

采用eBPF加速云原生环境中 Curve文件系统性能

网易数帆科技 向东



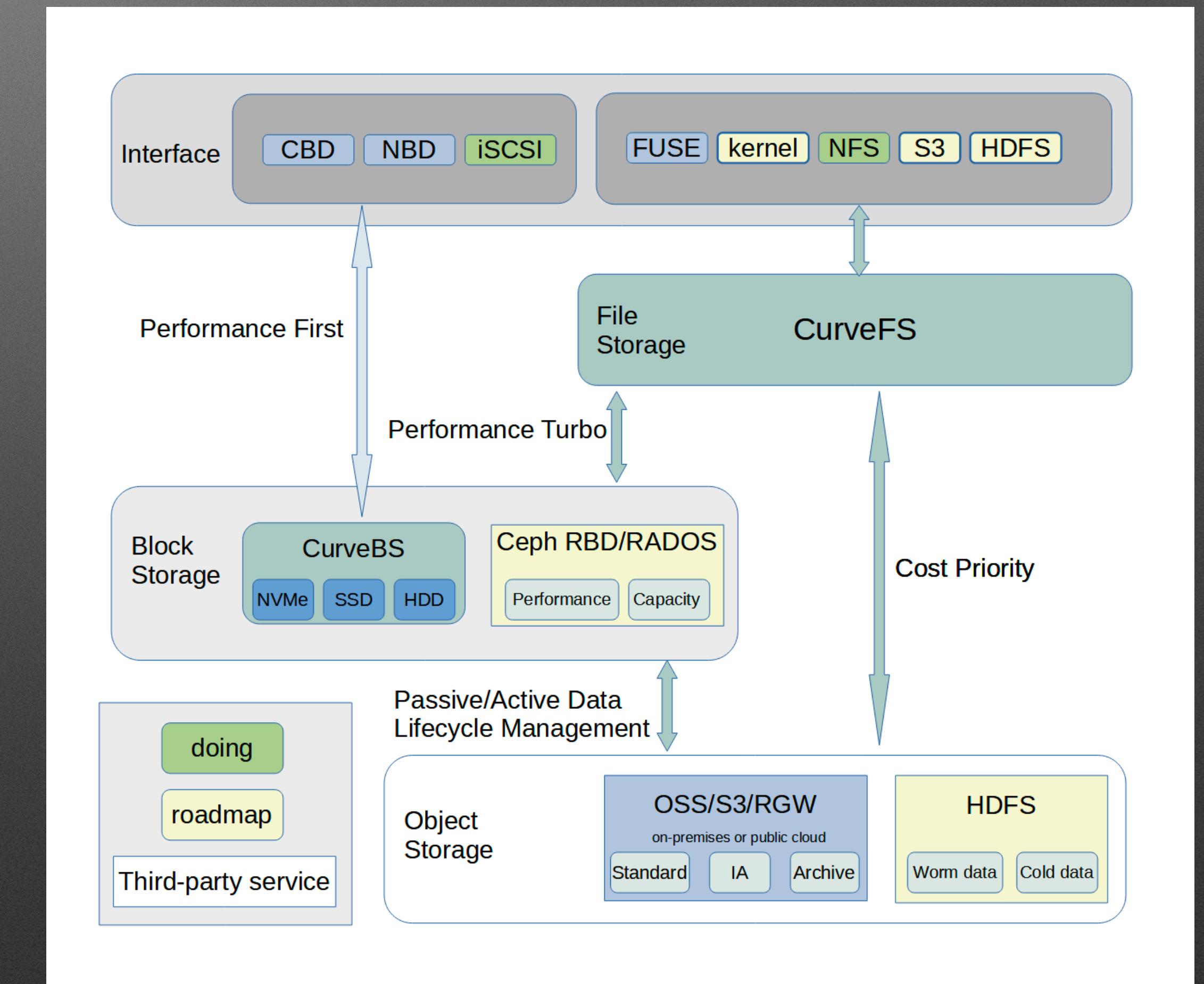
IAS 2022

提纲

- 什么是Curve
- Curve的应用场景及挑战
- Curve客户端面临问题及分析
- 什么是ebpf
- 基于epbf的Curve Cache设计
- Curve社区介绍

