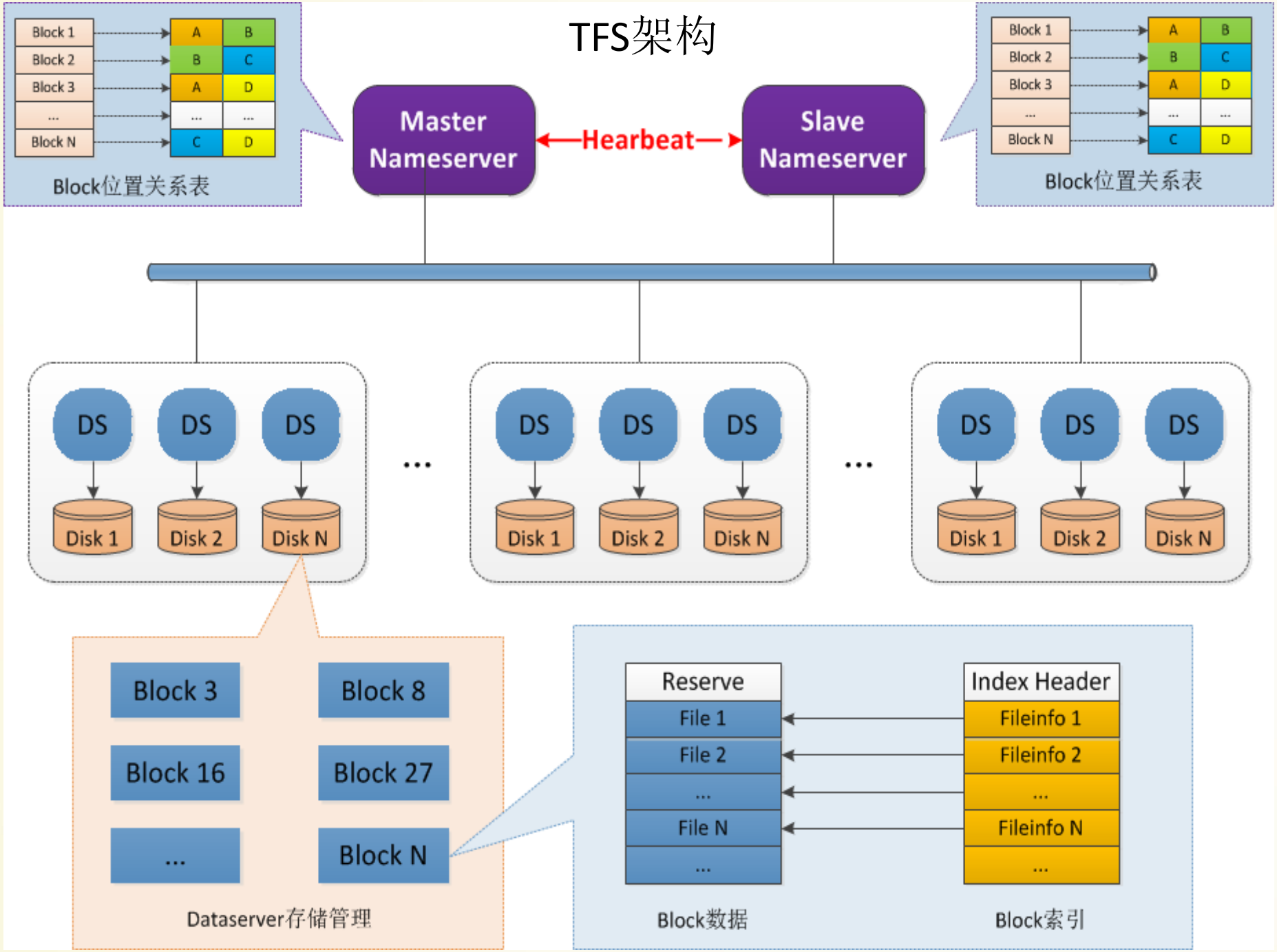
DTCC 2014中国数据库技术大会
DATABASE TECHNOLOGY CONFERENCE CHINA 2014
大数据技术探索和价值发现


