Google: Big Table

FaceBook: Cassandra and HBase

Yahoo: HBase

Amazon: DynamoDB

问什么既有 文件系统 又有 数据库系统

* 文件系统：
  + 输入： /home/jonyong/character\_name.txt。里面存了每个人的颜值身高
  + 输出： 文件内容
  + 如果需要 找到 令狐冲的 颜值，需要读取整个文件，for 循环，找到令狐冲颜值，如果query频繁，并没法很有效的找到信息。
  + 文件系统的缺点： 只能提供简单的读写文件操作。实际操作中有复杂的查询需求： 比如查询令狐冲颜值，查询颜值<5
  + 所以我们需要一个更复杂的系统 建立在文件系统之上。
* 数据库系统：
  + 建立在文件系统之上，
  + 负责组织把一些数据存到文件系统
  + 对外接口比较方便操作
  + 什么是数据库系统：
    - 操作：输入key，输出value
    - 数据库可以建index，来查询令狐冲颜值
* Scenario 需求：
  + 比较简单，给web server查询
* Storage:
  + 数据库最终都会存到文件里面，作为文件存到文件系统上
  + 在文件系统上，怎么样更好的搭建数据库系统？
    - 读取文件到内存里面，内存里面查询？ 这样不行，因为内存放不下所有数据
    - 先对数据进行排序？ -- 所以要在文件系统上排序？？ 使用外排序：一块一块load到内存进行排序，然后进行K Merge。排好序之后，查找数据就能在文件系统里进行二分查询。。这里排序的可能是按人的名字，每个人最后会有一个offset
    - 如何再大硬盘，小内存的情况下进行二分： 根据每个数据的offset，然后把二分点的chunk load到内存， 再进行二分，把二分点load到内存。
    - 查询解决了，有一天令狐冲整容了怎么办？从颜5渣要变成颜6
      * 三种方案：1
        + 1. 直接在文件修改：问题出现在如果把颜值从5改成10^8，会覆盖下面数据
        + 2. 读取文件，改了，再写回：太慢：把整个数据库文件读出来抬到了
        + 3. 在后面append令狐冲新的颜值： 写入速度快！！！问题出现在，查询的时候，找到了令狐冲的两条记录（根据时间戳）。Append的问题： 数据会不停增多， 并且没法再采用二分操作去查询数据（因为不是有序的了）

怎么解决：每过一小段时间，整理成有序的，并且删掉重复的。

为什么要高效写数据：BigTable以及NoSQL的理念是，能够大量快速的写，查询的时候，用别的方法进行优化，不至于太难。

有没有一种方法，既可以读的时候二分查询，又可以写的时候在最后append？-- 分块有序。每次

e.g. 以名字作为key进行排序，前面是有序的，只有最后一块是无序的。 前面保证用二分，后面是无序的，定期整理， 用for循环去搞最后一块。

块会越写越多， 每一块都有重复，怎么办？ 定期做一个K路归并：把两个有序块整理成1个有序块。K merge，同时可以删除重复数据。怎么把无序块整理成有序块？ 外排序。 看评论

* + - * 写入过程：
        + 直接append到无序文件末尾：memory中记录当前无序file对应的address(offset). 快!!!!!!
        + 无序文件积累到一定大小之后，就进行外排序

可不可以一开始就存在内存里面？ 可以。

怎么知道哪个文件是有序的，哪个文件是无序的？给最后一个文件编号作为无序文件。

内存排序　＋　１次硬盘写入。

持久化Serialization：怎么把内存中的数据写入到disk? Serialization-- LIntcode 上 Binary Tree Serialization 题做一下。 做一下binary tree serialization的题。

机器挂了，内存没了，怎么做备份：Write Ahead Log(WAL). 掉电后可以快速通过WAL进行恢复。既然写了WAL到硬盘，岂不是又要写硬盘。 WAL使用timestamp来记录operation，而serailize是 按照key排序整理好然后进行排序的。WAL会比直接写硬盘快，有一些优化操作。 看链接：HBase怎么实现WAL

* + - * 读出过程：
        + 现在看的都是一台机器上的操作，机器文件系统expose写一个key或读一个key的操作。
        + 令狐冲存在于Sorted list(memory）， File 0， file 1，哪个是真的

需要把3个全读吗，不用，先看memory的，memory没有令狐冲，就查file 1， 如果还不存在，查file 0

一个file里，怎么查询令狐冲？

１.　读出来ｆｏｒ循环

２.　硬盘二分

３.　更好的方法？　建立Index.

