

下列关于分布式文件系统的描述，哪些是正确的

A: 分布式文件系统一定是P2P架构的去中心化系统

B: 分布式文件系统能够解决一个文件太大在单台机器上存不下的问题

C: 分布式文件系统和普通的笔记本电脑的文件系统的原理是相通的

D: 分布式文件系统适合存储文件大小很小的文件

E: 分布式文件系统适合存储文件大小较大的文件

正确答案是 B C E，有24%的同学答对了，加油赶上他们!

GFS 采用的是 Master-slave 的架构。他的一些基本设计原理，如 chunk 的设计（对应文件系统里的 block）和普通文件系统是相通的。整套系统的设计是基于一些合理假设的（这些假设都是Google从大量工程实践中总结出来的），其中一条假设就是：“We expect a few million files, each typically 100MB or larger in size”。如果实在需要存很多小文件的话，其实可以将这些小文件打包成一个大文件然后再存储，只要记录下每个小文件在文件内部的起始字节和大小就行了。

我们在描述硬盘存储空间大小的时候，1P 是多大？

A: 1024 T

B: 1000 T

已回答

我不会

是有点难吗，答错了呢。正确答案是 B，有20%的同学超过了你，但是千万不要气馁。

这个题大部分同学都会答错，你答对没有呢？硬盘厂家的标准是十进制，而计算机系统中使用的是二进制。所以在硬盘的领域里，1P = 1000T, 1T = 1000G, 1G = 1000M。因此你买的 1T 的硬盘在计算机中查看大小时，实际只有 931G。祝黑心硬盘厂家们早日倒闭!

版权所有 © 九章算法 (杭州) 科技有限公司

# 系统设计 Distributed System Design (九章网站下载最新课件)

课程版本 v6.0

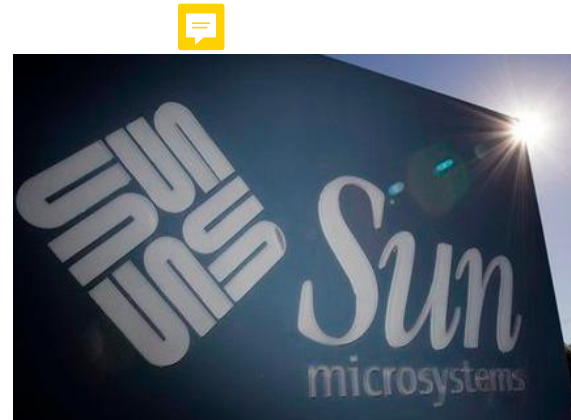
本节主讲人：北丐老师

：九章课程不允许录像，否则将追究法律责任，赔偿损失

# 什么是分布式系统？

一言以概之：用多台机器去解决一台机器上不能够解决的问题。

比如：存储不够？QPS太大？



- Distributed File System (Google File System)
  - 怎么有效存储数据？
  - No SQL 底层需要一个文件系统
- Map Reduce
  - 怎么快速处理数据？
- Bigtable = No-SQL DataBase
  - 怎么连接底层存储和上层数据

# Design Distributed File System

## 了解分布式文件系统后可以做什么？

1. Google, Microsoft面试可能会考到.
2. 学习经典系统, 对其他系统设计也有帮助.  
比如如何处理failure和recovery.

Distributed File System	Company	开源
GFS	Google 	No
HDFS	Yahoo(Altaba)Open Source of GFS 	Yes

# Distributed File System

Hadoop Distributed File System  
VS

Google File System(GFS)

