SBF 采用 counter 中的最小值来近似表示元素的出现频率。

问题实例：给你 A,B 两个文件，各存放 50 亿条 URL，每条 URL 占用 64 字节，内存限制是

4G，让你找出 A,B 文件共同的 URL。如果是三个乃至 n 个文件呢？

根据这个问题我们来计算下内存的占用，4G=2^32 大概是 40 亿\*8 大概是 340 亿，n=50

亿，如果按出错率 0.01 算需要的大概是 650 亿个 bit。现在可用的是 340 亿，相差并不多，

这样可能会使出错率上升些。另外如果这些 urlip 是一一对应的，就可以转换成 ip，则大

大简单了。

二、Hashing

适用范围：快速查找，删除的基本数据结构，通常需要总数据量可以放入内存

基本原理及要点：

hash 函数选择，针对字符串，整数，排列，具体相应的 hash 方法。

碰撞处理，一种是 open hashing，也称为拉链法；另一种就是 closed hashing，也称开

地址法，opened addressing。

扩展：

d-left hashing 中的 d 是多个的意思，我们先简化这个问题，看一看 2-left hashing。2-left

hashing 指的是将一个哈希表分成长度相等的两半，分别叫做 T1 和 T2，给 T1 和 T2 分别

配备一个哈希函数，h1 和 h2。在存储一个新的 key 时，同 时用两个哈希函数进行计算，

QQ942609288????，???????QQ??

QQ942609288????，???????QQ??

编者 QQ：1040195253 38

得出两个地址 h1[key]和 h2[key]。这时需要检查 T1 中的 h1[key]位置和 T2 中的 h2[key]

