大数据面试题（三）

1.HBase 和 HDFS 关系

HDFS是Hadoop分布式文件系统。
HBase的数据通常存储在HDFS上。HDFS为HBase提供了高可靠性的底层存储支持。
Hbase是Hadoop database即Hadoop数据库。它是一个适合于非结构化数据存储的数据库，HBase基于列的而不是基于行的模式。
HBase是Google Bigtable的开源实现，类似Google Bigtable利用GFS作为其文件存储系统，HBase利用Hadoop HDFS作为其文件存储系统；Google运行MapReduce来处理Bigtable中的海量数据，HBase同样利用Hadoop MapReduce来处理HBase中的海量数据。
HDFS为HBase提供了高可靠性的底层存储支持，Hadoop MapReduce为HBase提供了高性能的计算能力，Zookeeper为HBase提供了稳定服务和failover机制。Pig和Hive还为HBase提供了高层语言支持，使得在HBase上进行数据统计处理变的非常简单。 Sqoop则为HBase提供了方便的RDBMS（关系型数据库）数据导入功能，使得传统数据库数据向HBase中迁移变的非常方便。

2.HBase (架构)内部机制是什么？

HBase采用Master/Slave架构搭建集群，它隶属于Hadoop生态系统，由一下类型节点组成： HMaster 节点、HRegionServer 节点、 ZooKeeper 集群，而在底层，它将数据存储于HDFS中，因而涉及到HDFS的NameNode、DataNode等，总体结构如下：

HMaster节点用于：

- 管理HRegionServer，实现其负载均衡。
- 管理和分配HRegion，比如在HRegion Split时分配新的HRegion；
- 在HRegionServer退出时迁移其内的HRegion到其他HRegionServer上。
- 实现DDL操作（Data Definition Language，namespace和table的增删改，column familiy的增删改等）。
- 管理namespace和table的元数据（实际存储在HDFS上）。
- 权限控制（ACL）。

HRegionServer节点用于：

- 存放和管理本地HRegion。
- 读写HDFS，管理Table中的数据。
- Client直接通过HRegionServer读写数据（从HMaster中获取元数据，找到RowKey所在的HRegion/HRegionServer后）。

ZooKeeper集群用于：

- 存放整个 HBase集群的元数据以及集群的状态信息。
- 实现HMaster主从节点的failover

3.如何提高 HBase 客户端的读写性能？请举例说明

1 开启 bloomfilter 过滤器，开启 bloomfilter 比没开启要快 3、4 倍
2 Hbase 对于内存有特别的需求，在硬件允许的情况下配足够多的内存给它
3 通过修改 hbase-env.sh 中的
export HBASE_HEAPSIZE=3000 #这里默认为 1000m
4 增大 RPC 数量
通过修改 hbase-site.xml 中的 hbase.regionserver.handler.count 属性，可以适当的放大RPC 数量，默认值为 10 有点小。

4.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？

region 中的 rowkey 是有序存储，若时间比较集中。就会存储到一个 region 中，这样一个 region 的数据变多，其它的 region 数据很少，加载数据就会很慢，直到 region 分裂，此问题才会得到缓解。

5.请描述如何解决 HBase 中 region 太小和 region 太大带来的冲突？

Region 过大会发生多次compaction，将数据读一遍并重写一遍到 hdfs 上，占用io，region过小会造成多次 split，region 会下线，影响访问服务，最佳的解决方法是调整 hbase.hregion.max.filesize 为 256m。

6.Region 如何预建分区？

预分区的目的主要是在创建表的时候指定分区数，提前规划表有多个分区，以及每个分区的区间范围，这样在存储的时候 rowkey 按照分区的区间存储，可以避免 region 热点问题。
通常有两种方案：
方案 1:shell 方法
create 'tb_splits', {NAME => 'cf',VERSIONS=> 3},{SPLITS => ['10','20','30']}
方案 2: JAVA 程序控制
· 取样，先随机生成一定数量的 rowkey,将取样数据按升序排序放到一个集合里；
· 根据预分区的 region 个数，对整个集合平均分割，即是相关的 splitKeys；
· HBaseAdmin.createTable(HTableDescriptor tableDescriptor,byte[][]splitkeys)可以指定预分区的 splitKey，即是指定 region 间的 rowkey 临界值

