
SACC
SACC
2021
2021 中国系统架构师大会
SYSTEM ARCHITECT CONFERENCE CHINA 2021
数字转型 架构重塑
ITT168.com
ChinaUnix.net
ITPUB
云上会议 网络直播 | 2021.5.20-2021.5.22

架构设计演进

58同城 后端高级架构师 钟昌寿

目录

WFS架构演进

WFS是什么

简要介绍WFS的基本实现原理以及适用的业务场景及其优势。

WFS一期架构

基于58原有的对象存储WOS与原有的分布式KV存储WTable基础上搭建。

WFS二期架构

重新设计两层模型的对象存储以及基于Paxos数据同步的纯内存元数据管理。

分布式读缓存系统

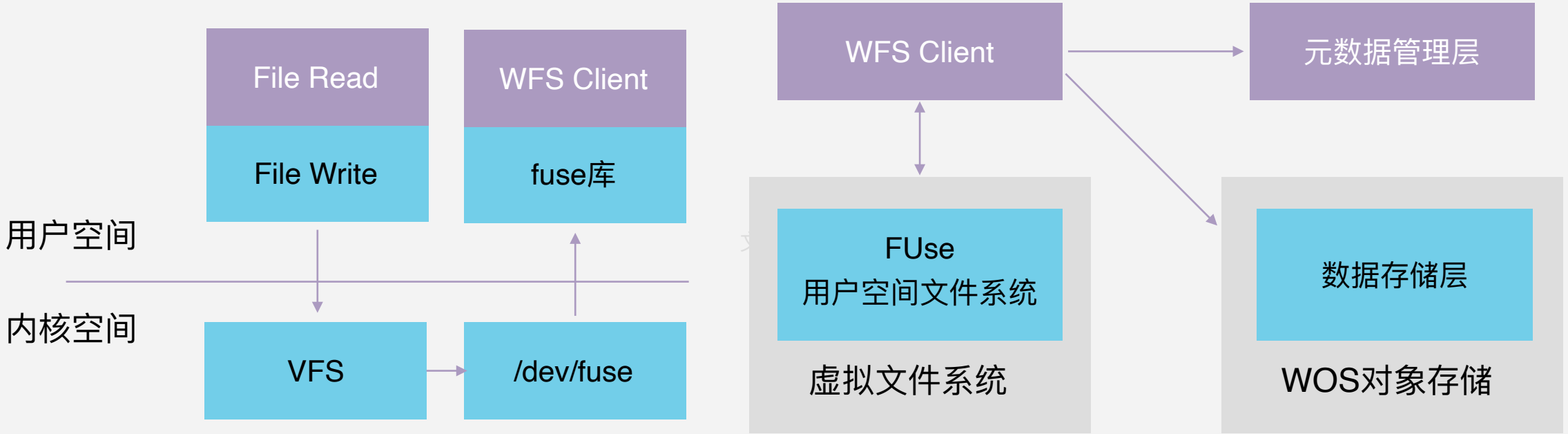
为了避免机械磁盘IO雪崩，在对象存储之上搭建基于SSD磁盘的读缓存，加速热点访问。

01 WFS是什么

简要介绍WFS的基本实现原理以及适用的业务场景及其相对于传统的对象存储的优势。

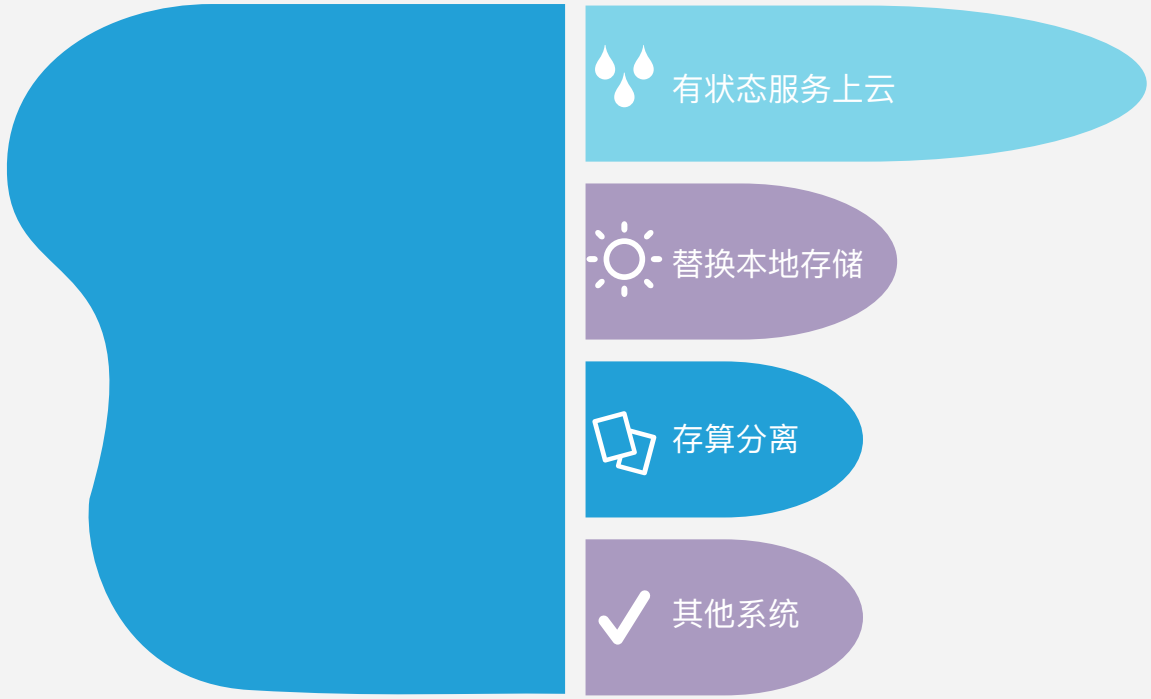
WFS基本原理

基本访问流程与主要相关系统



WFS适用业务场景

适用的业务场景及其优点简要介绍



由于Docker+K8S的服务特性，业务服务节点可能会根据系统调度等原因而在物理机之间漂移。对于那些有状态的服务，在他们服务节点漂移时需要把这台跟着转移。此时使用WFS将会十分方便，在创建新的Pod时自动挂载业务的存储节点即可。

比较常见的比如AI系统，将训练素材放在WFS中，训练节点放在物理机或者云上。由于WFS支持POSIX标准接口，不需要对程序做任何修改即可使用。还能够方便AI平台与业务部门共享训练素材和训练模型的上传与下载。