## 1. 按照4S分析

- **S**enario 场景分析
- **S**ervice 服务
- **S**torage 存储
- **S**cale 升级优化

## 2. 理清楚work solution

## 3. Scale升级优化

# Scenario 场景分析

需要设计哪些功能

- 需求1

- 用户写入一个文件， 用户读取一个文件.
- 支持多大的文件？
  - 越大越好？ 比如 >1000T

- 需求2

- 多台机器存储这些文件
- 支持多少台机器？
  - 越多越好？



# Service 服务

# Service 服务

Client

+

Server

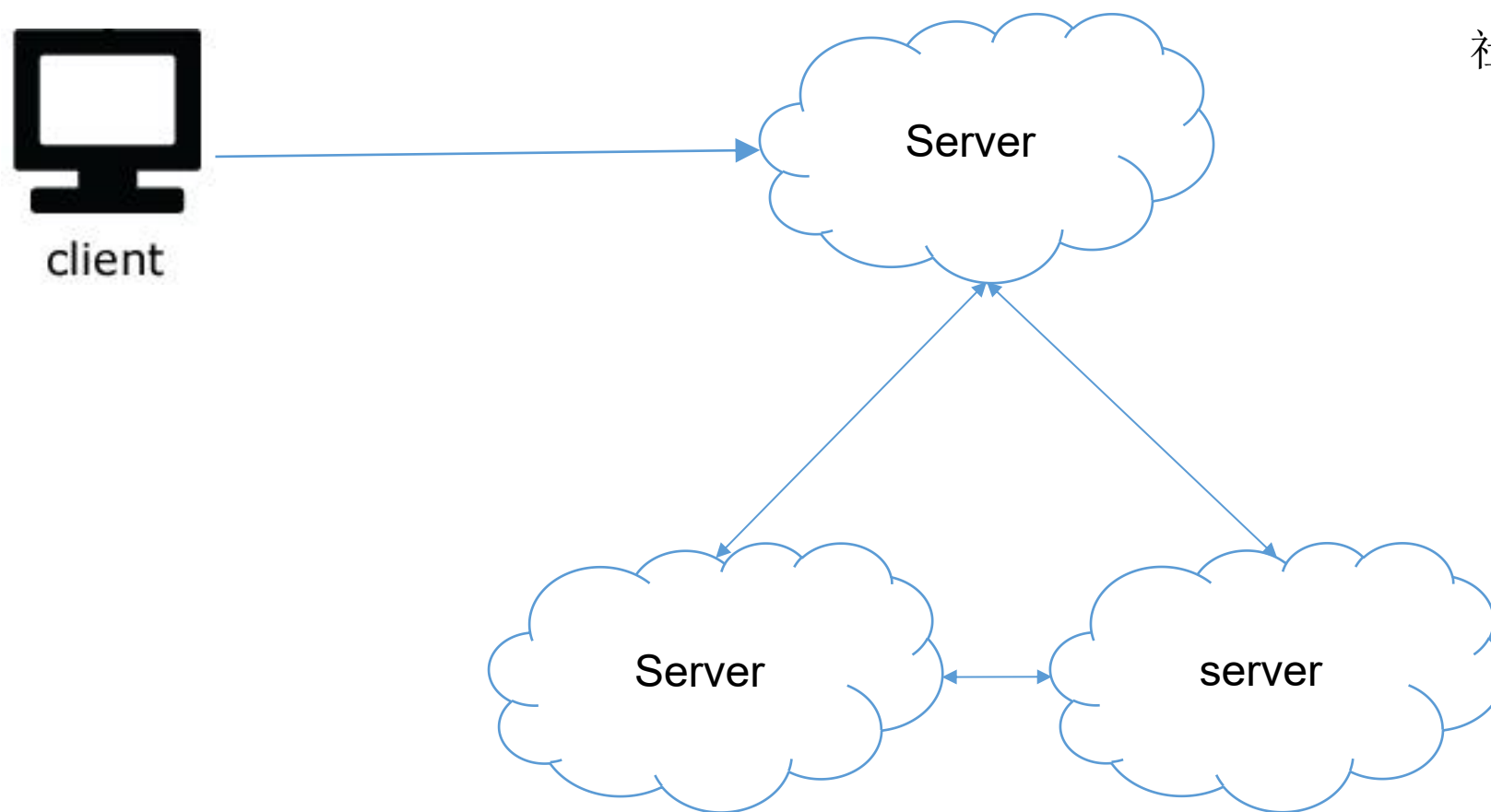


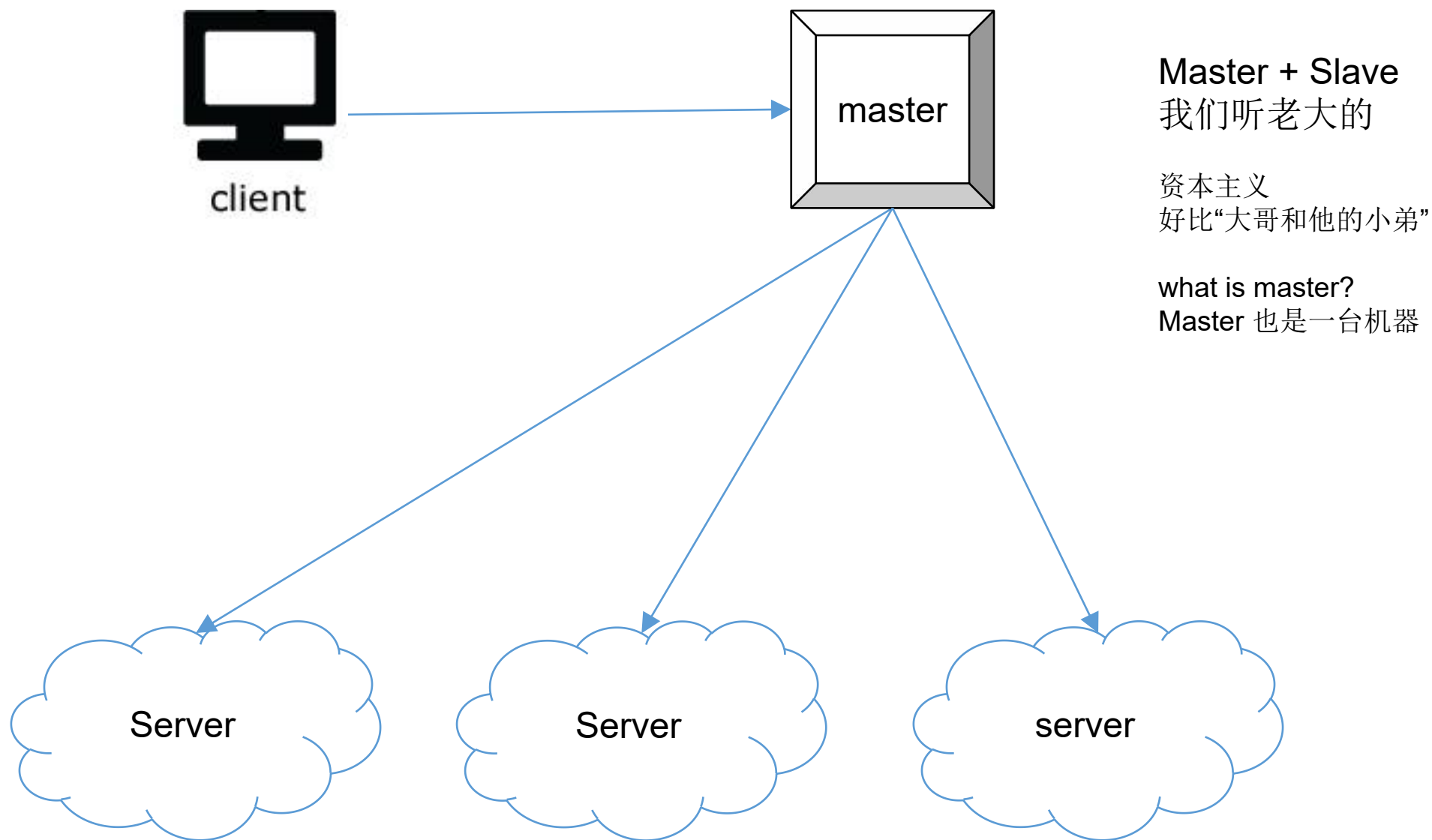
# 多台机器怎么沟通？

## 社会主义 or 资本主义

Peer to peer  
谁也看不惯谁

社会主义






- Peer 2 Peer
  - Advantage
    - 一台机器挂了还可以工作
  - Disadvantage
    - 多台机器需要经常通信保持他们数据一致
- Master Slave
  - Advantage
    - Simple Design
    - 数据很容易保持一致
  - Disadvantage
    - 单master要挂
- Final Decision 
  - Master + Slave
  - 单master挂了重启就是。挂的概率在0.1%

# Storage 存储

数据如何存储



- 大文件存在哪？
  - 内存？ 硬盘？

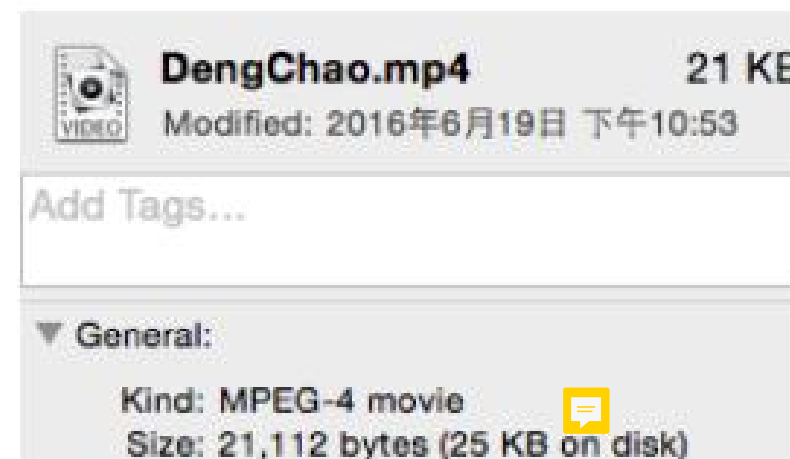
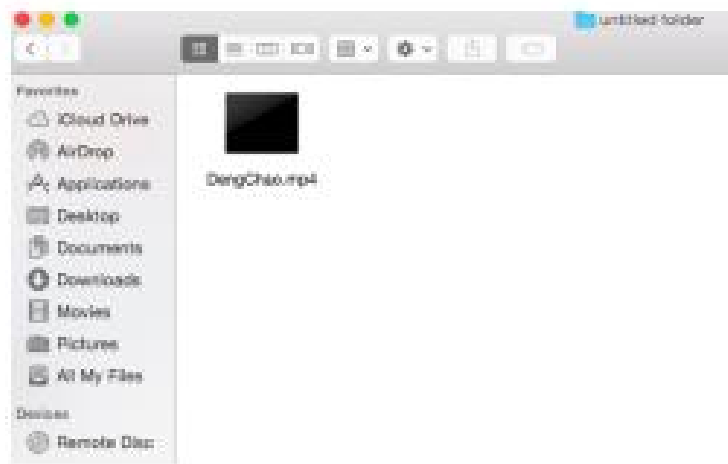
- 大文件存在哪？
  - 内存？硬盘？
- 怎么存在文件系统里面呢？
  - 怎么设计GFS？

