ZeroDrive

同步网盘文件系统

存储技术基础 陶天骅 陈张萌 2020.6.3

目 录

灵感来源

- 网盘是大家所熟悉的，比如有OneDrive、iCloud等。
- 目的是使不同设备上的文件操作尽可能同步，就像在同一个硬盘上一样。

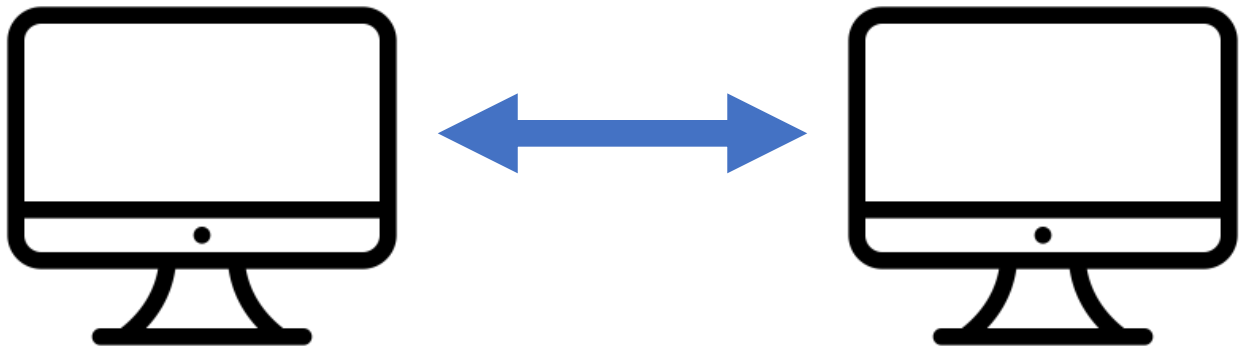
设计概述

- ZeroDrive是基于Linux，使用FUSE 3实现的网络同步盘文件系统
- 仅使用纯C++和FUSE库
- 支持多个用户共享操作
- 使用日志记录文件操作
- 使用 TCP socket 通信
- 可以灵活选择使用方式