比较简单的建index的方法：Hashtable：ｋｅｙ是名字，ｖａｌｕｅ是名字的地址。还是不太可行，占用内存太大，只建部分索引：以Ａ作为开头字母对应的地址是多少，Ｄ开头对应地址多少，Ｓ开头。。所以可以仅仅读一小块。　相似问题：ＬＩｎｔｃｏｄｅ题：　Ｉｎｔｅｒｓｅｃｔｉｏｎ　of two arrays II: 建一个内存中放得下的index， 之后用index + 硬盘二分来解决问题。

比较复杂的index： B true， B+ tree 进行index

读硬盘的时间 = 读内存 \* （30 ~ 100）

有没有更好的方法检查一个key在不在一个File里面？ **BloomFilter**：快速检查一个ｋｅｙ在不在ｆｉｌｅ里

构造hashtable。 Hash1(Ling)=1, Hash2(ling) = 2; Hash1(hu) = 5, Hash2(Hu) = 7。 把ling和hu插入了bloom filter的数组。

检查时（e.g. check if chen is in the table）Hash1（Chen） = 3， Hash2（Chen） = 5。 如果3下标的表对应为0，说明不在。

优点： key的空间很小，因为把keyhash过了。

问题： 有可能查询时，明明不存在，但是在数组中还是有entry。

Hash过后如果不在bloomFilter里，则一定不在。 如果在的话，则有可能在。

误判率： False is always false, True may be true. How many false is hidden in true? Hash函数个数越多，误判率越低。位数组越长，误判率越低。

工业界： 15个Hash Function， 维数组大小200w，加入的字符串10w， 判断2000w个字符， 大概3%误判率。 wiki上面有具体计算公式。

* + - * + 完整读入过程：

机器上有2个有序文件。内存中有两个文件对应的address和bloom filter以及index。 在内存的sorted list上 没有必要建立Bloom Filter，因为已经在memory了，没有必要。 文件系统中每个有序File也存对应的bloomFilter。

查看Read Key linghuchong. 先看sorted List，如果不存在，然后看File 1的bloomFilter，没有filter掉之后，看index数组，这样比硬盘直接二分要快。 BloomFilter快还是Indexcheck 快？ BloomFilter， 因为Index check不管怎样，还是要读硬盘，而bloomFilter全部在memory中。