# Interviewer: How to save a file in one machine?

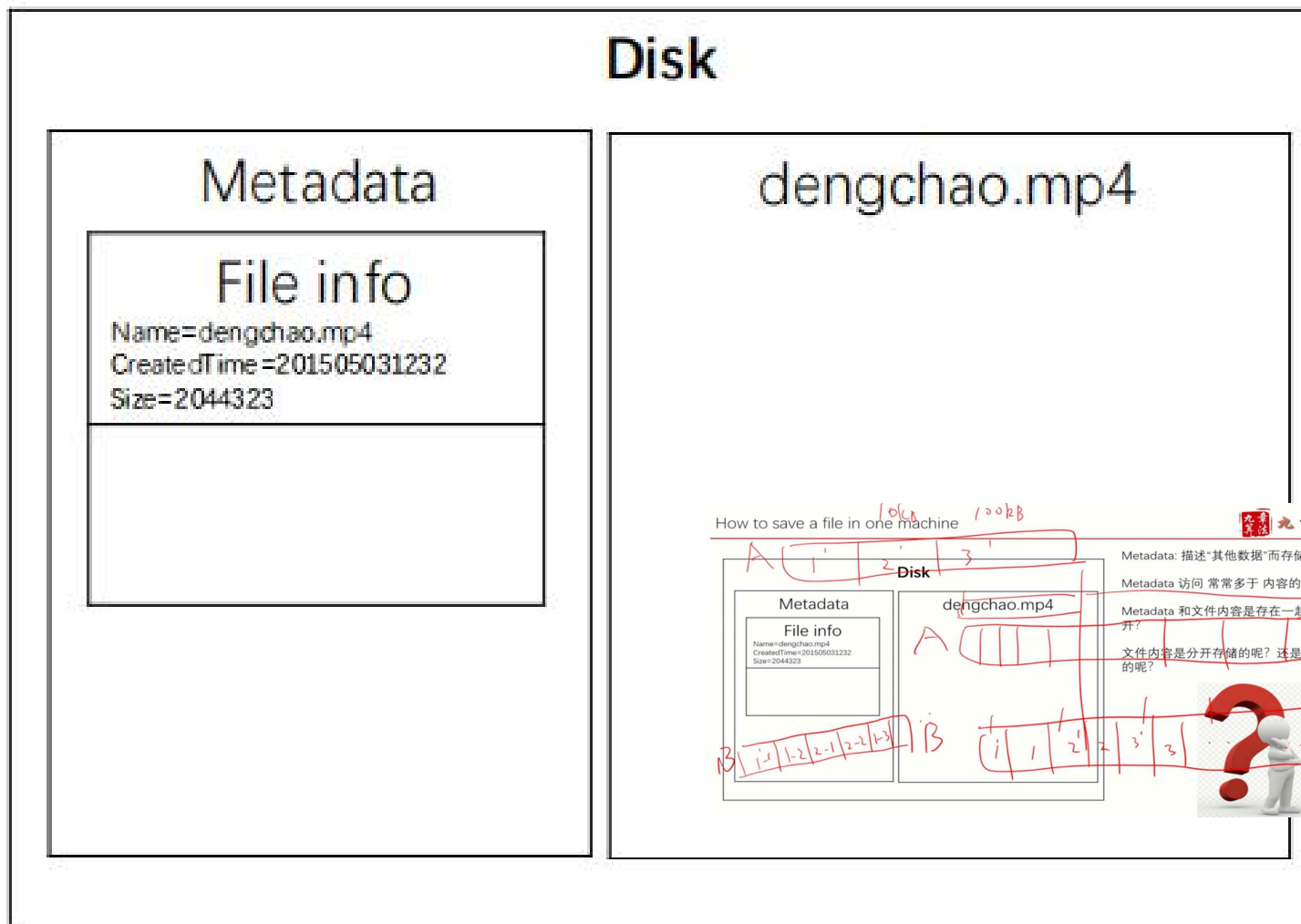
普通的操作系统是怎么做的呢？ 100G

# DengChao.mp4

## 一个文件有什么东西？




# How to save a file in one machine



Metadata: 描述“其他数据”而存储的信息

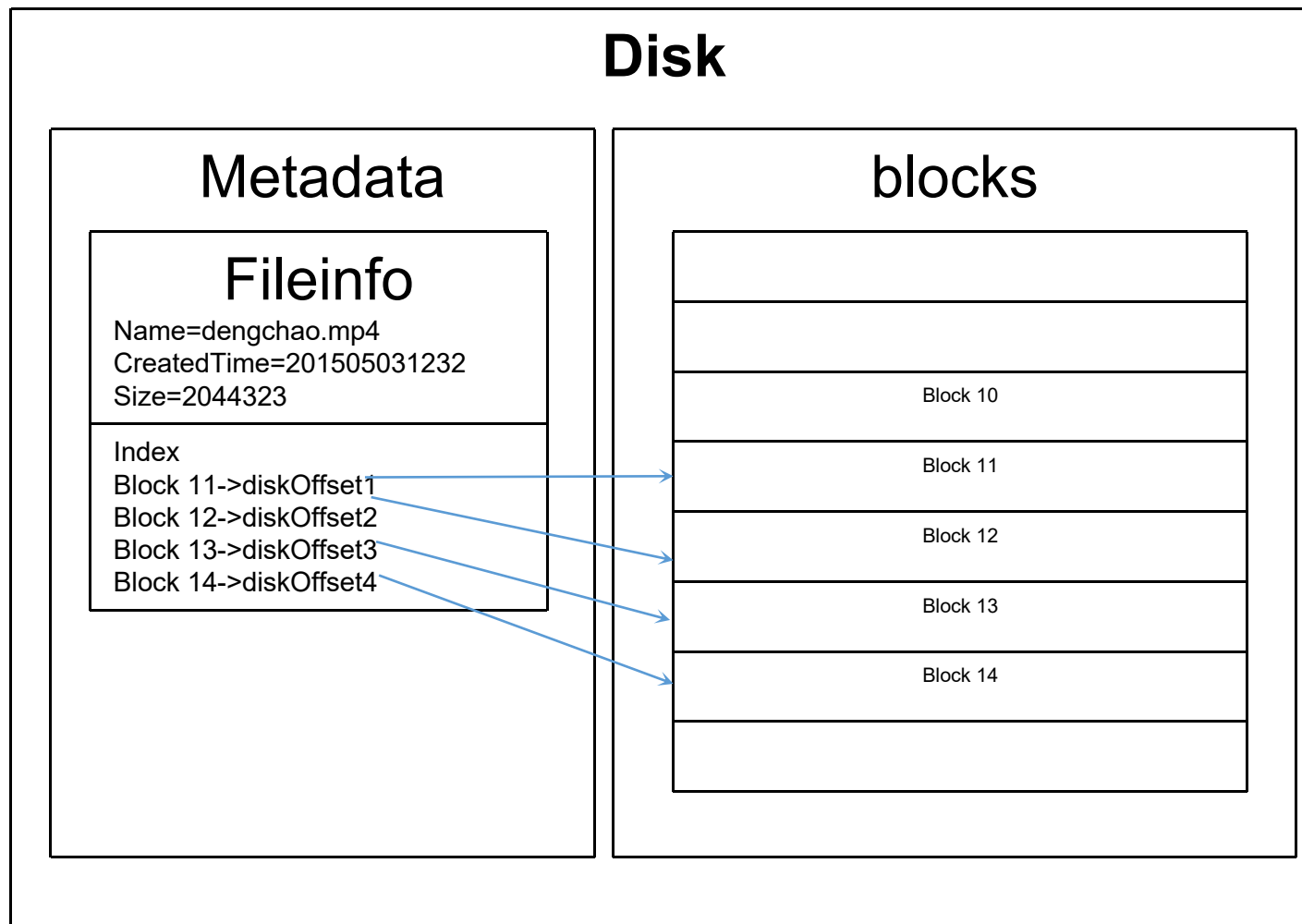
Metadata 访问 常常多于 内容的访问

Metadata 和文件内容是存在一起还是分开? 

文件内容是分开存储的呢? 还是连续存储的呢? 



# How to save a file in one machine



Key point

- 1 block = 4096Byte

# Interviewer: How to save a large file in one machine?

Is block size big enough?

100T(多文件)

