Curve

Curve核心组件 - SnapShotCloneServer

D I G I T A L S A I L

许超杰

网易数帆存储团队

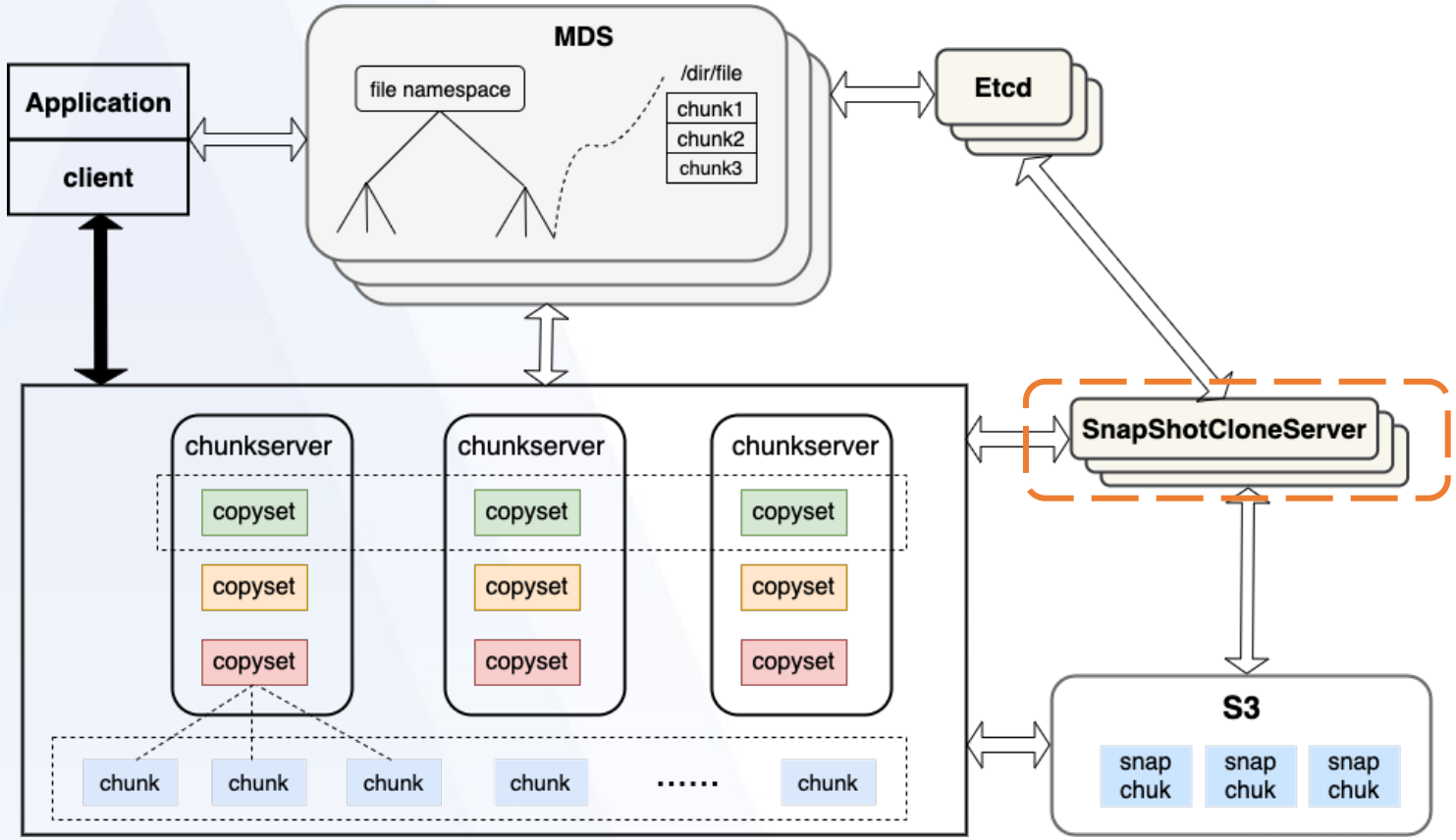
Curve

DIGITAL SAIL

目录

CURVE基本架构

- 元数据节点 MDS
  - 管理和存储元数据信息
  - 感知集群状态，合理调度
- 数据节点 Chunkserver
  - 数据存储
  - 副本一致性，raft
- 客户端 Client
  - 对元数据增删改查
  - 对数据增删改查
- **快照克隆服务器**
  - **快照**
  - **克隆**



快照和克隆的特点

Curve

- 快照的定义

快照是云盘数据在某个时刻完整的只读拷贝，是一种便捷高效的数据容灾手段，常用于数据备份、制作自定义镜像、应用容灾等。

- 快照的特点
  - 转储到s3对象存储
  - 异步转储快照，底层使用copy-on-write技术，读写不影响转储
  - 增量转储，第一次全量转储s3之后，后续只需转储增量部分
  - 高可用，快照任务中断自动拉起继续转储

快照和克隆的特点

Curve

- 克隆的定义
  - 克隆是指从卷复制出卷的功能，提供快速的复制卷的能力。
  - 这里的克隆还包括从快照回滚的功能

- 克隆的特点
  - 支持Lazy和非Lazy两种模式克隆
  - 支持从快照克隆和从镜像（卷）克隆
  - 支持从快照回滚
  - 高可用，克隆任务中断自动拉起继续克隆

快照克隆服务器架构

Curve

HttpService:

- 基于brpc提供restful API的对外http接口

SnapshotService & CloneService:

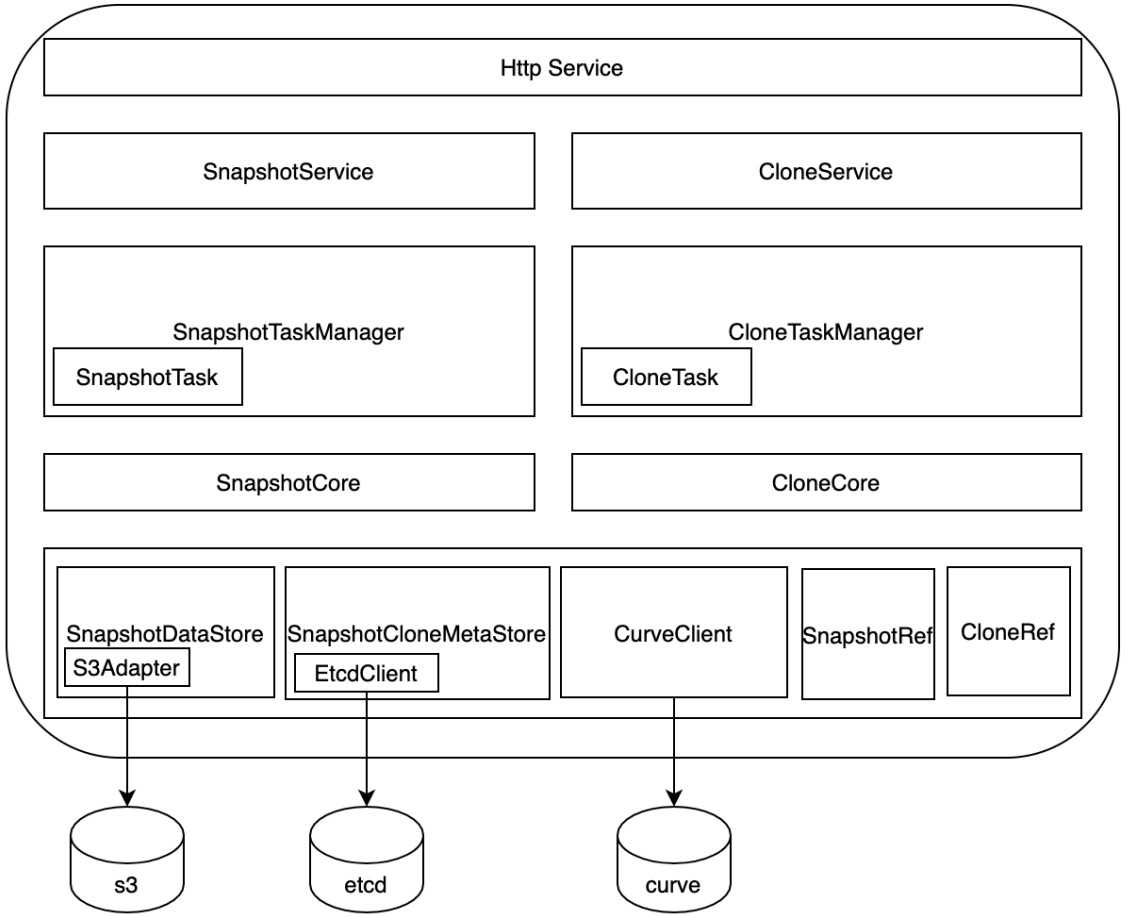
- Serivce层面区分上层请求为同步接口调用，还是异步接口调用，同步接口调用直接调用Core层接口实现功能，异步接口创建Task，并交由TaskManager调度。