位置，哪一个 位置已经存储的（有碰撞的）key 比较多，然后将新 key 存储在负载少的位

置。如果两边一样多，比如两个位置都为空或者都存储了一个 key，就把新 key  存储在左

边的 T1 子表中，2-left 也由此而来。在查找一个 key 时，必须进行两次 hash，同时查找

两个位置。

问题实例：

1).海量日志数据，提取出某日访问百度次数最多的那个 IP。

IP 的数目还是有限的，最多 2^32 个，所以可以考虑使用 hash 将 ip 直接存入内存，然后

进行统计。

三、bit-map

适用范围：可进行数据的快速查找，判重，删除，一般来说数据范围是 int 的 10 倍以下

基本原理及要点：使用 bit 数组来表示某些元素是否存在，比如 8 位电话号码

扩展：bloom filter 可以看做是对 bit-map 的扩展

问题实例：

1)已知某个文件内包含一些电话号码，每个号码为 8 位数字，统计不同号码的个数。

8 位最多 99 999 999，大概需要 99m 个 bit，大概 10 几 m 字节的内存即可。

2)2.5 亿个整数中找出不重复的整数的个数，内存空间不足以容纳这 2.5 亿个整数。

将 bit-map 扩展一下，用 2bit 表示一个数即可，0 表示未出现，1 表示出现一次，2 表示

出现 2 次及以上。或者我们不用 2bit 来进行表示，我们用两个 bit-map 即可模拟实现这个

2bit-map。

四、堆

适用范围：海量数据前 n 大，并且 n 比较小，堆可以放入内存

基本原理及要点：最大堆求前 n 小，最小堆求前 n 大。方法，比如求前 n 小，我们比较当

前 元素与最大堆里的最大元素，如果它小于最大元素，则应该替换那个最大元素。这样最

后得到的 n 个元素就是最小的 n 个。适合大数据量，求前 n 小，n 的大小比较 小的情况，

这样可以扫描一遍即可得到所有的前 n 元素，效率很高。

扩展：双堆，一个最大堆与一个最小堆结合，可以用来维护中位数。

问题实例：

1)100w 个数中找最大的前 100 个数。

用一个 100 个元素大小的最小堆即可。

五、双层桶划分—-其实本质上就是【分而治之】的思想，重在“分”的技巧上！

适用范围：第 k 大，中位数，不重复或重复的数字

基本原理及要点：因为元素范围很大，不能利用直接寻址表，所以通过多次划分，逐步确定

范围，然后最后在一个可以接受的范围内进行。可以通过多次缩小，双层只是一个例子。

扩展：

问题实例：

1).2.5 亿个整数中找出不重复的整数的个数，内存空间不足以容纳这 2.5 亿个整数。

有点像鸽巢原理，整数个数为 2^32,也就是，我们可以将这 2^32 个数，划分为 2^8 个区

域(比如用单个文件代表一个区域)，然后将数据分离到不同的区域，然后不同的区域在利用

bitmap 就可以直接解决了。也就是说只要有足够的磁盘空间，就可以很方便的解决。

2).5 亿个 int 找它们的中位数。

编者 QQ：1040195253 40

这个例子比上面那个更明显。首先我们 将 int 划分为 2^16 个区域，然后读取数据统计落

到各个区域里的数的个数，之后我们根据统计结果就可以判断中位数落到那个区域，同时知

道这个区域中的第 几大数刚好是中位数。然后第二次扫描我们只统计落在这个区域中的那

些数就可以了。

实际上，如果不是 int 是 int64，我们可以经过 3 次这样的划分即可降低到可以接受 的程度。

即可以先将 int64 分成 2^24 个区域，然后确定区域的第几大数，在将该区域分成 2^20 个

子区域，然后确定是子区域的第几大数，然后子区域里 的数的个数只有 2^20，就可以直

接利用 direct addr table 进行统计了。

六、数据库索引

适用范围：大数据量的增删改查

基本原理及要点：利用数据的设计实现方法，对海量数据的增删改查进行处理。

七、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。

扩展：

问题实例：文档检索系统，查询那些文件包含了某单词，比如常见的学术论文的关键字搜索。

八、外排序

适用范围：大数据的排序，去重

基本原理及要点：外排序的归并方法，置换选择败者树原理，最优归并树

扩展：

问题实例：

1).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 个字节，内存限

制大小是 1M。返回频数最高的 100 个词。

这个数据具有很明显的特点，词的大小为 16 个字节，但是内存只有 1m 做 hash 有些不够，

所以可以用来排序。内存可以当输入缓冲区使用。

九、trie 树

适用范围：数据量大，重复多，但是数据种类小可以放入内存

基本原理及要点：实现方式，节点孩子的表示方式

扩展：压缩实现。

问题实例：

1).有 10 个文件，每个文件 1G，每个文件的每一行都存放的是用户的 query，每个文件的

query 都可能重复。要你按照 query 的频度排序。

2).1000 万字符串，其中有些是相同的(重复),需要把重复的全部去掉，保留没有重复的字符

串。请问怎么设计和实现？

3).寻找热门查询：查询串的重复度比较高，虽然总数是 1 千万，但如果除去重复后，不超

过 3 百万个，每个不超过 255 字节。

十、分布式处理 mapreduce

适用范围：数据量大，但是数据种类小可以放入内存

基本原理及要点：将数据交给不同的机器去处理，数据划分，结果归约。

扩展：

问题实例：

1).The canonical example application of MapReduce is a process to count the

appearances of

each different word in a set of documents:

2).海量数据分布在 100 台电脑中，想个办法高效统计出这批数据的 TOP10。

3).一共有 N 个机器，每个机器上有 N 个数。每个机器最多存 O(N)个数并对它们操作。如

何找到 N^2 个数的中数(median)？

经典问题分析

上千万 or 亿数据（有重复），统计其中出现次数最多的前 N 个数据,分两种情况：可一次读

入内存，不可一次读入。

可用思路：trie 树+堆，数据库索引，划分子集分别统计，hash，分布式计算，近似统计，

外排序

    所谓的是否能一次读入内存，实际上应该指去除重复后的数据量。如果去重后数据可以

放入内存，我们可以为数据建立字典，比如通过 map，hashmap，trie，然后直接进行统

计即可。当然在更新每条数据的出现次数的时候，我们可以利用一个堆来维护出现次数最多

的前 N 个数据，当 然这样导致维护次数增加，不如完全统计后在求前 N 大效率高。

    如果数据无法放入内存。一方面我们可以考虑上面的字典方法能否被改进以适应这种情

形，可以做的改变就是将字典存放到硬盘上，而不是内存，这可以参考数据库的存储方法。

    当然还有更好的方法，就是可以采用分布式计算，基本上就是 map-reduce 过程， 首

先可以根据数据值或者把数据 hash(md5)后的值，将数据按照范围划分到不同的机子，最

好可以让数据划分后可以一次读入内存，这样不同的机子负责处 理各种的数值范围，实际

上就是 map。得到结果后，各个机子只需拿出各自的出现次数最多的前 N 个数据，然后汇

总，选出所有的数据中出现次数最多的前 N 个数 据，这实际上就是 reduce 过程。

QQ942609288????，???????QQ??

