BigTable nosql的数据库

文件系统：给一个文件名和文件内容，写入到系统当中。读取时，根据文件名读取出内容来。

文件系统的不足：只提供简单的读写功能，对于复杂的查询则比较麻烦，需要自己写代码，性能不一定最优，而且还要考虑各种异常处理。

数据库系统是建立在文件系统之上，负责组织数据存储在文件系统上，提供对外的接口比较方便操作数据。

BigTable的修改：不会修改原来的数据，而是在文件后面追加一条数据。带来的问题：

1. 读取数据怎么识别哪个是最新的记录

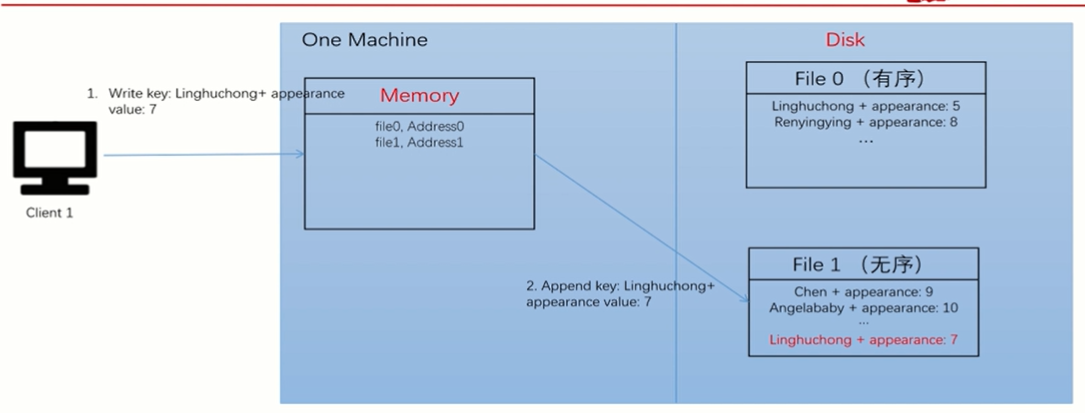
BigTable每条插入的数据都带有一个时间戳，时间戳最大的那个就是真正的数据。

1. 没有顺序，怎么进行类似二分的快速查找

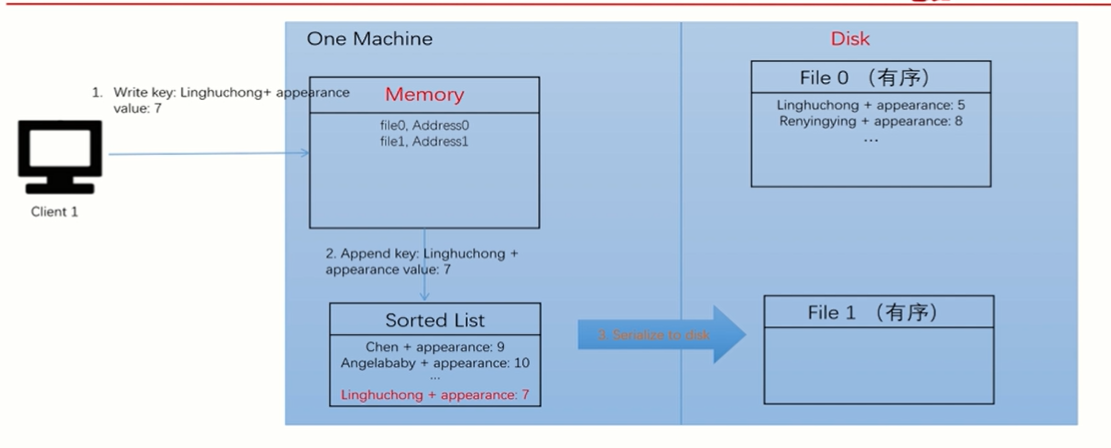
将数据进行分块，每一个块都是内部有序的，插入数据时，只有最后一块是无序的，并且每隔一段时间整理成有序的。BigTable中块的大小通常是256M，当到达这个阈值时，就不会再插入，而会进行整理成有序块，之后再有数据写入，就新建一个块。