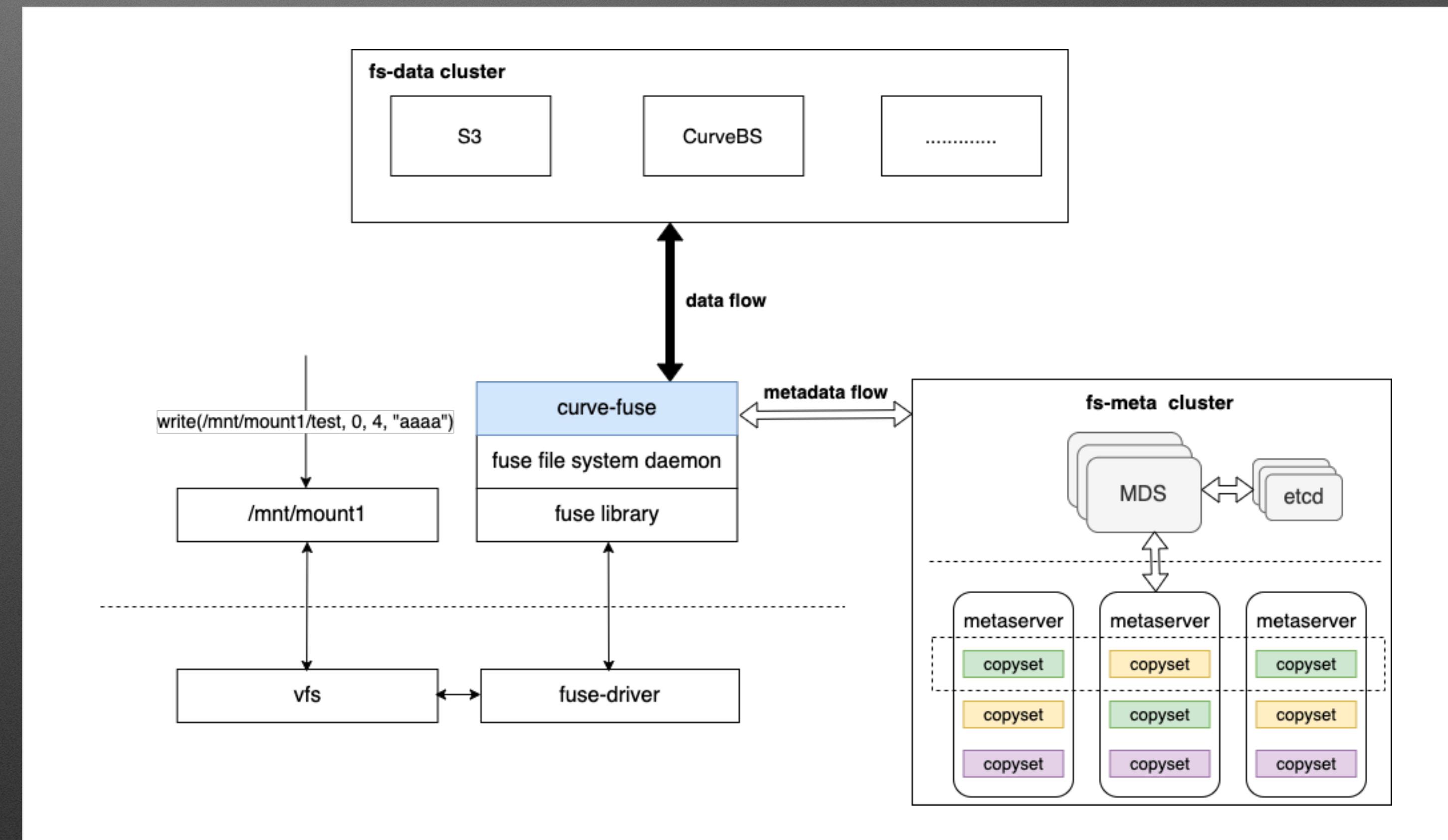
Curve是什么？

- Curve云原生软件定义存储
- Curve分布式块存储
- Curve分布式文件存储
- 高性能、易运维、云原生



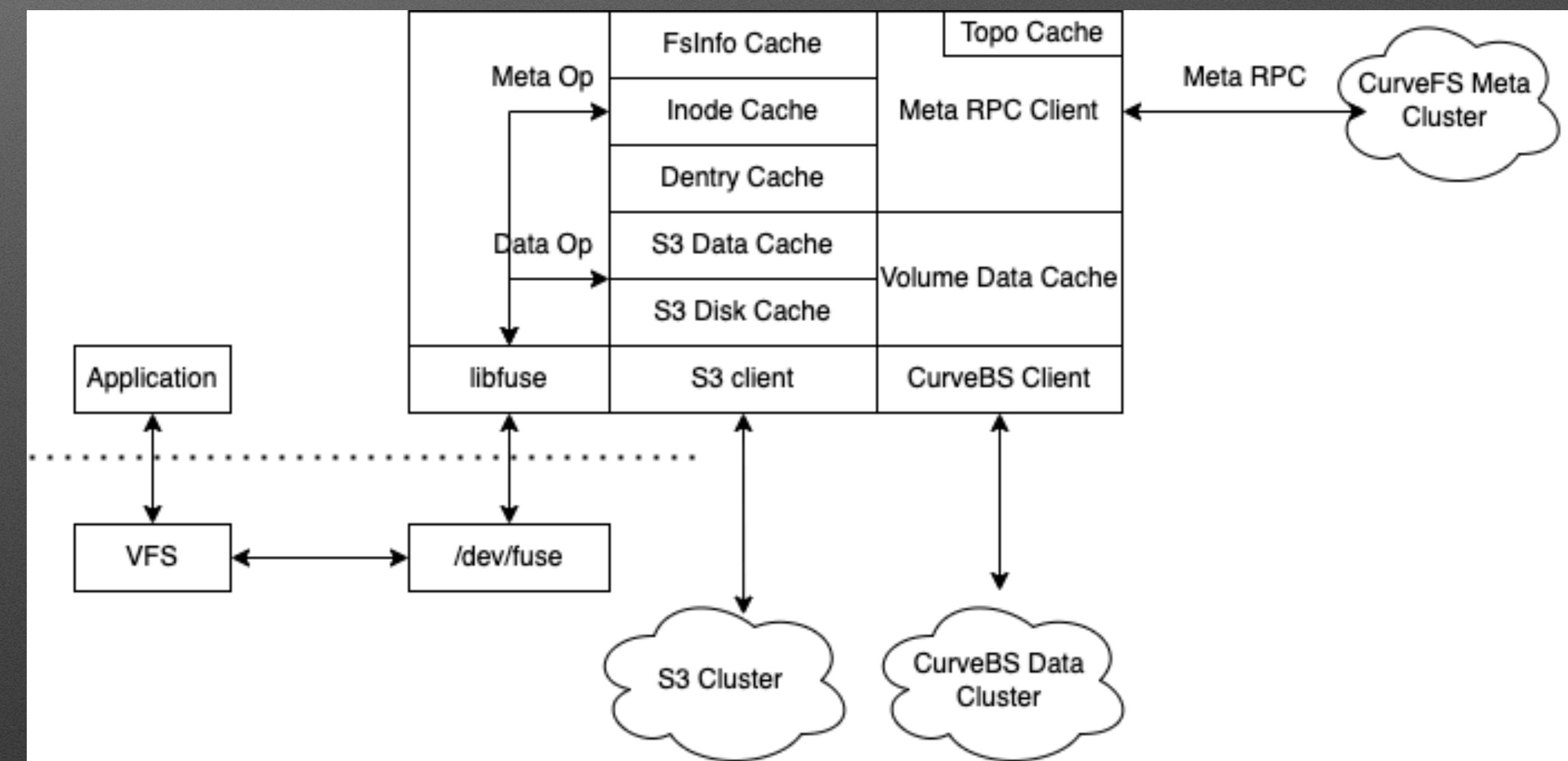
Curve文件系统框架和主要应用场景

- AI机器学习场景
- 大数据计算场景
- 中间件数据存储场景