# TFS Erasure Code应用实践

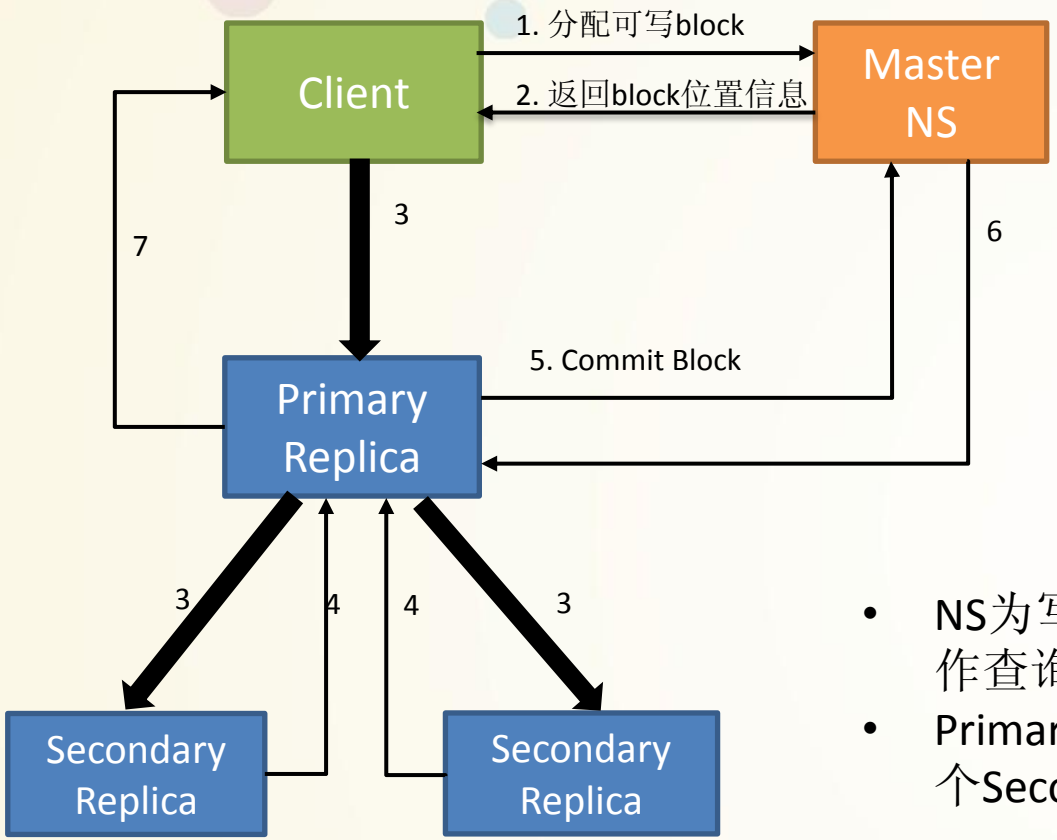
阿里云核心系统存储组 张友东
技术博客：http://yunnotes.net
新浪微博：@HUST张友东

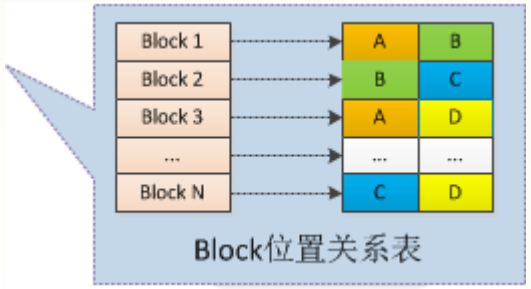
# TFS应用现状

- Taobao File System
  - 分布式文件存储系统
- 数千台存储节点
  - 单台11或12块SATA盘
  - 600G、1T、2T、4T
  - 使用裸盘，不做RAID
- 部署总容量数十PB
  - 使用容量约85%
- 存储文件数量超过千亿
  - 文本、图片、音乐、APP、视频等


TFS架构
Block 1
Block 2
Block 3
...
Block N
A
B
C
D
Block位置关系表
Master Nameserver
Hearbeat
Slave Nameserver
Block位置关系表
DS
Disk 1
Disk 2
Disk N
Block 3
Block 8
Block 16
Block 27
Block N
Dataserver存储管理
Reserve
File 1
File 2
File N
Index Header
Fileinfo 1
Fileinfo 2
Fileinfo N
Block数据
Block索引