7.HBase的特点是什么？

(1)hbase是一个分布式的基于列式存储的数据库，基于hadoop的HDFS存储，zookeeper进行管理。

(2)hbase适合存储半结构化或非结构化数据，对于数据结构字段不够确定或者杂乱无章很难按一个概念去抽取的数据。

(3)基于的表包括rowkey，时间戳和列族。新写入数据时，时间戳更新，同时可以查询到以前的版本。

(4)hbase是主从结构。Hmaster作为主节点，hregionserver作为从节点。

- 大：一个表可以有上百亿行，上百万列
- 面向列：面向列表（簇）的存储和权限控制，列（簇）独立检索。
- 结构稀疏：对于为空（NULL）的列，并不占用存储空间，因此，表可以设计的非常稀疏。
- 无模式：每一行都有一个可以排序的主键和任意多的列，列可以根据需要动态增加，同一张表中不同的行可
  以有截然不同的列。
- 数据多版本：每个单元中的数据可以有多个版本，默认情况下，版本号自动分配，版本号就是单元格插入时

的时间戳。
- 数据类型单一：HBase中的数据在底层存储时都是`byte[]`，可以存放任意类型的数据。

8.Hbase行键列族的概念，物理模型，表的设计原则？

行键：是hbase表自带的，每个行键对应一条数据。

列族：是创建表时指定的，为列的集合，每个列族作为一个文件单独存储，存储的数据都是字节数组，其中数据可以有很多，通过时间戳来区分。

物理模型：整个hbase表会拆分成多个region，每个region记录着行键的起始点保存在不同的节点上，查询时就是对各个节点的并行查询，当region很大时使用.META表存储各个region的起始点，-ROOT又可以存储.META的起始点。

Rowkey的设计原则：各个列族数据平衡，长度原则、相邻原则，创建表的时候设置表放入regionserver缓存中，避免自动增长和时间，使用字节数组代替string，最大长度64kb，最好16字节以内，按天分表，两个字节散列，四个字节存储时分毫秒。

列族的设计原则：尽可能少(按照列族进行存储，按照region进行读取，不必要的io操作)，经常和不经常使用的两类数据放入不同列族中，列族名字尽可能短。

9.HBase 优化

（1）高可用
在 HBase 中 Hmaster 负责监控 RegionServer 的生命周期，均衡 RegionServer 的负载，如果 Hmaster 挂掉了，那么整个 HBase 集群将陷入不健康的状态，并且此时的工作状态并不会维持太久。所以 HBase 支持对 Hmaster 的高可用配置。

（2）预分区
每一个 region 维护着 startRow 与 endRowKey，如果加入的数据符合某个 region 维护的rowKey 范围，则该数据交给这个 region 维护。那么依照这个原则，我们可以将数据所要投放的分区提前大致的规划好，以提高 HBase 性能 .

（3）RowKey 设计
一条数据的唯一标识就是 rowkey，那么这条数据存储于哪个分区，取决于 rowkey 处于哪个一个预分区的区间内，设计 rowkey 的主要目的 ，就是让数据均匀的分布于所有的 region中，在一定程度上防止数据倾斜。接下来我们就谈一谈 rowkey 常用的设计方案

（4）7.4 内存优化
HBase 操作过程中需要大量的内存开销，毕竟 Table 是可以缓存在内存中的，一般会分配整个可用内存的 70%给 HBase 的 Java 堆。但是不建议分配非常大的堆内存，因为 GC 过程持续太久会导致 RegionServer 处于长期不可用状态，一般 16~48G 内存就可以了，如果因为框架占用内存过高导致系统内存不足，框架一样会被系统服务拖死。

10.简述 HBase 中 compact 用途是什么，什么时候触发，分为哪两种，有什么区别，有哪些相关配置参数？

在 hbase 中每当有 memstore 数据 flush 到磁盘之后，就形成一个 storefile，当 storeFile的数量达到一定程度后，就需要将 storefile 文件来进行 compaction 操作。
Compact 的作用：
① 合并文件
② 清除过期，多余版本的数据
③ 提高读写数据的效率

HBase 中实现了两种 compaction 的方式：minor and major. 这两种 compaction 方式的区别是：
1、Minor 操作只用来做部分文件的合并操作以及包括 minVersion=0 并且设置 ttl 的过期版本清理，不做任何删除数据、多版本数据的清理工作。
2、Major 操作是对 Region 下的 HStore 下的所有 StoreFile 执行合并操作，最终的结果是整理合并出一个文件。

11.HBase 的特点是什么？

1）大：一个表可以有数十亿行，上百万列；
2）无模式：每行都有一个可排序的主键和任意多的列，列可以根据需要动态的增加，同一张表中不同的行可以有截然不同的列；
3）面向列：面向列（族）的存储和权限控制，列（族）独立检索；
4）稀疏：空（null）列并不占用存储空间，表可以设计的非常稀疏；
5）数据多版本：每个单元中的数据可以有多个版本，默认情况下版本号自动分配，是单元格插入时的时间戳；
6）数据类型单一：Hbase 中的数据都是字符串，没有类型。