- 支持POSIX兼容的文件API
- 支持低延迟的文件数据访问



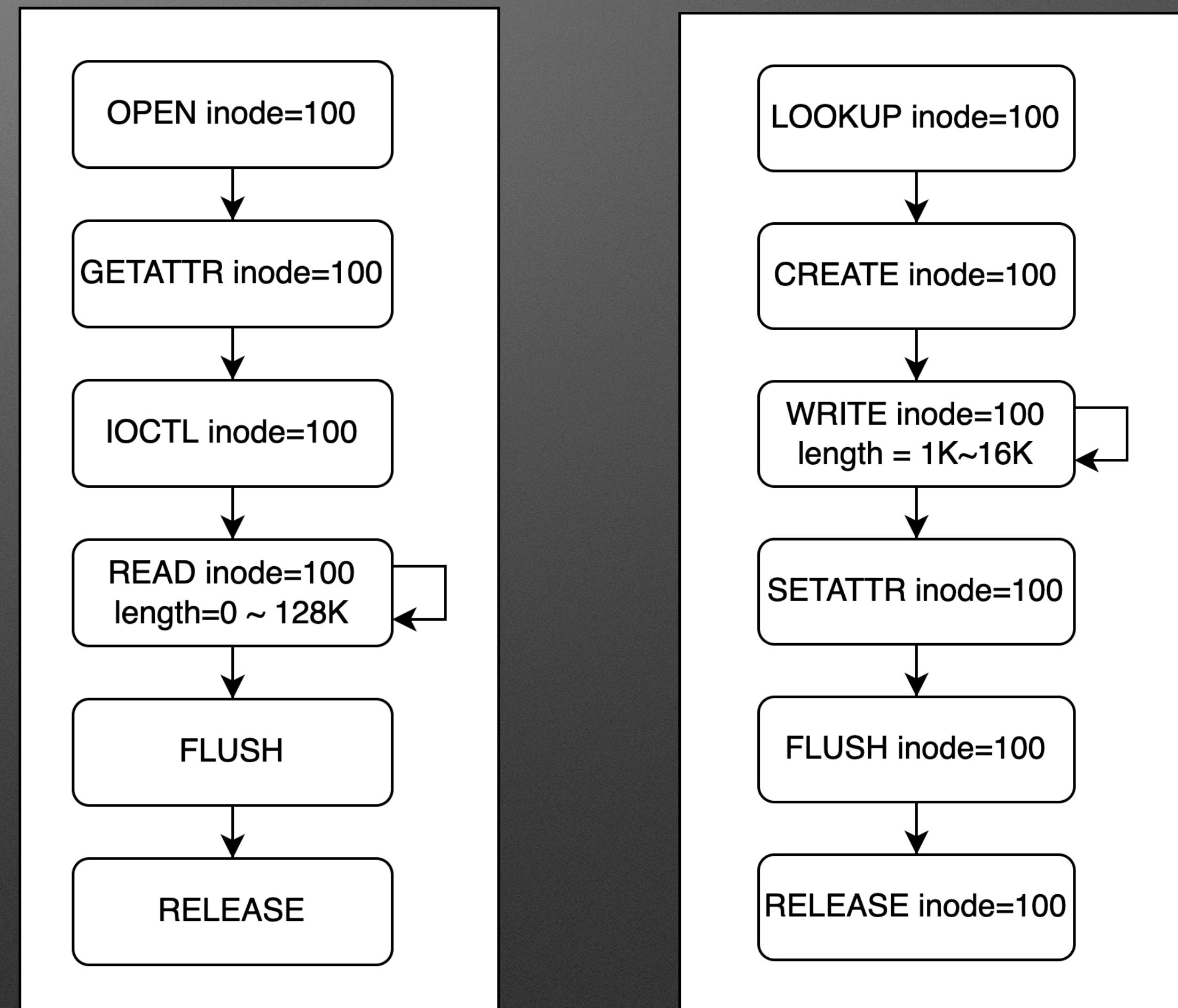
Curve文件系统面临的问题

- 用户态实现
 - 稳定性/可靠性高
 - 容易更新及维护
- 基于FUSE提供POSIX兼容文件接口
- 问题
 - 相对kernel文件系统的实现(ext4, xfs)性能差异大，延迟高