QQ942609288????，???????QQ??

编者 QQ：1040195253 44

    实际上可能想直接将数据均分到不同的机子上进行处理，这样是无法得到正确的解的。

因为一个数据可能被均分到不同的机子上，而另一个则可能完全聚集到一个机子上，同时还

可能存在具有相同数目的数据。比如我们要找出现次数最多的前 100 个，我 们将 1000 万

的数据分布到 10 台机器上，找到每台出现次数最多的前 100 个，归并之后这样不能保证

找到真正的第 100 个，因为比如出现次数最多的第 100 个可能有 1 万个，但是它被分到了

10 台机子，这样在每台上只有 1 千 个，假设这些机子排名在 1000 个之前的那些都是单独

分布在一台机子上的，比如有 1001 个，这样本来具有 1 万个的这个就会被淘汰，即使我们

让每台机子选 出出现次数最多的 1000 个再归并，仍然会出错，因为可能存在大量个数为

1001 个的发生聚集。因此不能将数据随便均分到不同机子上，而是要根据 hash 后的值将

它们映射到不同的机子上处理，让不同的机器处理一个数值范围。

    而外排序的方法会消耗大量的 IO，效率不会很高。而上面的分布式方法，也可以用于

单机版本，也就是将总的数据根据值的范围，划分成多个不同的子文件，然后逐个处理。处

理完毕之后再对这些单词的及其出现频率进行一个归并。实际上就可以利用一个外排序的归

并过程。

另外，还可以考虑近似计算，也就是我们可以通过结合自然语言属性，只将那些真正实

际中出现最多的那些词作为一个字典，使得这个规模可以放入内存。

十六、来自 aboutyun 的面试题 6 道：

1.说说值对象与引用对象的区别？

2.谈谈你对反射机制的理解及其用途？

3.ArrayList、Vector、LinkedList 的区别及其优缺点？HashMap、HashTable 的区别及其

优缺点？

3.列出线程的实现方式？如何实现同步？

4.sql 题,是一个图表，具体忘了

5.列出至少五种设计模式？用代码或 UML 类图描述其中两种设计模式的原理？

6.谈谈你最近正在研究的技术，谈谈你最近项目中用到的技术难点及其解决思路。

十七、来自巴图提供的算法面试题 1 道：

用户手机号  出现的地点  出现的时间                 逗留的时间

111111111     2          2014-02-18 19:03:56.123445   133

222222222     1          2013-03-14 03:18:45.263536    241

333333333     3          2014-10-23 17:14:23.176345    68

222222222     1          2013-03-14 03:20:47.123445    145

333333333     3          2014-09-15 15:24:56.222222    345

222222222     2          2011-08-30 18:13:58.111111    145

222222222     2          2011-08-30 18:18:24.222222    130

按时间排序

期望结果是：

222222222     2           2011-08-30 18:13:58.111111    145

222222222     2          2011-08-30 18:18:24.222222     130

222222222     1           2013-03-14 03:18:45.263536    24

111111111   ~~~~~~~~

333333333  ~~~~~~~

QQ942609288????，???????QQ??

QQ942609288????，???????QQ??