# TFS写流程



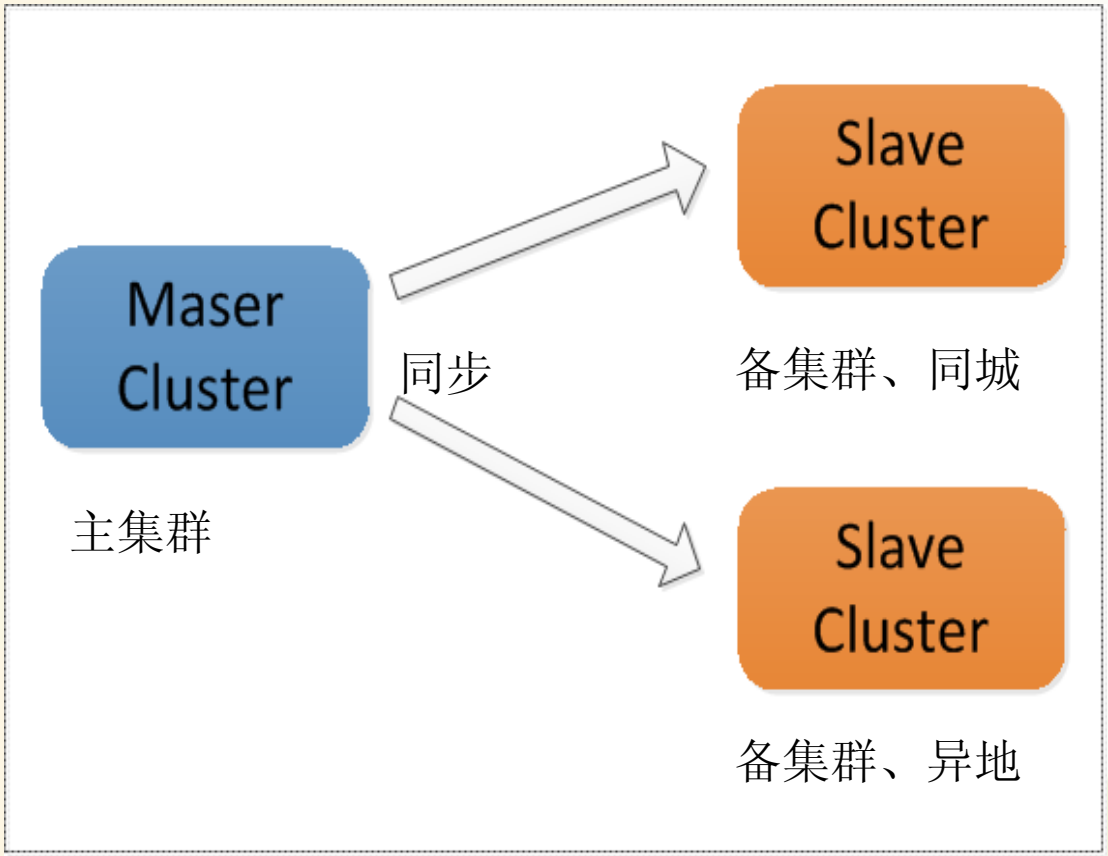




- NS为写操作分配可写块；为读操作查询block位置信息
- Primary Replica负责将数据写到多个Secondary Replica

控制流

数据流

# 存储成本



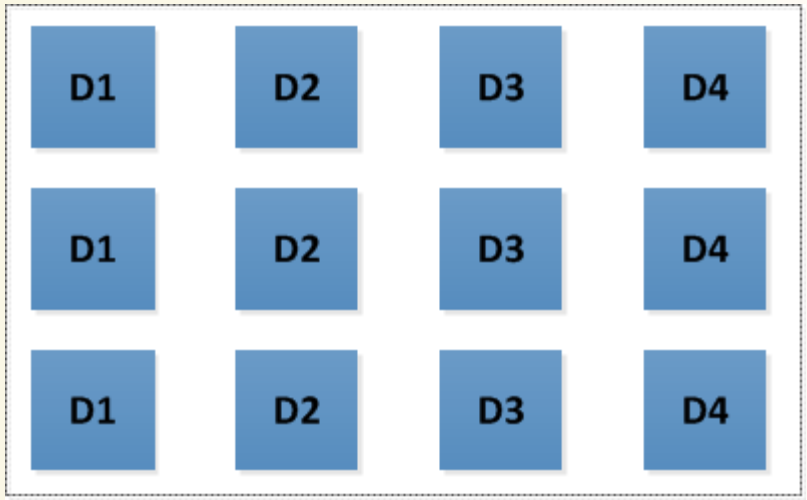

## 存储成本

- 集群内多副本
- 多IDC容灾

## 降低成本

- 优化存储结构 5%
- 利用系统盘 5%
- **应用erasure code 25%**

# 多副本 VS 存储编码



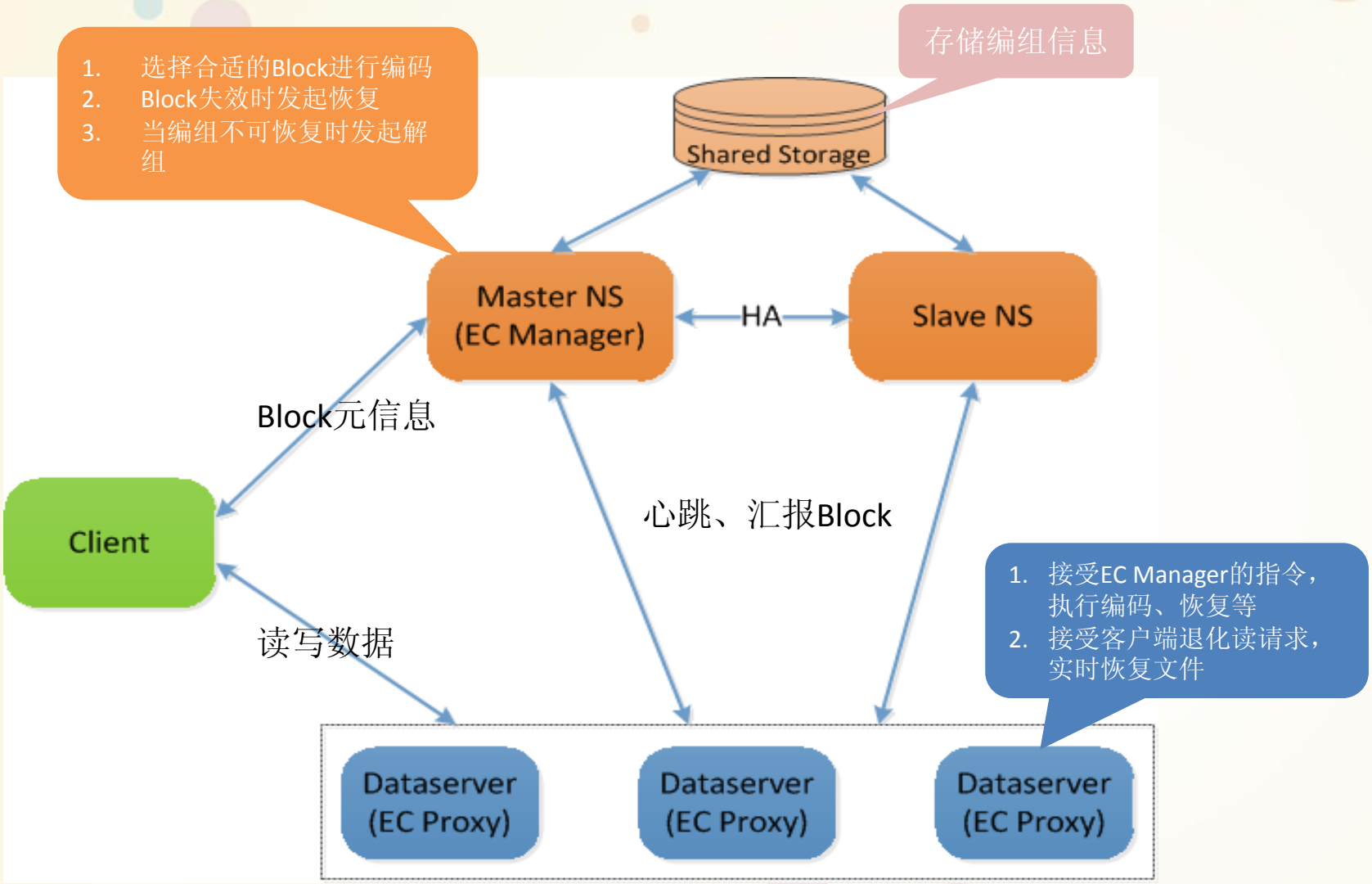

- 3副本
- 容忍2副本失效
- 存储成本 3X


- 4 + 2 Reed Solomon