设计优势

灵活部署

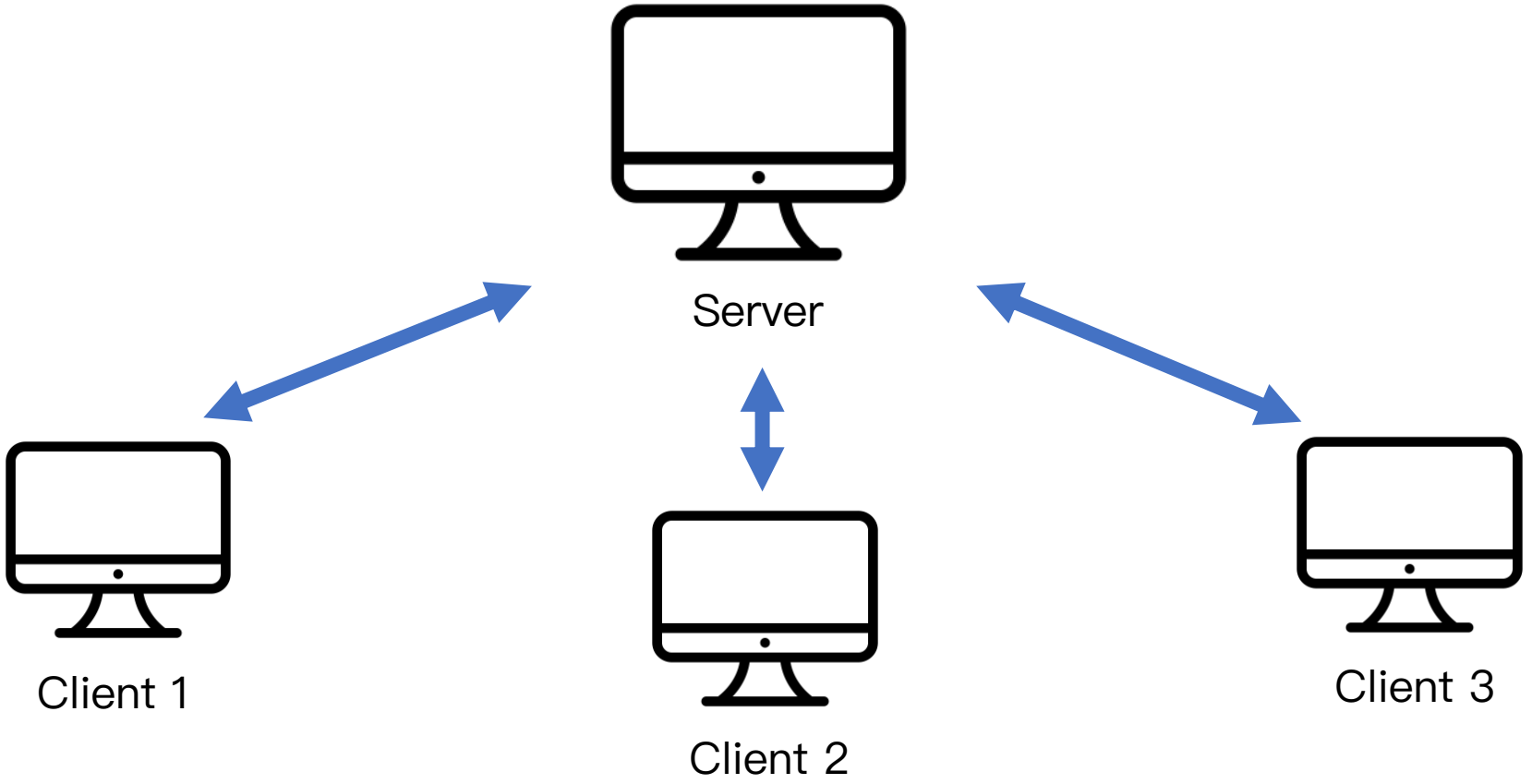
- 分为 Server 和 Client
- 可以用于普通两台电脑之间的共享



设计优势

灵活部署

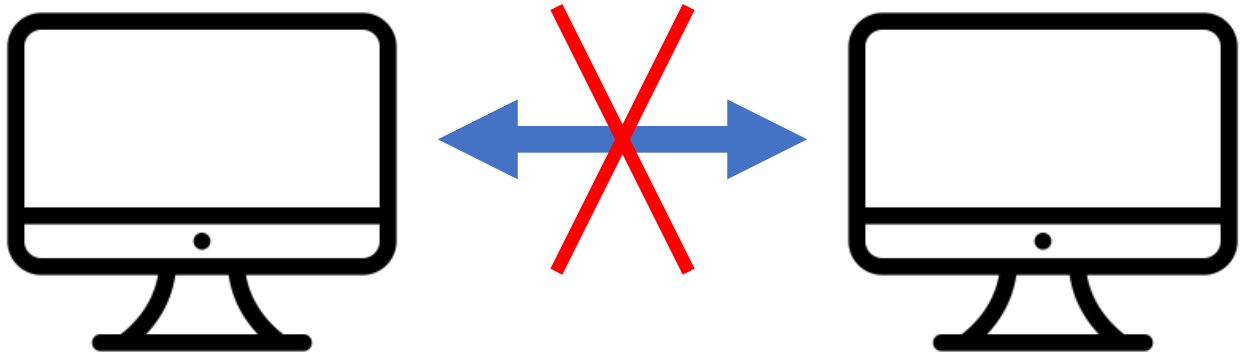
- 也可以用多客户端模式
- 文件时刻保持最新，保存多份副本
- 一处更改，处处更新
- Server 端可以是普通电脑，也可产生数据



设计优势

灵活部署

- 若不联网时，各自就可以当一个正常的文件系统使用
- 联网之后选取差异文件进行同步



视频Demo

存储技术基础 陶天骅 陈张萌 2020.6.3

视频演示

操作序列

- 启动Server
- 启动Client
- 往 Server 复制文件，Client 获得更新
- 往 Client 复制文件， Server 获得更新
- 在 Server 中修改文件名
- 往 Client 复制含文件的文件夹
- 在 Client 中删除文件
- 往 Client 复制嵌套文件夹
- 往 Client 复制大文件，检查传输可靠性
- 打开 Client 2，初始为空，过一会同步一致
- 在 Client 2 新增文件，会出现在 Server 和 Client 1

工作机制

PULL

PUSH

- Client 每隔一段时间（如10秒）会向 Server 发送两种消息：
  - PULL：请求下载过去一段时间的更新
  - PUSH：将本地过去一段时间的更新上传
- 因此并不在每次修改之后，立即传输
- 需要计算
  - 过去一段时间的更改
  - 将更改进行合并