比如ES系统，ClickHouse，Hadoop系统，将这些系统的底层存储替换成WFS即可。

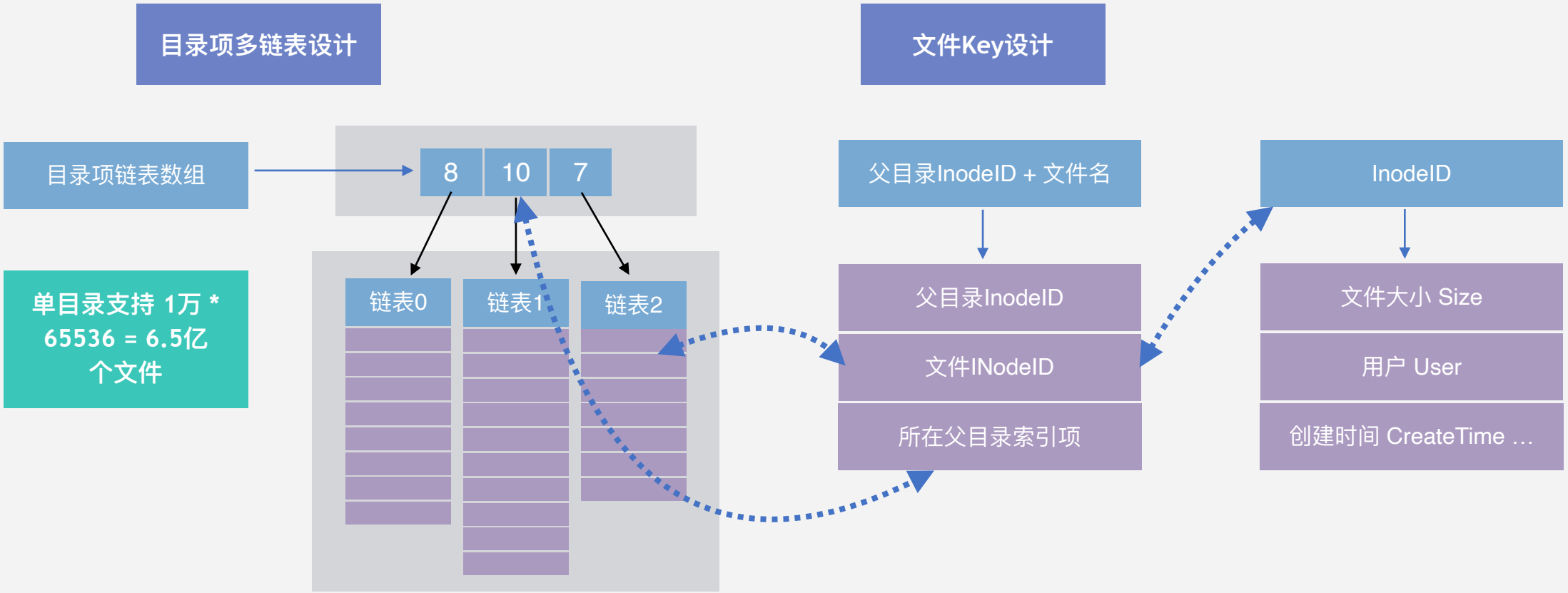
原来存储到NFS，MFS，CephFS上的数据可以转移过来，可用于存储Nginx日志，私有云日志等。