- 容忍2副本失效
- 存储成本 1.5X

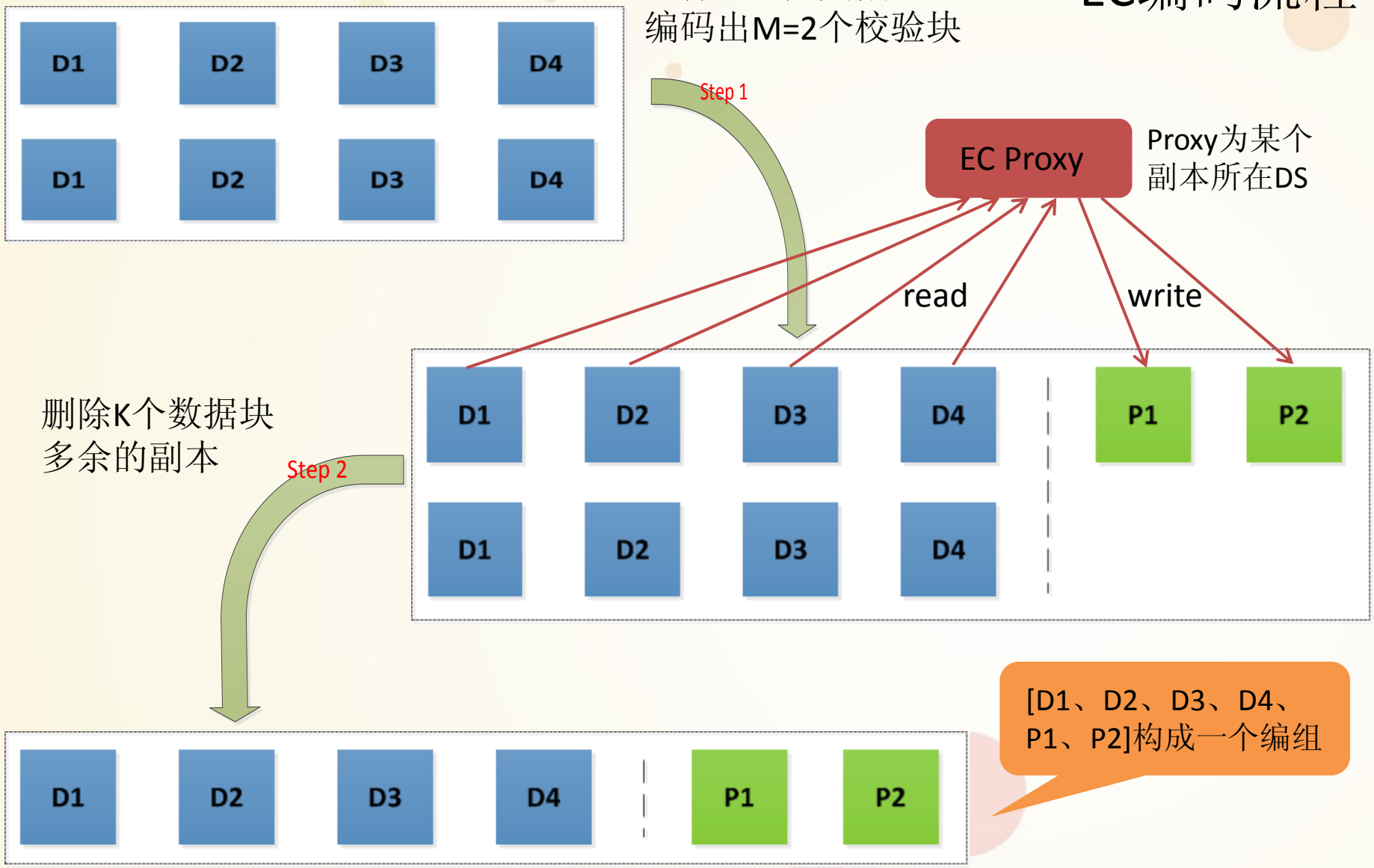
# 技术方案选择

- Caucy Reed Solomon(CRS)算法
  - Jerasure开源库
- 后台异步编码
  - 与文件写入流程完全解耦
  - 以Block为单位进行编码管理
- 编组信息持久化存储
  - 主备NS共享存储
  - 编码算法、参数可随时调整
- 后台Block恢复 + 实时文件恢复(退化读)
- Block编码后支持文件更新

# EC实现方案


1. 选择合适的Block进行编码
2. Block失效时发起恢复
3. 当编组不可恢复时发起解组
存储编组信息
Shared Storage
Master NS
(EC Manager)
HA
Slave NS
Block元信息