SnapshotTaskManager & CloneTaskManager:

- 任务管理层负责调度SnapshotTask和CloneTask，并向上提供如cancel task等功能。

SnapshotCore & CloneCore:

- 快照克隆核心模块，负责向下调用DataStore，MetaStore等底层模块，实现快照和克隆的具体功能。



快照克隆服务器架构

Curve

SnapshotDataStore:

- SnapshotDataStore负责管理快照转储的数据块，通过调用S3Adaptor（一个封装了s3 client的接口层）与S3交互，存取s3中的对象。

SnapshotCloneMetaStore:

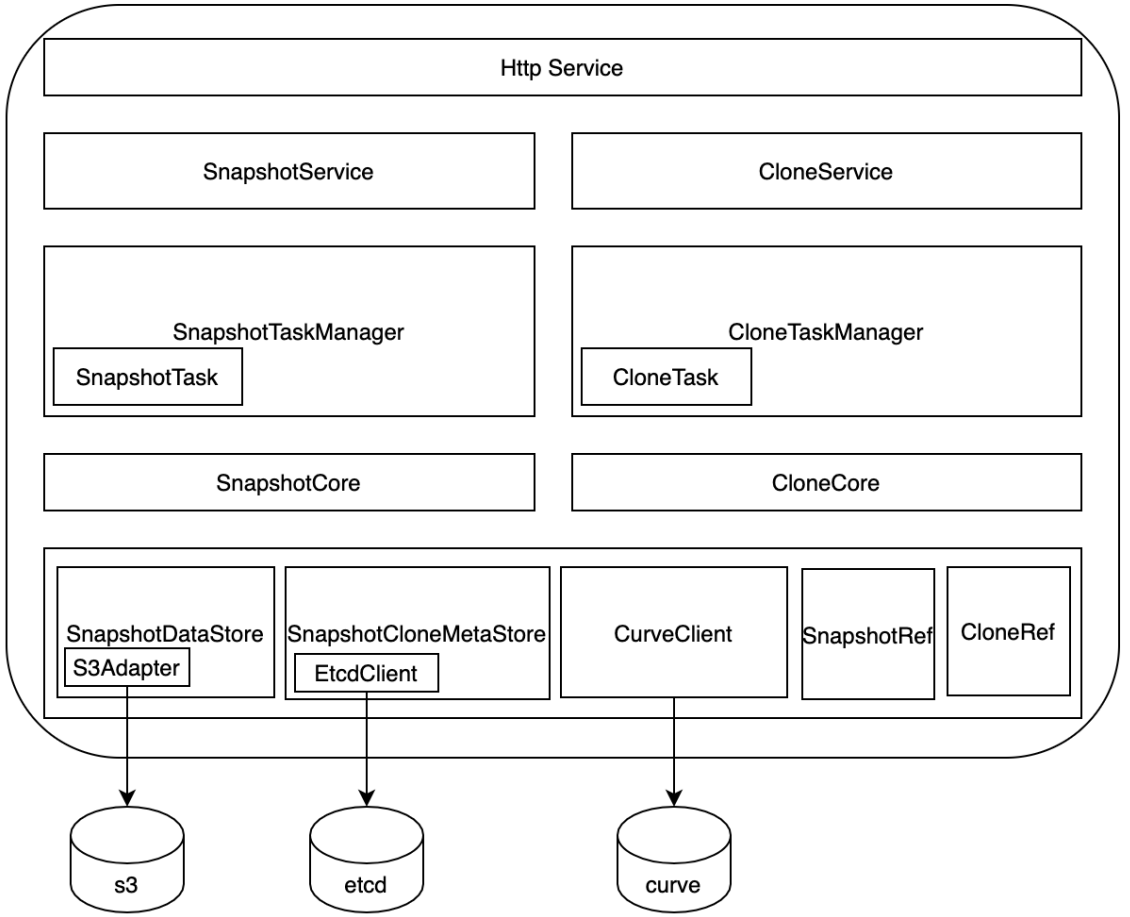
- SnapshotCloneMetaStore负责管理快照和克隆任务等元数据，通过调用etcdclient，与etcd存储交互，存取etcd中的快照和克隆元数据。

CurveClient:

- CurveClient封装了Client接口，负责与MDS和ChunkServer交互。

SnapshotRef & CloneRef:

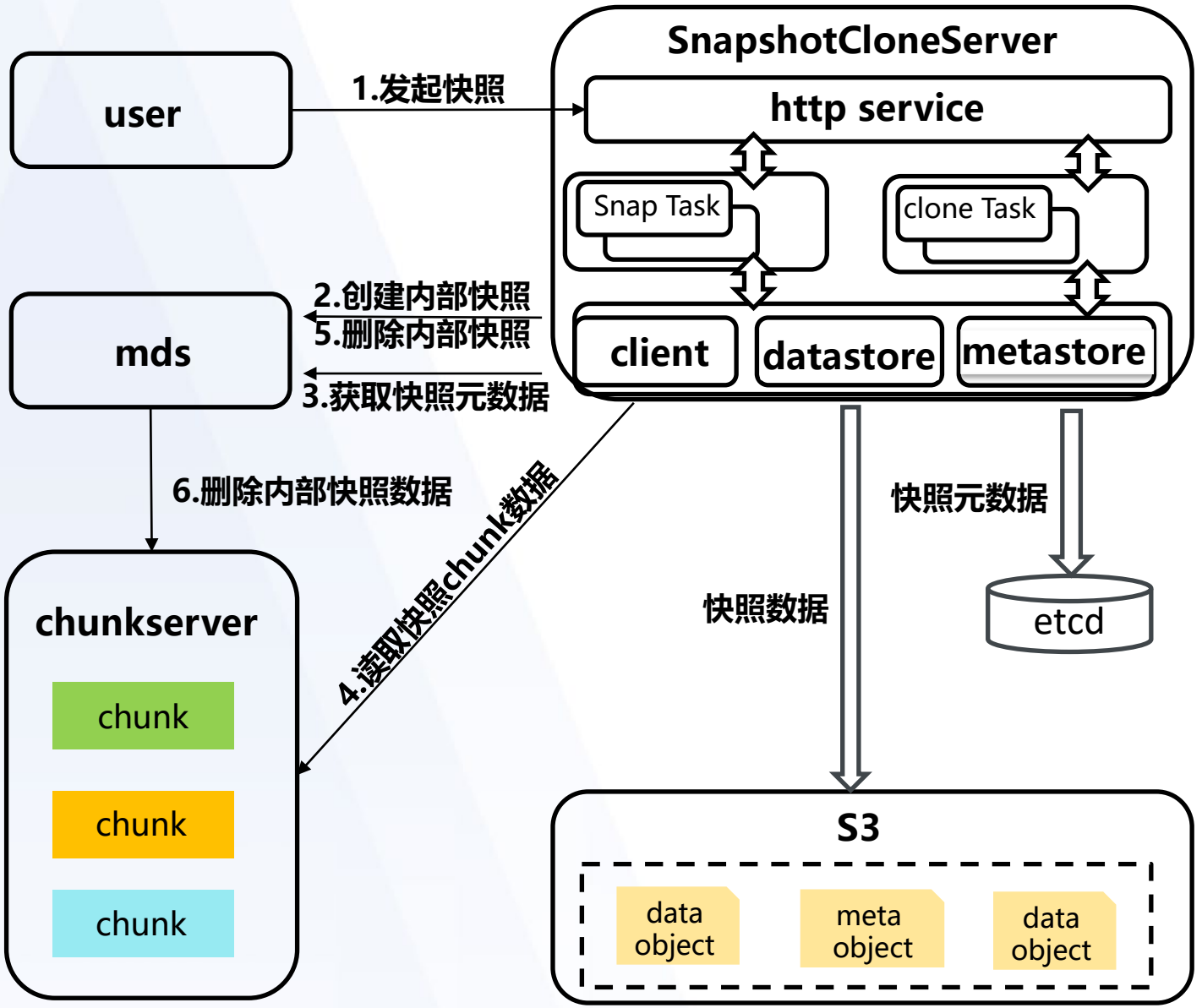
- 负责管理快照和克隆源卷的引用计数。



快照总体流程

快照流程：

- 1.用户发起快照，生成快照任务，并持久化到etcd，开始执行快照任务。
- 2.在curve中创建内部快照，并返回快照信息，然后将快照信息更新到etcd。此时，即返回用户快照成功，可以进行读写。
- 3.向mds查询快照的元数据，转储快照元数据块metaObject。
- 4.根据快照元数据信息，转储快照数据块dataObject。
- 5.调用mds接口，移除curve内部的快照。
- 6.mds调用chunkserver接口，删除内部快照数据

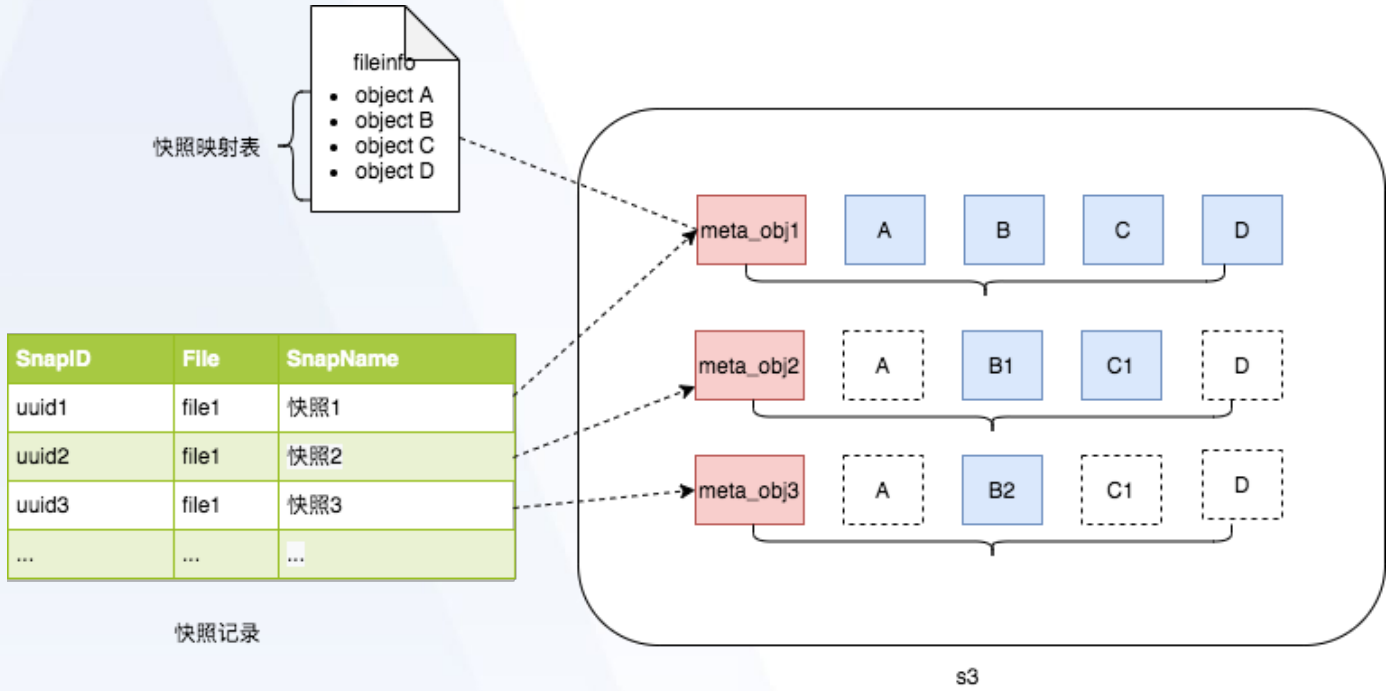


快照的元数据和数据组织

Curve

Etcd中的快照元数据:

字段	类型	说明
uuid	string	快照唯一Id
user	string	所属用户
fileName	string	快照目标卷名
snapshotName	string	快照名
seqNum	uint64_t	快照版本号
chunkSize	uint32_t	chunk的size
segmentSize	uint64_t	segment的size
fileLength	uint64_t	卷的大小
time	uint64_t	快照创建时间
status	enum	快照的创建状态