工作机制

PULL

- PULL：
- Server 保存一个uint64的时间戳server_stamp记录当前的同步状态
- Client 保存一个uint64的时间戳last_sync记录上次同步时Server的状态

PUSH

- PUSH：
- 将所有新的未同步的修改文件操作（如CREATE，WRITE，MKDIR）记录到一个队列中，标记为unstaged
- 稍后记录到日志文件中

工作机制

PULL

PUSH

- 每次PULL的时候：

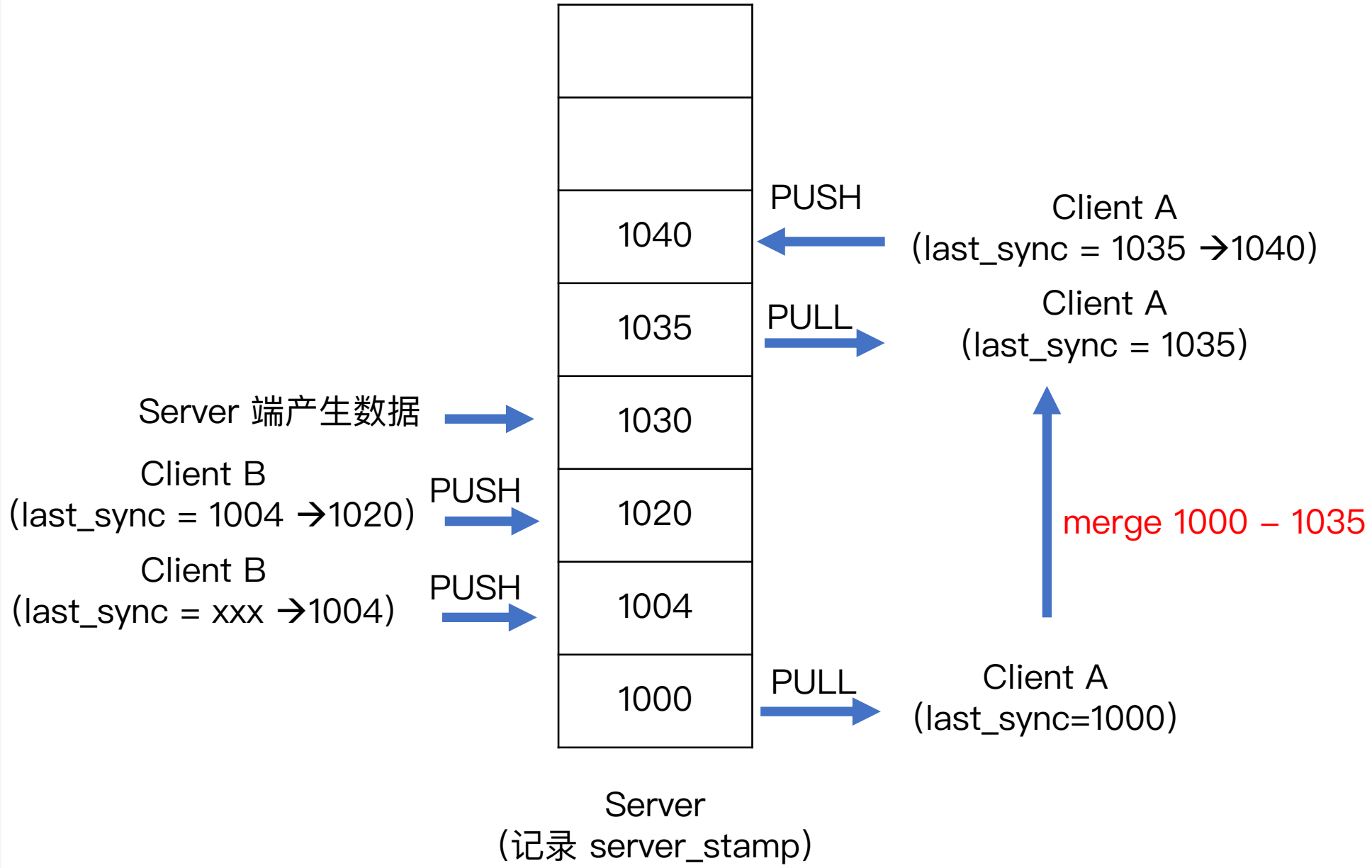
```
if (last_sync == 0)
    完全新的 Client，下载全部
else if (server_stamp == last_sync)
    已是最新
else if (server_stamp > last_sync)
    下载从last_sync到server_stamp 的变化
```