Client
心跳、汇报Block
读写数据
1. 接受EC Manager的指令，执行编码、恢复等
2. 接受客户端退化读请求，实时恢复文件
Dataserver
(EC Proxy)
Dataserver
(EC Proxy)
Dataserver
(EC Proxy)

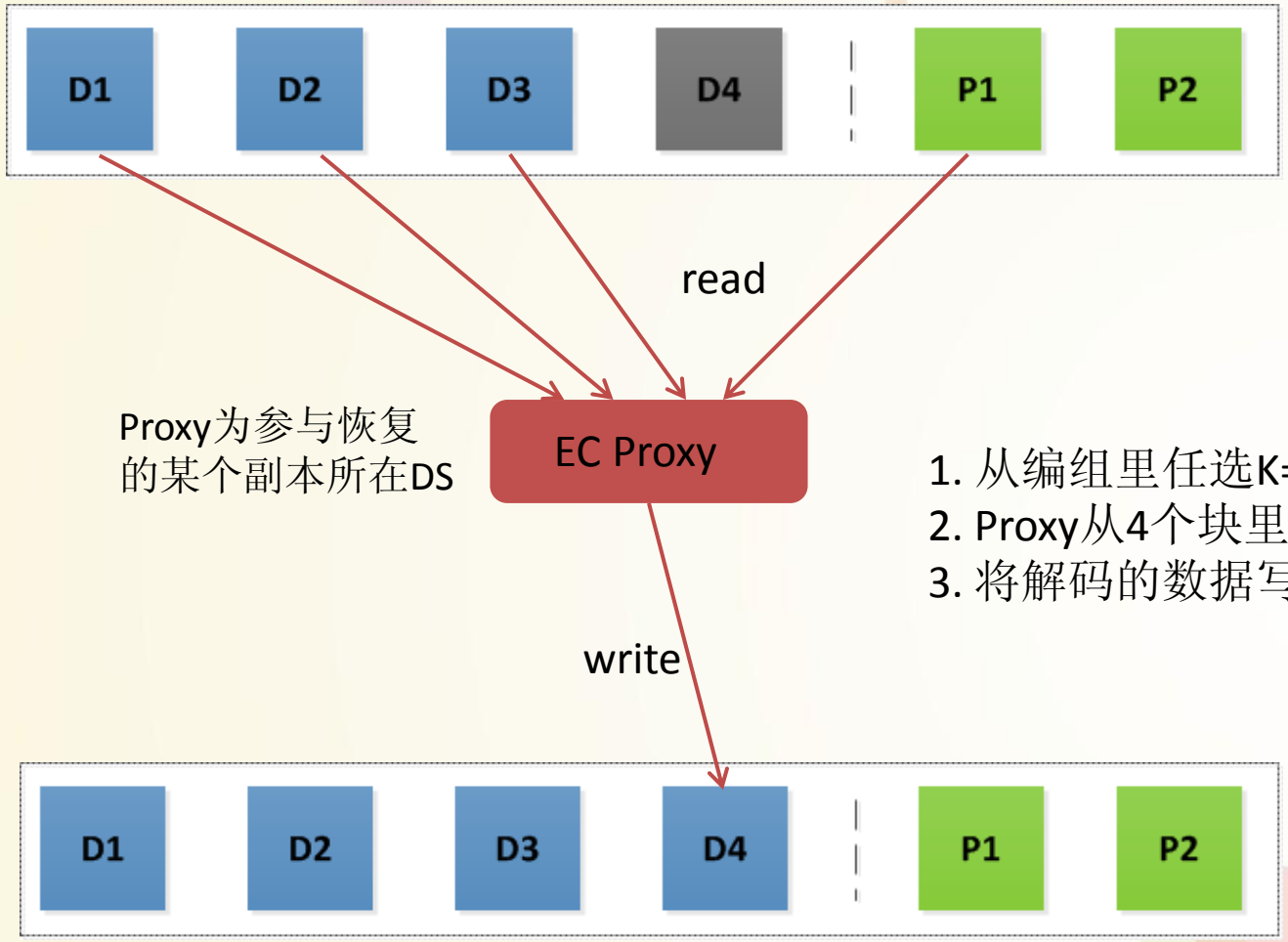
# EC编码流程


选择K=4个数据块
编码出M=2个校验块
D1
D2
D3
D4
D1
D2
D3
D4
Step 1
EC Proxy
Proxy为某个
副本所在DS
read
write
D1
D2
D3
D4
P1
P2
D1
D2
D3
D4
删除K个数据块
多余的副本
Step 2
D1
D2
D3
D4
P1
P2
[D1、D2、D3、D4、
P1、P2]构成一个编组