快照的元数据和数据组织

Curve

MetaObject：

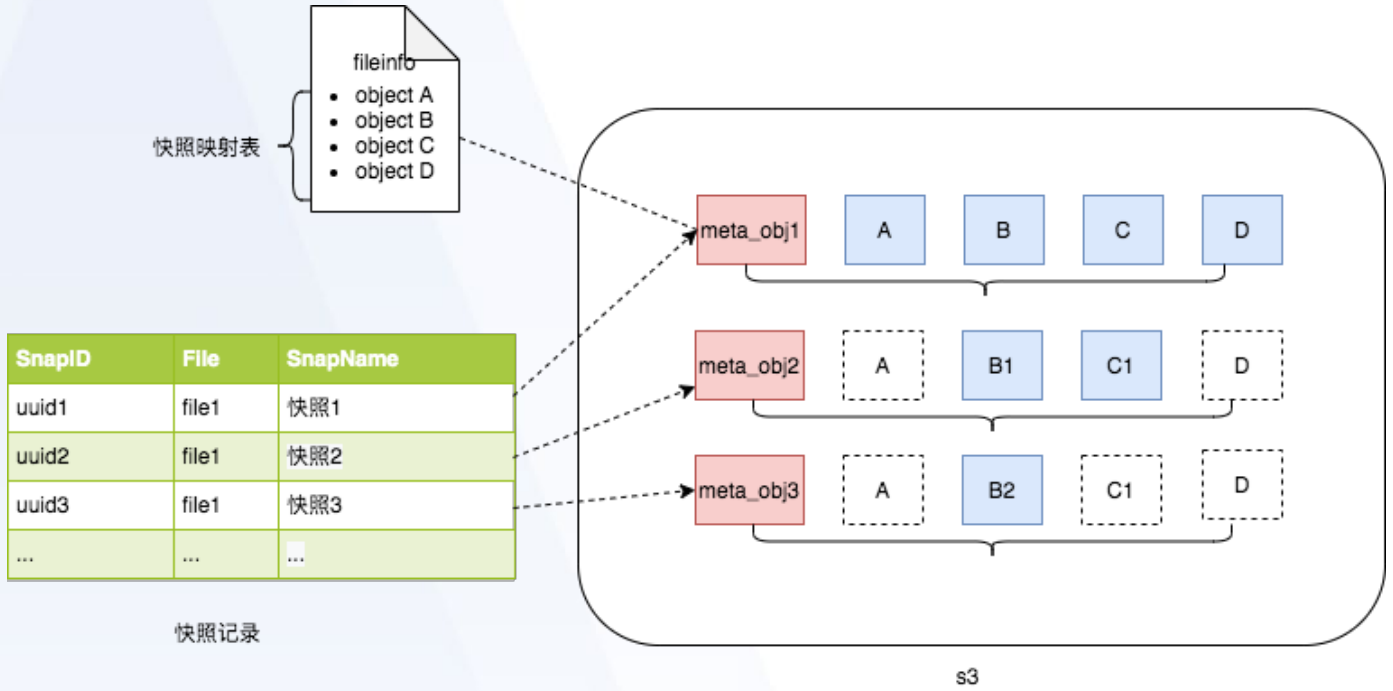
- fileInfo 快照目的卷的卷名等信息
- chunkMap 快照chunk映射表

DataObject：

- 保存完整的chunk数据，大小为一个Chunk的大小，即16MB

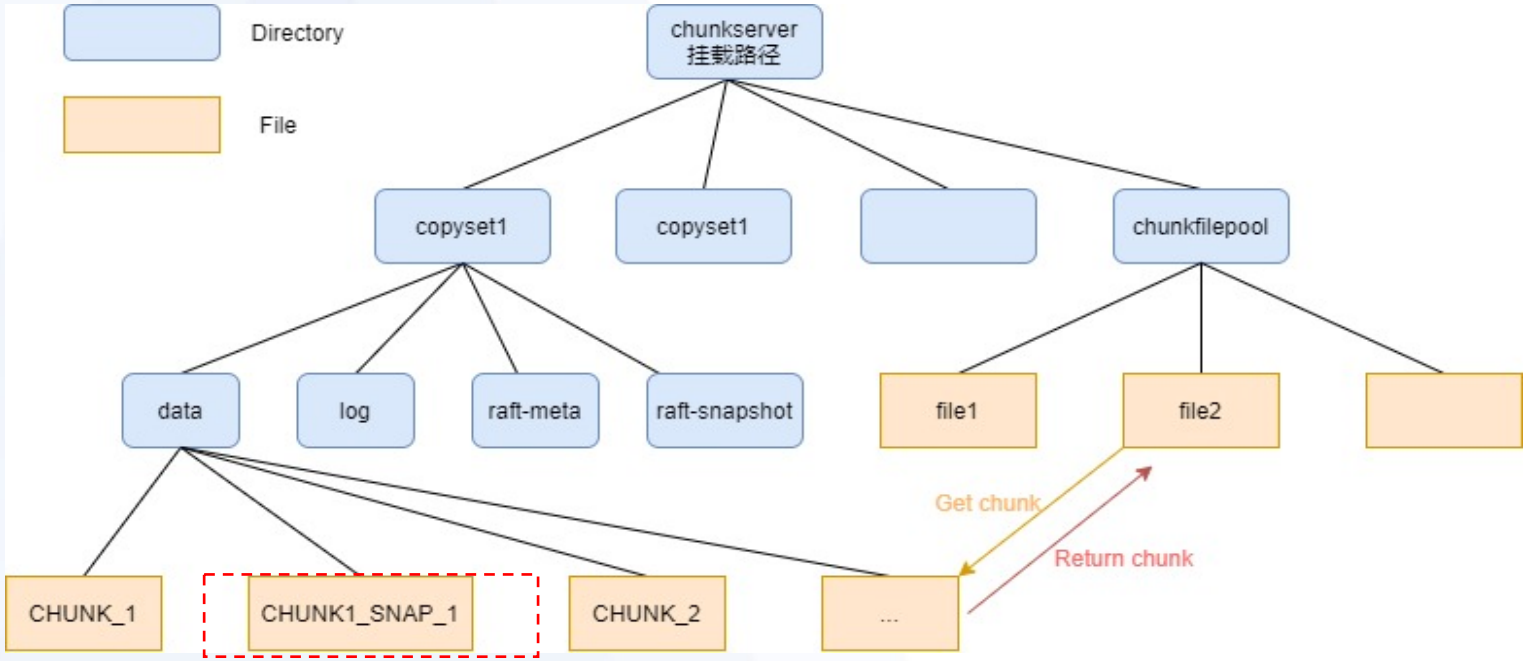
增量转储原理：

- 打快照时读取当前目标卷的所有快照的全部metaObject
- 根据本快照的chunk映射表，判断当前的快照chunk是否需要转储



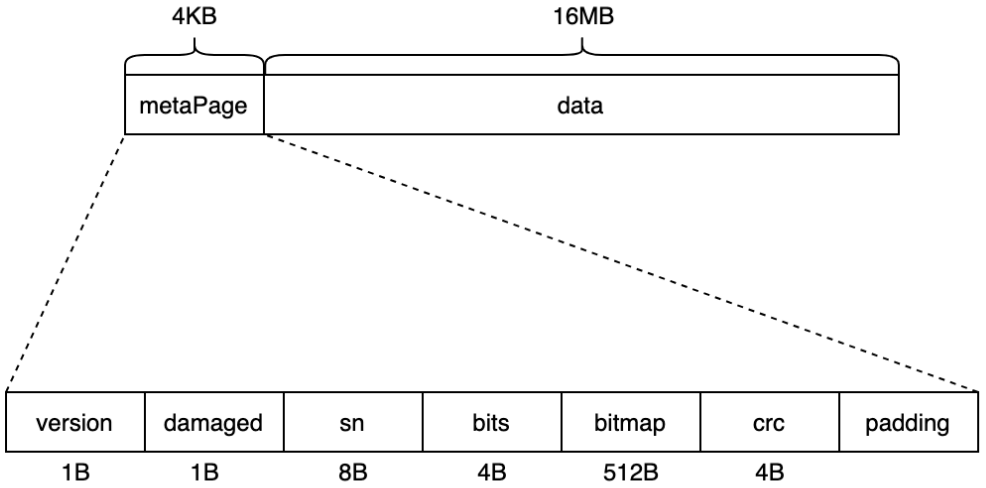
快照在CHUNKSERVER上的数据组织

- 快照chunk和普通chunk，都是ChunkServer上的ext4文件系统中的文件，称为SnapFile和ChunkFile；
- SnapFile 与ChunkFile是同构的,都来自ChunkFilePool；
- SnapFile与ChunkFile在同一个目录；
- SnapFile的命名方式为“chunk_” + ChunkId + “_snap_” + seqNum的形式，以区别于ChunkFile。



