

系统设计

Distributed System Design 2

（九章网站下载最新课件）

本节主讲人：韩愈老师

版权声明：九章课程不允许录像，否则将追究法律责任，赔偿损失

- Design a Bigtable
 - NoSQL database 设计框架和原理
 - SStable 读和写
 - Bloom Filter

Interviewer: What is bigtable?

\aleph_0 脸茫然
 $\left\{ \text{脸茫然} \right\}_{n=0}^{\infty}$

What is bigtable?

NoSQL DataBase	Company
Bigtable	Google
Hbase	Open Source of Bigtable
Cassandra	Facebook
DynamoDB	Amazon

Comparison of different No SQL database:

<https://www.digitalocean.com/community/tutorials/a-comparison-of-nosql-database-management-systems-and-models>

为什么我们要讲bigtable 的实现？

1. Google面试题
2. 解决相类似系统设计题,比如:Look up service
3. 追问NoSQL How to scale的原理



文件系统 vs 数据库系统

什么是文件系统？

操作：

输入： /home/jinyong/character_name.txt

输出：文件内容

如果有下面需求 找到“令狐冲”的“颜值”

- 1、打开文件
- 2、For循环扫描文件的内容
然后找令狐冲的颜值

```
{  
  {姓名:'令狐冲', '颜值':5, '身高':'160cm'},  
  {姓名:'郭靖', '颜值':9, '身高':'180cm'},  
  {姓名:'东邪', '颜值':7, '身高':'170cm'},  
}
```

/home/jinyong/character_name.txt

文件系统不足？

文件系统提供一些简单的读写文件操作

实际查询当中有复杂的查询需求：

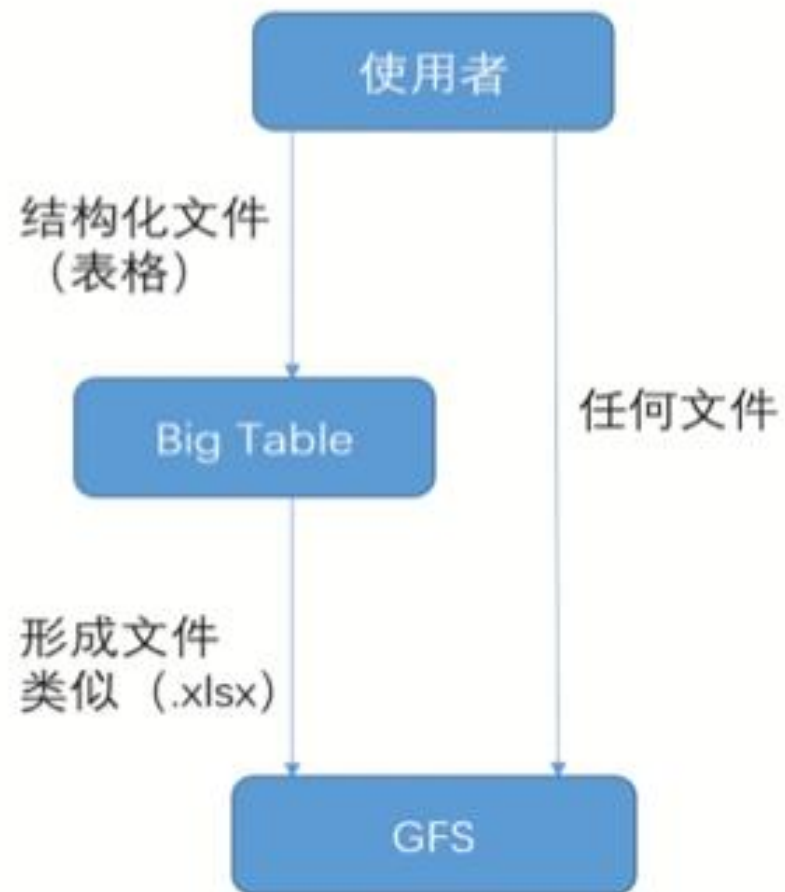
比如：查询令狐冲颜值

查询颜值小于5的

所以我们需要一个更复杂的系统建立在文件系统之上

Big Table 与 GFS关系

	GFS	Big Table
本质	分布式文件系统	分布式NoSQL Database
开发公司	Google	Google
是否开源	否	否
类似开源产品	HDFS	HBase
基本操作	读 & 写文件	增删改查记录
类似什么	家用电脑的文件系统	大数据版的Excel



设计数据库系统

Scenario 需求

查询: key (令狐冲 + 颜值)

返回: value (5)

因为后端系统通常给web server使用, Scenario比较单一

Storage

数据库怎么存储 以表的形式?

姓名	颜值	身高
令狐冲	5	160
郭靖	9	180
东邪	7	170

数据最终都会存到文件里面

```
{  
  {'姓名': 'Linghuchong', '颜值': 5, '身高': '160cm'},  
  {'姓名': 'Guojing', '颜值': 9, '身高': '180cm'},  
  {'姓名': 'Dongxie', '颜值': 7, '身高': '170cm'},  
}
```


从文件系统基础上思考 搭建数据库系统

在文件里面 怎么更好支持查询操作？

```
{  
  {'姓名': 'Linghuchong', '颜值': 5, '身高': '160cm'},  
  {'姓名': 'Guojing', '颜值': 9, '身高': '180cm'},  
  {'姓名': 'Dongxie', '颜值': 7, '身高': '170cm'},  
}
```

先读取文件到内存里面
然后排序+二分查询？

有什么问题？

直接在硬盘中对数据进行排序 + 硬盘中二分？

那么怎么在硬盘里面进行排序和二分呢？

查询解决了，
有一天令狐冲整容了怎么办？

查询解决了，
有一天令狐冲整容了怎么办？
修改令狐冲颜值，从5变到6

查询解决了， 有一天令狐冲整容了怎么办？

修改令狐冲颜值，从5变到6

1. 直接在文件里面修改
2. 读取整个文件，修改好了，把原文件删除，重新写入新文件
3. 不修改，直接append操作追加一条记录“令狐冲颜值=6”在文件最后面

1. 直接在文件里面修改
很难做到直接修改内容，如果原来是4个字节，现在修改成8个字节，那么之后的内容都需要移动位置。
2. 读取整个文件，修改好了，把原文件删除，重新写入新文件
非常耗费时间，每次要读出写入 其他多余不变的内容
3. 不修改，直接append操作追加一条记录“令狐冲颜值=6”在文件最后面
好处： 特别快

BigTable为了写优化 选择了直接Append

坏处： 读取数据怎么办： 1.怎么识别哪个是最新的记录 2.没有顺序怎么二分？

怎么识别哪个是最新的记录？

时间戳最大的那个就是真正的数据

没有顺序怎么二分？

分块有序

1. 每一块都是内部有序
2. 写的时候只有最后一块是无序的，并且隔一段时间整理成有序



块会越写越多，会有很多重复
(令狐冲经常做整形手术)

块会越写越多，会有很多重复

(令狐冲经常做整形手术)