# Block编组条件

- Block已经写满
  - 不会再新写入文件，只会有少量更新
- Block里删除文件比例不超过阈值(如5%)
  - 延时删除策略
  - 减少存储空间浪费
- Block里的文件一段时间(如1个月)未被访问
  - 冷数据，减小实时恢复发生概率
- 组内Block机架安全

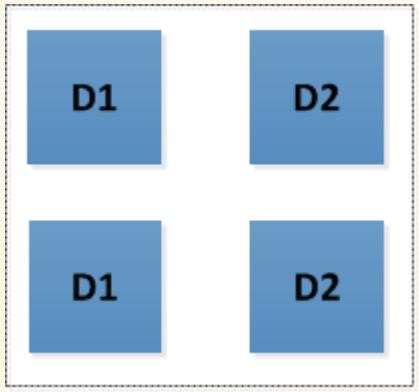
# EC恢复/退化读流程




1. 从编组里任选K=4个块，如[D1,D2,D3,P1]
2. Proxy从4个块里读取数据并解码
3. 将解码的数据写到新的D3

# EC解除编组流程

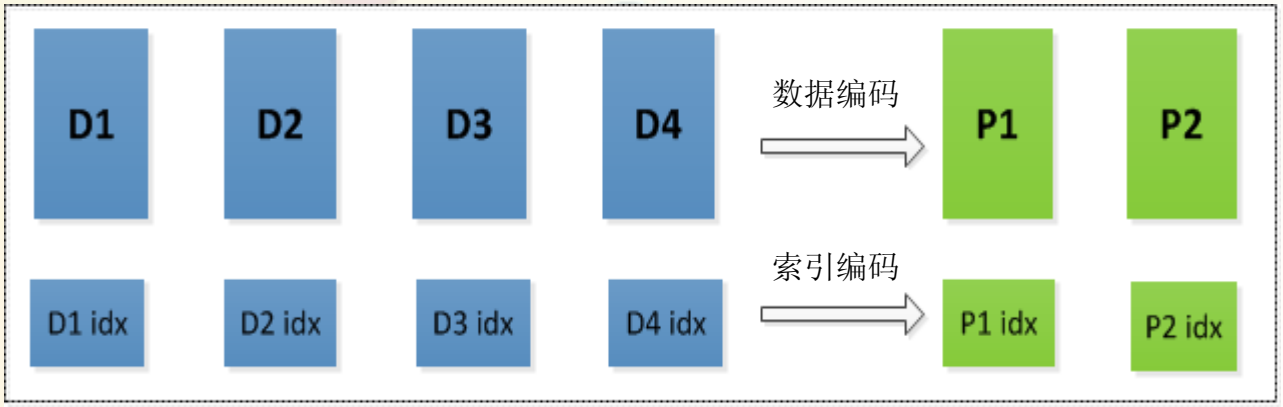