* + - Specific Name in big table：
      * String is stored in the file，所以用SsTable表示排好序的文件 = Sorted String Table
      * Sorted List 用 Skip List实现。 可以快速排序。 中文名字：跳跃表
    - 我们已经学会了一台机器Big Table的操作： 读/写的过程。 Skip LIst code已经开源了。 在github上。
  + Scale：sharding?
    - Horizontal vs. Verticle sharding？
      * Horizontal 横向去切，一行一行分sharding
    - Column Family：
      * 两个column关系比较密切
    - 使用consistent hashing来实现sharding：
    - 一台机器搞不定，那么需要多台机器：
      * Master + Slave（GFS和BIg Table的Master/Slave 是在一个机器群体，还是不一样的群体： 一般是不一样的机群）
      * Master manage all slave server.
    - How to read/write?
      * 先找到令狐冲这个key对应的server id是多少，然后写入到硬盘。请问：这里的硬盘是Server上的硬盘，还是写入GFS？ 其实是写入到GFS上
      * 如果令狐冲名字改了，index要不要变？我们说的index是sstable上的index，写入了就不会变的。因为我们在append，后面 的index变了。
    - BigTable -> GFS. big table最后数据是存在Big Table。 所说的硬盘是GFS上的硬盘。 那 本地硬盘还要不要？　要！　ＷＡＬ要写在本地硬盘，同时本地硬盘可能需要作为Buffer
    - SsTable是怎么写到GFS里的呢？
      * 每个SsTable可以当做一个文件，以文件形式，拼成一个chunk存在GFS里面。
    - What is Tablet？ 每个小表。所以每个Slave server叫做Tablet Server
  + Big Table 给用户提供的框架？
    - {Row key1: Column Key 1}: Column Value 1
  + 还有什么问题没有解决：
    - 对于big table来说，Race Condition。如果既被读，又被写 怎么办？
      * 加锁！Distributed Lock Service： Chubby, Zookeeper
        + Lock Server. Consistent Hash Map 给hash Map。 通过lock server得知key是否被Lock
    - 对于GFS，为什么没有锁
      * 因为GFS是不修改的，而对Big Table来说，是会被修改的。
  + 对于海量数据，快速过滤：
    - BloomFilter
  + 参考阅读。
  + 先有BigTable，生下了HBase。Facebook偷学了HBase，和DynamoDB，写出来了Cassandra

Discussion:

# **Audience Question**

# **Q: mongo 呢**

A: mongoDB也是nosql，nosql很多种，我们就不一一提到了

# **Audience Question**

# **Q: 想问下面试里问的key-value store是cassandra之类的还是memcached之类的？**

A: 后者 cassandra 有row key和column key 更加复杂一些，memcached是key-value的缓存，不是数据库，redis则是key-value的内存型的Nosql