02 WFS一期架构

WFS一期系统是构建在原有系统之上，存储层放置到原有的对象存储，元数据则存储到分布式KV之中。

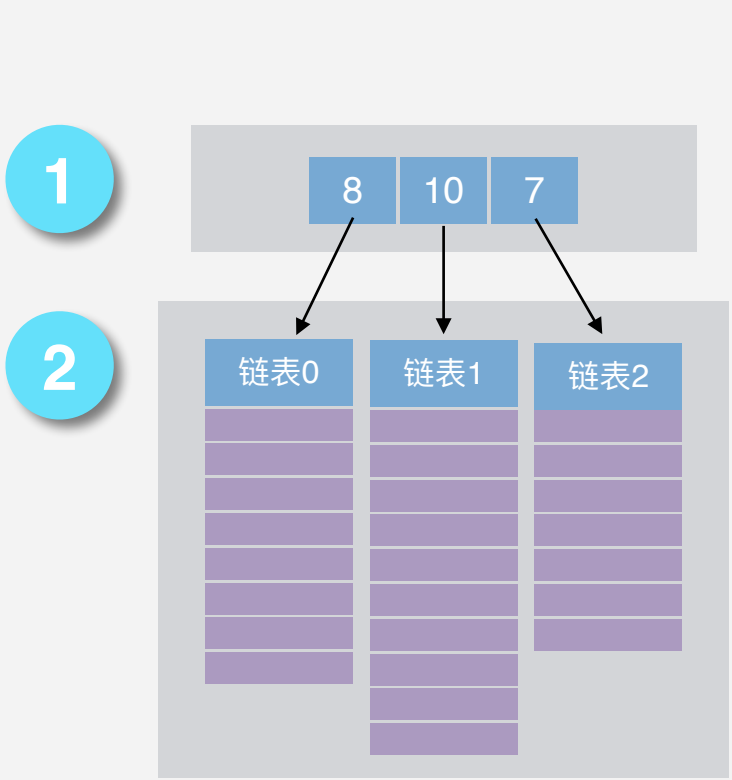
元数据Key设计

采用成熟的分布式KV作为存储



元数据实现难点-相关Key一致性

采用成熟的分布式KV作为存储



3

父目录InodeID + 文件名

父目录InodeID

文件INodeID

所在父目录索引项

4

InodeID

文件大小 Size

用户 User

创建时间 CreateTime …

5

文件分片列表

Offset -> NeedleID

Offset -> NeedleID

……

元数据实现难点-互斥锁

采用成熟的分布式KV作为存储

01

删除空目录加锁

Rename&Remove时需要确保只删除空目录，采用Cas与设置《目录锁》。

02

创建文件检查目录锁

允许并发创建文件，但是需要确保创建文件时父目录没有被删除，采用两阶段检查《目录锁》。

03

文件并发写

写一个文件块涉及到写多个Key，由于没有原子更新能力，因此也需要加锁，确保数据一致性。

元数据设计总结

采用成熟的分布式KV作为存储

元数据管理使用分布式KV存储的缺点：

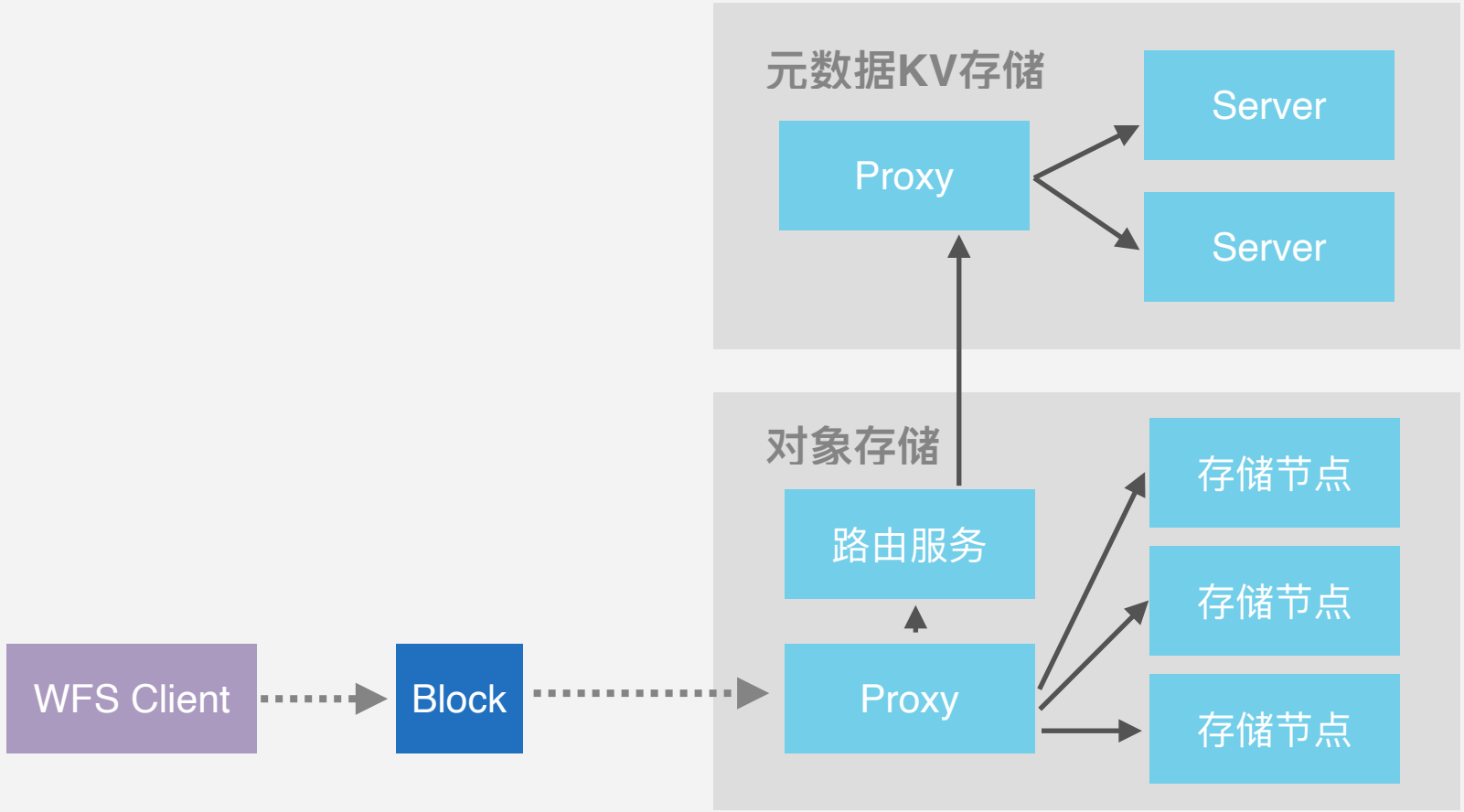
一、没有分布式事务，对多个Key修改无法作为一个原子实现，需要设计合适容错机制或锁机制。