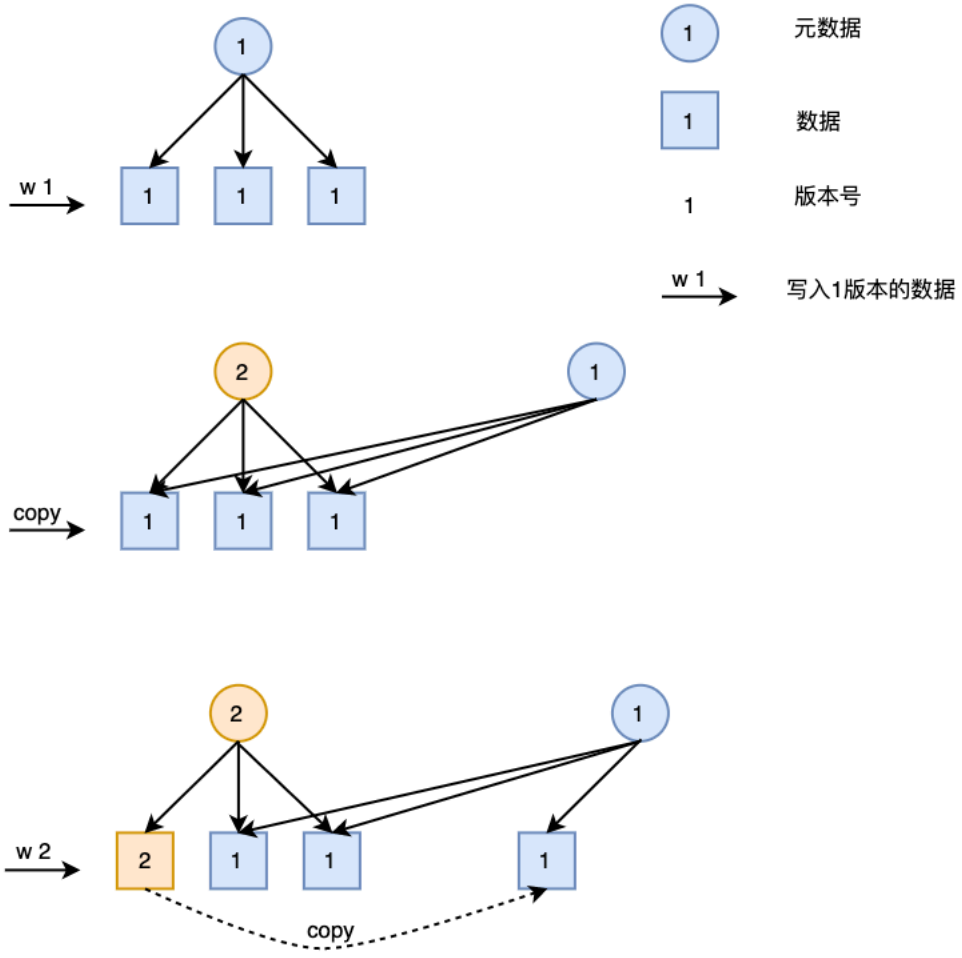
CHUNKSERVER端快照实现-SNAPFILE

字段	类型	说明
version	uint8_t	文件格式协议版本号
demaged	bool	损坏标记
sn	uint64_t	快照版本号
bits	uint32_t	位图的位数
bitmap	char[]	位图
crc	uint32_t	上述字段的crc校验码
padding	/	填0，以补足4KB