块会越写越多，部分key会出现大量的重复，可以定期对有序块进行k路归并排序，这样相同的key就会排在一起，然后在根据时间戳筛选，删除掉过期的数据。

BigTable一个可行的写入过程：

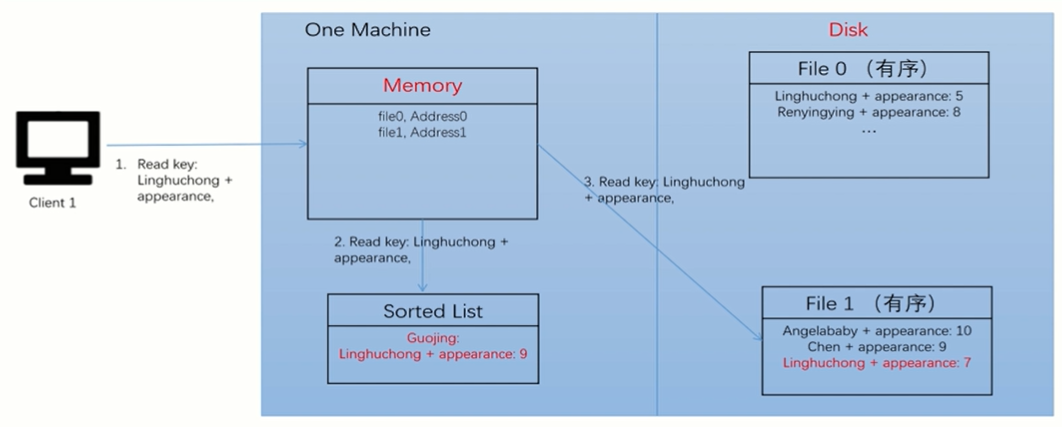


最后一个无序的File可以从一开始就可以存在内存中，因为一个块最大256M，完全可以放在内存中，可以使用skip list这样的数据结构，让它也是有效的，这样当到达阈值256M时，直接一次写入硬盘中。

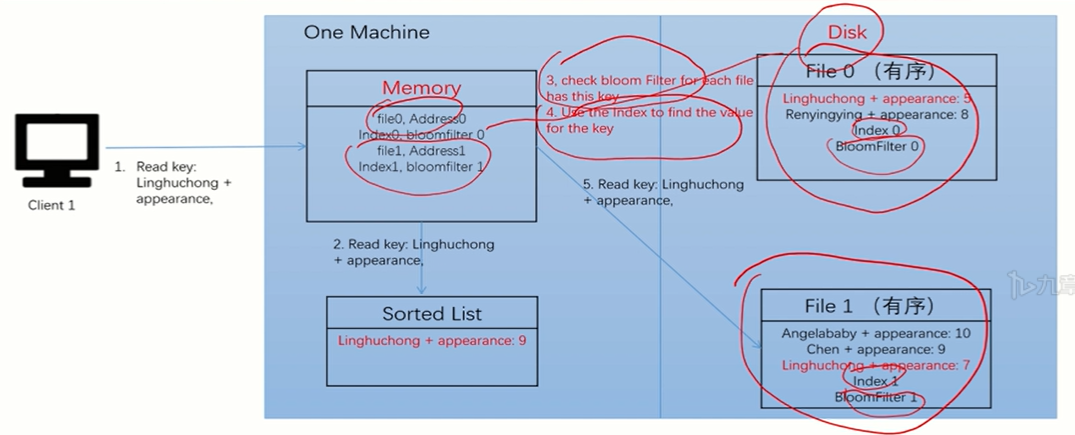


最后一个块先放到内存中，如果这时机器挂了，内存就没了。BigTable采用Write Ahead Log的方式，每次写入数据到内存时，先写一次硬盘，记录一个log。这里不用担心WAL会拖慢速度，WAL非常简单方便，仅仅是一个append操作，这是非常快的。

BigTable的数据读取：先从最后那个内存中的无序块查找，因为数据带有时间戳，如果在最后的无序块中查找到对应的key，则它肯定是我们想要的数据。如果找不到，就去倒数第2个有序块中查找，如果找到就停止，如果没找到就继续在前面的有序块中查找。



添加Index和布隆过滤器后的读取过程：



BigTable中有序块就是SSTable，即Sorted String Table

布隆过滤器：是一个数据结构，用来检测某个元素在一个集合中存不存在。

布隆过滤器是个大的bit array.是将输入的key经过一个或多个hash函数并对bit array的长度取模，算出一个下标索引值，然后把bit array对应的位置设置为1.