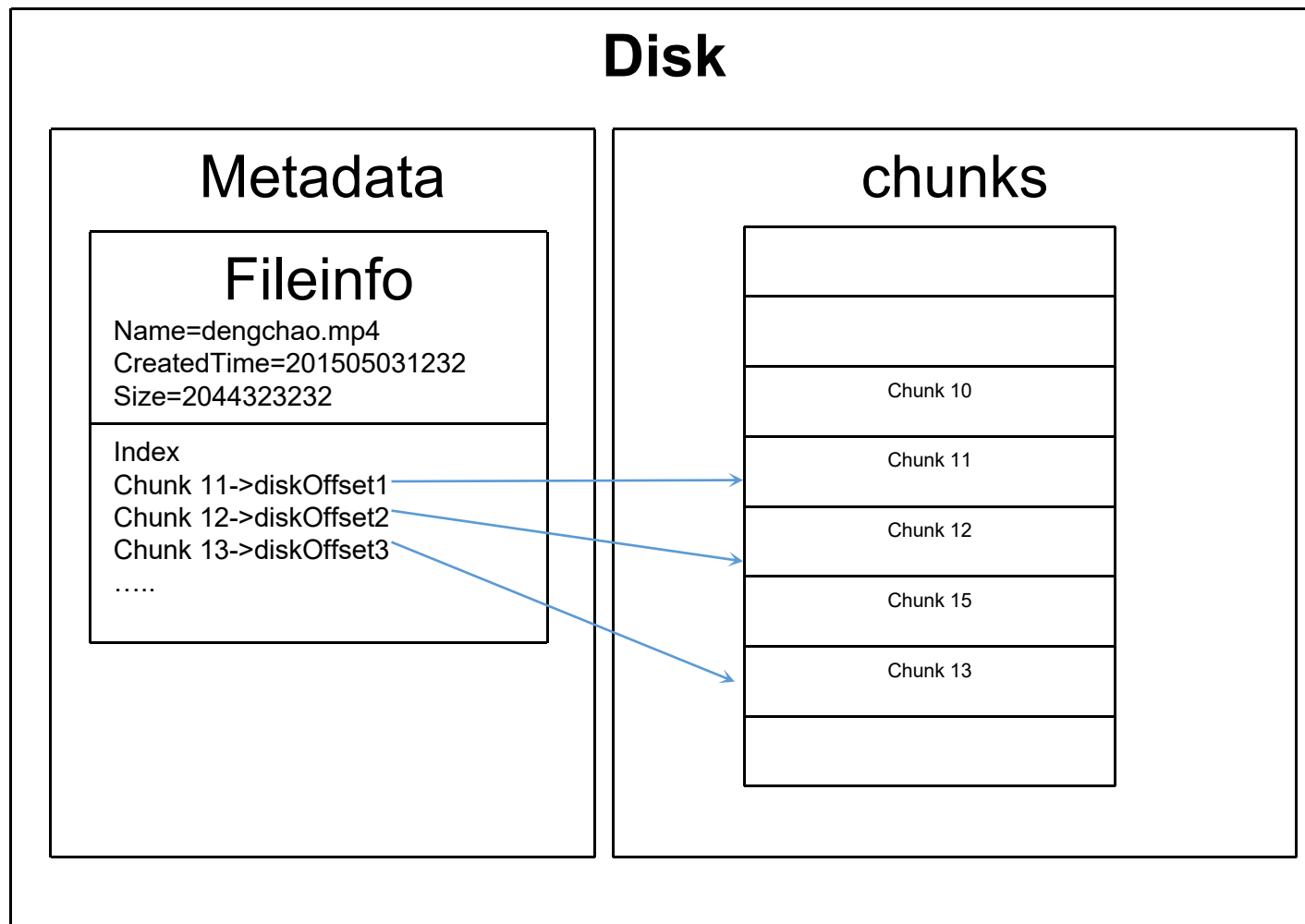
=100\*1000G

=100\*1000\*1000M

=100\*1000\*1000\*1000K

=100\*1000\*1000\*1000block

# Interviewer: How to save a large file in one machine?



Key point

- 1 chunk= 64M  
= 64\*1024K

Advantage

- Reduce size of metadata
- Reduce traffic

Disadvantage

- Waste space for small files

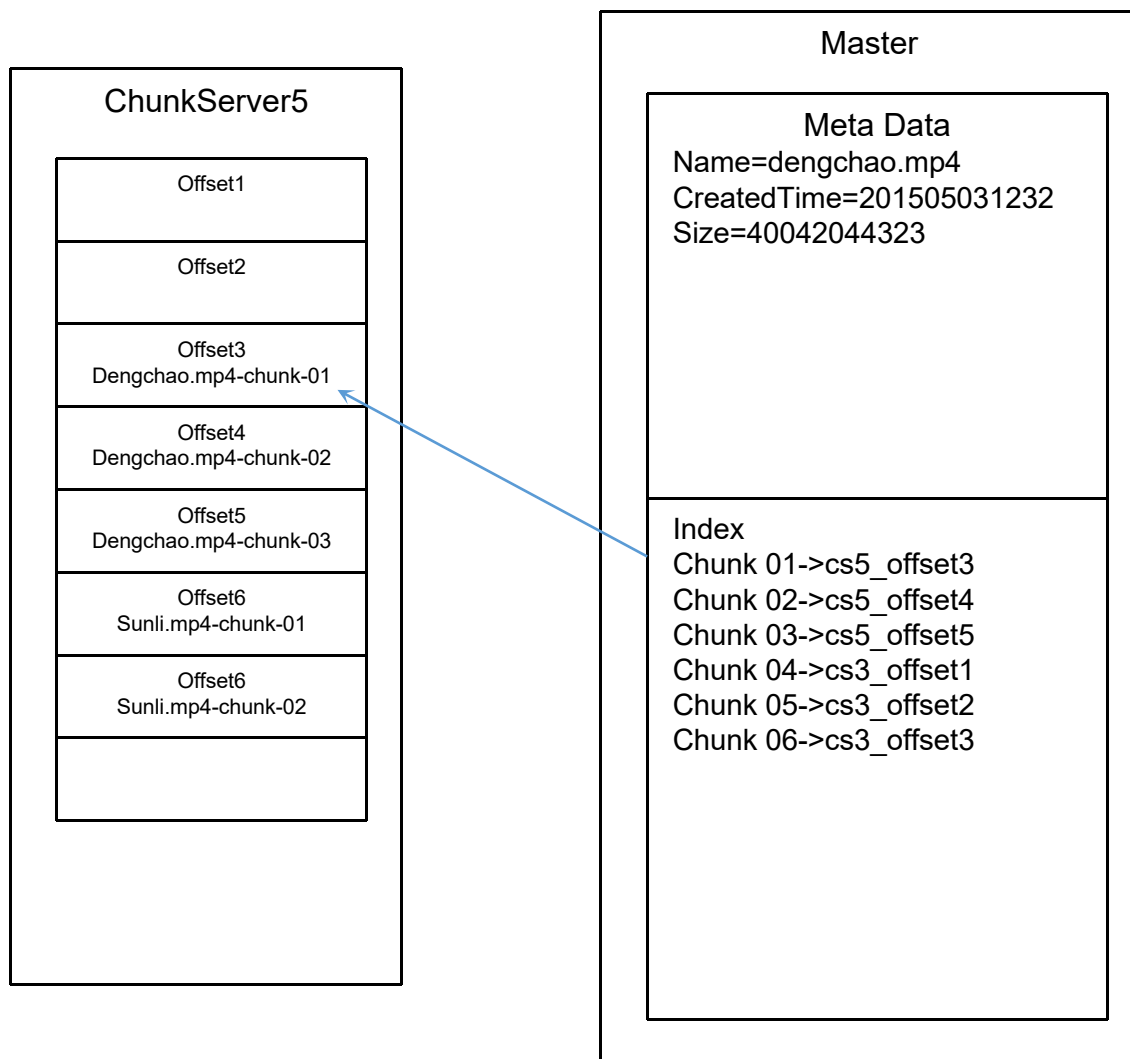


# Interviewer: How to save extra-large file in several machine?

10P

Is one machine big enough?

这里的文件并不是指一个dengchao.mp4就那么大  
而是很多个文件

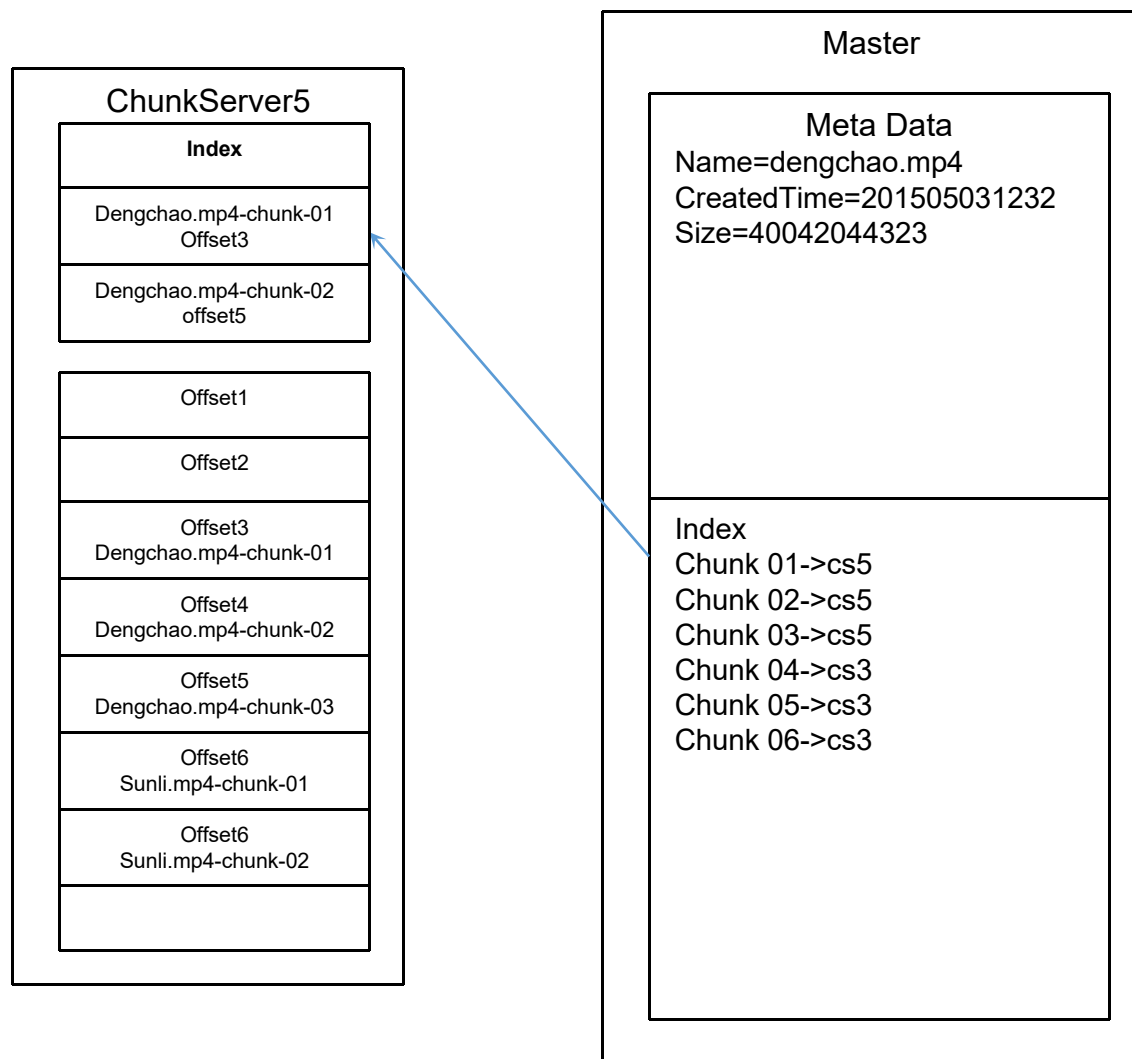


## Key point

- One master + many ChunkServers

Slave Servers = Chunk Servers

每个chunk的Offset偏移量可不可以不存  
在master上面？



## Key point

- The master don't record the diskOffset of a chunk

## Advantage

- Reduce the size of metadata in master
- Reduce the traffic between master and ChunkServer (chunk offset改变不需要通知master)

# Master 存储10P 文件的metadata 需要多少容量?

1 chunk = 64MB needs 64B.(经验值)

10P=16\*10<sup>6</sup> chunk needs 10 G

- 按照4S分析
  - **S**enario 场景分析
  - **S**ervice 服务
  - **S**torage 存储
  - **S**cale 升级优化
- 理清楚work solution
- **S**cale 升级优化