FUSE文件IO读写流程

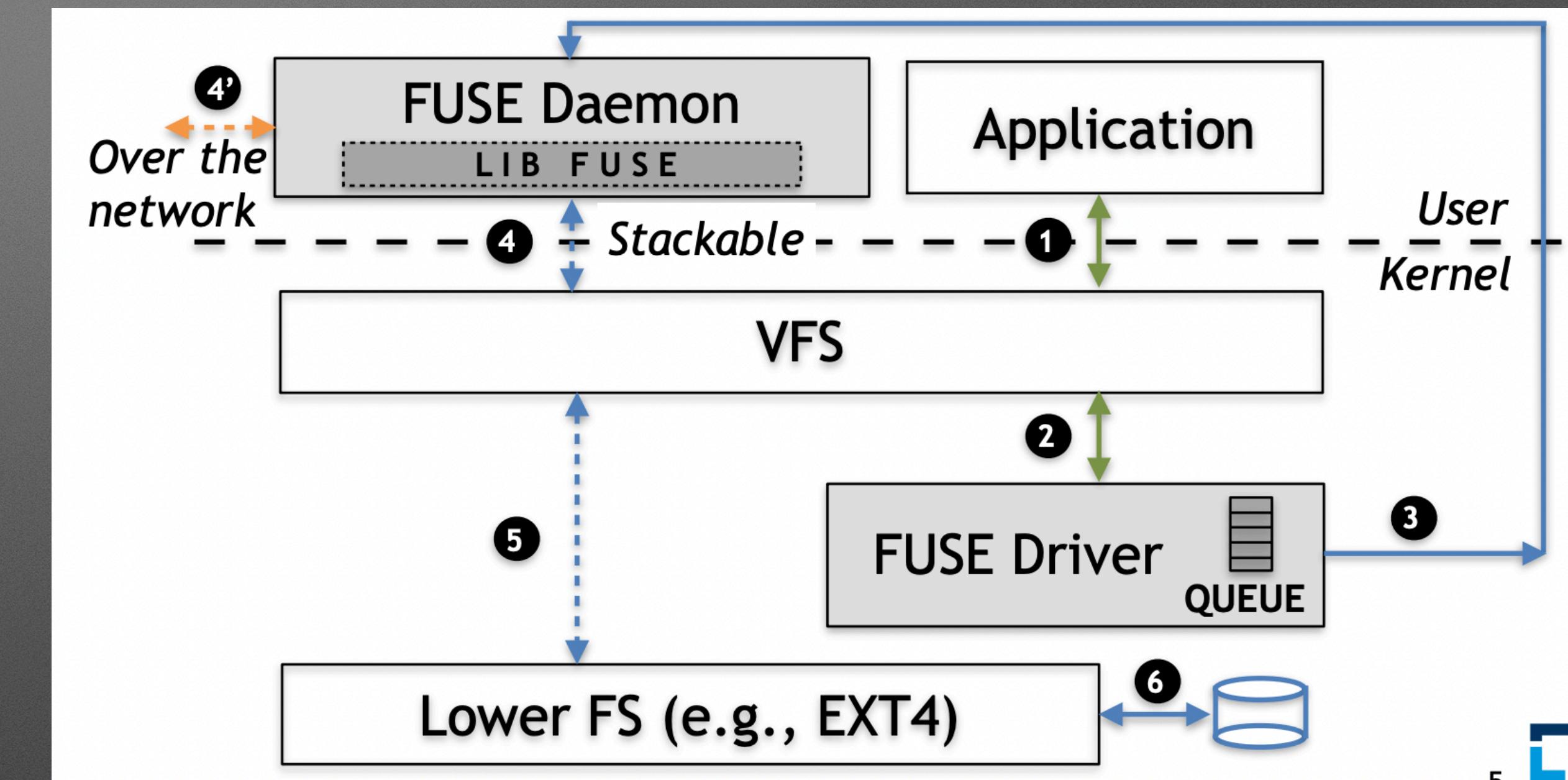
- 场景1 pytorch example word_language_model
 - LOOKUP inode 返回 fstat + timeout设置
 - OPEN 打开 inode返回ok
 - GETATTR 返回fstat
 - READ inode 读取的内容不等从16KB到128KB
 - 关闭文件时会发送FLUSH请求和RELEASE请求
- 场景2 解压压缩包场景
 - LOOKUP inode 没有该inode
 - CREATE创建文件句柄并返回fstat + timeout设置
 - WRITE 写入内容从0~16KB不等
 - SETATTR inode 根据UID, ATIME, CTIME, length来设置属性
 - 关闭文件时会发送FLUSH请求和RELEASE请求