重复非常多

每次查询所有的块非常消耗时间

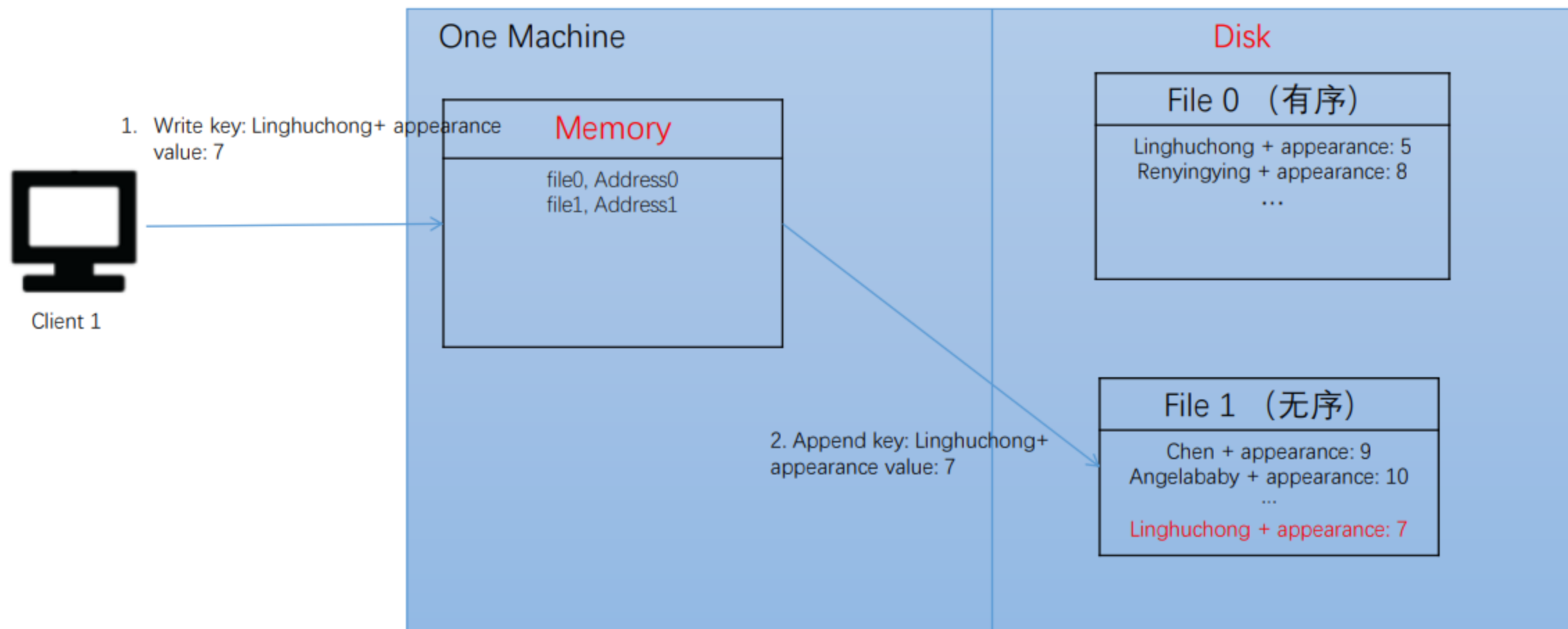
定期K路归并

<http://www.lintcode.com/en/problem/merge-k-sorted-arrays/>

完整系统读/写过程

One Work Solution

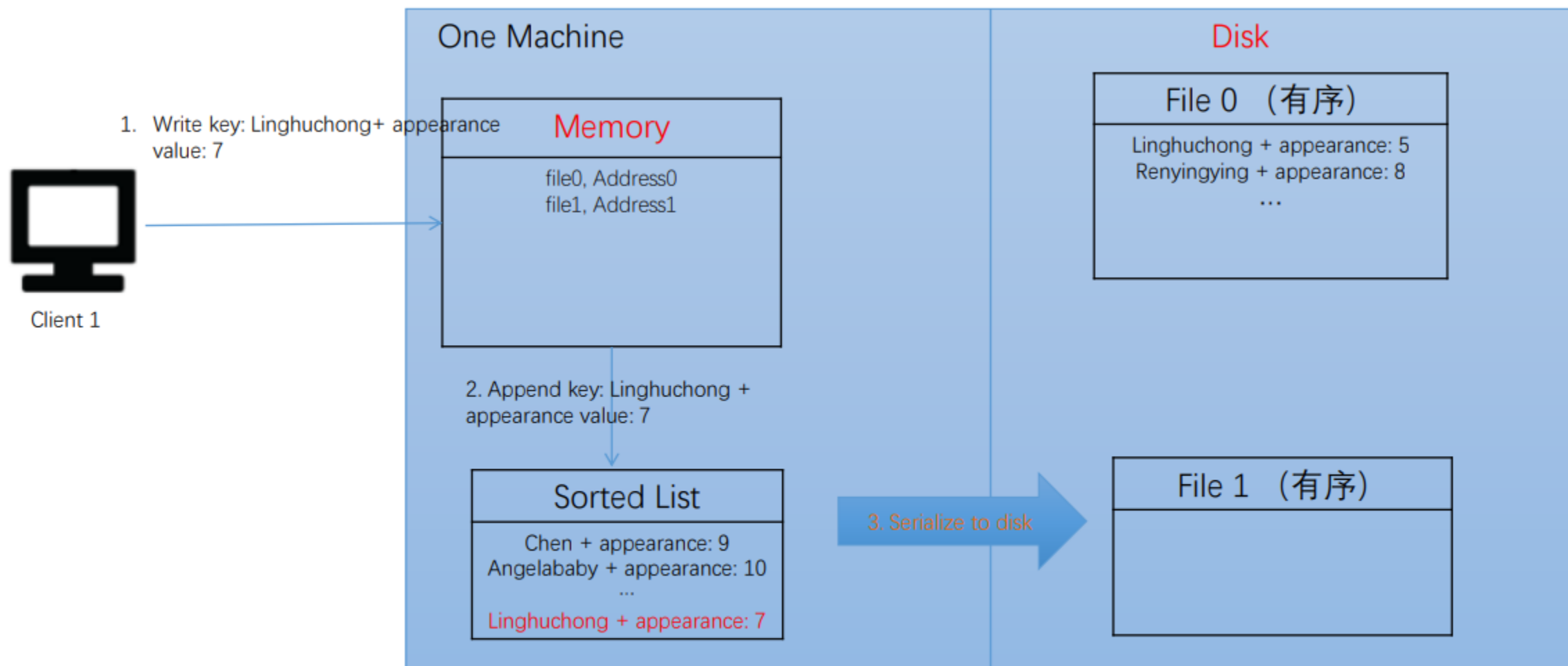
写入过程



怎么把最后一个File 从无序变成有序？

1. 读入到内存快速排序
2. 硬盘外部排序
3. 可不可以一开始就存在内存里面？

1. 读入到内存快速排序。
所有数据1次硬盘写入，1次硬盘统一读取+内存排序+1次硬盘统一写入
2. 硬盘外部排序
有必要么？
3. 可不可以一开始就存在内存里面？
内存排序+1次硬盘写入



Serialization

<http://www.lintcode.com/en/problem/binary-tree-serialization/>

Interviewer: 机器挂了，内存没了？

Write Ahead Log (WAL)