二、每次文件操作涉及到多个Key与锁检查，需要与KV存储多次交互，影响性能。

三、代码逻辑复杂，需要处理各种网络超时、并发等导致的失败，部分情况需要回滚修改。

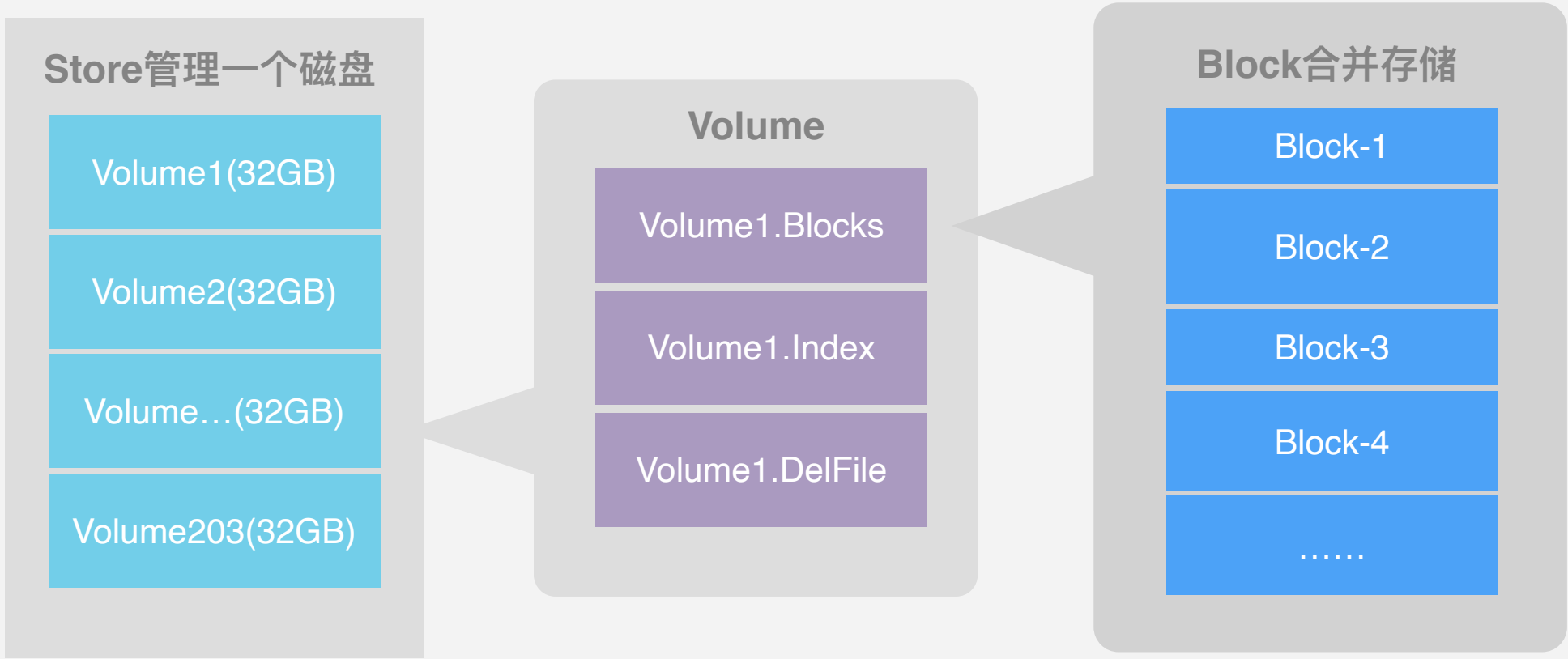
数据存储层-对象存储

采用对象存储作为存储层服务



一期存储管理

采用对象存储作为存储层服务



数据存储层总结

采用对象存储作为存储层服务

存储层直接使用分布式对象存储的缺点：

一、读写Block的IO路径太长，网络延迟较高。

二、存在较多的随机写场景，写入性能较差。

三、删除数据后，空间回收需要消耗大量IO资源并且速度很慢。

03 WFS二期架构

WFS二期在架构上更注重读写的性能，不再局限于原有的系统而是针对WFS业务特点进行定制化开发。

WFS二期开发背景

海量小文件性能瓶颈，数据回收瓶颈

海量小文件高性能读写，存储空间快速回收：

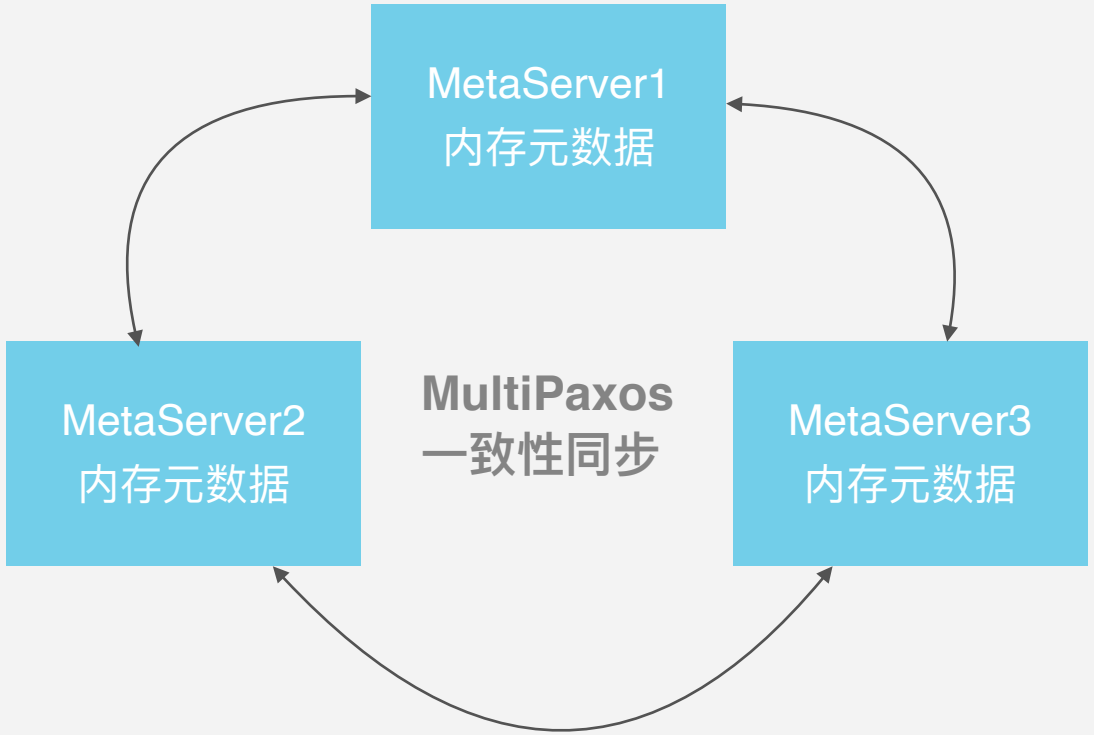
一、海量小文件场景下，元数据操作十分频繁，成为主要瓶颈。

二、文件删除频繁，要求具备快速的空间回收能力。

三、存储层网络IO路径较长，文件调度等原因导致性能较差。

WFS二期元数据管理