FUSE文件IO读写流程

FUSE的IO路径及瓶颈分析

- 对比测试
 - 文件访问测试直接访问ext4
 - 通过FUSE访问passthrough_ll底层ext4
- 内核调用延迟测试
 - 与FUSE Daemon通讯120us左右， FUSE Daemon大概10us以内
 - 瓶颈在/dev/fuse通讯开销



```
curve-fuse down_write end: down_write[55142] , 1273, 1655177594688845897  
curve-fuse generic_write_checks end: generic_write_checks[55142] , 1020, 1655177594688849207  
curve-fuse queue_request end: queue_request[55142] , 2103, 1655177594688874907  
curve-fuse request_wait_answer end: request_wait_answer[55142] , 20825, 1655177594688897428  
curve-fuse __fuse_request_send end: __fuse_request_send[55142] , 27965, 1655177594688900042  
curve-fuse file_remove_privs end: file_remove_privs[55142] , 32157, 1655177594688903055  
curve-fuse file_update_time end: file_update_time[55142] , 1242, 1655177594688906408  
curve-fuse fuse_wait_on_page_writeback end: fuse_wait_on_page_writeback[55142] , 1504, 1655177594688911957  
curve-fuse queue_request end: queue_request[55142] , 2602, 1655177594688920077  
curve-fuse request_wait_answer end: request_wait_answer[55142] , 34023, 1655177594688956194  
curve-fuse __fuse_request_send end: __fuse_request_send[55142] , 43384, 1655177594688958452  
curve-fuse fuse_send_write end: fuse_send_write[55142] , 46456, 1655177594688960654  
curve-fuse fuse_perform_write end: fuse_perform_write[55142] , 54363, 1655177594688962967  
curve-fuse up_write end: up_write[55142] , 1050, 1655177594688965628  
curve-fuse write end: fuse_file_write_iter[55142] , 123740, 1655177594688967707
```

基于FUSE可能的优化点

- 降低内核与libfuse通讯延迟
- 基于文件属性的操作内核直接返回?
- 基于文件数据的操作先内核读写cache?