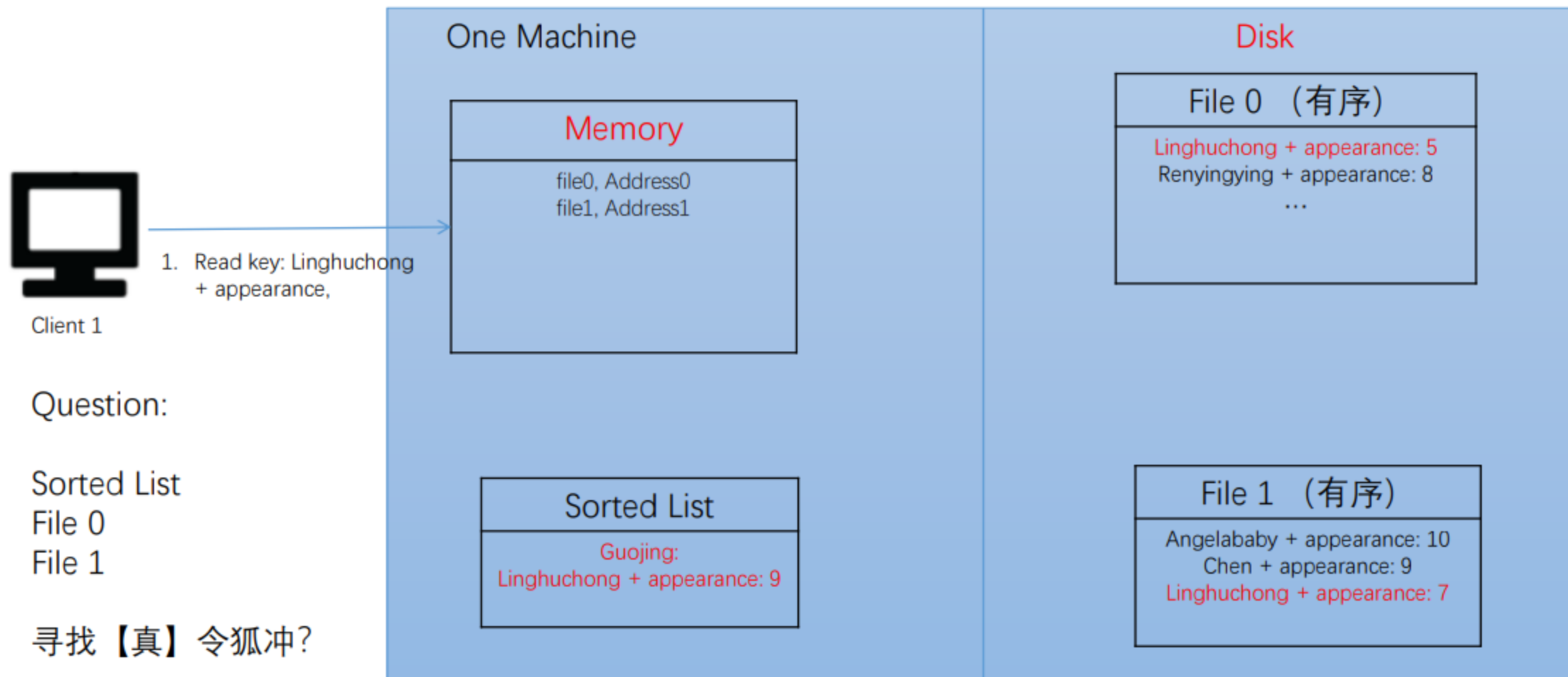
那写log岂不是又要写硬盘

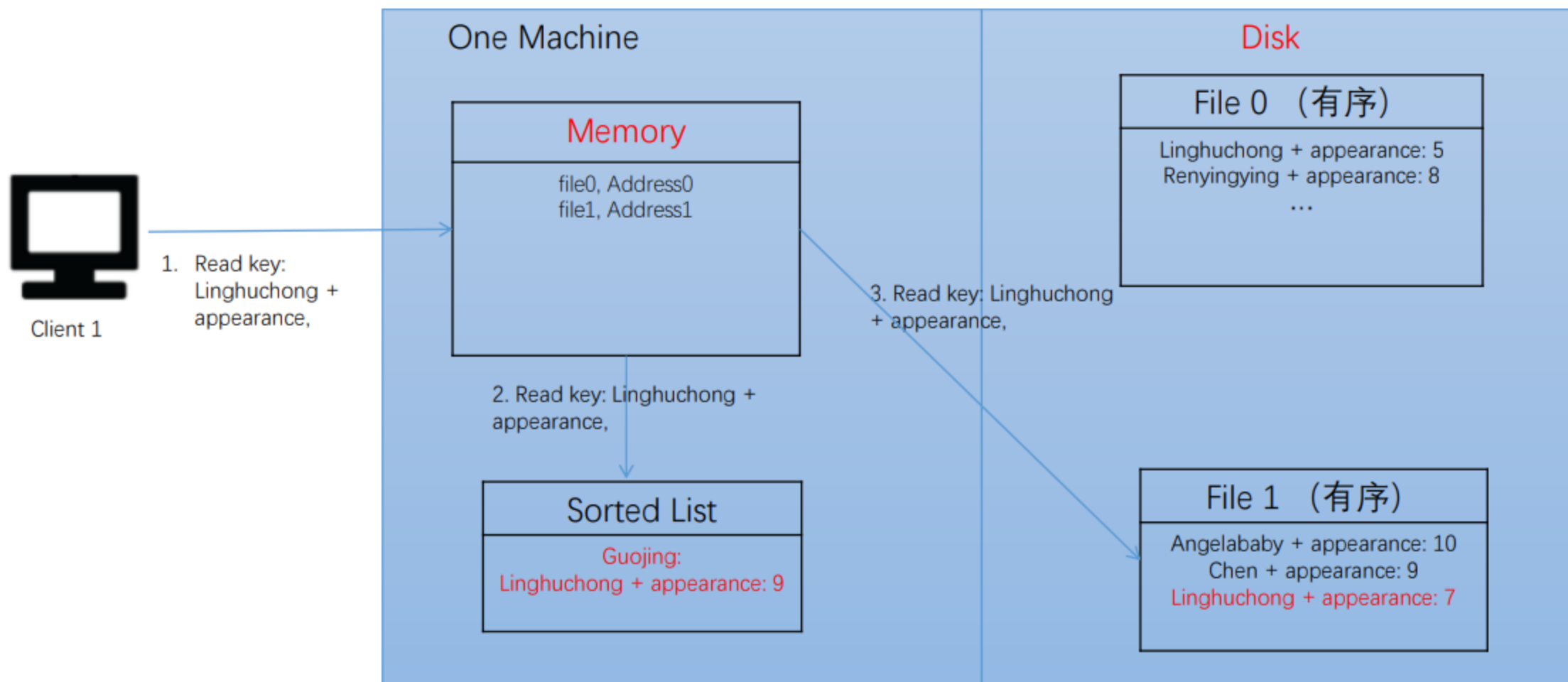
WAL 非常方便,不像 重要数据需要整理

内存排序+1次硬盘统一写入+1次硬盘写Log

Link: <http://www.larsgeorge.com/2010/01/hbase-architecture-101-write-ahead-log.html>

读出过程



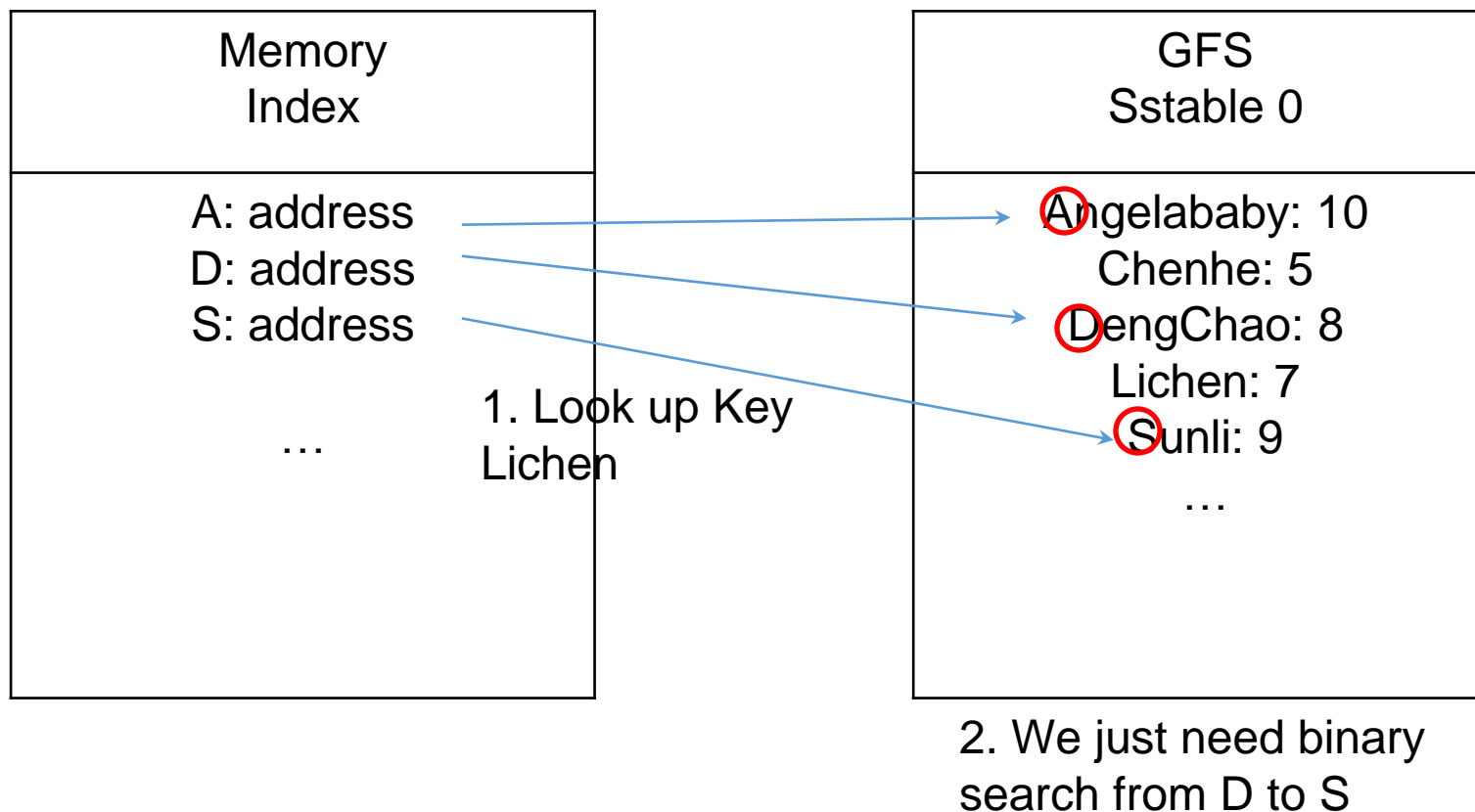


一个File里面怎么查询令狐冲?

1. 硬盘二分
2. 有没有更好的方法?

建立Index

目的： 加快查询
怎么建？



Key

- 把一些Key放入内存作为Index
- Index有效减少磁盘读写次数

Intersection of Two Arrays ii Follow Up

<http://www.lintcode.com/en/problem/intersection-of-two-arrays-ii/>

这道题的challenge题目

Read More:

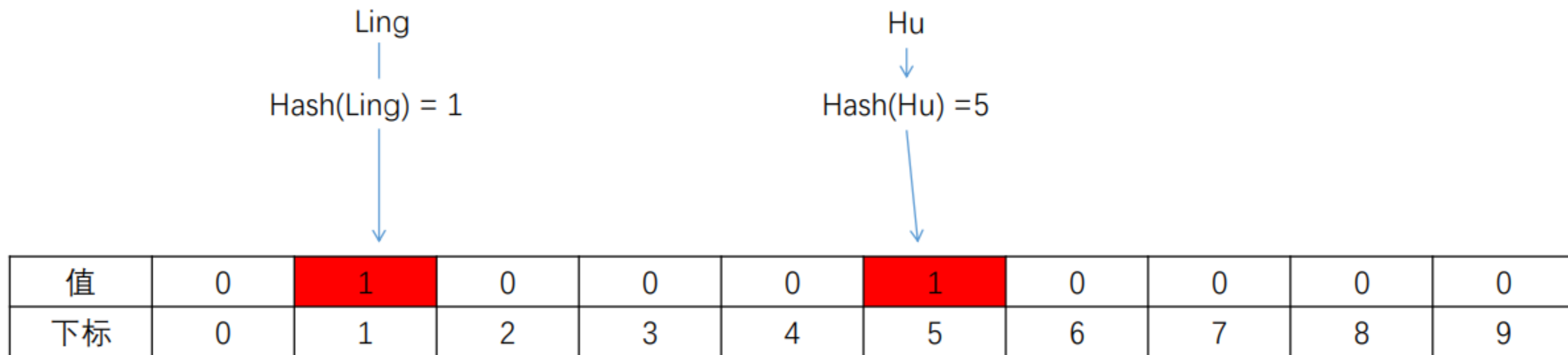
B tree index: <http://bit.ly/2bTwhZC>

有木有更好的方法检查一个key在不在一个File里面？

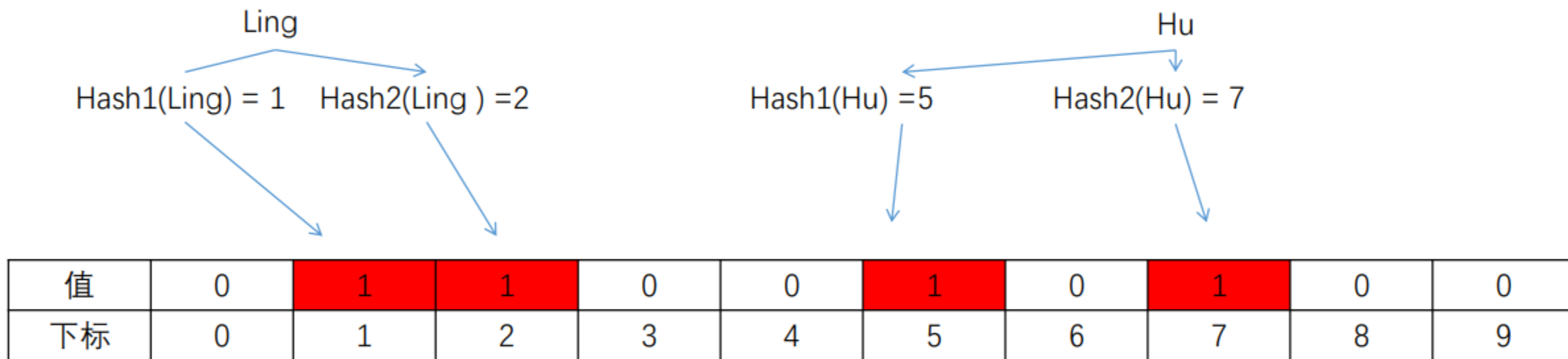
为什么要做如此多的读优化？

因为在写的时候做了Append优化，才会想办法加快读的速度。

BloomFilter

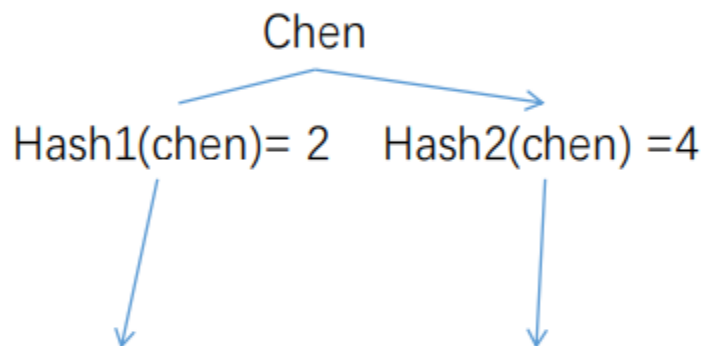


块中的Key: Ling Hu



块中的Key: Ling Hu

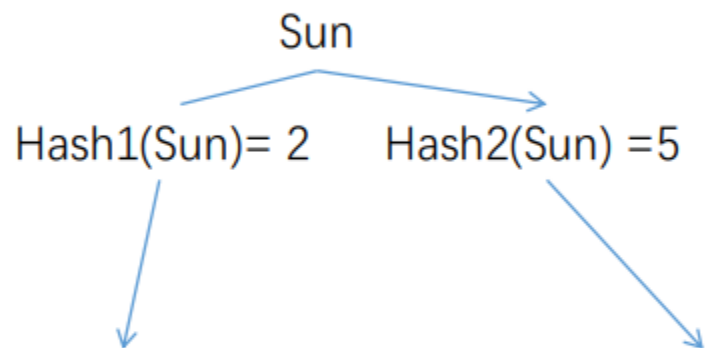
- 如何检查“chen” 在bloom filter 里面？



值	0	1	1	0	0	1	0	1	0	0
下标	0	1	2	3	4	5	6	7	8	9

块中的Key: Ling Hu

- 如何检查“sun” 在bloom filter 里面?



值	0	1	1	0	0	1	0	1	0	0
下标	0	1	2	3	4	5	6	7	8	9

sun

Bloom Filter 误判率

False is always False.

True may be True.

How many false is hidden in true?

Bloom Filter精确度跟什么有关？

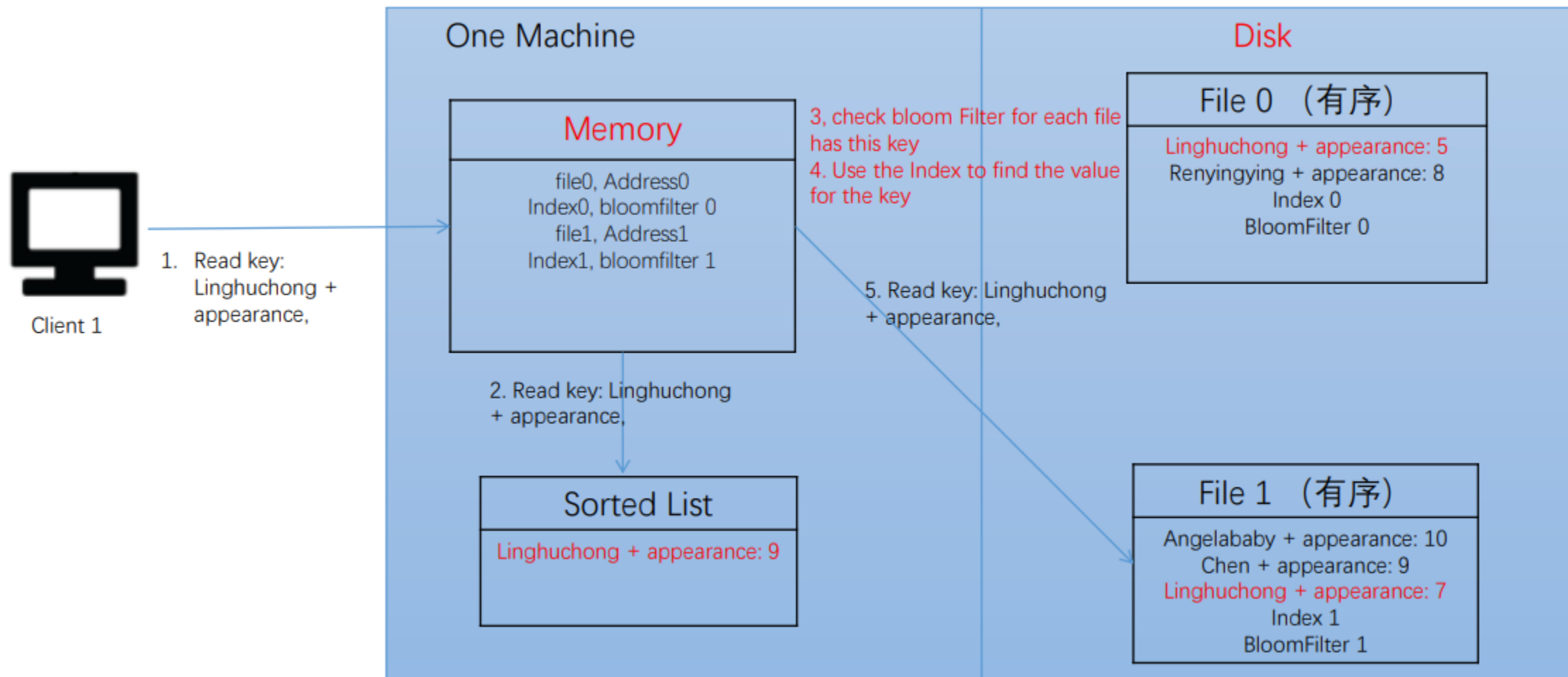
1. 哈希函数个数
2. 位数组长度
3. 加入的字符串数目

Bloom Filter 误判率

举个例子如果
哈希函数的个数15个、
位数组大小200w、
加入的字符串数量10w个的话、
判断2000w个新的字符串
误判率在 3~4% 左右

计算误判率公式: https://en.wikipedia.org/wiki/Bloom_filter

完整的读出过程(with Index, BF)



Specific Name in BigTable

String is Store in the File.

SSTable = Sorted String Table

Sorted List 用 Skip List 实现

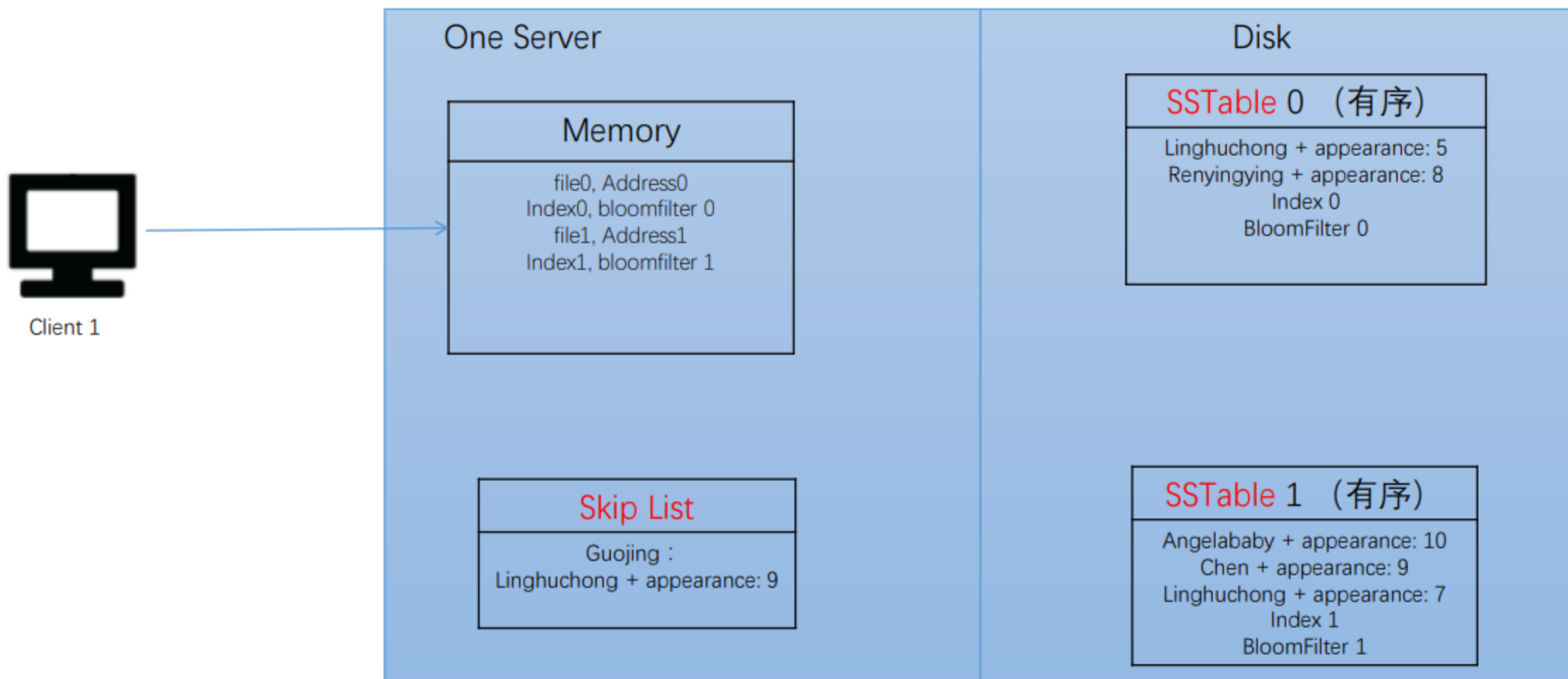
1. Skip List

Code: https://github.com/petegoodliffe/skip_list

Wiki: <http://bit.ly/2g0C29a>

2. SSTable

Google SSTable Page: <http://bit.ly/1kqwrFe>



我们已经学会了 一台机器BigTable的操作

读/写

Key: 令狐冲+颜值

Value: 5

姓名	颜值	身高
令狐冲	5	160
郭靖	9	180
东邪	7	170

Interviewer: How to read/write
key: value from 1PB file

Scale

Sharding?