采用MultiPaxos同步的纯内存存储

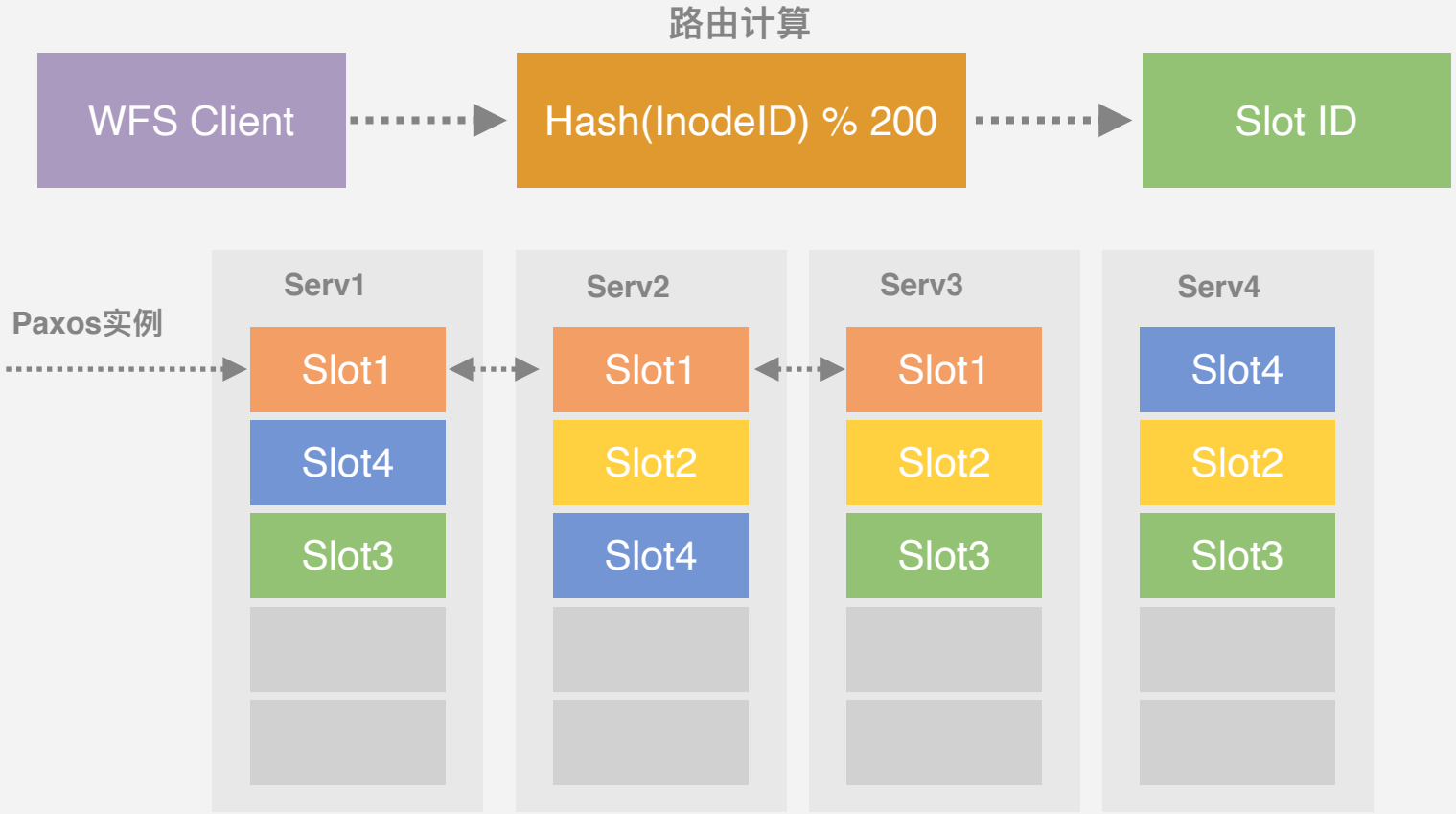


纯内存读写
定期创建快照
PaxosLog

快照位置

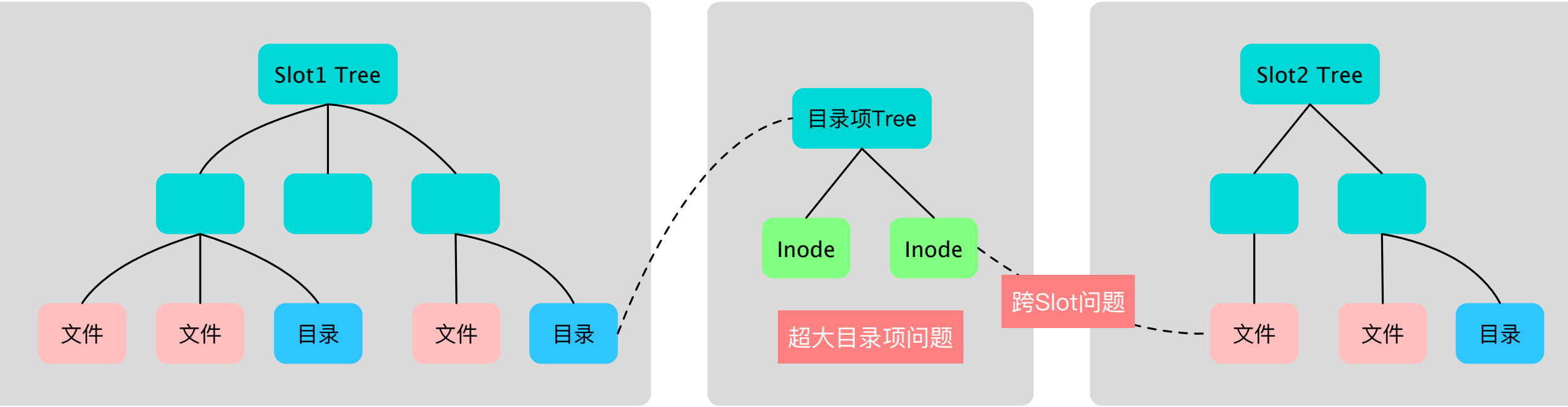
WFS二期元数据管理

采用MultiPaxos同步的纯内存存储



WFS二期元数据管理

采用MultiPaxos同步的纯内存存储



WFS二期元数据管理

采用MultiPaxos同步的纯内存存储

跨Slot操作（Create、Remove、Rename），以Remove为例

一、在InodeInfo中增加锁标志位，modifyTime作为加锁时间。

二、以原子方式去查询待删除目录是否为空，若为空则加一个锁（5每秒过期）返回。

三、到父目录中删除目录项。

四、真正删除目标目录，此时需要检查锁还在（即使已经过期）则允许删除。

五、任何修改操作都需要去检查锁，若锁存在并且未过期则不允许操作。

元数据管理优化后的效果：

一、平均一个元数据操作耗时下降了80%。

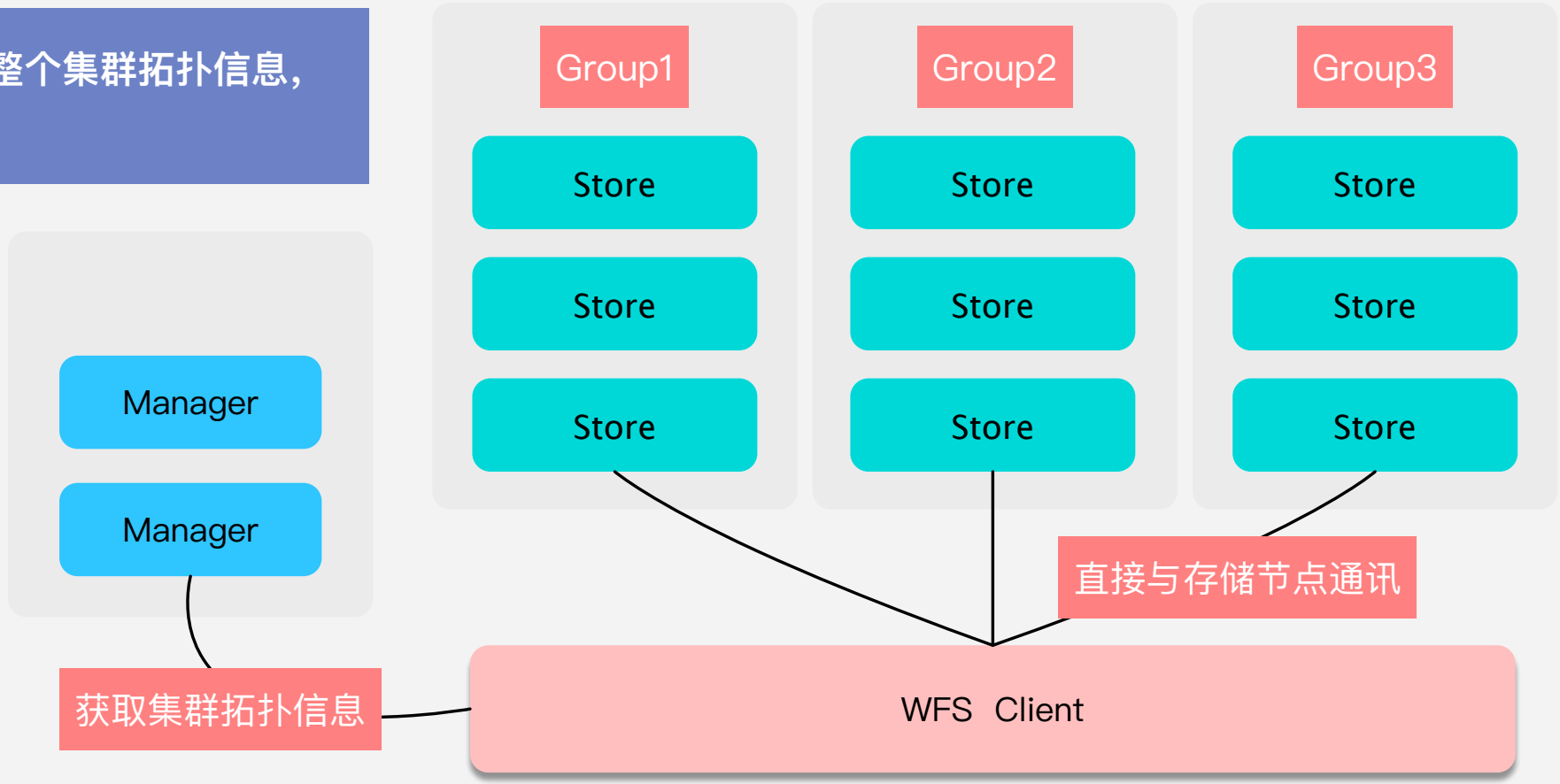
二、内存使用优化后，一亿个小文件一个MetaServer节点大概需要40GB内存。