- 每次PUSH的时候：
  - 要先 PULL，保持最新
  - 更新 Server 的 server_stamp

工作机制

PULL

PUSH



日志文件

- 目录格式

```
zerodrive_internal_s.journal
├── 1591126147293
├── 1591126166413
├── 1591126183295
├── 1591126201556
├── 1591126226691
├── 1591126251793
├── 1591126281940
├── 1591126324161
└── 1591126344203
```

- 日志内容

日志1591126201556的内容

3

/trace

511

1

/trace/astar.trace

1

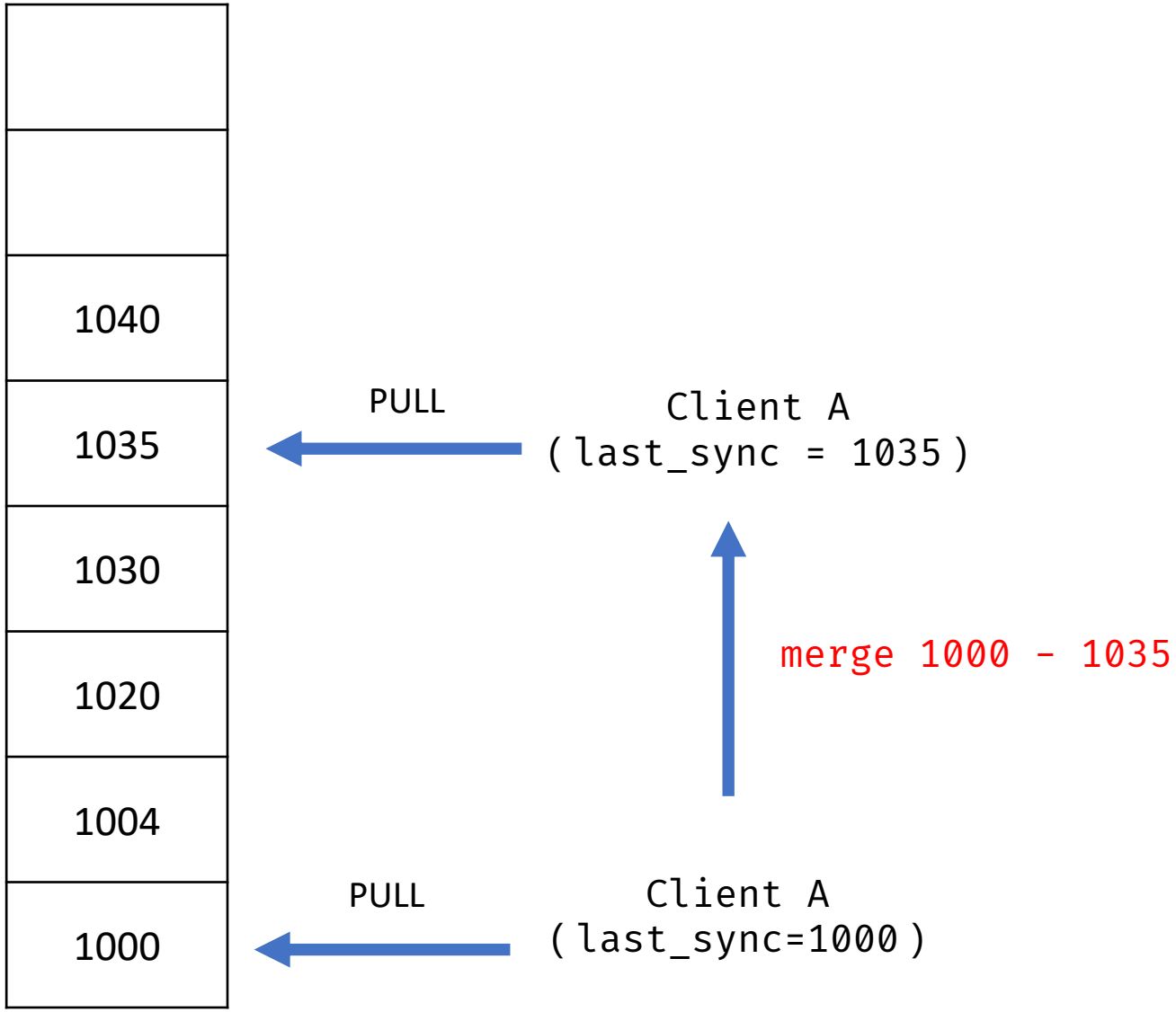
/trace/bodytrack_1m.trace

1

/trace/bzip2.trace

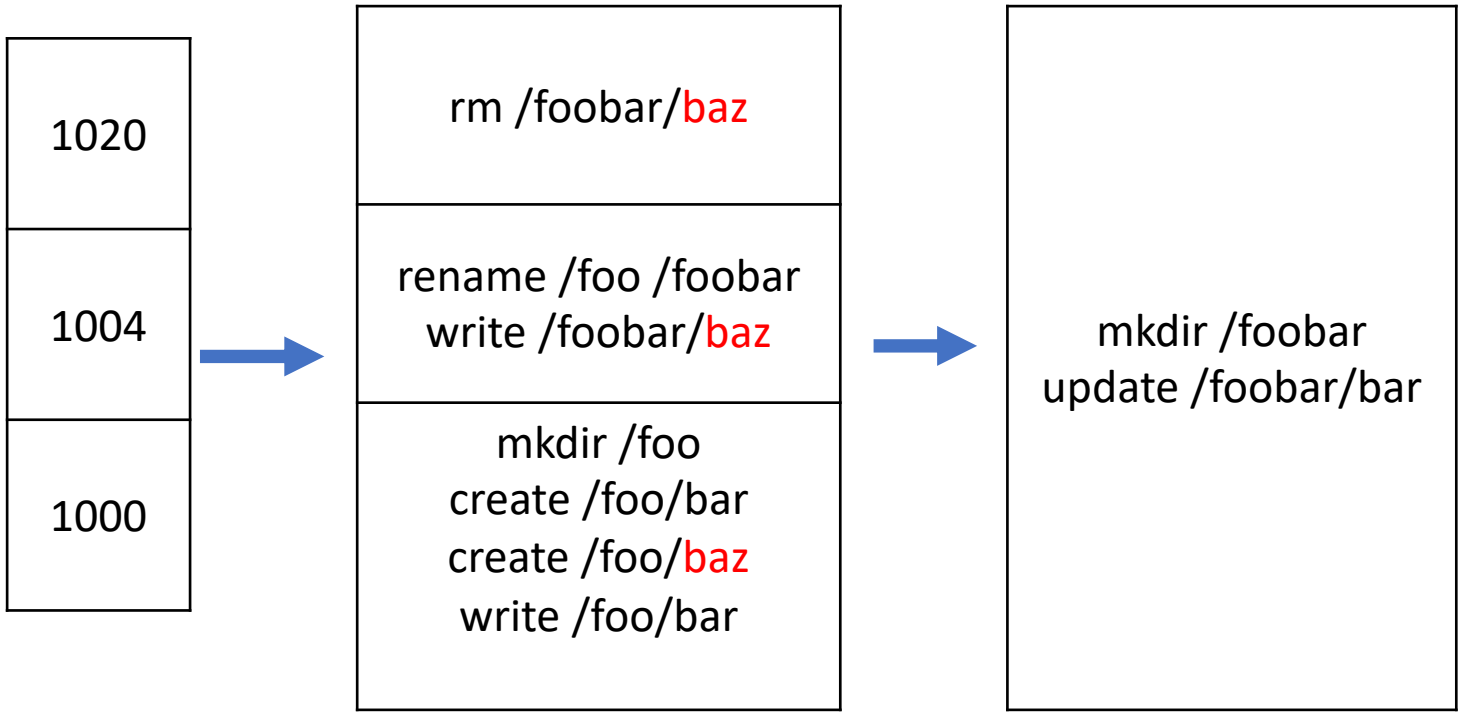
...