解除编组




当编组中超过K个块失效时，编组不可恢复，为将损失最小化

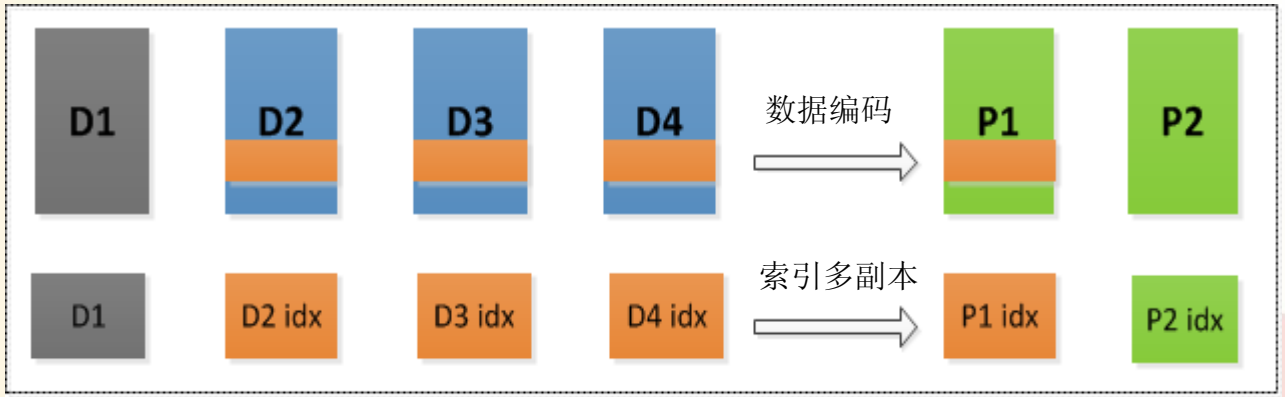
1. 复制所有的数据块
2. 删除所有的校验块

# 索引存储方案




## 索引编码存储

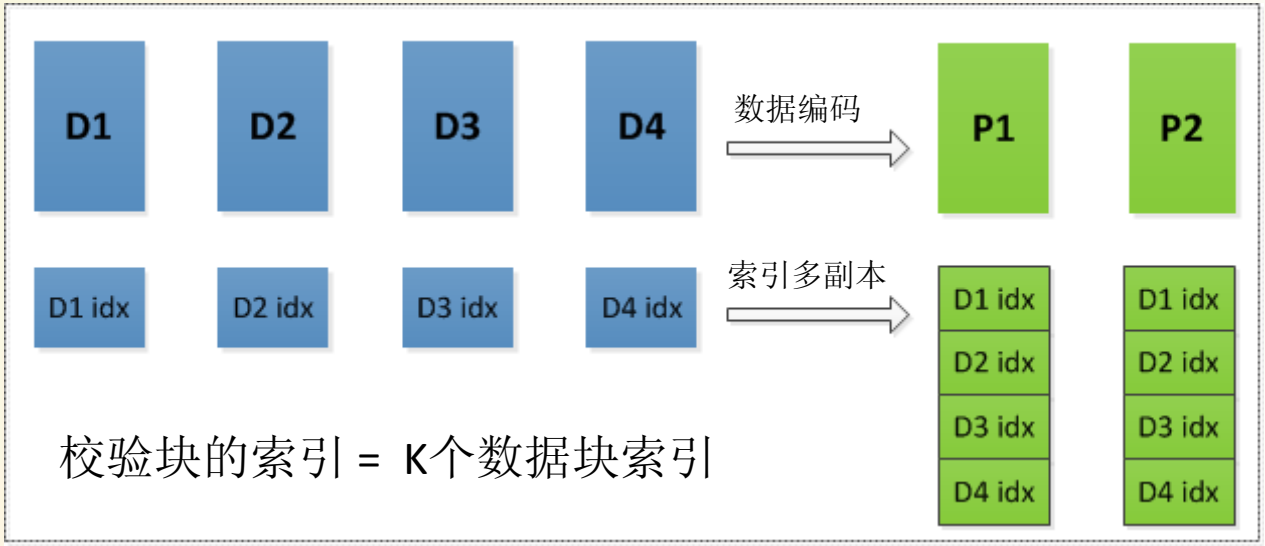
- 简单、直观
- 退化读效率低




## 退化读D1里的文件

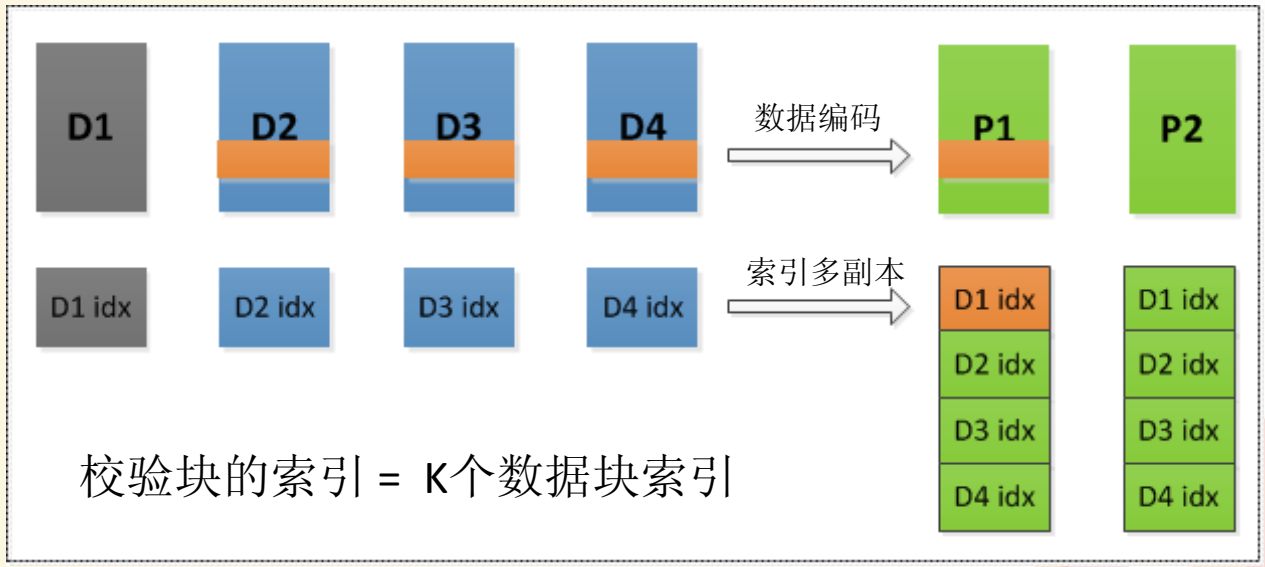
1. K次网络读取index
2. 计算恢复index
3. K次网络读取data
4. 计算恢复data