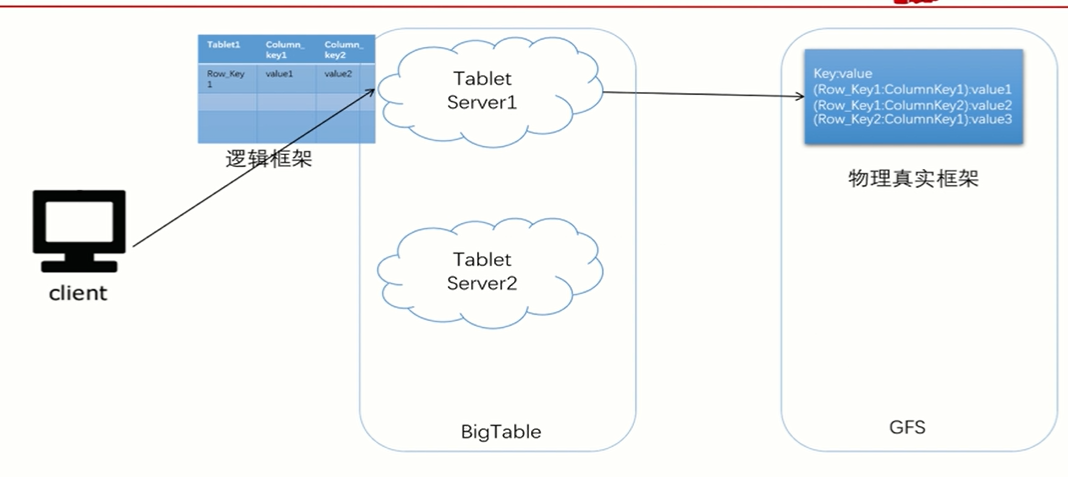
使用布隆过滤器时，把输入的key经过多个hash函数后得到的下标索引在bit array上看看对应的位置是否为1，如果不全部为1，则该key肯定是不存在的，如果全部都为1，则表示该key有可能存在。

相比于hash table，布隆过滤器占用的空间要小很多。而且写到硬盘上时，布隆过滤器在硬盘上的形式和内存中没有变化，就是一个二进制数组，而hash table很多key存储的都是链表，序列化要花更多的时间。



布隆过滤器的膨胀：如果key值增多，有可能之前的存储布隆过滤器的二进制数组会不够用，这样就需要建立一个新的bit array。但不能直接把之前的bit array中的值直接复制过来，因为布隆过滤器的哈希算法会对bit array的长度取模，旧的bit array中的值不能直接映射到新的bit array上，这样就需要把旧的bit array保留下来，在判断key是否存在时，需要在旧的bit array和新的bit array上都判断一次。