姓名	颜值	身高
令狐冲	5	160
郭靖	9	180
东邪	7	170

Sharding

Vertical Sharding?
Horizontal Sharding?

难点：取令狐冲“颜值”是不是会取“身高”，“武功”相关属性

Diagram illustrating a BigTable structure with Row and Column labels.

Row key	颜值	身高
令狐冲	5	160
郭靖	9	180
东邪	7	170

	颜值	身高
令狐冲	5	160
郭靖	9	180
东邪	7	170

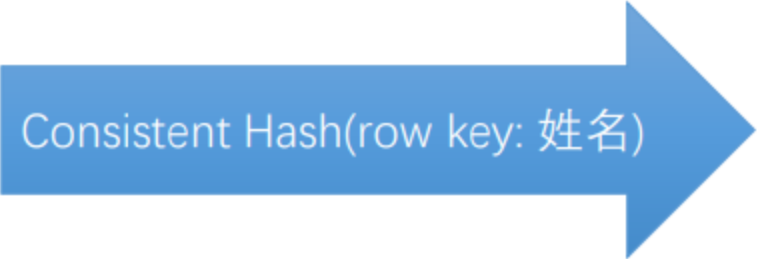


表1	颜值	身高
令狐冲	5	160

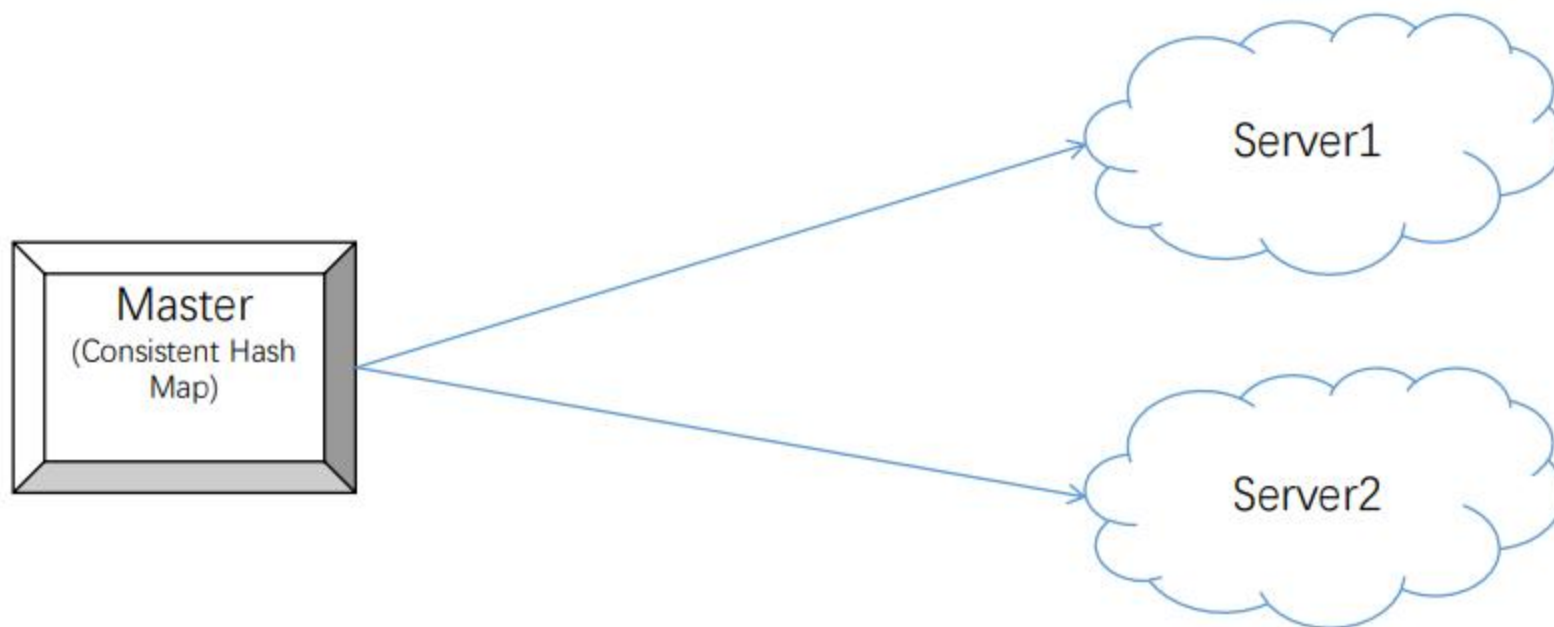
表2	颜值	身高
郭靖	9	180
东邪	7	170

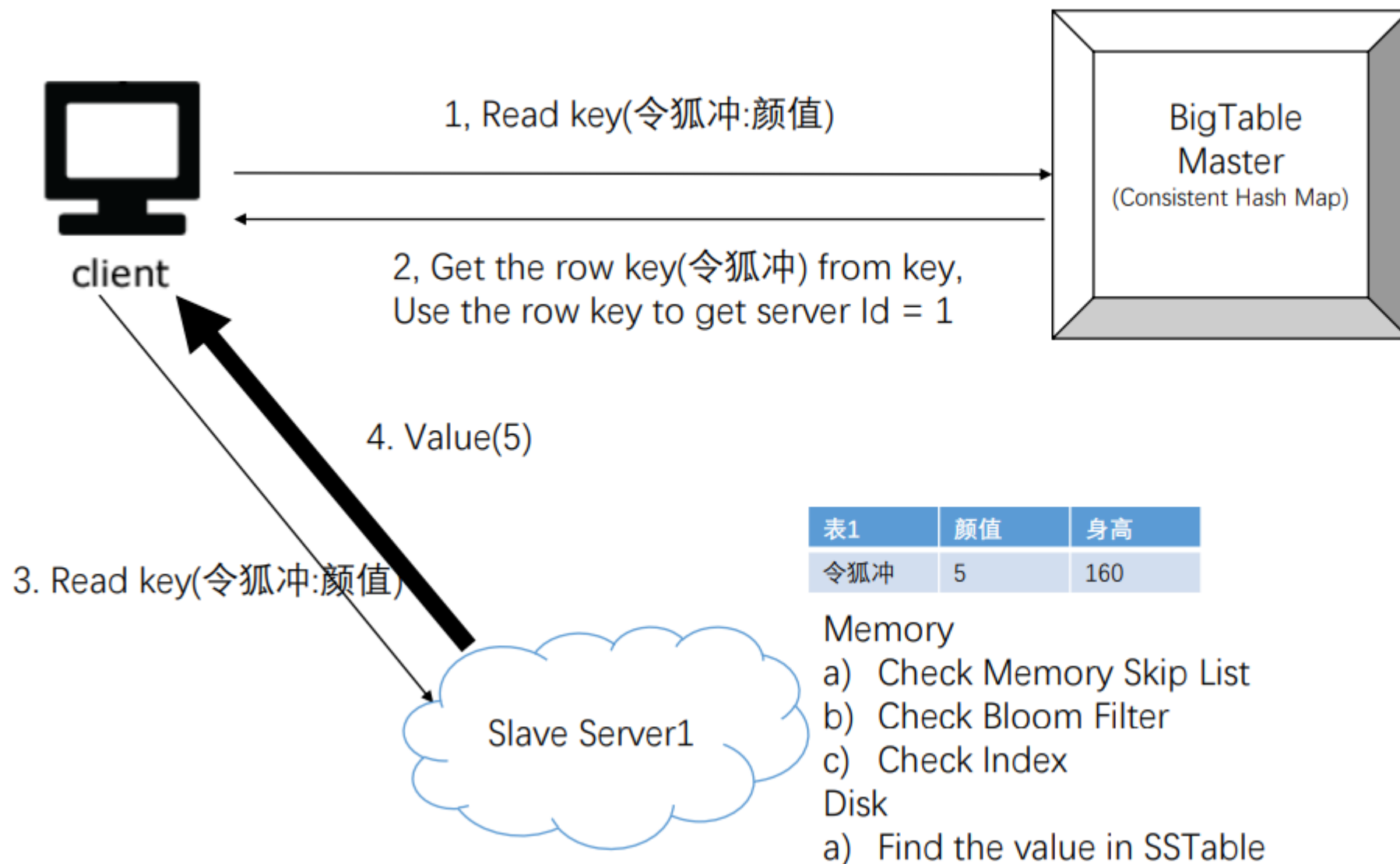
一台机器搞不定，那么需要多台
机器了

Master + Slave

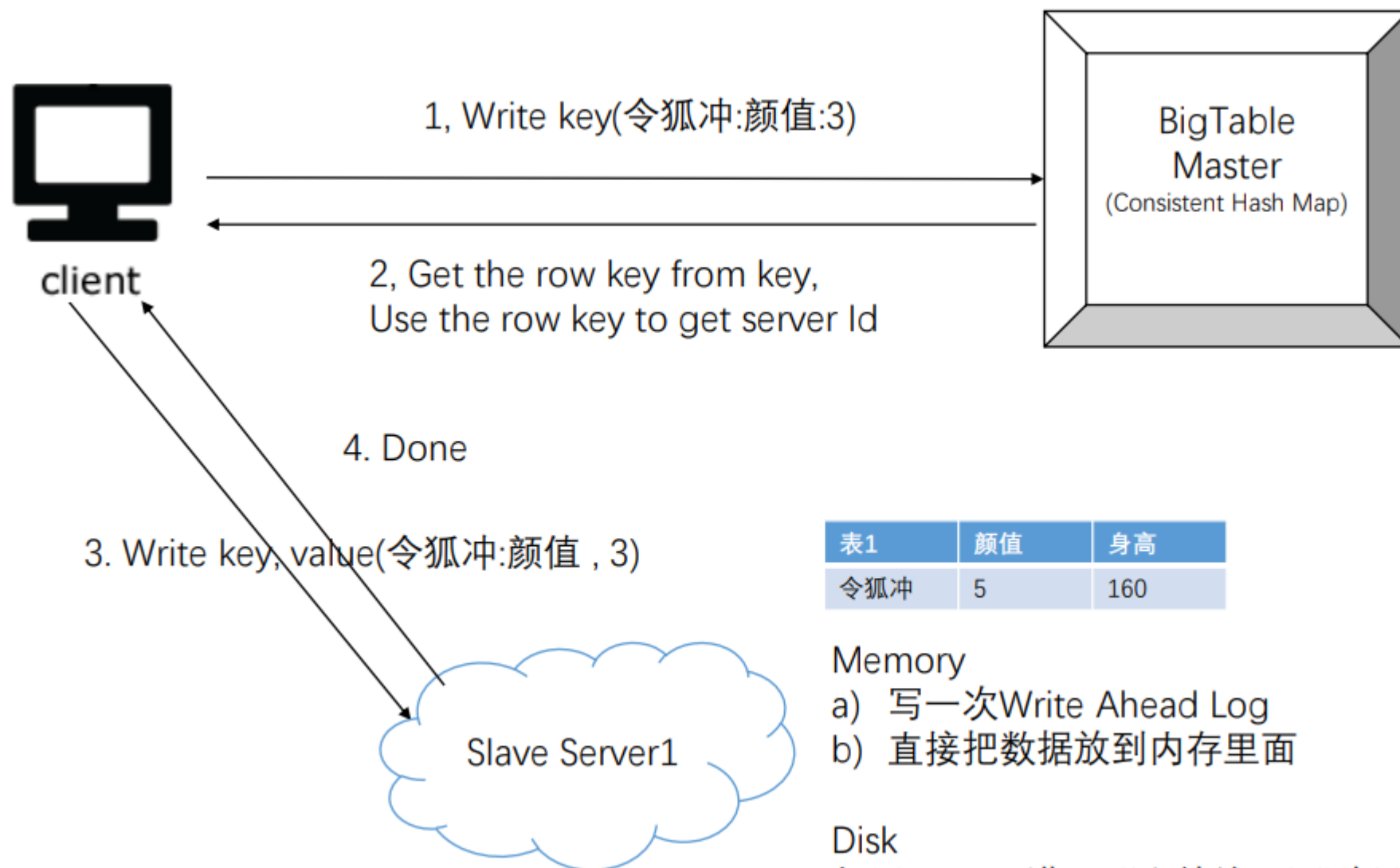
Interviewer: How to manager server?

- Key
 - Master + Slave
 - Master has HashMap [key, server address]





Interviewer: How do we write
BigTable?



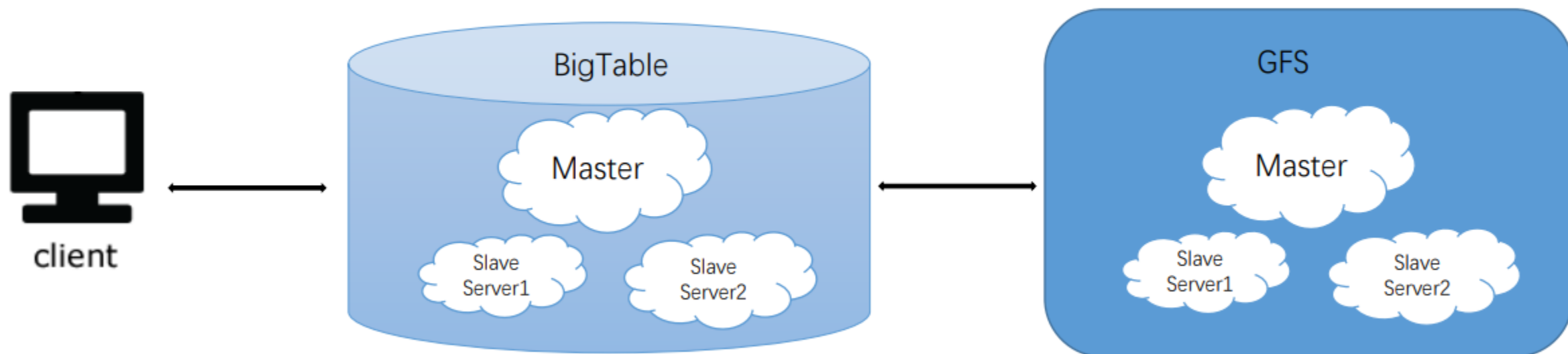
Interviewer: 每台机器数据越写越多存不下怎么办?

现在所有的数据都存在Slave Server local disk里面

把所有数据存到GFS里面

Advantage:

1. Disk Size
2. Replica
3. Failure and Recovery

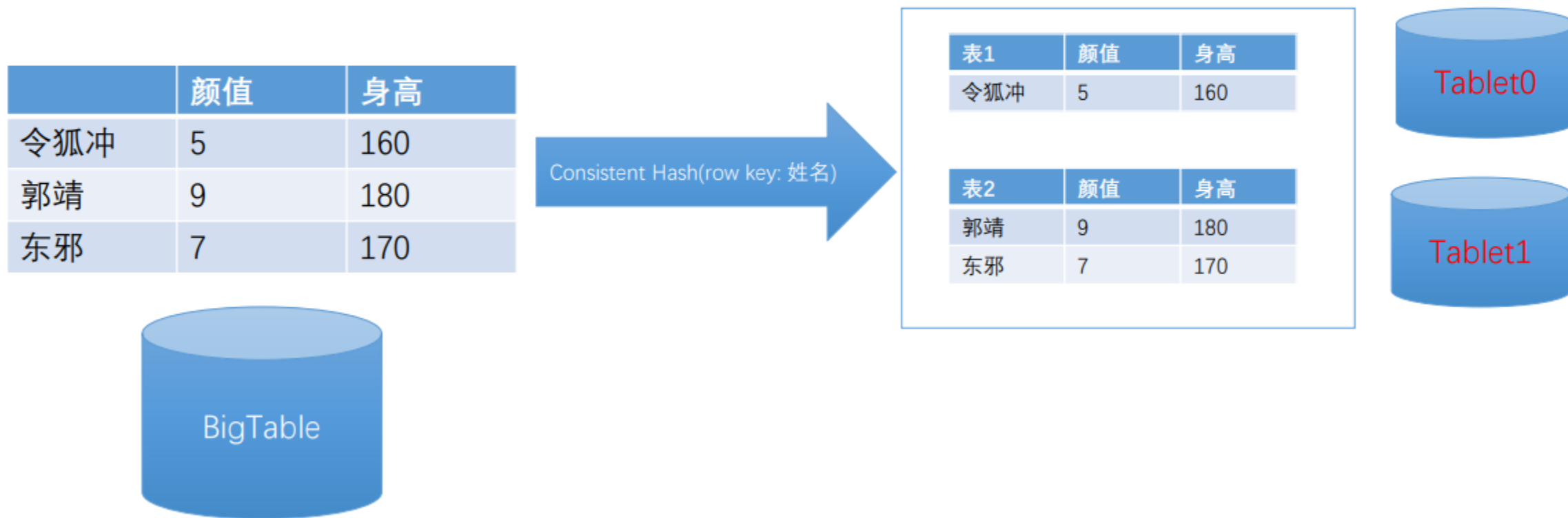


BigTable vs GFS

都是Master + Slave

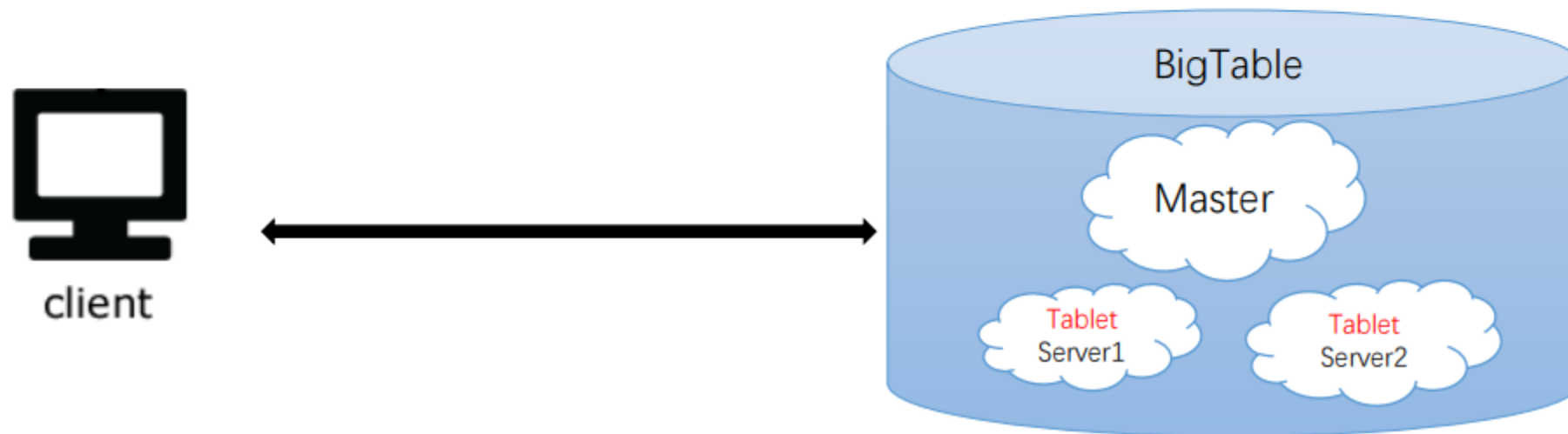
是否就用一个Master + Slave把两个都实现了？

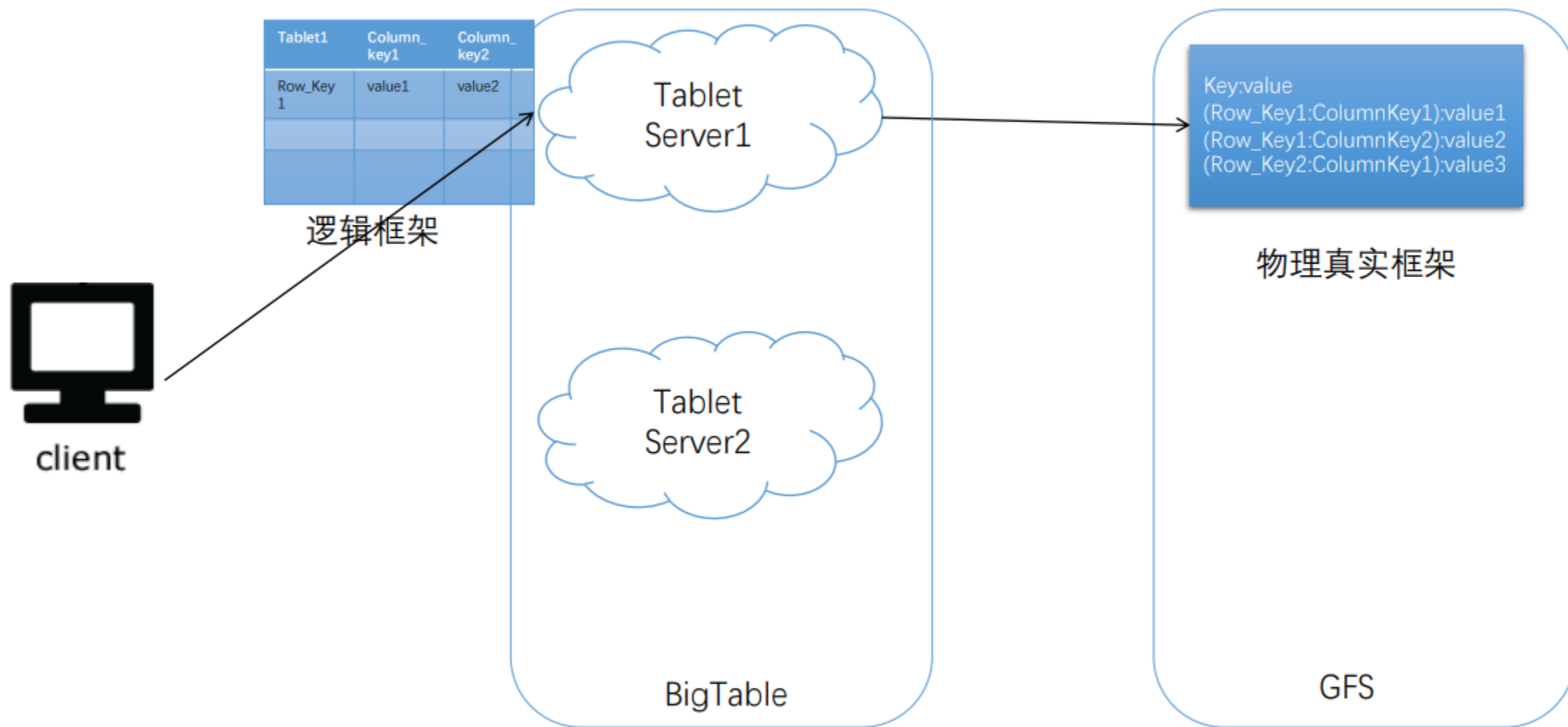
BigTable Naming



What is Tablet Server

Tablet Server = Store Tablet Slave Server





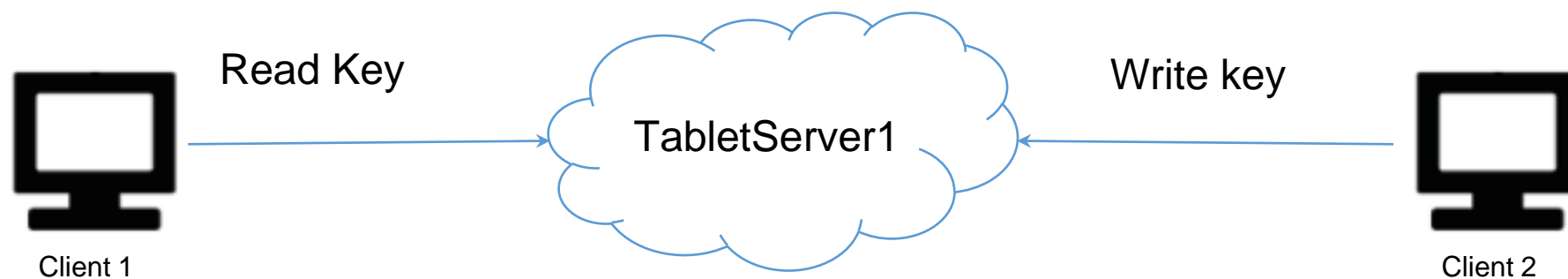
看看还有什么问题没有解决

读和写同时发生？

写的过程当中，有读请求？

Race Condition

Interviewer: What if we read and write the same key at same time?



Race Condition

- <http://bit.ly/25FBHM4>



We need a lock

We need a distributed lock.

由多台机器组成的分布式锁服务

- Chubby
- Zookeeper
- Read More:
 - <http://bit.ly/1Pukiyt>
 - <http://bit.ly/1TOWIsR>