PULL
合并

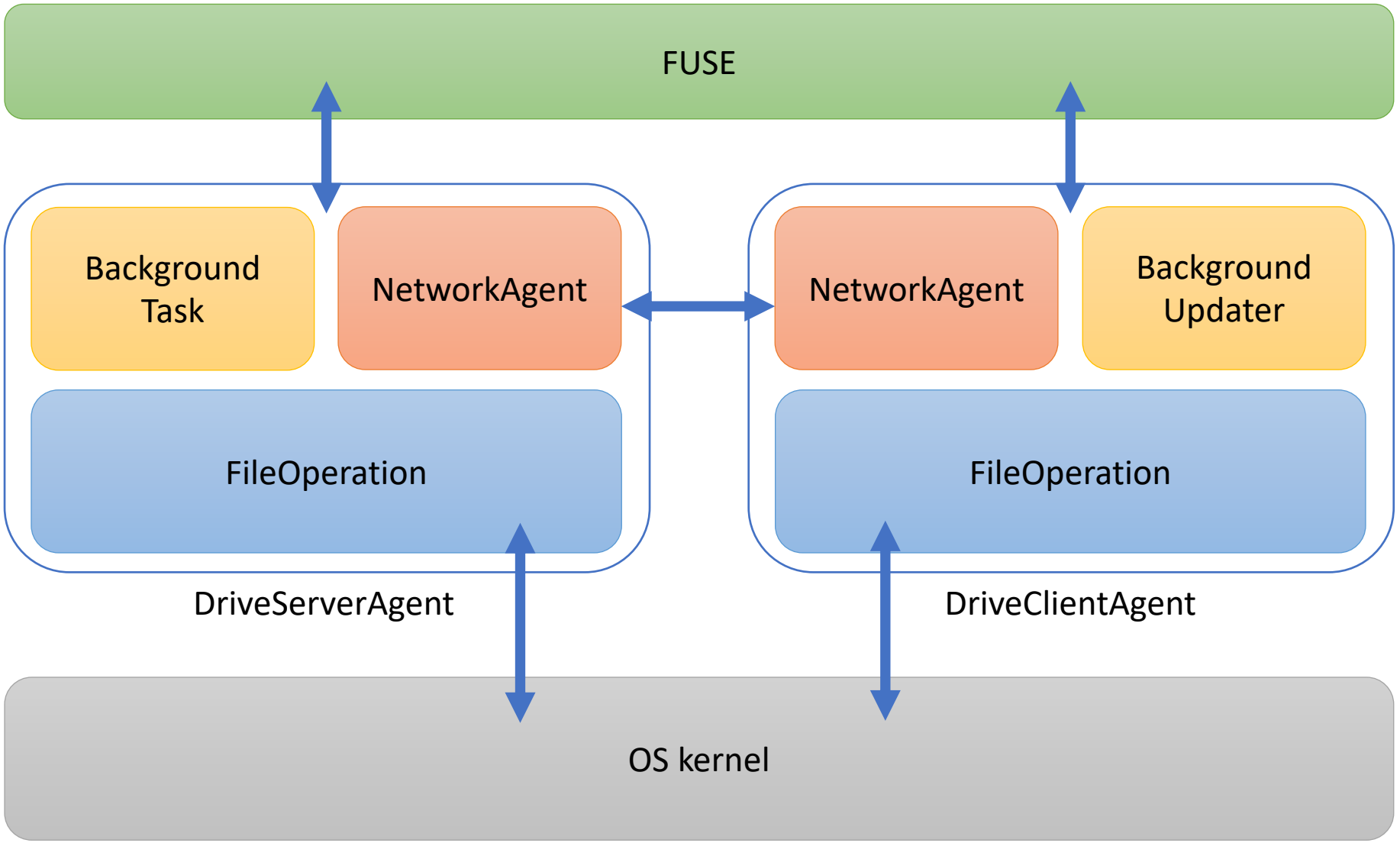

1040
1035
1030
1020
1004
1000
PULL
Client A
(last_sync = 1035)
merge 1000 - 1035
PULL
Client A
(last_sync=1000)