# 索引存储方案(续)




索引多副本存储

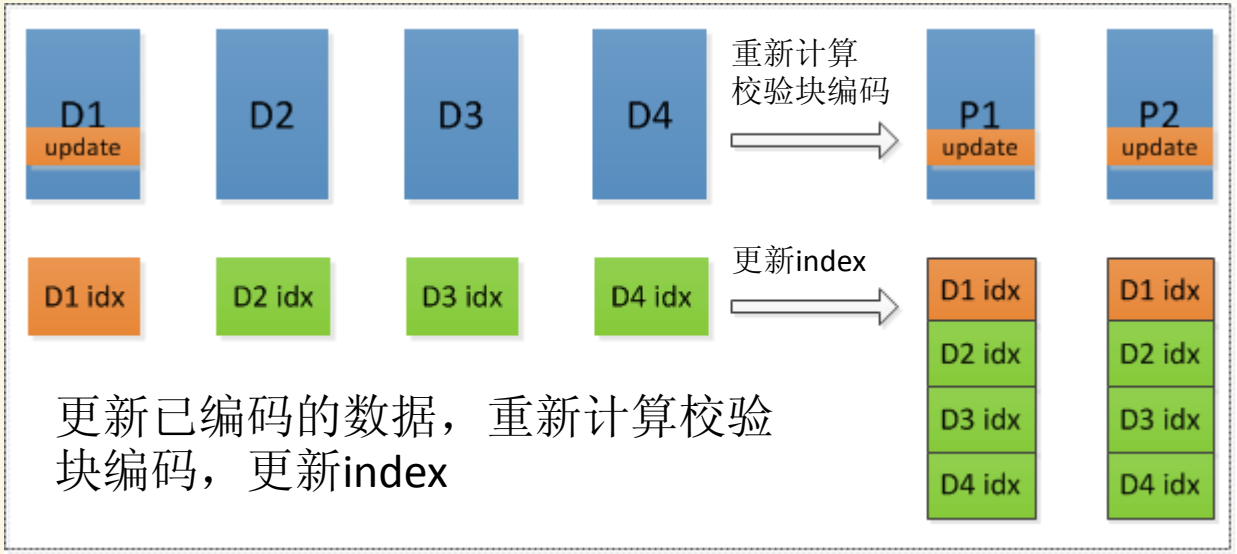
- 索引与数据容错度相同
- Index额外存储开销很小
- 退化读效率高



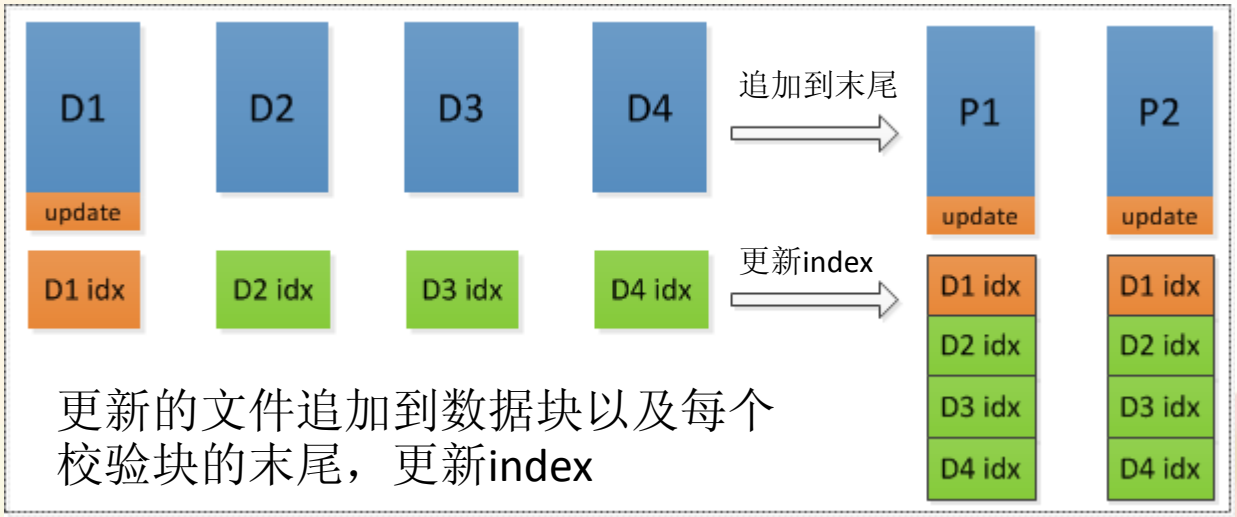

退化读D1里的文件

1. 1次网络查询index
2. K次网络读取data
3. 计算恢复data

# 文件更新支持




- 保证更新原子性
- 实现成本高




- 更新的文件多副本存储
- 更新弱一致性

# 总结

- 方案设计
  - 系统自身特性：Block存储多个小文件
  - 业务场景需求：退化读+更新的支持
- 当前应用情况
  - 成本下降约5%，4+2长期看成本下降趋近25%
- 未来工作
  - 优化编码算法
  - 提升恢复效率
  - 冷数据存储

# Q&A

# THANKS