# One Work Solution for Read / Write



# Interviewer: How to write a file?



# 一次写入 还是拆分成多份多次写入？



client

把大胖子直接写入呢？  
还是把大胖子碎尸万段了后写入呢？

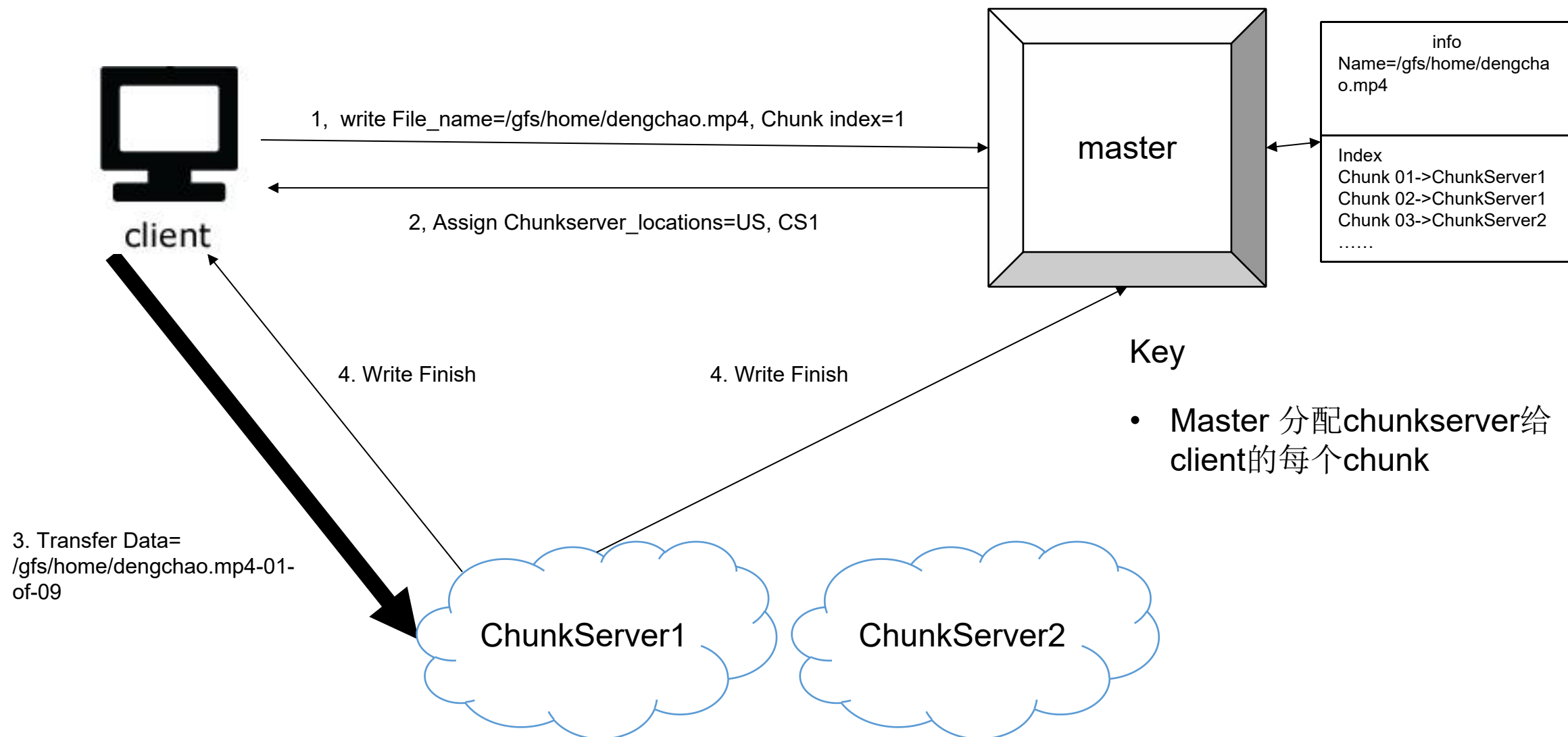
- 写入过程中出错了，那么需要重新写入，哪一种方法更好？
  - 一次传输得重新传输整个文件，多次只用重新传一小份。
- 如果是分成多份多次写入，那么每一份的大小？
  - 文件本来是按照Chunk来存储的，所以传输单位也是Chunk

# 那每一个chunk是怎么写入server的呢？

直接写到chunk server? 

需要先和master沟通，再写入chunk server? 

# How to write a file?



# 要修改Dengchao.mp4怎么办？

/gfs/home/dengchao.mp4

要修改的部分在哪个chunk？

修改了过后chunk变大了要怎么处理？

修改了过后chunk变小了要怎么处理？

# 要修改Dengchao.mp4怎么办？'

One time to write, Many time to read.

先删掉/gfs/home/dengchao.mp4

重新把整个文件重写一份

# Interviewer: How to read a file?

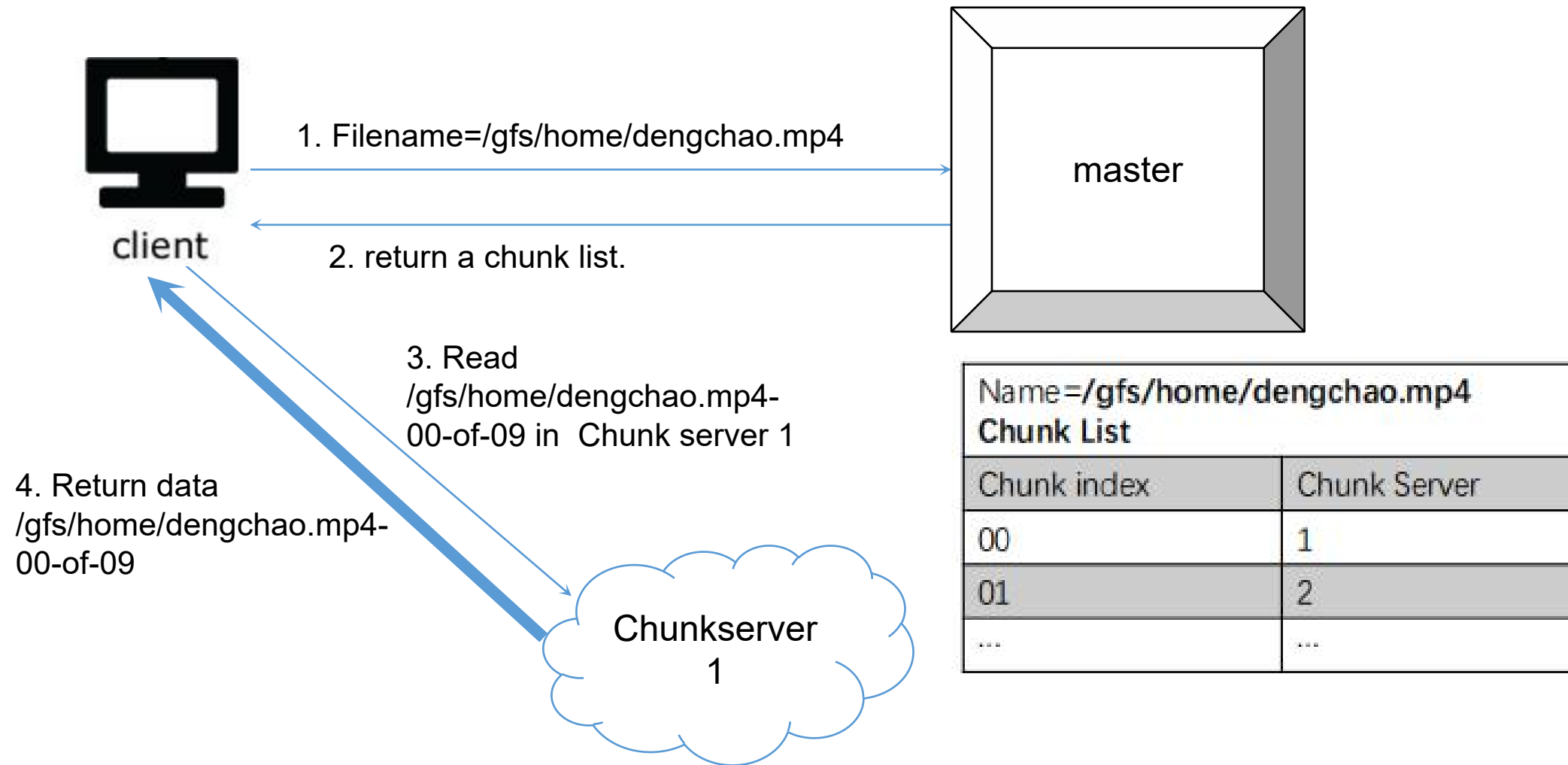
一次读整个文件？  
还是拆分成多份多次读入？





那么client怎么知道dengchao.mp4  
被切成了多少块?

# How to read from a file?



- 存储各个文件数据的metadata
- 存储Map(file name + chunk index -> chunk server)
  - 读取时找到对应的chunkserver
  - 写入时分配空闲的chunkserver

- 存储
  - 普通文件系统 Meta Data, Block
  - 大文件存储: Block-> Chunk
  - 多台机器超大文件: Chunk Server + Master
- 写入
  - Master+Client+ChunkServer 沟通流程
  - Master 维护metadata 和 chunkserver 表
- 读出
  - Master+Client+ChunkServer 沟通流程

# Scale 升级

系统如何优化与维护  
GFS的精髓

# 单Master 够不够？

# 单Master 够不够？

工业界90%的系统都采用单master

Simple is perfect

# Single Master Failure

Double Master

Paper: [Apache Hadoop Goes Realtime at Facebook](#)

Multi Master

Paper: [Paxos Algorithm](#)



# Scale about the Failure and Recover



Interviewer: How to identify whether  
a chunk on the disk is broken?