三、代码逻辑复杂度大幅下降，代码量下降了60%。

WFS二期存储层

采用双层模型，简化网络IO路径

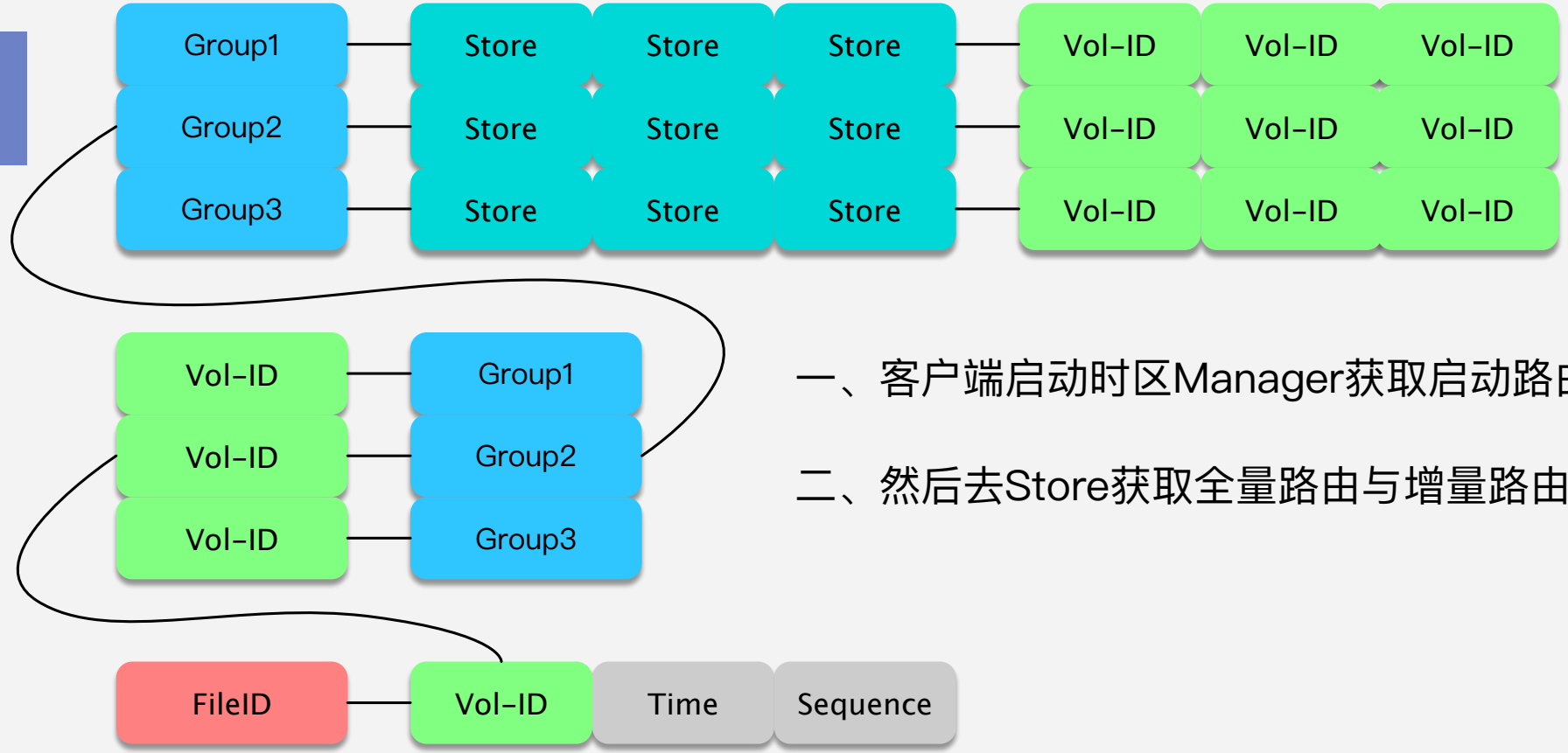
两层模型：客户端具有整个集群拓扑信息，直接与存储节点通讯：



WFS二期存储层

采用双层模型，简化网络IO路径

集群拓扑信息：
1000台机器集群拓扑信息4MB



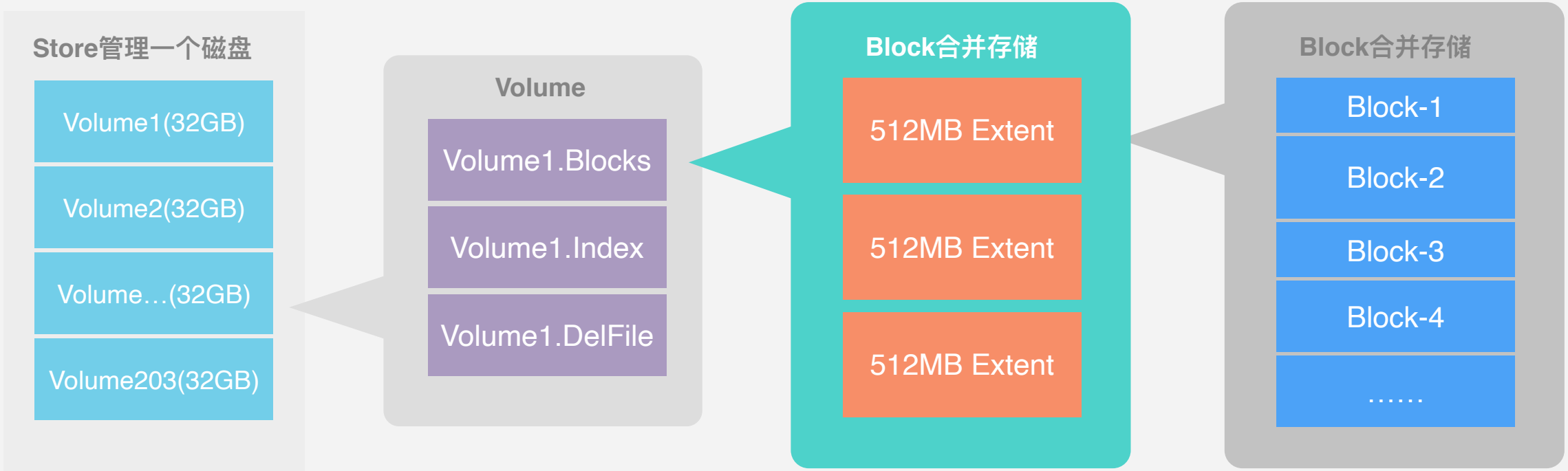
一、客户端启动时区Manager获取启动路由。

二、然后去Store获取全量路由与增量路由。

一期存储管理

采用对象存储作为存储层服务

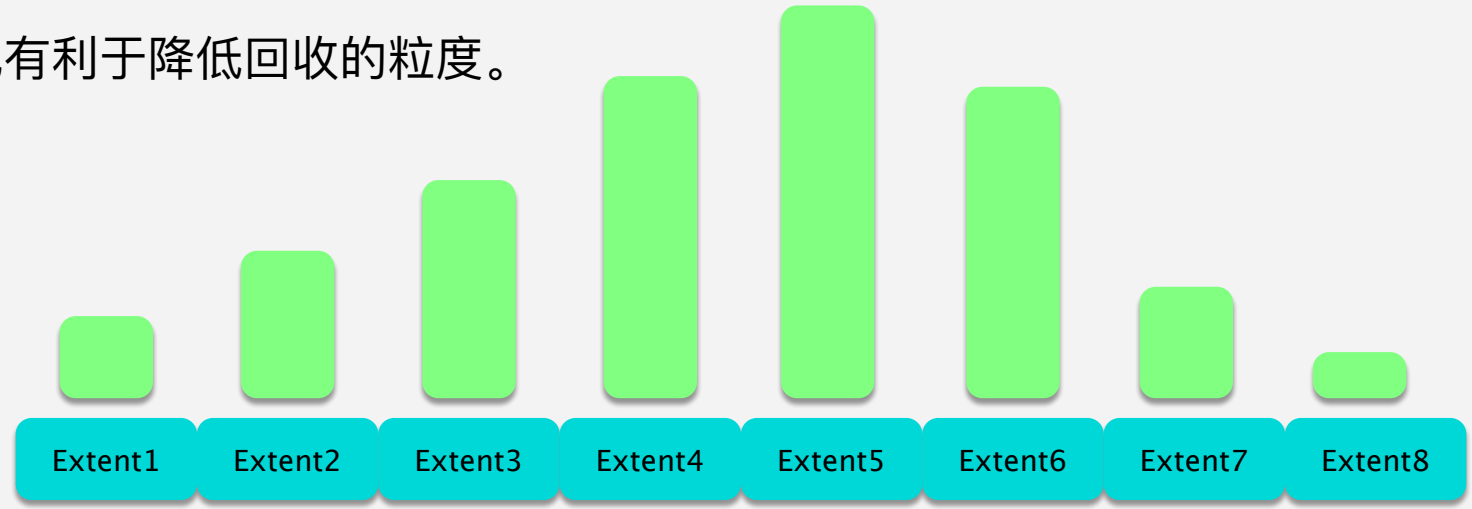
空间回收优化，增加一层Extent：



空间回收优化：