编者 QQ：1040195253 46

十八、来自象夫提供的面试题 7 道：

Hdfs:

1.文件大小默认为 64M，改为 128M 有啥影响？

2.RPC 原理？

3.NameNode 与 SecondaryNameNode 的区别与联系？

MapReduce:

4.介绍 MadpReduce 整个过程，比如把 WordCount 的例子的细节将清楚（重点讲解

Shuffle）？

5.对 Hadoop 有没有调优经验，没有什么使用心得？（调优从参数调优讲起）

6.MapReduce 出现单点负载多大，怎么负载平衡？（可以用 Partitioner）

7.MapReduce 怎么实现 Top10？

十九、来自 mo•mo•ring 提供的面试题 13 道：

xxxx 软件公司

1.你胜任该职位有什么优势

2.java 优势及原因（至少 3 个）

3.jvm 优化

4.写一个冒泡程序

5.hadoop 底层存储设计

6.职业规划

xxx 网络公司

1.数据库

1.1 第一范式，第二范式和第三范式

1.2 给出两张数据表，优化表（具体字段不记得了，是关于商品定单和供应商方面的）

1.3 以你的实际经验，说下怎样预防全表扫描

2.网络七层协议

3.多线程

4.集合 HashTable 和 HashMap 区别

5.操作系统碎片

6.zookeeper 优点，用在什么场合

7.Hbase 中的 metastore 用来做什么的？

二十、来自 Clouds 提供的面试题 18 道：

1，在线安装 ssh 的命令以及文件解压的命令？

2，把公钥都追加到授权文件的命令？该命令是否在 root 用户下执行？

3，HadoopHA 集群中哥哥服务的启动和关闭的顺序？

4，HDFS 中的 block 块默认保存几份？默认大小多少？

5，NameNode 中的 meta 数据是存放在NameNode 自身，还是 DataNode 等其他节点？

DatNOde 节点自身是否有 Meta 数据存在？

6，下列那个程序通常与 NameNode 在一个节点启动？

7，下面那个程序负责 HDFS 数据存储？

8， 在 HadoopHA 集群中国 Zookeeper 的主要作用，以及启动和查看状态的命令？

9， HBase 在进行模型设计时重点在什么地方？一张表中国定义多少个 Column Family

QQ942609288????，???????QQ??

QQ942609288????，???????QQ??

编者 QQ：1040195253 48

最合适？为什么？

10，如何提高 HBase 客户端的读写性能？请举例说明。

11，基于 HadoopHA 集群记性 MapReduce 开发时，Configuration 如何设置

hbase.zookeeper,quorum 属性的值？

12， 在 hadoop 开发过程中使用过哪些算法？其应用场景是什么？

13， MapReduce 程序如何发布？如果 MapReduce 中涉及到了第三方的 jar 包，该如何

处理？

14， 在实际工作中使用过哪些集群的运维工具，请分别阐述期作用。

15， hadoop 中 combiner 的作用?

16， IO 的原理，IO 模型有几种?

17， Windows 用什么样的模型，Linux 用什么样的模型？

18，一台机器如何应对那么多的请求访问，高并发到底怎么实现，一个请求怎么产生的，

在服务端怎么处理的，最后怎么返回给用户的，整个的环节操作系统是怎么控制的？

二十一、来自\*\*\*\*提供的面试题 11 道：

1.hdfs 的 client 端，复制到第三个副本时宕机，hdfs 怎么恢复保证下次写第三副本?block

块信息是先写 dataNode 还是先写 nameNode?

2.快排现场写程序实现？

3.jvm 的内存是怎么分配原理？

4.毒酒问题---1000 桶酒，其中 1 桶有毒。而一旦吃了，毒性会在 1 周后发作。问最少需要

多少只老鼠可在一周内找出毒酒？

5.用栈实现队列？

QQ942609288????，???????QQ??

QQ942609288????，???????QQ??

编者 QQ：1040195253 49

6.链表倒序实现？

7.多线程模型怎样（生产，消费者）？平时并发多线程都用哪些实现方式？

8.synchonized 是同步悲观锁吗？互斥？怎么写同步提高效率？

9.4 亿个数字，找出哪些重复的，要用最小的比较次数，写程序实现。

10.java 是传值还是传址？

11.java 处理多线程，另一线程一直等待？

二十二、来自\*\*\*\*提供的面试题 18 道：

1.一个网络商城 1 天大概产生多少 G 的日志？

2.大概有多少条日志记录（在不清洗的情况下）？

3.日访问量大概有多少个？

4.注册数大概多少？

5.我们的日志是不是除了 apache 的访问日志是不是还有其他的日志？

6.假设我们有其他的日志是不是可以对这个日志有其他的业务分析？这些业务分析都有什

么？

7、问：你们的服务器有多少台？

8、问：你们服务器的内存多大？

9、问：你们的服务器怎么分布的？（这里说地理位置分布，最好也从机架方面也谈谈）

10、问：你平常在公司都干些什么（一些建议）

下面是 HBASE 我非常不懂的地方：

11、hbase 怎么预分区？

QQ942609288????，???????QQ??