# Checksum

A: Check Sum 存在 False Positive	B: Check Sum 存在 False Negative
C: Check Sum 只能通过数据的和进行校验	D: 使用 Check Sum 的目的是因为磁盘存储的数据可能会因为磁性的变化导致数据出错
<input type="button" value="已回答"/>	<input type="button" value="我不会"/>

Mailtag

我选择了 D

没有答对哦！正确答案是 B D，有34%的同学做对了这道题目哦，继续努力！

False Negative 是指 check sum 认为是没问题的，但是可能存在问题。这个是会发生的，比如 1,4 两个数据，check sum = 5，但是 2 + 3 也是 5。所以 check sum 相同不能证明数据一定没有发生变化。但是 check sum 不同就表示原始数据肯定发生了变化。因此 check sum 是存在 False Negative 但不存在 False Positive 的。

原来

数据	1	2	3	Checksum(xor)
二进制表示	01	10	11	00

错误后

数据	1	3	3	Checksum(xor)
二进制表示	01	11	11	01

- Checksum Method (MD5, SHA1, SHA256 and SHA512)
- Read More: <https://en.wikipedia.org/wiki/Checksum>

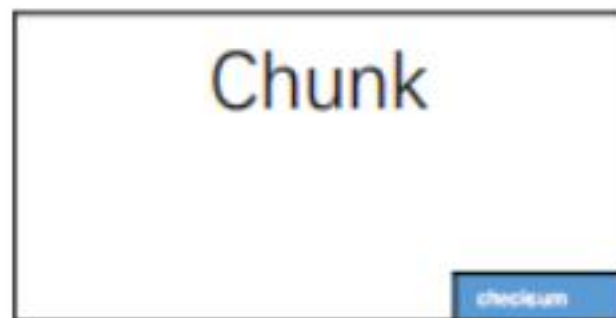
# How to identify whether a chunk on the disk is broken?

- 1 checksum size?
- 4bytes = 32bit
- 1 chunk = 64MB
- Each block has a checksum
- The size of checksum of 1T file
- $1\text{P}/64\text{MB} \times 32\text{bit} = 62.5\text{ MB}$
- Add check sum for blocks is acceptable.

# 什么时候写入checksum?

# 什么时候写入checksum?

Answer: 写入一块chunk的时候顺便写入



# 什么时候检查checksum?



# 什么时候检查checksum?

Answer: 读入这一块数据的时候检查

1. 重新读数据并且计算现在的checksum
2. 比较现在的checksum和之前存的checksum是否一样

Interviewer: How to avoid chunk data loss when a ChunkServer is down/fail?

# Interviewer: How to avoid data loss when a ChunkServer is down/fail?

Answer: Replica (专业词汇)  
做备份

需要多少个备份？  
每个备份放在哪？

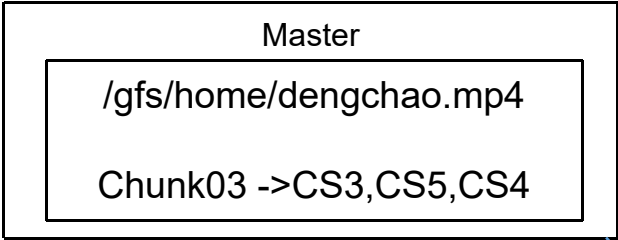
# 需要多少个备份？ 每个备份放在哪？

1. 三个备份都放在一个地方(加州)。
2. 三个备份放在三个相隔较远的地方（加州，滨州，纽约州）
3. 两个备份相对比较近，另一个放在较远的地方（2个加州，1个滨州）

# Interviewer: How to recover when a chunk is broken?

# Interviewer: How to recover when a chunk is broken?

Answer: Ask master for help



- Ask master for help

2. Ask CS3 and CS5 ?

## ChuckServer4

Dengchao.mp4-  
chunk-03

Sunli.mp4-chunk-02

4. There  
u go



# How to find whether a Chunk Server is down?

# How to find whether a ChunkServer is down?

单选题

在 Heartbeat 的模式中如何知道用户已经离线了?

Heartbeat 的意思是, 客户端每隔一段时间 (如 4s) 向发送一次 request, 告知服务器仍然在线。服务器记录下最近一次收到 request 的时间 (last\_updated\_at)。

A: 比较 last\_updated\_at 的时间戳和当前时间, 看看是不是超过一定容忍的时间范围

B: 定期扫描数据库, 将 last\_updated\_at 比较老的数据标记为 is\_offline

已回答

我不会

Interviewer: HeartBeat.

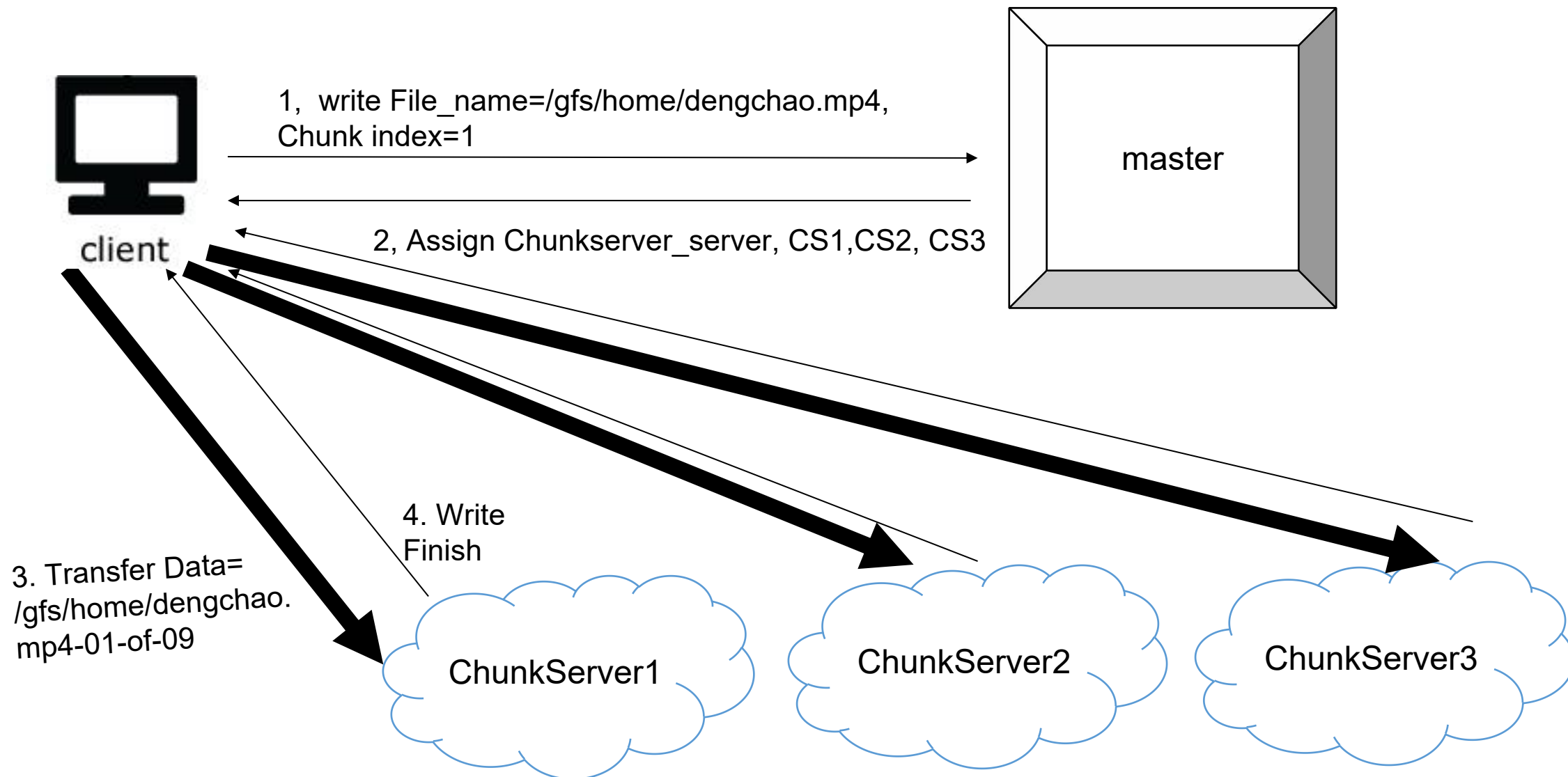
A: master -> chunkservers?

B: chunkservers->master? 