实现POSIX兼容API途径及问题

- 基于FUSE的实现

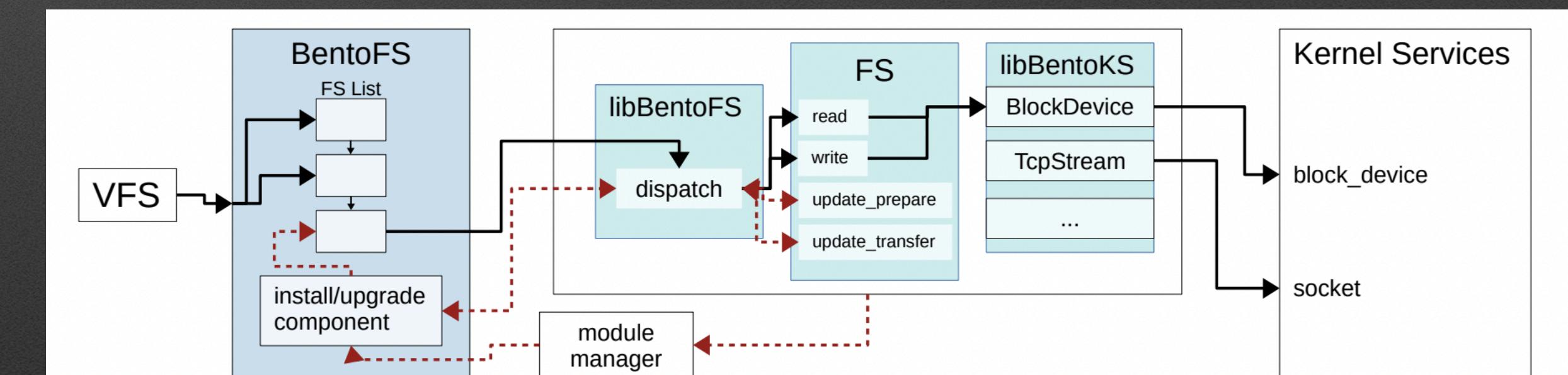
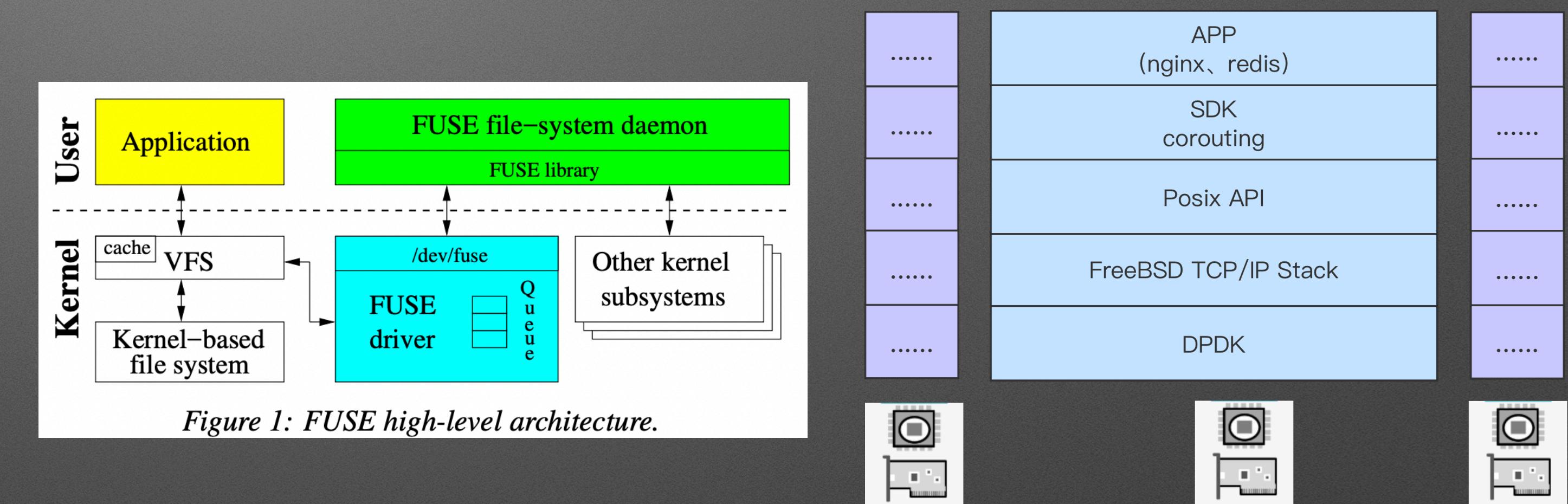
- curve / ceph / gluster