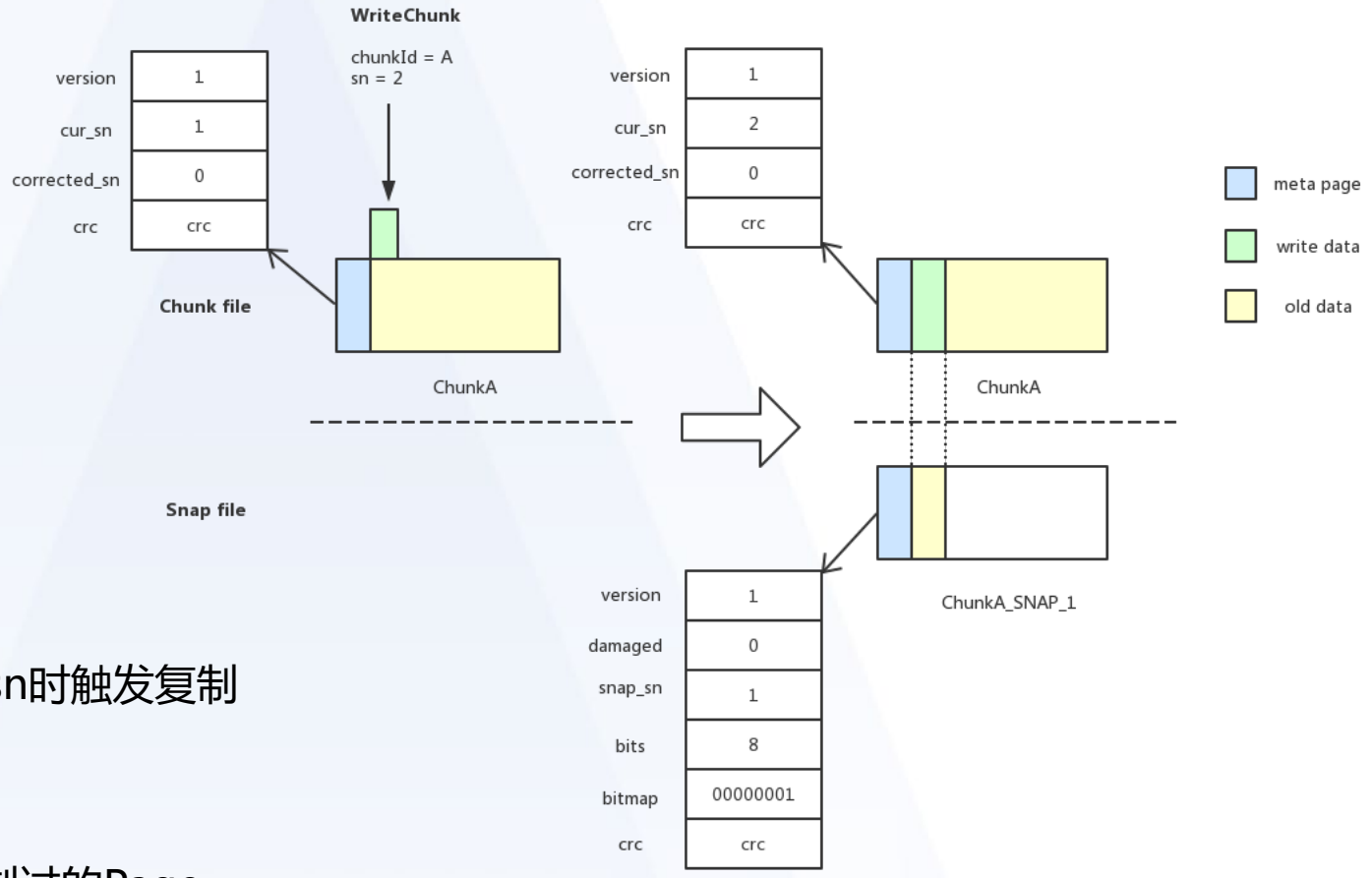


CHUNKSERVER端快照实现-写时复制原理

- 写时复制通常使用版本号实现
- 复制时仅复制元数据，并增加版本号
- 写入时，先复制要写入的数据块，然后再写入



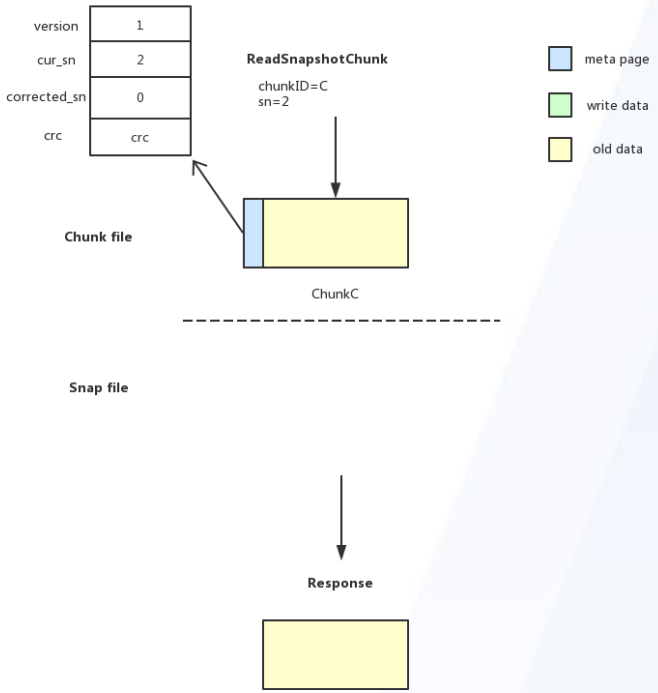
CHUNKSERVER端快照实现-写时复制



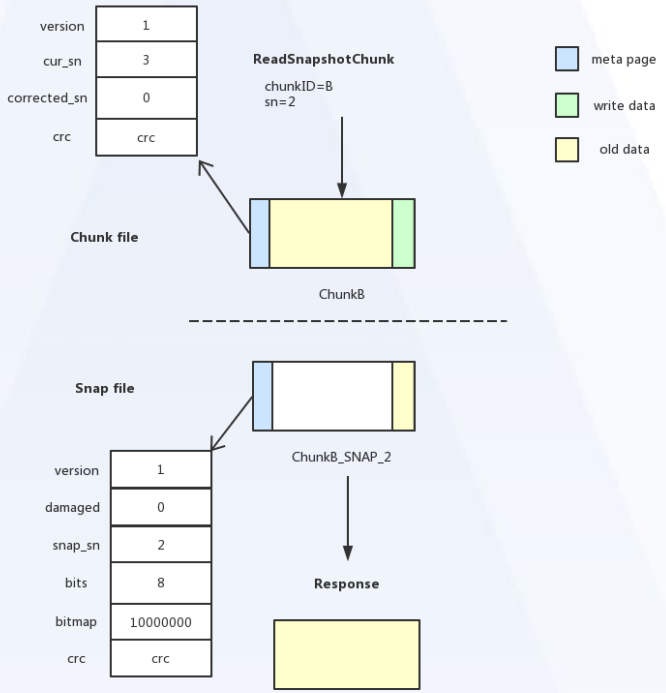
- 使用copy-on-write
- 当前写请求的sn > chunk的cur_sn时触发复制
- 拷贝的单位是一个Page，即4KB
- 使用snapfile中的bitmap标记复制过的Page