# Scale about the Write

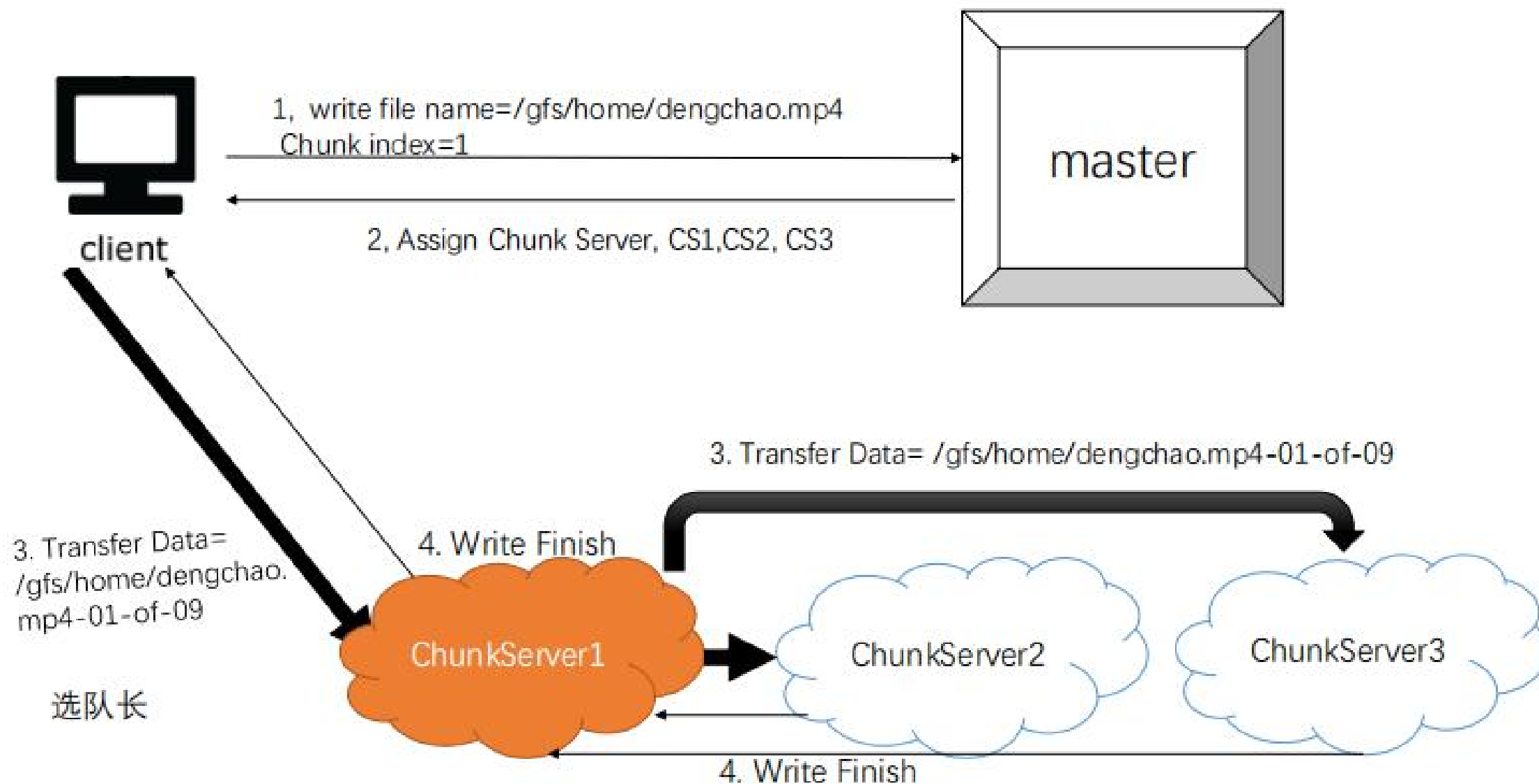
Interviewer: Whether write to only one server is safe?

# How to write a file?



# Interviewer: How to solve Client bottleneck?

# How to solve Client bottleneck?



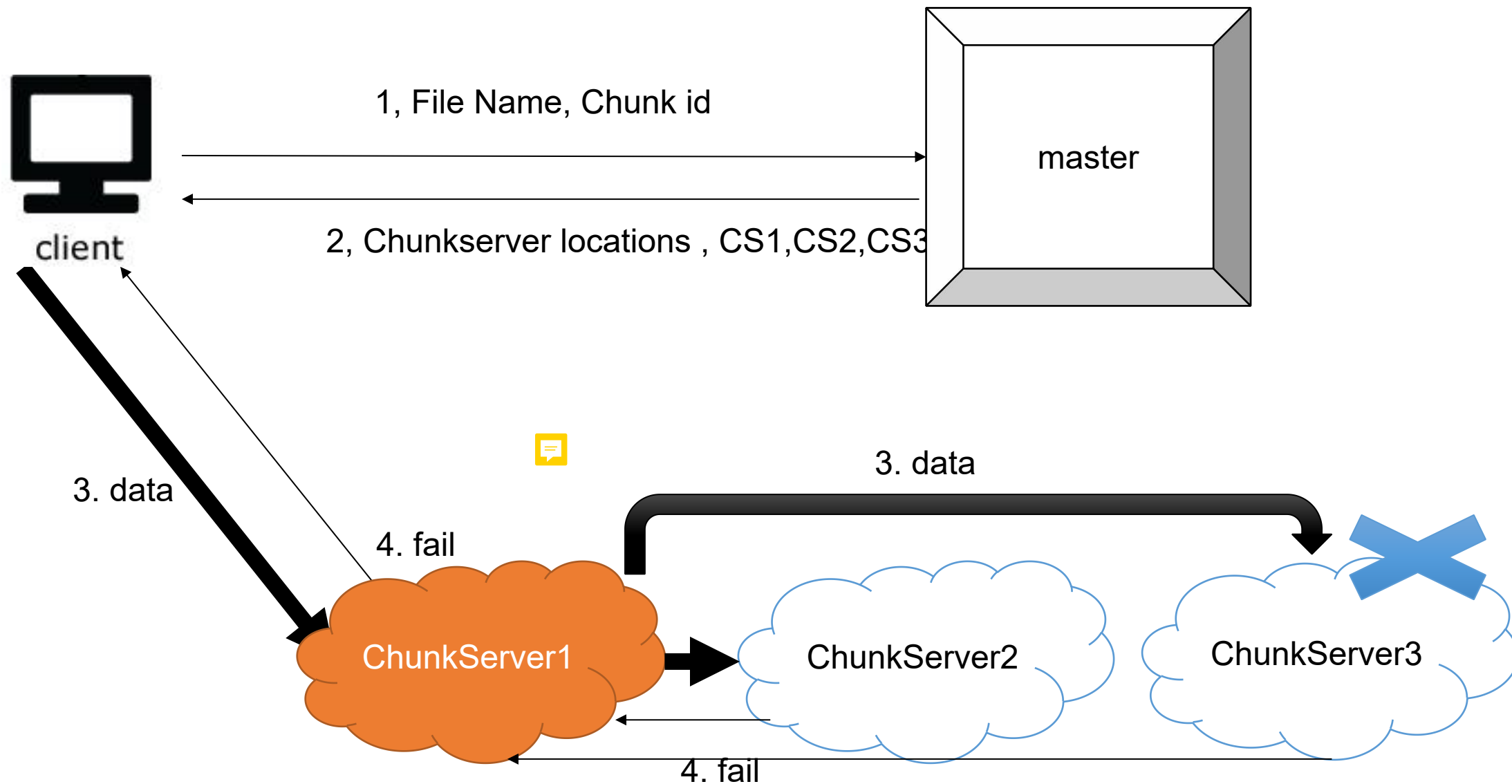
# Interviewer: 怎么样选队长?

1. 找距离最近的 (快)
2. 找现在不干活的 (平衡traffic)