PULL

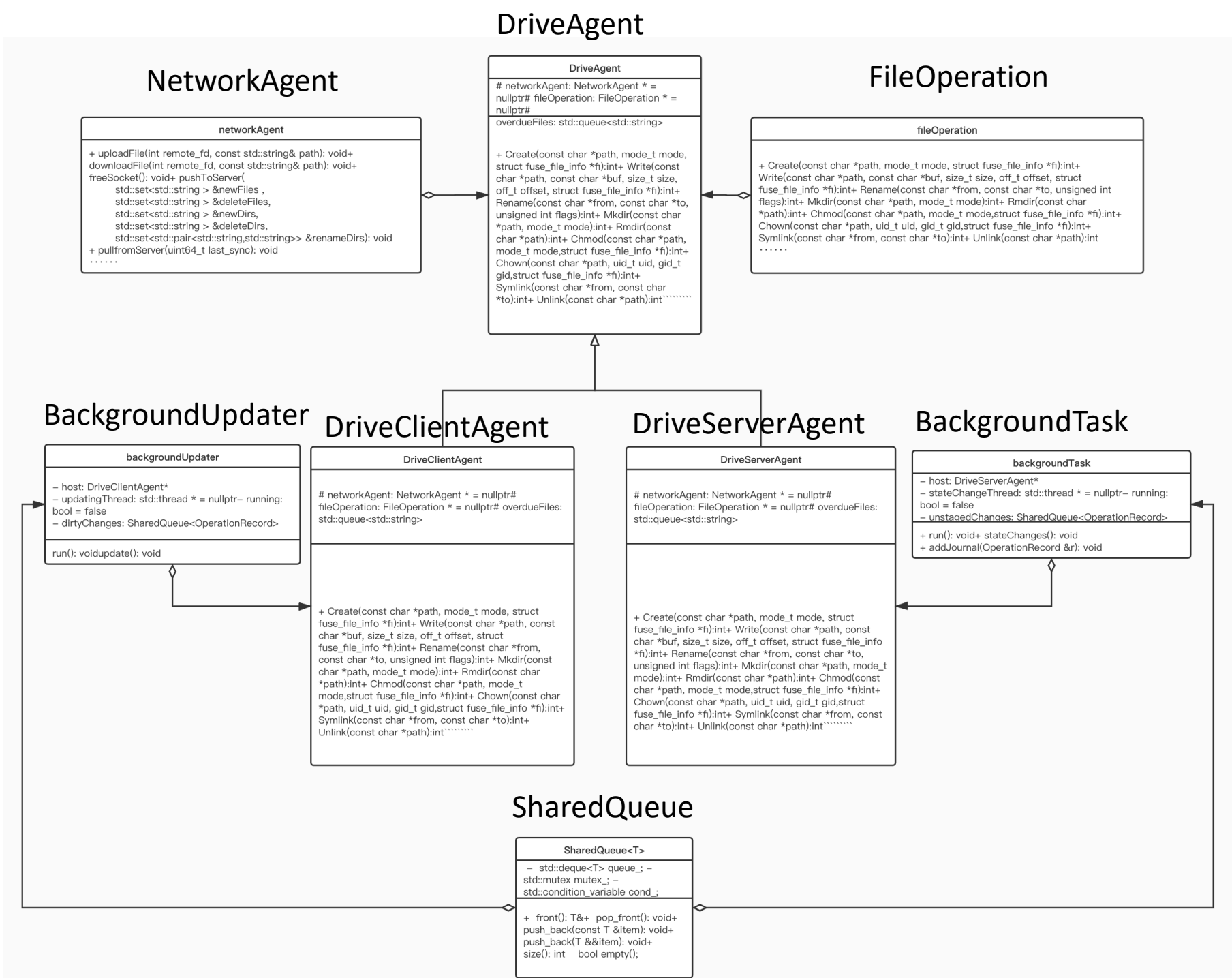
合并


1020
1004
1000
rm /foobar/baz
rename /foo /foobar
write /foobar/baz
mkdir /foo
create /foo/bar
create /foo/baz
write /foo/bar
mkdir /foobar
update /foobar/bar

代码架构


FUSE
Background Task
NetworkAgent
FileOperation
DriveServerAgent
NetworkAgent
Background Updater
FileOperation
DriveClientAgent
OS kernel