# **Audience Question**

# **Q: 令狐冲个子好矮！：）**

A: 估计被砍了一刀

# **Audience Question**

# **Q: nosql索引也是b树么？**

A: 不是的，b树为硬盘的索引准备的，目的是减少磁盘的访问次数

# **Audience Question**

# **Q: 数据库的数据是不是还是存在硬盘里？**

A: 是的，文件系统和数据库都是数据的组织方式的，都是抽象的逻辑

# **Q: no**

# **Audience Question**

# **Q: 那么nosql的索引的原理是什么呢**

A: 不同的Nosql对index的处理也不一样，比如MongoDB他的索引和mysql的有些类似，redis是key-valye储存，他并不直接支持索引

# **Audience Question**

# **Q: client和server是不是都是相对性的？一个client有可能是另一个client的server？**

A: 是的，理解的非常正确

# **Audience Question**

# **Q: 我不太理解“数据库是建立砸文件系统之上的‘这句话**

A: 就是说 数据库最后的数据都是存在文件系统上

# **Q: 2**

# **Audience Question**

# **Q: 看过，本科还实现过，作业。。**

A: 赞

# **Audience Question**

# **Q: 那k merge的時候又怎麽能保證内存夠用呢？**

A: k路归并，只需要在内存存大小为k个heap就可以了

# **Audience Question**

# **Q: 上面这个问题，在内存存大小为k个heap是什么意思？k个heap在内存能存下吗**

A: 因为你要做k路归并，那么需要一个大小为k的heap来归并，k如果太大存不下，那么就缩小知道k的范围能够使得内存能够放下这个heap

# **Audience Question**

# **Q: 数据库的数据类型不是固定的吗,怎么能int变long？**

A: 可以修改数据库的schema

# **Q: 还有排序问题怎么解决？**

# **Audience Question**

# **Q: 所以如果K路归并，如果k大于内存size, 那么就先对一部分做k merge, 之后k变小了，再进行k merge，是这个意思吗？**

A: 对的，可以一部分一部分合并

# **Audience Question**

# **Q: 那旧的令狐冲数据不需要delete吗？**

A: 旧的可以定时删除，或者你设置我们需要保存多少个version，version多了，多余的删除

# **Q: 修改文件后需要重新排序吗？**

A: 都是append进去的sstable有序即可

# **Audience Question**

# **Q: 那以前的记录就一直放在数据库永远不动吗**

A: 好问题，可以引进一些方法处理这个问题，比如保存若干版本的数据，好多数据库都是可以设置你保留n的版本的数据

# **Audience Question**

# **Q: 硬盘外排序在什么时候发生？有个单独线程周期性做吗？**

A: 在数据append进去的时候 以及多个有序文件做合并的时候 需要用到

# **Audience Question**

# **Q: 所以SQL和NOSQL数据库后面都一定有一个文件系统作支持么？**

A: 他们的数据持久化后都是写在文件系统上的

# **Audience Question**

# **Q: 请问这个设计的是nosql还是包括sql和nosql?**

A: 这个设计是对于bigtale的，不同的nosql的设计也是不一样的，比如mongoDB就有类似于mysql的index。redis直接不支持index

# **Audience Question**

# **Q: 那么每个column都要存一个排好序的copy？**

A: 我们对不同的column 其实会有不同的存储，这个叫做列式存储，也就是row 对应一个column这是第一个存储，row 对应第二个column是第二个存储，就是纵向划分，不一定是copy，只是column被分开来存了，是否有点感觉？ （Bigtable不是完全的列式存储，Hbase是完全意义的列式存储），有关内容可以课后扩展。

# **Audience Question**

# **Q: 无序的那一块如何整合到有序呢？还是一直无序？**

A: 无序的那块是被放在内存里的，之后过多了 就被有序化 输出到硬盘上

# **Audience Question**

# **Q: 一般一块是多大呢？**

A: 根据你实际的数据可以优化块的大小

# **Audience Question**

# **Q: 如果数据被持久化之后，之前存在内存中的数据还在么？还是会清空？**

A: 不会的，除非断电或者内存不工作了，然后我们既可以从硬盘上开始恢复数据

# **Audience Question**

# **Q: 如何知道哪个块是有序快哪个块是无序块呢？**

A: 最后一个快是无序的，已经输出到硬盘上的都是有序的，

# **Audience Question**

# **Q: 有序指的是文件之间有序 还是文件里的数据有序啊？**

A: 文件内的数据是有序的

# **Audience Question**

# **Q: 如果机器down了怎么办，那一块就丢失了**

A: 所以数据要有备份，来容灾

# **Audience Question**

# **Q: 写入到内存，在硬盘做持久化备份，和直接写入硬盘有什么优势呢？不都是要写到硬盘里吗**

A: 这里有一个整理和排序的过程，是直接写入硬盘不能完成的

# **Audience Question**

# **Q: 也就是说无序的内容都存在最后一个文件里 ？ 还是无序的数据存在多个文件里 ？**

A: 只存在最后一个，其他输出整块的内存都是有序的

# **Audience Question**

# **Q: serialization是什么**

A: 序列化操作，你可以理解为就是把一个数据结构变成一个字符串

# **Audience Question**

# **Q: 具体说说写的什么数据到write ahead log**

A: 把写操作 写入到log里，这样就能恢复数据了，注意只需要写操作

# **Audience Question**

# **Q: white ahead log 没明白是什么log? 有什么作用呢？**

A: log就是写的操作，你把所有写的操作都记录下来了，从头执行一遍，数据不就恢复了么

# **Audience Question**

# **Q: 老师老师，没听懂为啥必须要有write ahead log**

A: 恢复数据，容灾

# **Audience Question**

# **Q: 这个log存在硬盘里？**

A: 是的

# **Audience Question**

# **Q: 内存里的数据写入硬盘后，WAL还有用吗**

A: 有用，序列化丢数据多，log丢数据少，log顶多就丢几次写操作，序列化可能会丢一段时间的数据

# **Audience Question**

# **Q: log是用什么存储的 log写入到硬盘里 内存也写到硬盘里**

A: 有很多log系统，他也有会优化，比如几个操作，合并成一个写操作，压缩log，这个你可以网上搜相关的log系统，log相比快照或者序列化可以确保丢失的数据极少，也就是最近的几个写操作可能丢失。