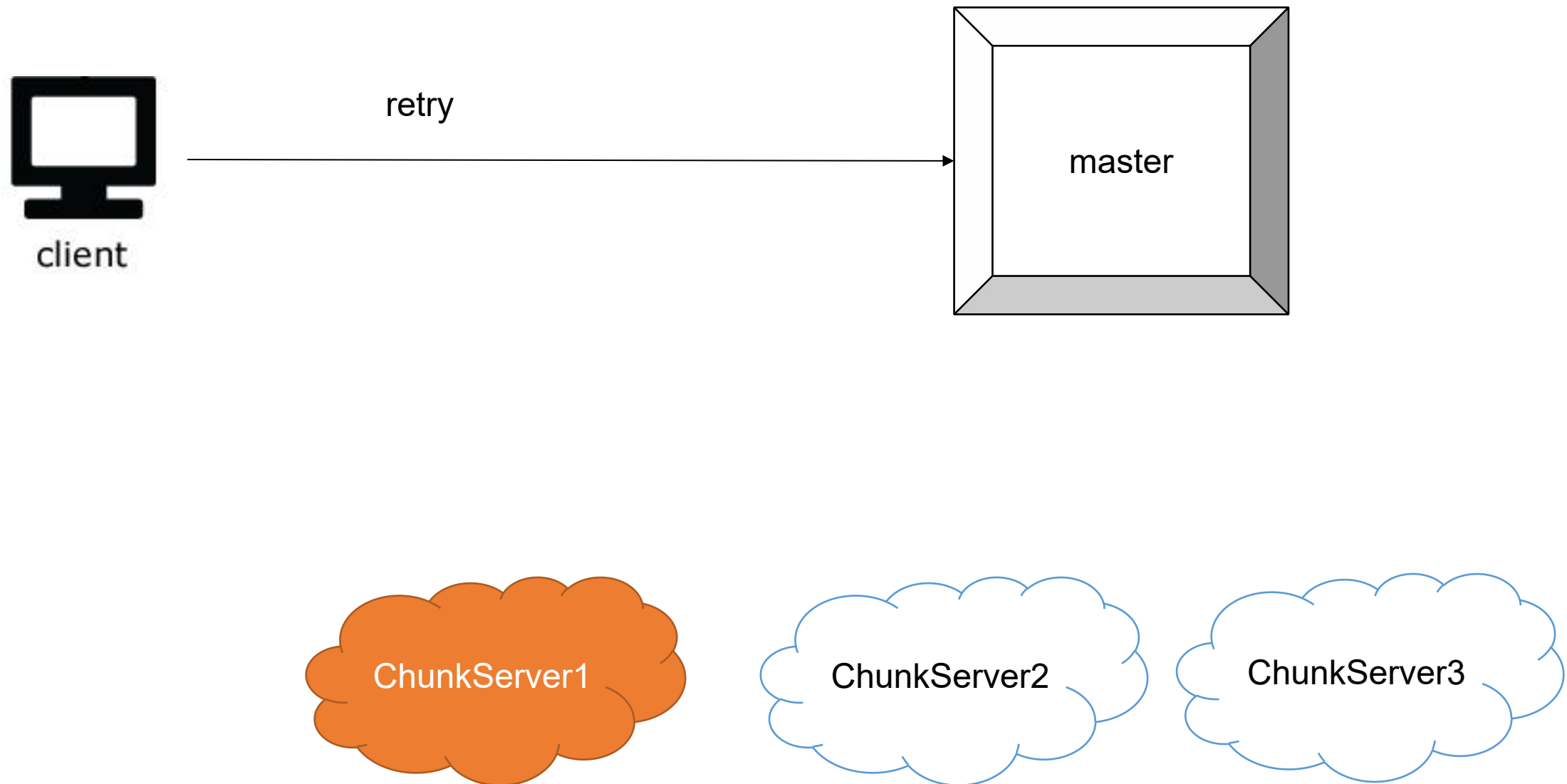
# Interviewer: How to solve Chunk Server failure?



# How to solve ChunkServer failure?



# How to solve ChunkServer failure?



- Key Point: Master-Slave
- Storage:
  - Save a file in one machine -> a big file in one machine -> a extra big file in multi-machine
  - Multi-machine
    - How to use the **master**?
    - How to traffic and storage of master?
- Read:
  - The process of reading a file
- Write:
  - The process of writing a file
  - How to reduce master traffic?
    - Client 和 Chunk Server沟通
  - How to reduce client traffic?
    - Leader Election
- Failure and Recover (key)
  - Discover the failure a chunk?
    - **Check Sum**
  - Avoid the failure a chunk?
    - **Replica**
  - Recover the failure?
    - Ask master
  - Discover the failure of the chunkserver?
    - **Heart Beat**
  - Solve the failure of writing ChunkServer?
    - Retry

# Google onsite non-abstract large scale system design 真题

<https://www.jiuzhang.com/qa/627/>

- Expert/Master, <http://url.cn/dOLFCs>
  - Expert/Master, <http://url.cn/eErkhm>
  - Expert/Master, <http://url.cn/LqTkoa>
- 
- 为什么说学习**GFS**对我们其他的系统设计也有好处呢？
    - Master Slave Pattern
    - How to handle failure
    - How to use GFS

# GFS实战

## 设计lookup service

真实面经：

- 设计一个只读的lookup service. 后台的数据是10 billion个key-value pair, 服务形式是接受用户输入的key, 返回对应的value。已知每个key的size是0.1kB, 每个value的size是1kB。要求系统QPS  $\geq 5000$ , latency  $< 200\text{ms}$ .
- server性能参数需要自己问, 我当时只问了这些, 可能有需要的但是没有问到的……
  - commodity server
  - 8X CPU cores on each server
  - 32G memory
  - 6T disk
- 使用任意数量的server, 设计这个service

同学解答:

given 10 billion key-value pair

=> total key size  $\sim 10 \text{ billion} * 0.1\text{kB} = 1\text{T}$

=> total value size  $\sim 10 \text{ billion} * 1\text{kB} = 10\text{T}$

with 6T disk , a server with two disks will be enough



同学解答:

For every request, 1 value, which is 1kB needs to be returned

total time for reading one value will be  $10\text{ms}(\text{disk seek}) + 1\text{kB}/1\text{MB} * 30\text{ms}(\text{reading 1kB sequentially from disk})$   
 $= 10\text{ms}.$

同学解答:

QPS on 1 server will be  $1s/10ms * 2 \text{ disk} = 200$

required QPS support is 5000. So we need  $5000/200 = 25$  servers.

同学解答：

Finding the key, read the value.

Using binary search  $\log(n)$

For each time, the disk latency is 1 seek + 1 read.

Reading key is really small, so can be ignored.

Total time for find the key :  $\log(10\text{billion}) * 10\text{ms} = 100\text{ms}$ .

Reading a key will take another disk seek , 10ms.

1 round trip in the same data center is 0.5ms.

Total latency is  $100 + 10 + 0.5 = 110.5\text{ms}$ .

|, so can be ignored.  
 $\log(10\text{billion}) * 10\text{ms} = 100\text{ms}$ .  
30 300ms  
other disk seek , 10ms.  
lata center is 0.5ms.  
+ 0.5 = 110.5ms  
310.5ms!

QPS on 1 server will be  $1s/10ms * 2 \text{ disk} = 200$

required QPS support is 5000. So we need  $5000/200 = 25$  servers.

## GFS实战

QPS on 1 server will be  $1s/10ms * 2 \text{ disk} = 200$

required QPS support is 5000. So we need  $5000/200 = 25$  servers.

1s / 300ms }

6

(200)

300ms

- 我们希望减少什么的时间:
  - finding the key 的300ms

- 什么没有用上?
  - 内存

- 一台机器32G内存

- 40台机器就可在内存中装下所有的<key, 硬盘地址>这样的键值对
- 内存中二分查找, 30次, 时间可以忽略不计
- so total latency is  $10 + 0.5 = 10.5\text{ms}$

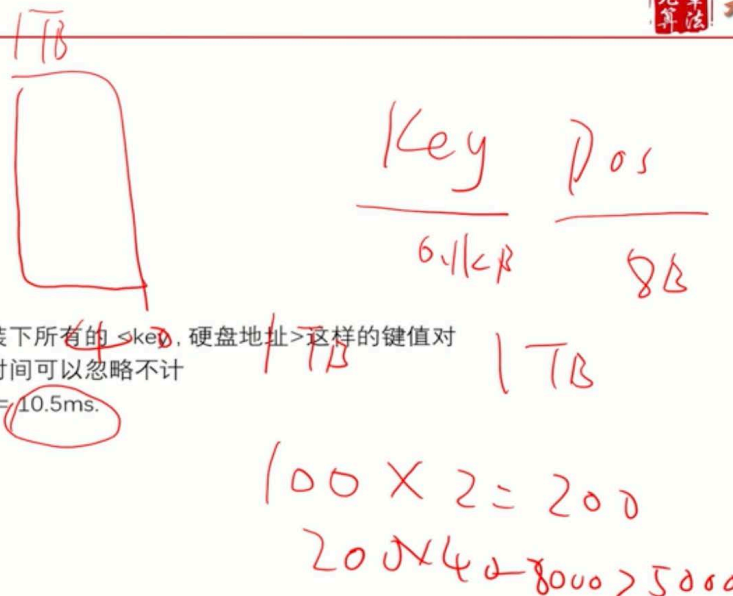
## GFS实战

- 我们希望减少什么的时间:
  - finding the key 的300ms

- 什么没有用上?
  - 内存

- 一台机器32G内存

- 40台机器就可在内存中装下所有的<key, 硬盘地址>这样的键值对
- 内存中二分查找, 30次, 时间可以忽略不计
- so total latency is  $10 + 0.5 = 10.5\text{ms}$ .



Copyright © www.jiuzhang.com



# GFS常见问题解答

问：什么是文件系统中的block？

0	1	2	3	4	5	6	7	8
4kb	4kb	4kb	4kb	4kb	4kb	4kb	4kb	4kb

1 block

问：什么是异或（XOR）操作？

XOR	0	1
0	0	1
1	1	0

相同为0，不同为1



问：再解释下Check Sum?

思考：如果你记录一串数在硬盘 1 2 3 ..... 8 9 10, 怎么保证10年后记录不出错?

1 2 3 4 ..... 8 9 10 55

"Ilovecoding" CCAC0ED4DFAFFA2A

"I am happy to join with you today in what will go down in history as the greatest demonstration for freedom in the history of our nation"

9C72CD9A76B45B04