CHUNKSERVER端快照实现-转储内部快照

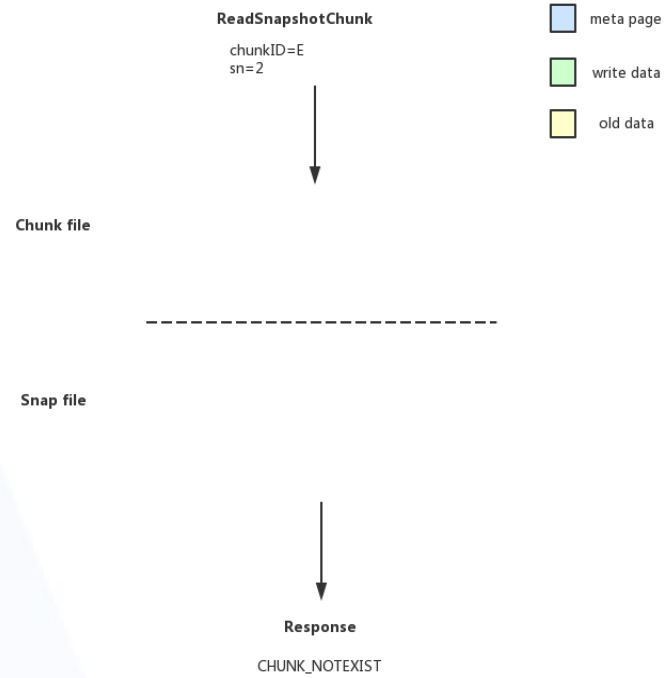
转储内部快照，即读内部快照的三种情况：



a) 打快照后未写过，未触发cow，
无snap file产生，直接读取chunk file



b) 打快照后写过，触发了cow，
有snap file， 合并读取

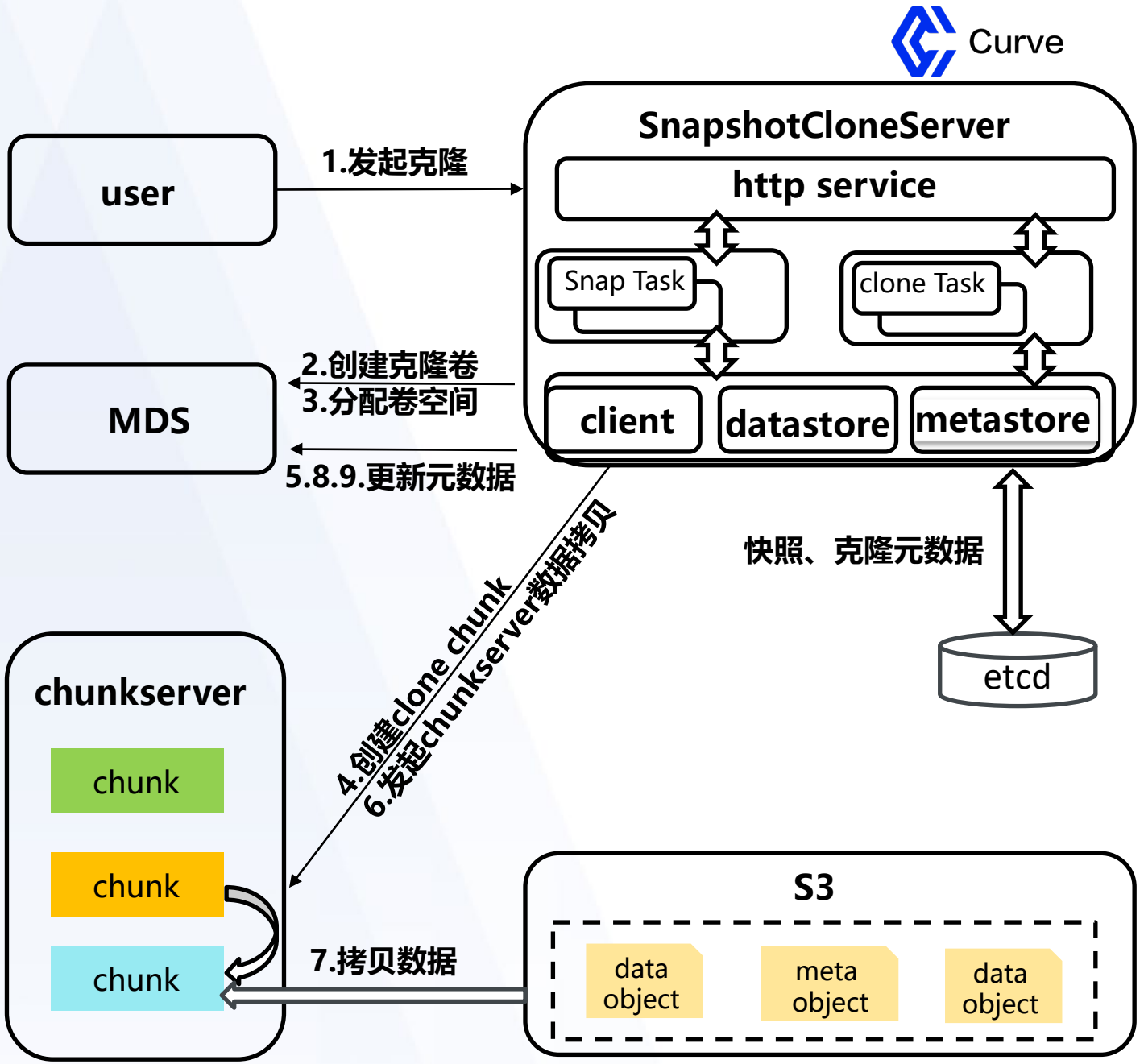


c) 卷从未写过，
两者都没有，返回NOTEXIST

克隆总体流程

克隆流程：

- 1. 用户发起克隆，生成克隆任务，并持久化任务元数据到etcd，开始执行克隆任务。
- 2. 调用mds接口创建clone卷信息，该clone卷是个临时卷，位于/clone目录下。
- 3. 调用mds接口为目的卷分配空间。
- 4. 根据目的卷的分配信息，调用chunkserver接口创建CloneChunk。
- 5. 更新克隆卷状态为metaInstalled。
- 6. 发起ChunkServer数据拷贝
- 7. ChunkServer从克隆源拷贝数据。
- 8. 将卷从临时卷rename为克隆目标卷名。
- 9. 更新克隆卷状态为Cloned。



克隆卷状态变化

Curve

Cloning:

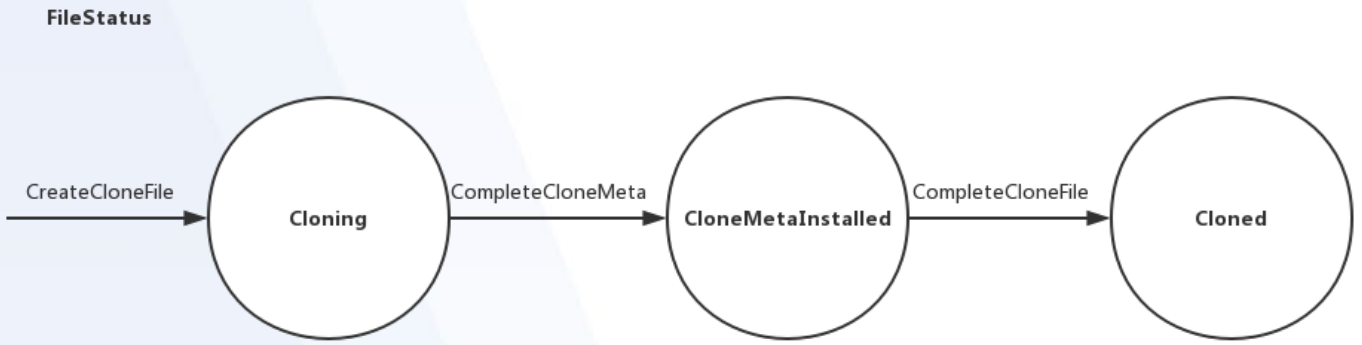
- 初始状态；
- 正在安装元数据或拷贝数据中；
- 用户不可见。

MetaInstalled:

- 元数据安装成功；
- Lazy方式下可见，用户可用；
- 非Lazy不可见。

Cloned:

- 数据拷贝完成；
- 可提供所有服务。



LAZY 克隆

Curve

Lazy克隆

较快，秒级克隆：

MetaInstalled状态可用，即完成元数据安装，就从临时目录rename，用户可见。

Lazy Alloc Chunk，利于超售：

Lazy克隆不直接分配chunk，而是等到client来写时才分配chunk

额外接口：

不进行数据复制，而是提供额外的Flatten接口，完成数据复制。

适用场景：

适用于从镜像快速创建云主机场景

非Lazy克隆

较慢，分钟级：

Cloned状态可用，即完成整个数据克隆，才从临时目录rename，用户才可见。

无Lazy Alloc chunk：

安装元数据时即分配好chunk。

无额外接口：

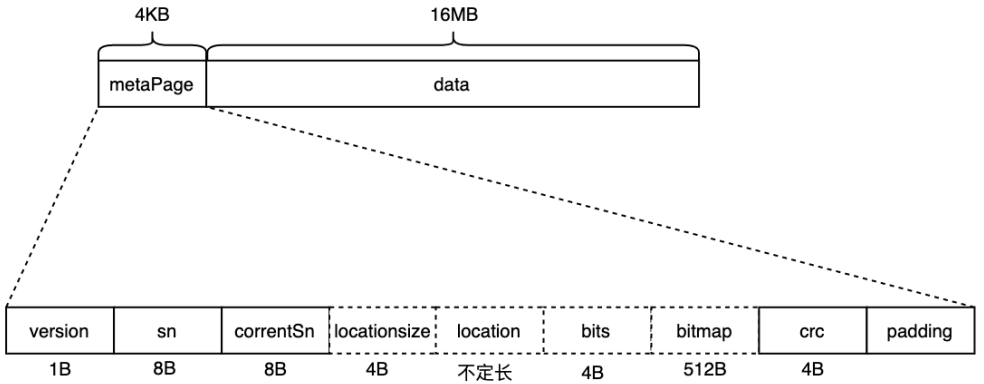
无需Flatten接口。

适用场景：

适用于从云主机或快照创建镜像

CHUNKSERVER端克隆实现-CHUNKFILE

字段	类型	说明
version	uint8_t	文件格式协议版本号
sn	uint64_t	chunk文件的版本号
correntSn	uint64_t	chunk的修正版本号
locationSize	size_t	可缺省，当前为CloneChunk时表示location占用的字节数
location	char[]	可缺省，当前为CloneChunk时表示克隆源字段
bits	uint32_t	可缺省，当前为CloneChunk时表示bitmap的位数
bitmap	char[]	可缺省，位图表
crc	uint32_t	上述字段的crc校验码
padding	/	填0，以补足4KB