QQ942609288????，???????QQ??

编者 QQ：1040195253 50

12、hbase 怎么给 web 前台提供接口来访问（HTABLE 可以提供对 HTABLE 的访问，但是

怎么查询同一条记录的多个版本数据）？

13、.htable API 有没有线程安全问题，在程序中是单例还是多例？

14、我们的 hbase 大概在公司业务中（主要是网上商城）大概都几个表，几个表簇，大概

都存什么样的数据？

15、hbase 的并发问题？

下面的 Storm 的问题：

16、metaq 消息队列 zookeeper 集群 storm 集群（包括 zeromq,jzmq,和 storm 本身）

就可以完成对商城推荐系统功能吗？还有没有其他的中间件？

17、storm 怎么完成对单词的计数？（个人看完 storm 一直都认为他是流处理，好像没有

积攒数据的能力，都是处理完之后直接分发给下一个组件）

18、storm 其他的一些面试经常问的问题？

二十三、飞哥(hadoop 月薪 13k)提供的面试题 18 道：

1、你们的集群规模？

开发集群：10 台（8 台可用）8 核 cpu

2、你们的数据是用什么导入到数据库的？导入到什么数据库？

处理之前的导入：通过 hadoop 命令导入到 hdfs 文件系统

处理完成之后的导出：利用 hive 处理完成之后的数据，通过 sqoop 导出到 mysql 数据库

中，以供报表层使用。

3、你们业务数据量多大？有多少行数据？(面试了三家，都问这个问题)

开发时使用的是部分数据，不是全量数据，有将近一亿行（8、9 千万，具体不详，一般开

QQ942609288????，???????QQ??

QQ942609288????，???????QQ??

编者 QQ：1040195253 51

发中也没人会特别关心这个问题）

4、你们处理数据是直接读数据库的数据还是读文本数据？

将日志数据导入到 hdfs 之后进行处理

5、你们写 hive 的 hql 语句，大概有多少条？

不清楚，我自己写的时候也没有做过统计

6、你们提交的 job 任务大概有多少个？这些 job 执行完大概用多少时间？(面试了三家，都

问这个问题)

没统计过，加上测试的，会与很多

7、hive 跟 hbase 的区别是？

8、你在项目中主要的工作任务是？

利用 hive 分析数据

9、你在项目中遇到了哪些难题，是怎么解决的？

某些任务执行时间过长，且失败率过高，检查日志后发现没有执行完就失败，原因出在

hadoop 的 job 的 timeout 过短（相对于集群的能力来说），设置长一点即可

10、你自己写过 udf 函数么？写了哪些？

这个我没有写过

11、你的项目提交到 job 的时候数据量有多大？(面试了三家，都问这个问题)

不清楚是要问什么

12、reduce 后输出的数据量有多大？

13、一个网络商城 1 天大概产生多少 G 的日志？ 4tb

14、大概有多少条日志记录（在不清洗的情况下）？ 7-8 百万条

15、日访问量大概有多少个？百万

QQ942609288????，???????QQ??

QQ942609288????，???????QQ??

编者 QQ：1040195253 52

16、注册数大概多少？不清楚  几十万吧

17、我们的日志是不是除了 apache 的访问日志是不是还有其他的日志？关注信息

18、假设我们有其他的日志是不是可以对这个日志有其他的业务分析？这些业务分析都有

什么？

二十四、来自 aboutyun 提供的面试题 1 道：

有一千万条短信，有重复，以文本文件的形式保存，一行一条，有重复。

请用 5 分钟时间，找出重复出现最多的前 10 条。

分析：

常规方法是先排序，在遍历一次，找出重复最多的前 10 条。但是排序的算法复杂度最低为

nlgn。

可以设计一个 hash\_table, hash\_map<string, int>，依次读取一千万条短信，加载到

hash\_table 表中，并且统计重复的次数，与此同时维护一张最多 10 条的短信表。

这样遍历一次就能找出最多的前 10 条，算法复杂度为 O(n)。

二十五、北京-南桑（hadoop 月薪 12k）提供的面试题 5 道：

1、job 的运行流程(提交一个 job 的流程)？

2、Hadoop 生态圈中各种框架的运用场景？

3、还有很多的选择题

4、面试问到的

hive 中的压缩格式 RCFile、TextFile、SequenceFile 各有什么区别？

以上 3 种格式一样大的文件哪个占用空间大小..等等

QQ942609288????，???????QQ??