一、由于删除数据的时间属性较强，不同Extent被删除数据量不同，优先回收删除量较大的Extent。

二、即使删除数据没有时间属性，也有利于降低回收的粒度。



其他性能优化：随机写转换成顺序写

一、从轮询调度Volume修改成单个Volume写满调度下一个Volume。

二、从8字节对齐，修改成4KB对齐。

WFS二期元数据管理

采用MultiPaxos同步的纯内存存储

效果总结：

一、读写性能相较于老版本WFS提升4倍。

二、架构大幅简化，后期运维容易很多。

三、存储层空间回收速度大幅提升。

04 分布式读缓存系统

为了避免机械磁盘IO高时，延迟大幅升高导致性能降低，设计SSD存储的读缓存系统缓解读压力。

分布式读缓存设计

参考两层存储模型做设计

分布式读缓存设计要点：

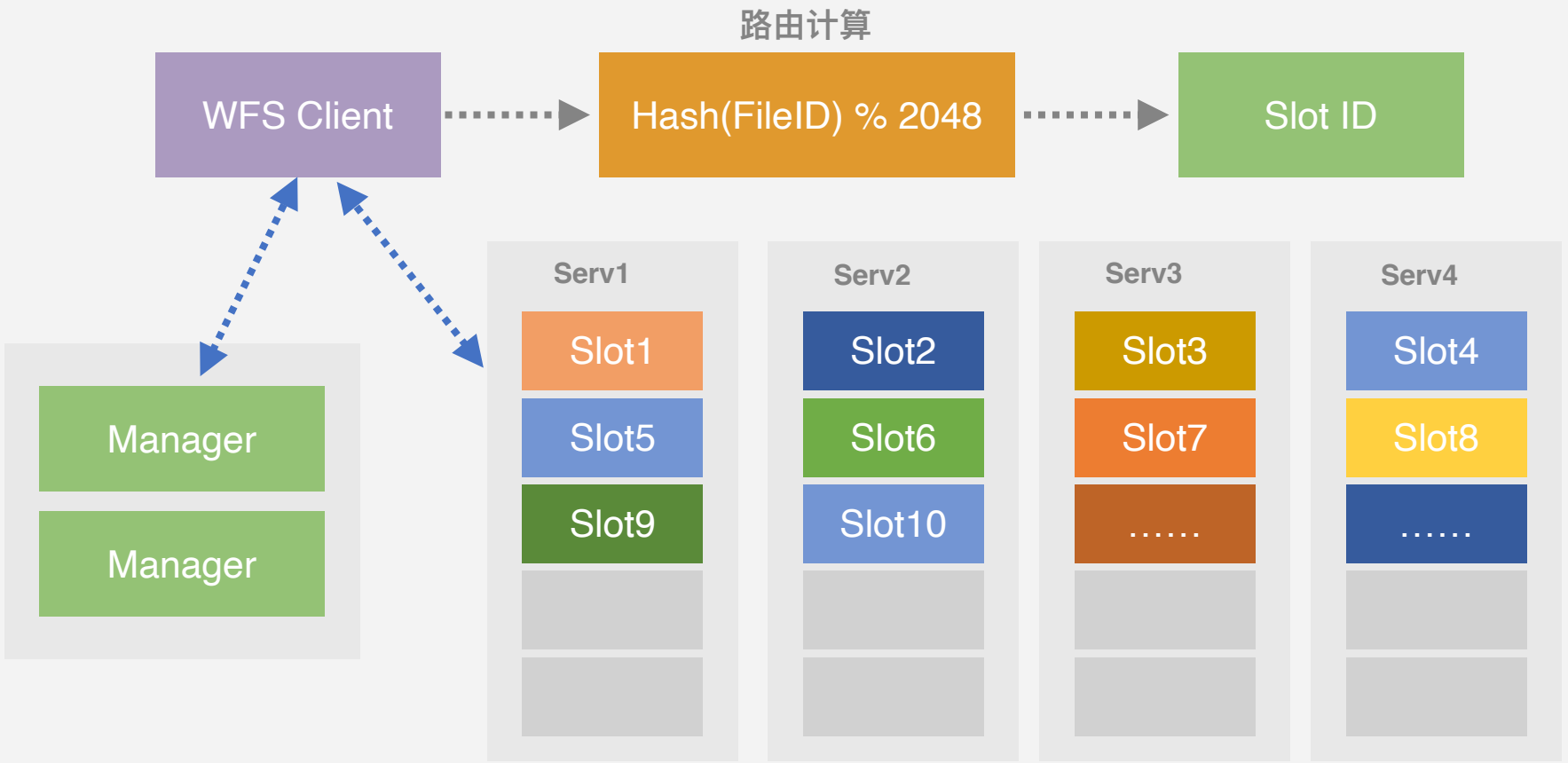
一、依然采用两层存储模型，客户端缓存所有路由信息，直接与存储节点通讯。

二、存储节点采用LRU模型淘汰冷数据。

三、每份数据只有一个备份，不用考虑数据容灾。

分布式缓存设计

集群管理与路由设计



分布式缓存设计

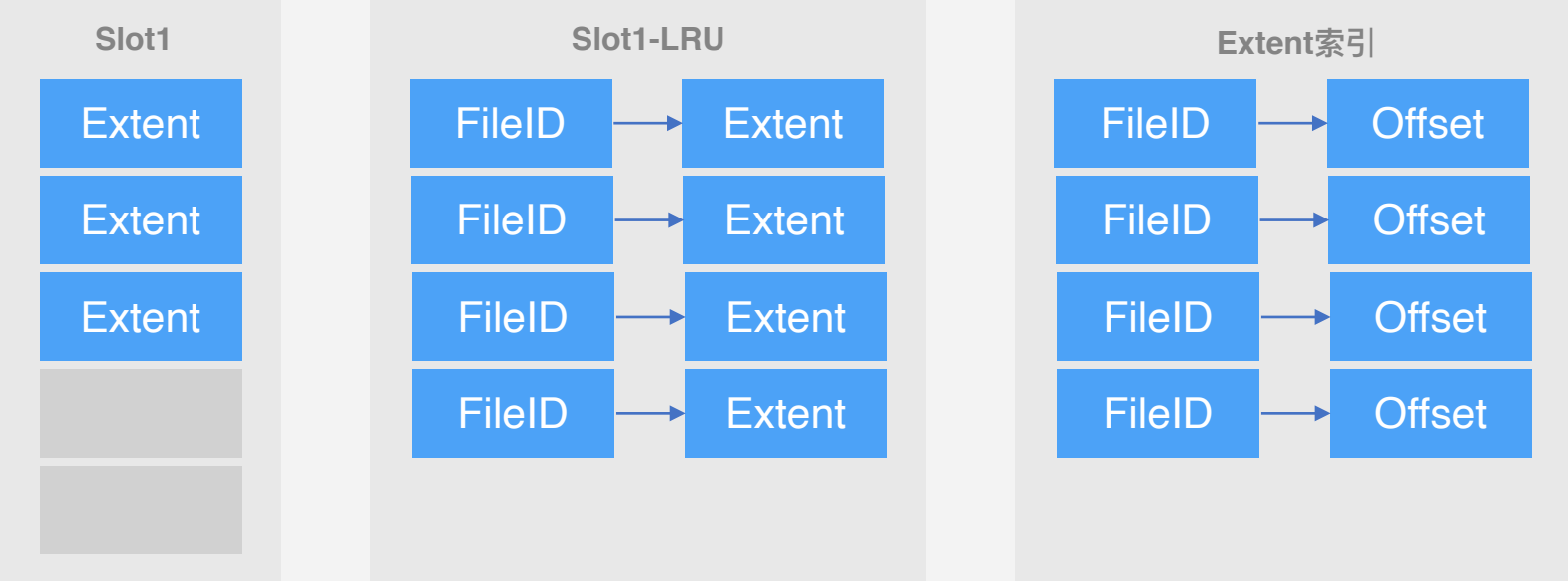
存储节点LRU与文件管理

存储节点设计要点：

一、Slot有Extent组成Extent有Block+Index文件组成

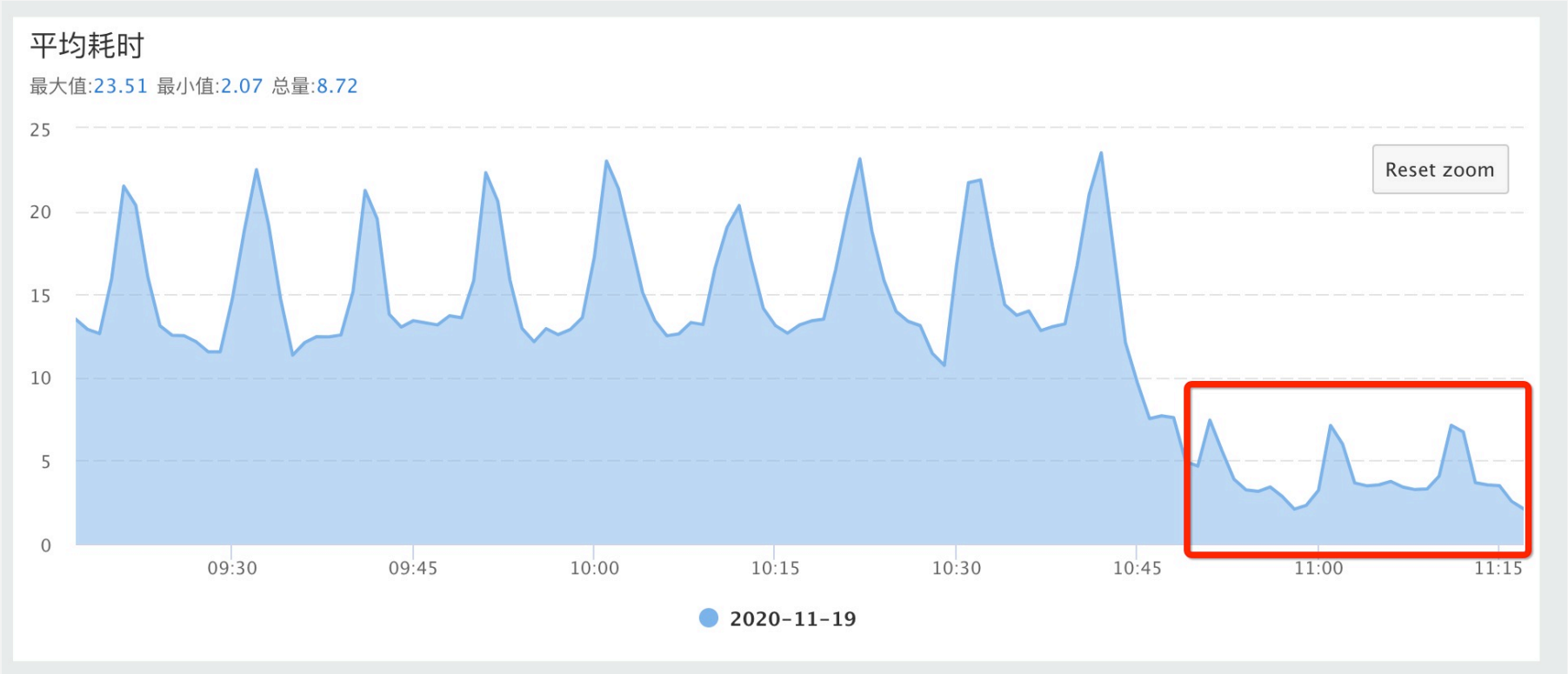
二、Slot内存中有一个LRU

三、Exent内存中有索引



分布式缓存设计

存储节点LRU与文件管理




SACC
THANKS