12.大的HFile为什么要Split成小的HFile？

compact将多个HFile合并单个HFile文件，随着数据量的不断写入，单个HFile也会越来越大，大量小的HFile会影响数据查询性能，大的HFile也会，HFile越大，相对的在HFile中搜索的指定rowkey的数据花的时间也就越长，HBase同样提供了region的split方案来解决大的HFile造成数据查询时间过长问题。

其实，split只是简单的把region从逻辑上划分成两个，并没有涉及到底层数据的重组，split完成后，Parent region并没有被销毁，只是被做下线处理，不再对外部提供服务。而新产生的region Daughter A和Daughter B，内部的数据只是简单的到Parent region数据的索引，Parent region数据的清理在Daughter A和Daughter B进行major compact以后，发现已经没有到其内部数据的索引后，Parent region才会被真正的清理。

13.如何规避hbase热点写问题

检索habse的记录首先要通过row key来定位数据行,当大量的Cient访问Hbase集群的一个或少数几个节点，会造成少数RegionServer的读/写请求过多、负载过大，而其他RegionServer负载却很小，就造成了“热点”现象.

大量访问会使热点region所在的单个主机负载过大，引起性能下降甚至region不可用。
（1）加盐
这里所说的加盐不是密码学中的加盐，而是在rowkey的前面增加随机数，具体就是给rowkey分配一个随机前缀以使得它和之前的rowkey的开头不同。给多少个前缀？ 这个数量应该和我们想要分散数据到不同的region的数量一致（类似hive里面的分桶）。
（ 自己理解： 即region数量是一个范围，我们给rowkey分配一个随机数，前缀（随机数）的范围是region的数量）
加盐之后的rowkey就会根据随机生成的前缀分散到各个region上，以避免热点。
（2）哈希
哈希会使同一行永远用一个前缀加盐。哈希也可以使负载分散到整个集群，但是读却是可以预测的。使用确定的哈希可以让客户端重构完整的rowkey，可以使用get操作准确获取某一个行数据。

（3）反转
第三种防止热点的方法是反转固定长度或者数字格式的rowkey。这样可以使得rowkey中经常改变的部分（最没有意义的部分）放在前面。这样可以有效的随机rowkey，但是牺牲了rowkey的有序性。

反转rowkey的例子：以手机号为rowkey，可以将手机号反转后的字符串作为rowkey，从而避免诸如139、158之类的固定号码开头导 致的热点问题。

（4）时间戳反转
一个常见的数据处理问题是快速获取数据的最近版本，使用反转的时间戳作为rowkey的一部分对这个问题十分有用，可以用Long.Max_Value - timestamp追加到key的末尾，例如[key][reverse_timestamp]，[key] 的最新值可以通过scan [key]获得[key]的第一条记录，因为HBase中rowkey是有序的，第一条记录是最后录入的数据。

（5）尽量减少行和列的大小
在HBase中，value永远和它的key一起传输的。当具体的值在系统间传输时，它的rowkey，列名，时间戳也会一起传输。 如果你的rowkey和列名很大，HBase storefiles中的索引（有助于随机访问）会占据HBase分配的大量内存，因为具体 的值和它的key很大。可以增加block大小使得storefiles索引再更大的时间间隔增加，或者修改表的模式以减小rowkey 和列名的大小。压缩也有助于更大的索引。

14.Hbase行键列族的概念，物理模型，表的设计原则？

行键：是hbase表自带的，每个行键对应一条数据。

列族：是创建表时指定的，为列的集合，每个列族作为一个文件单独存储，存储的数据都是字节数组，其中数据可以有很多，通过时间戳来区分。

物理模型：整个hbase表会拆分成多个region，每个region记录着行键的起始点保存在不同的节点上，查询时就是对各个节点的并行查询，当region很大时使用.META表存储各个region的起始点，-ROOT又可以存储.META的起始点。

Rowkey的设计原则：各个列族数据平衡，长度原则、相邻原则，创建表的时候设置表放入regionserver缓存中，避免自动增长和时间，使用字节数组代替string，最大长度64kb，最好16字节以内，按天分表，两个字节散列，四个字节存储时分毫秒。

列族的设计原则：尽可能少(按照列族进行存储，按照region进行读取，不必要的io操作)，经常和不经常使用的两类数据放入不同列族中，列族名字尽可能短。

15.HBase简单读写流程？

读：

找到要读数据的region所在的RegionServer，然后按照以下顺序进行读取：先去BlockCache读取，若BlockCache没有，则到Memstore读取，若Memstore中没有，则到HFile中去读。

写：

找到要写数据的region所在的RegionServer，然后先将数据写到WAL(Write-Ahead Logging，预写日志系统)中，然后再将数据写到Memstore等待刷新，回复客户端写入完成。