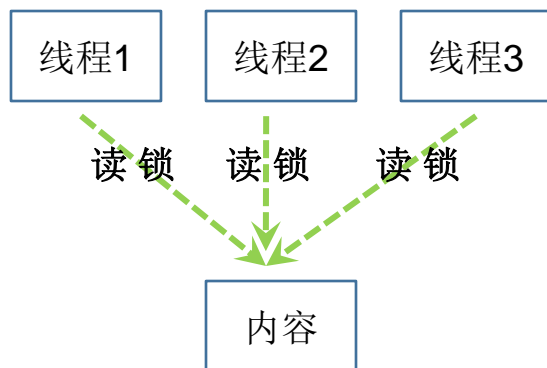


读写锁：读属性、写属性（一把锁）

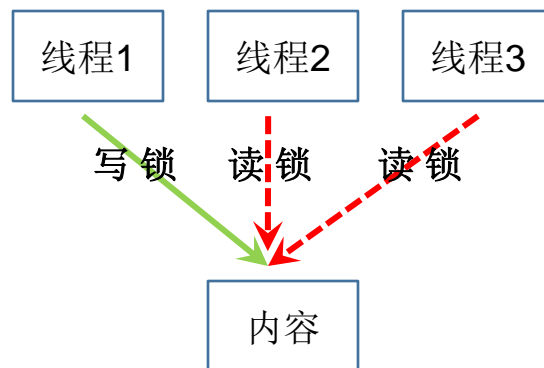
读锁：对内存做读操作的时候上读锁。

写锁：对内存做写操作的时候上写锁。

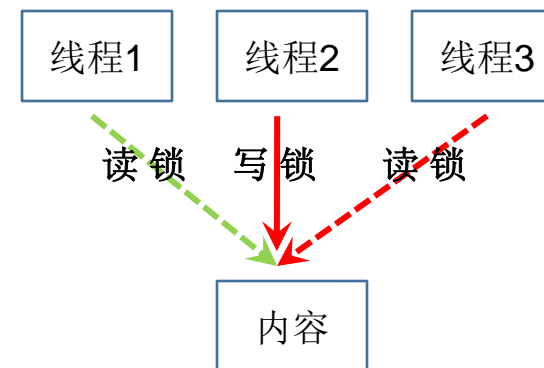
线程1加读锁成功，又来两个线程，
做读操作，可以加锁成功



线程1加写锁成功，又来两个线程，
做读操作，两个线程阻塞



线程1加写锁成功，又来了2线程加写锁
阻塞，又来了3线程加读锁阻塞



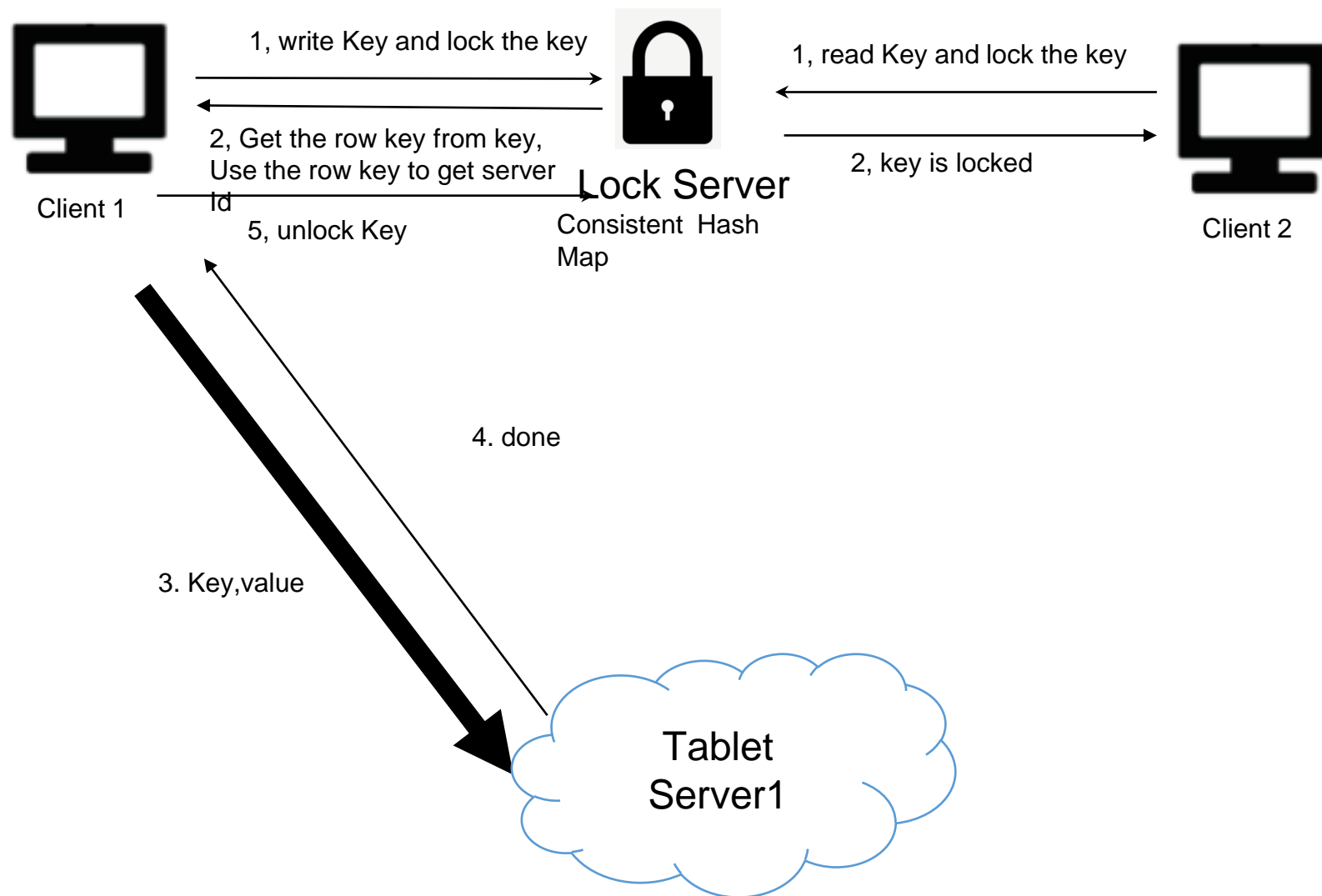
——读锁（共享锁）——▶

——写锁（排他锁）——▶

——绿色，非堵塞——▶

——红色，堵塞——▶

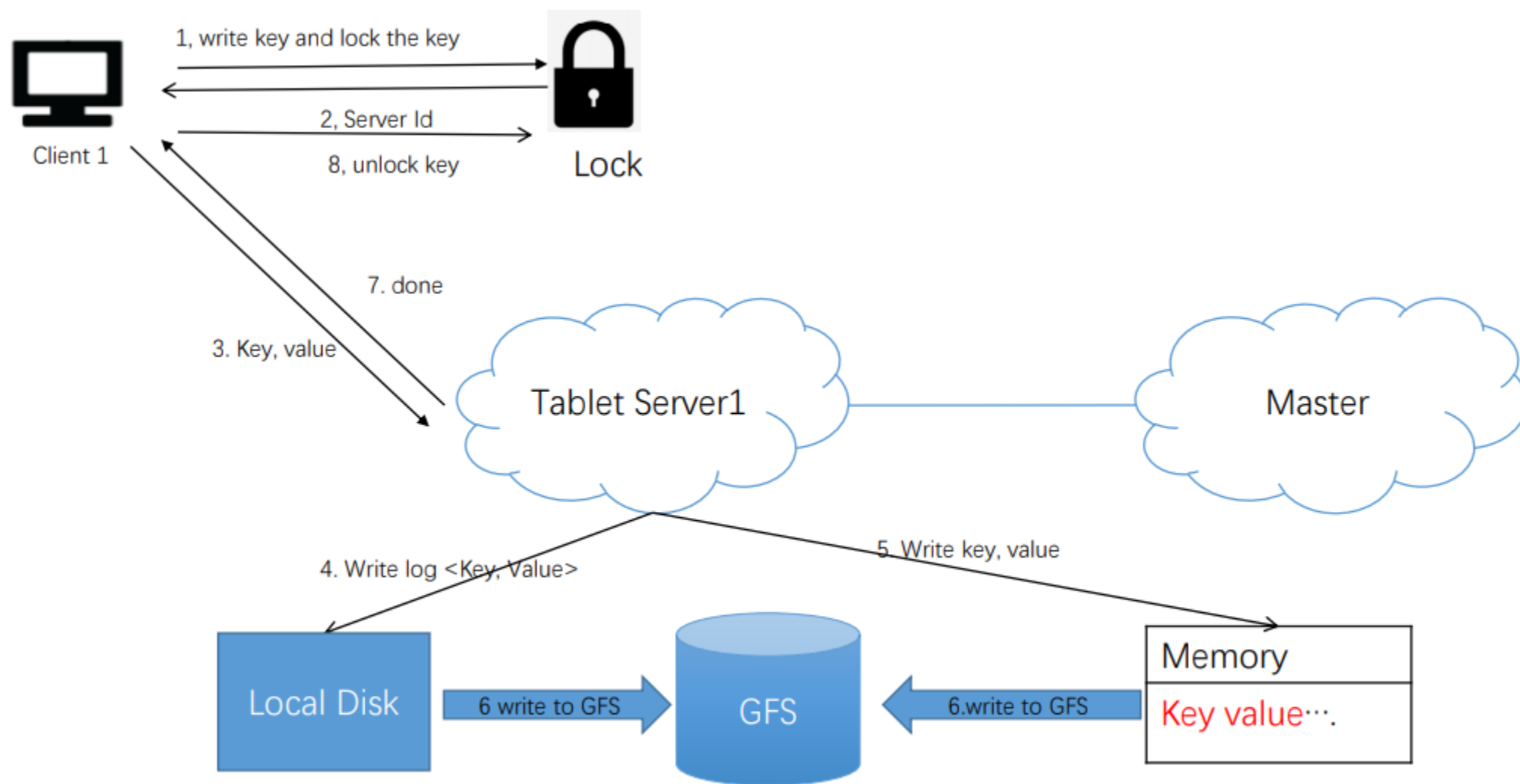
Interview: How to write a key?



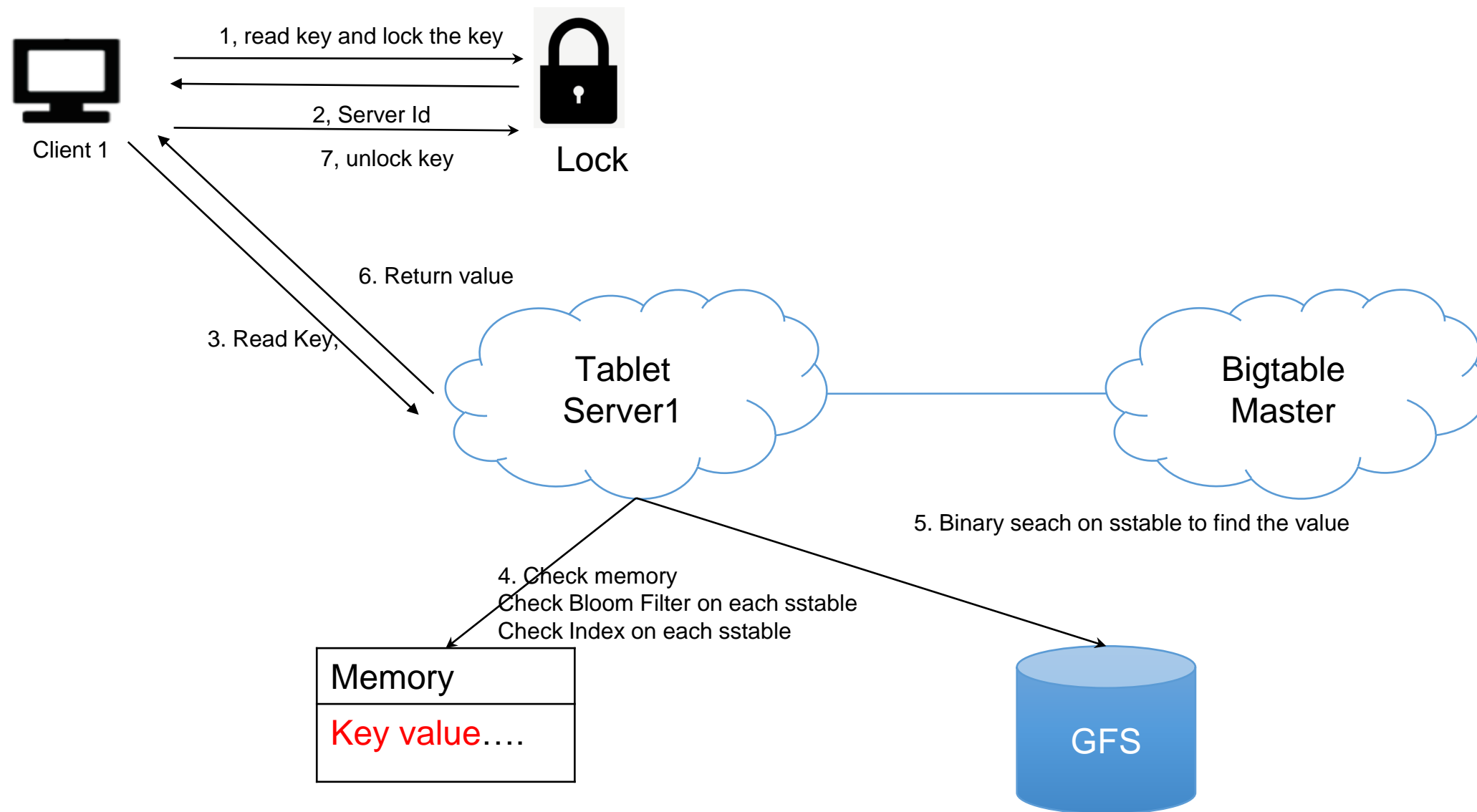
Advantage Distributed Lock

- The Metadata is store on the Lock
- Lock 本来要存储Metadata那master就不需要存储MetaData了

Summary of Write



Summary of Read



Summary of Read

- Design
 - Client + Master + Tablet Server + Distributed Lock
- Client
 - Read + Write
- Tablet Server
 - Maintain the Data (Key value pairs)
- Master
 - Shard the file
 - Manage the servers health
- Distributed Lock
 - Update Metadata
 - Maintain the Metadata
 - Lock Key



BigTable

- <http://www.cse.buffalo.edu/~mpetropo/CSE736-SP10/slides/seminar100409b1.pdf>

LevelDB + LSM Tree

- <http://zouzls.github.io/2016/11/23/LevelDB%E4%B9%8BLSM-Tree/>
- <http://www.cnblogs.com/fxjwind/archive/2012/08/14/2638371.html>