QQ942609288????，???????QQ??

编者 QQ：1040195253 53

还有 Hadoop 中的一个 HA 压缩。

5、假如：Flume 收集到的数据很多个小文件,我需要写 MR 处理时将这些文件合并

(是在 MR 中进行优化,不让一个小文件一个 MapReduce)

他们公司主要做的是中国电信的流量计费为主,专门写 MR。

二十六、来自炎帝初始化提供的面试题 2 道：

以下题目不必都做完，挑最擅长的即可。

题一：RTB 广告 DSP 算法大赛

请按照大赛的要求进行相应的建模和分析，并详细记录整个分析处理过程及各步骤成果物。

算法大赛主页：http://contest.ipinyou.com/cn/index.shtml

算法大赛数据下载地址：

http://pan.baidu.com/share/link?shareid=1069189720&uk=3090262723#dir

题二：cookieID 识别

我们有 M 个用户 N 天的的上网日志：详见 58.sample

字段结构如下：

ip                   string 客户端 IP

ad\_id                string 宽带 ADSL 账号

time\_stamp           string 上网开始时间

url                  string URL

ref                  string referer

ua                   string User Agent

dest\_ip              string 目标 IP

cookie               string cookie

day\_id               string 日期

58.com 的 cookie 值如：

bangbigtip2=1; bdshare\_firstime=1374654651270;

CNZZDATA30017898=cnzz\_eid%3D2077433986-1374654656-http%253A%252F%252Fsh.58.com

%26ntime%3D1400928250%26cnzz\_a%3D0%26ltime%3D1400928244483%26rtime%3D63;

Hm\_lvt\_f5127c6793d40d199f68042b8a63e725=1395547468,1395547513,1395758399,13957594

68; id58=05dvZ1HvkL0TNy7GBv7gAg==;

Hm\_lvt\_3bb04d7a4ca3846dcc66a99c3e861511=1385294705;

\_\_utma=253535702.2042339925.1400424865.1400424865.1400928244.2;

\_\_utmz=253535702.1400424865.1.1.utmcsr=(direct)|utmccn=(direct)|utmcmd=(none); city=sh;

pup\_bubble=1; \_\_ag\_cm\_=1400424864286; myfeet\_tooltip=end; ipcity=sh%7C%u4E0A%u6D77

QQ942609288????，???????QQ??

QQ942609288????，???????QQ??

编者 QQ：1040195253 54

其中有一个属性能标识一个用户，我们称之为 cookieID。

请根据样例数据分析出 58.com 的 cookieID。

要求详细描述分析过程。

二十七、来自 aboutyun 提供的面试题 7 道：

1、解释“hadoop”和“hadoop 生态系统”两个概念。

2、说明 Hadoop 2.0 的基本构成。

3、相比于 HDFS1.0, HDFS 2.0 最主要的改进在哪几方面？

4、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。

5、“MapReduce 2.0”与“YARN”是否等同，尝试解释说明。

6、MapReduce 2.0 中，MRAppMaster 主要作用是什么，MRAppMaster 如何实现任务

容错的？

7、为什么会产生 yarn,它解决了什么问题，有什么优势？

二十八、来自然月枕流君提供的面试题 6 道：

1、集群多少台,数据量多大,吞吐量是多大,每天处理多少 G 的数据？

2、自动化运维了解过吗,你们是否是自动化运维管理？

3、数据备份,你们是多少份,如果数据超过存储容量,你们怎么处理？

4、怎么提升多个 JOB 同时执行带来的压力,如何优化,说说思路？

5、你们用 HBASE 存储什么数据？

6、你们的 hive 处理数据能达到的指标是多少？