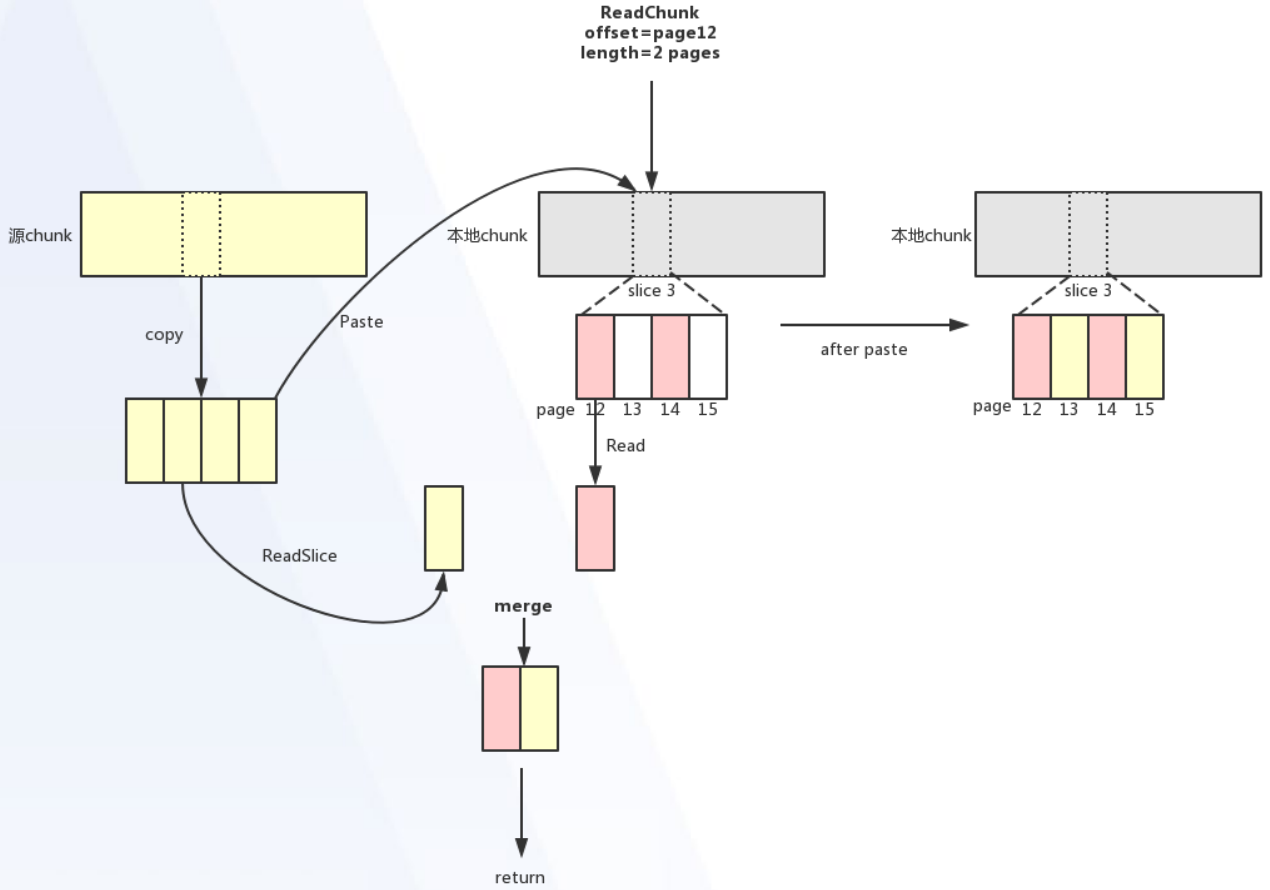
location定义为A@B的形式:

- 如果源卷在s3上，则location格式为objectName@s3，例如: objectxxx@s3
- 如果源卷在curve内部，则location格式为fileName:offset@cs，例如: /test1:0@cs

CHUNKSERVER端克隆实现-读时复制原理

读时复制原理:

- 使用chunkfile的bitmap来标记写过的Page，一个Page大小为4KB
- 读请求到来时，根据bitmap中的信息，
  - 对于已写过的区域，从本地chunk file读
  - 对于未写过的区域，从远端源chunk file读
- 之后，将两者合并返回。
- 同时把源chunk读到的数据异步写入到本地chunk，并标记bitmap，这个过程称之为PasteChunk



CHUNKSERVER端克隆实现-读时复制实现

Curve

需要从源chunk读取：

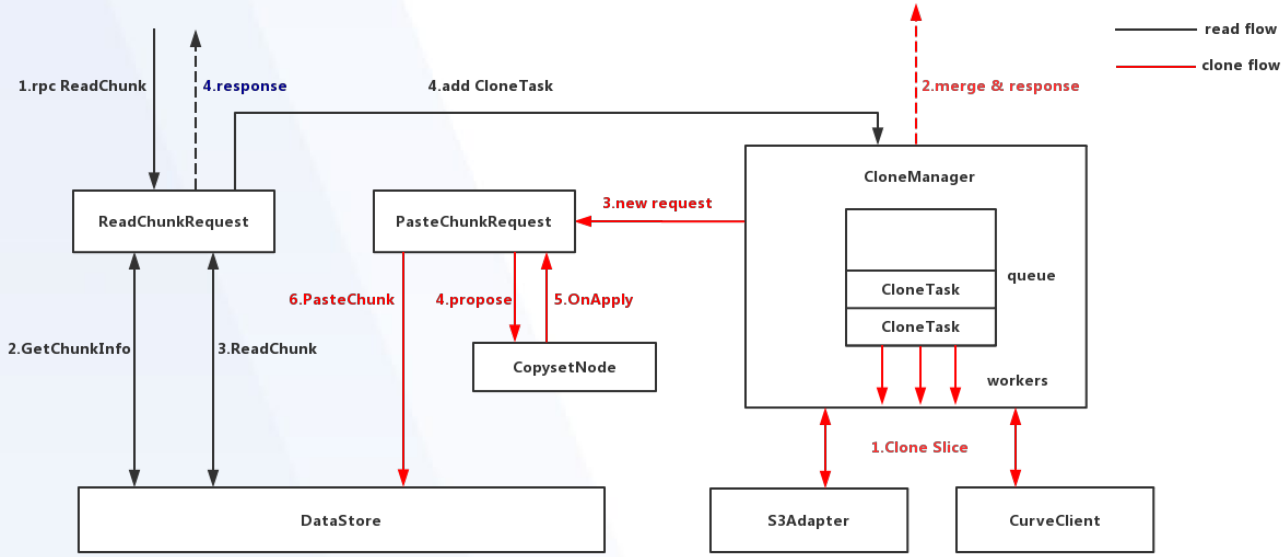
- 判断是非clone chunk 或者读取的区域已经被拷贝过（根据bitmap）
- 那么，直接读取

不需要从源chunk读取:

- 判断是Clone Chunk且需要读取的区域还未被拷贝过
- 那么，生成CloneTask，交给CloneManager
- Read & Merge，
- 同时生成PasteChunkRequest

异步完成源chunk读到的数据写入到本地chunk：

- 异步完成，不在IO主路径
- 类似与发起一个写请求，经CopysetNode走一致性协议完成
- 完成写入后，并标记bitmap，如果全部写过，则取消clone chunk标记。



欢迎大家参与CURVE项目！

- github主页： https://opencurve.github.io/
- github代码仓库： https://github.com/opencurve/curve
- 系列讲座：https://space.bilibili.com/700847536/channel/detail?cid=153949

THANK YOU

D I G I T A L S A I L

扫码即可关注