# **Audience Question**

# **Q: 文件不是1024M么，不能直接读入内存再二分？**

A: 这个load 不就是扫描了所有数据么，那么排序就没有意义了

# **Audience Question**

# **Q: 但是index是给不怎么更新的数据建吧，如果经常更新，index反而效率低是吗？**

A: 是的，经常修改的数据加index效率很低

# **Audience Question**

# **Q: bloom filter**

A: 是的 非常正确

# **Q: 3**

# **Audience Question**

# **Q: bloom filter**

A: 是的 Bloom filter说有 也可能没有这个数据，但是Bloom filter说没有 则肯定没有这个数据

# **Audience Question**

# **Q: 为啥Hash两次**

A: 其实是hash k次，bloom filter 由k个hash函数，根据k 和 size的大小 可以计算出一个容错率 这里以2个hash 举例

# **Audience Question**

# **Q: 有没有可能Chen的hash1 和 hash2 与Ling的是一样的**

A: 有，所以bloom filter 有一个容错率，他说有，则可能是没有的，但是BF说没有，则一定没有

# **Audience Question**

# **Q: 为什么做两个hash？**

A: 实际k个，这里两个hash举例子

# **Audience Question**

# **Q: 问什么两次hash出不同的值呢？**

A: 因为不同的hash函数，实际有k个\*不同的\*hash函数

# **Audience Question**

# **Q: 尴尬。。。bloomfilter这还是我本科毕设。。。竟然不记得了**

A: 不错，想起来了就好

# **Audience Question**

# **Q: 哈希函数越多，误判率越低；数位组越高，误判率越低；？**

A: 不，size一定的时候，可以根据你需要的容错率，计算一个合适的k，不是hash函数越多越好，你想如果hash函数很多，整个数组都是1 了

# **Audience Question**

# **Q: 然后call同一个hash 函数实际上是随机从K个中挑一个吗**

A: k个hash函数是固定的，依次执行计算k个下标

# **Audience Question**

# **Q: 为什么先用bloomfilter判断？**

A: 快啊

# **Audience Question**

# **Q: 为啥每个file需要bloomfilter，不是之间memory里面有bloomfilter就行了？**

A: 如果bloom filter无法告诉你在哪个块里面，他只能告诉你有木有

# **Audience Question**

# **Q: sstable 是用value排序?**

A: 按照key排序的

# **Audience Question**

# **Q: 哦 横向切**

A: Bigtable是横向和纵向都有，Hbase是完全纵向切

# **Audience Question**

# **Q: 刚才走神了 这里的bloom filter 是指的那一个那数组嘛**

A: 01数组+若干个hash函数

# **Audience Question**

# **Q: 工作中什么环节会让我真正去自己配置一个master slave的server system呢**

A: 如果你需要用到这些系统的话

# **Q: 硬盘本身是不是就是GFS？**

# **Audience Question**

# **Q: 写操作不是Master完成吗？slave读？**

A: 不是master 完成，如果master参与写，那么master将成为瓶颈

# **Audience Question**

# **Q: 为什么nosql的数据要根据key排序存在硬盘上，有没有可能存不排序呢？**

A: 排序主要是为了加速查询

# **Audience Question**

# **Q: sstable永远不变，那令狐冲的每份写记录都会记录下来？之前不是说会整理么？**

A: 会整理的，会有合并数据的步骤

# **Audience Question**

# **Q: lock server是要等到client1的任务全部完成再给2 吗**

A: 不是的，你只是锁在一个操作上，不是整个client的全部操作

# **Audience Question**

# **Q: GFS为什么不用lock？ 能再讲讲吗？老师老师**

A: 因为GFS支持的是一个无锁并发append

# **Q: distributed lock是一台单独的服务器？**

A: 不是的，这也是一个分布式系统，你可以了解一下chubby