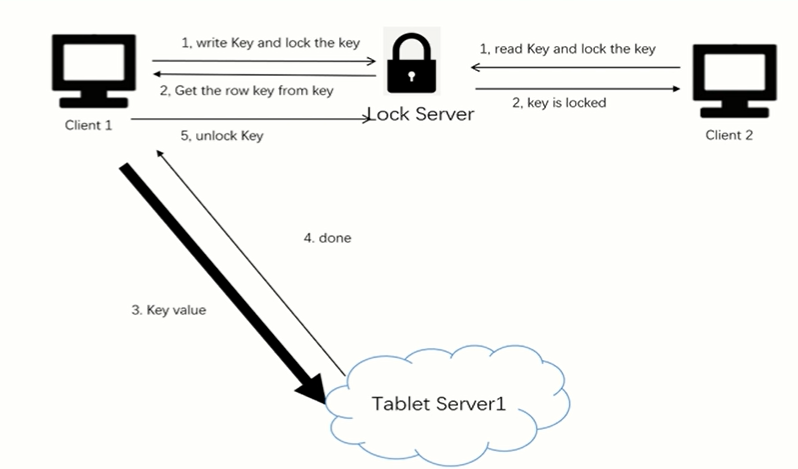
整体的BigTable的构架：



在逻辑层面上的一张表在bigtable中会拆分成不同的Tablet Server，而实际会存储在GFS上。

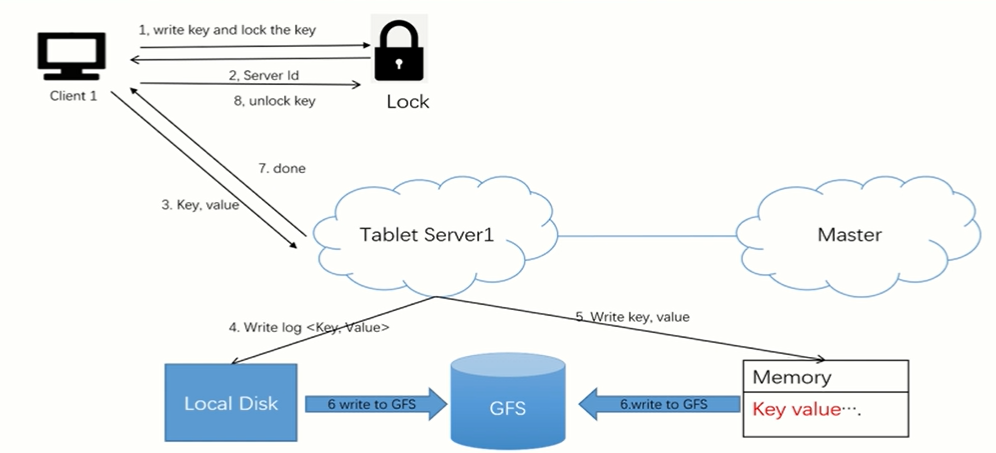
nosql数据库的拆分，可以对key进行一致性哈希，然后拆成不同的表。

BigTable的分布式锁：chubby和zookeeper。



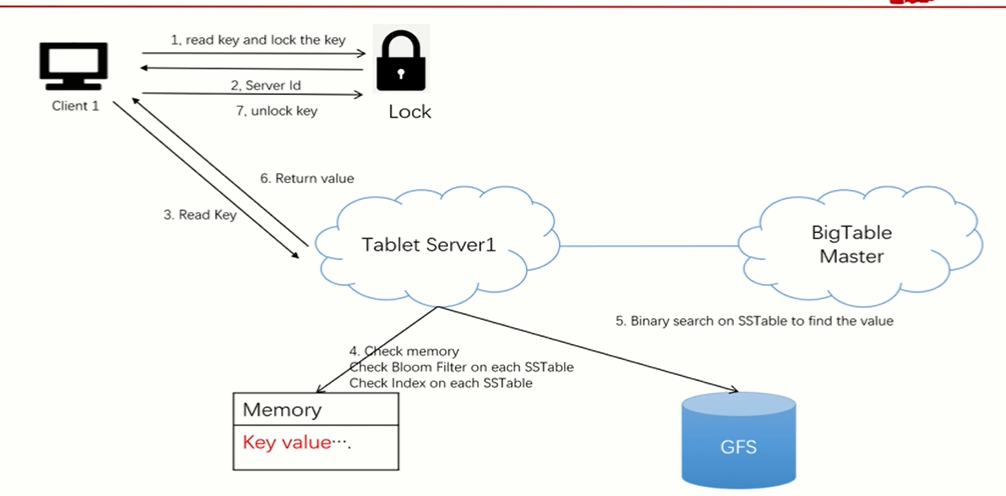
Metadata可以从master上迁移到Lock server上。

完整的分布式写入：

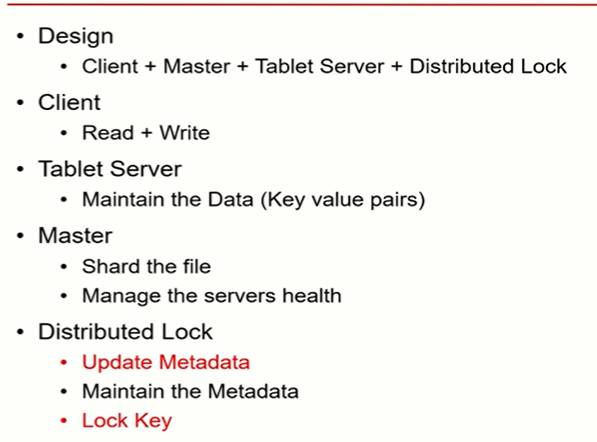


1. client去lock server请求锁
2. lock server锁住这个key，并且返回key所在服务器的server id
3. client把key,value写入到对应的tablet server
4. tablet sever会写写一条log，防止宕机导致内存数据丢失，
5. 然后把key.value写入到内存中