代码架构


DriveAgent
NetworkAgent
networkAgent
+ uploadFile(int remote_fd, const std::string& path): void+ downloadFile(int remote_fd, const std::string& path): void+ freeSocket(): void+ pushToServer(std::set<std::string > &newFiles , std::set<std::string > &deleteFiles, std::set<std::string > &newDirs, std::set<std::string > &deleteDirs, std::set<std::pair<std::string,std::string>> &renameDirs): void+ pullfromServer(uint64_t last_sync): void
DriveAgent
networkAgent: NetworkAgent * = nullptr# fileOperation: FileOperation * = nullptr#
overdueFiles: std::queue<std::string>
+ Create(const char *path, mode_t mode, struct fuse_file_info *fi):int+ Write(const char *path, const char *buf, size_t size, off_t offset, struct fuse_file_info *fi):int+ Rename(const char *from, const char *to, unsigned int flags):int+ Mkdir(const char *path, mode_t mode):int+ Rmdir(const char *path):int+ Chmod(const char *path, mode_t mode,struct fuse_file_info *fi):int+ Chown(const char *path, uid_t uid, gid_t gid,struct fuse_file_info *fi):int+ Symlink(const char *from, const char *to):int+ Unlink(const char *path):int
FileOperation
fileOperation
+ Create(const char *path, mode_t mode, struct fuse_file_info *fi):int+ Write(const char *path, const char *buf, size_t size, off_t offset, struct fuse_file_info *fi):int+ Rename(const char *from, const char *to, unsigned int flags):int+ Mkdir(const char *path, mode_t mode):int+ Rmdir(const char *path):int+ Chmod(const char *path, mode_t mode,struct fuse_file_info *fi):int+ Chown(const char *path, uid_t uid, gid_t gid,struct fuse_file_info *fi):int+ Symlink(const char *from, const char *to):int+ Unlink(const char *path):int
BackgroundUpdater
backgroundUpdater
– host: DriveClientAgent*– updatingThread: std::thread * = nullptr– running: bool = false– dirtyChanges: SharedQueue<OperationRecord>
run(): voidupdate(): void
DriveClientAgent
DriveClientAgent
networkAgent: NetworkAgent * = nullptr# fileOperation: FileOperation * = nullptr# overdueFiles: std::queue<std::string>
+ Create(const char *path, mode_t mode, struct fuse_file_info *fi):int+ Write(const char *path, const char *buf, size_t size, off_t offset, struct fuse_file_info *fi):int+ Rename(const char *from, const char *to, unsigned int flags):int+ Mkdir(const char *path, mode_t mode):int+ Rmdir(const char *path):int+ Chmod(const char *path, mode_t mode,struct fuse_file_info *fi):int+ Chown(const char *path, uid_t uid, gid_t gid,struct fuse_file_info *fi):int+ Symlink(const char *from, const char *to):int+ Unlink(const char *path):int
DriveServerAgent
DriveServerAgent
networkAgent: NetworkAgent * = nullptr# fileOperation: FileOperation * = nullptr# overdueFiles: std::queue<std::string>
+ Create(const char *path, mode_t mode, struct fuse_file_info *fi):int+ Write(const char *path, const char *buf, size_t size, off_t offset, struct fuse_file_info *fi):int+ Rename(const char *from, const char *to, unsigned int flags):int+ Mkdir(const char *path, mode_t mode):int+ Rmdir(const char *path):int+ Chmod(const char *path, mode_t mode,struct fuse_file_info *fi):int+ Chown(const char *path, uid_t uid, gid_t gid,struct fuse_file_info *fi):int+ Symlink(const char *from, const char *to):int+ Unlink(const char *path):int
BackgroundTask
backgroundTask
– host: DriveServerAgent*– stateChangeThread: std::thread * = nullptr– running: bool = false– unstagedChanges: SharedQueue<OperationRecord>
+ run(): void+ stateChanges(): void+ addJournal(OperationRecord &r): void
SharedQueue
SharedQueue<T>
– std::deque<T> queue_; – std::mutex mutex_; – std::condition_variable cond_;
+ front(): T&+ pop_front(): void+ push_back(const T &item): void+ push_back(T &&item): void+ size(): int bool empty();

实现难点

- 设计困难
  - 功能逻辑复杂，需要考虑各种情况
  - 需要设计日志系统
  - 需要设计应用网络通信协议
- 代码编写困难
  - FUSE调试困难，难以获得中间变量
  - 需要了解各种系统调用
  - 多客户端联网调试
  - 不使用其他库
  - 代码量在3000行左右（含空行和注释）
- 并发实现困难
  - 多线程调试混乱
  - 服务器和多客户端连接
  - 同步和网络操作不能阻塞文件操作
  - 要实现 thread-safe 的队列

感谢聆听

存储技术基础 陶天骅 陈张萌 2020.6.3