6. 日志和内存中的数据都会写入到GFS中
7. 全部完成后会告知client
8. client通知lock server释放锁

完整的读取流程：



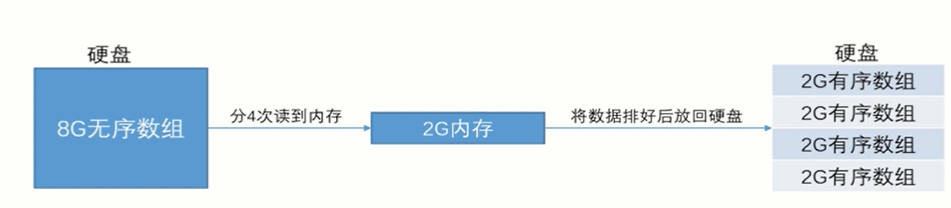
BigTable的组成：



外排序：



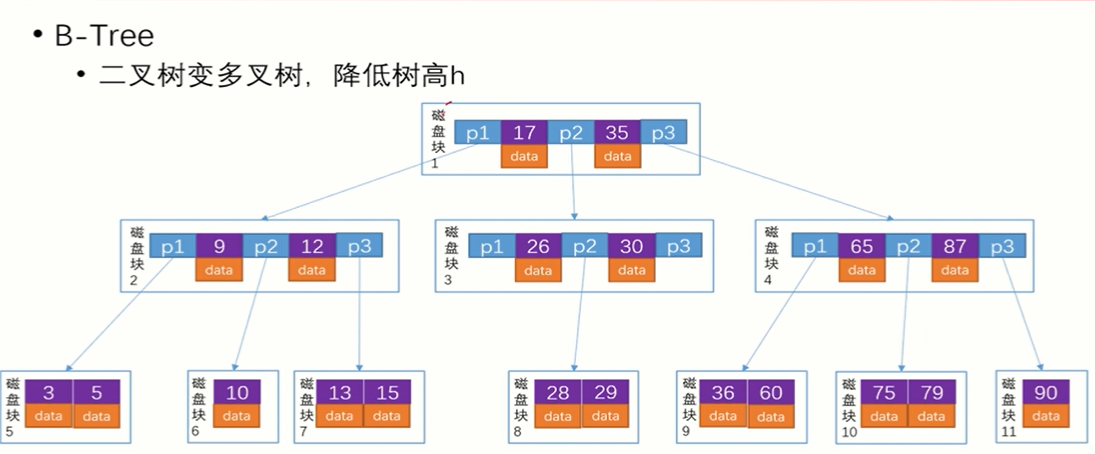
第一步：



第二步：



如果数据量非常大，使用二叉查找树时，其时间复杂度为，如果N为10亿，则需要查找9log2(10)，大约要30次，在硬盘上查找30次的话，一次寻轨为10ms，这样查找一次就要300ms，还是比较慢的。这是需要把二叉树转换为B树，以降低树高。



上面的B树，由二叉变成了三叉，小于17的放在根结点的左边，大于17小于35的放中间，大于35的放右边。

B树的问题：非叶结点data(上图中的黄色块)占用太多空间，能存储的key(上图中的蓝色和紫色块)的数量少，导致叉树比较少，因为一个磁盘块的大小是有限的。

B+树：非叶子结点不存data：这样非叶子结点所在的磁盘块都可以存key。把所有的data都放在叶子结点中。