- LD_PRELOAD重载文件系统系统调用

- vpp / f-stack / DirectFUSE

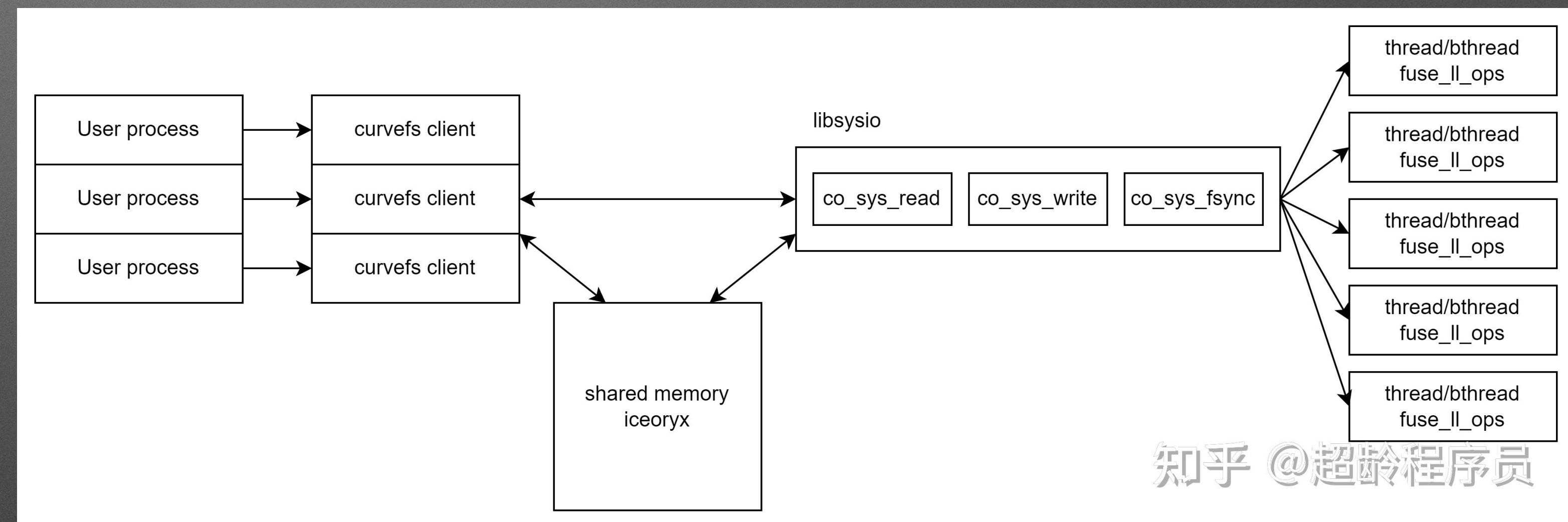
- Kernel版本实现

- BentoFS 基于rust的实现



采用LD_Preload方式瓶颈分析

- 环境
 - FUSE daemon使用 passthrough_ll 调用底层ext4
 - 进程共享内存通信延迟10us+



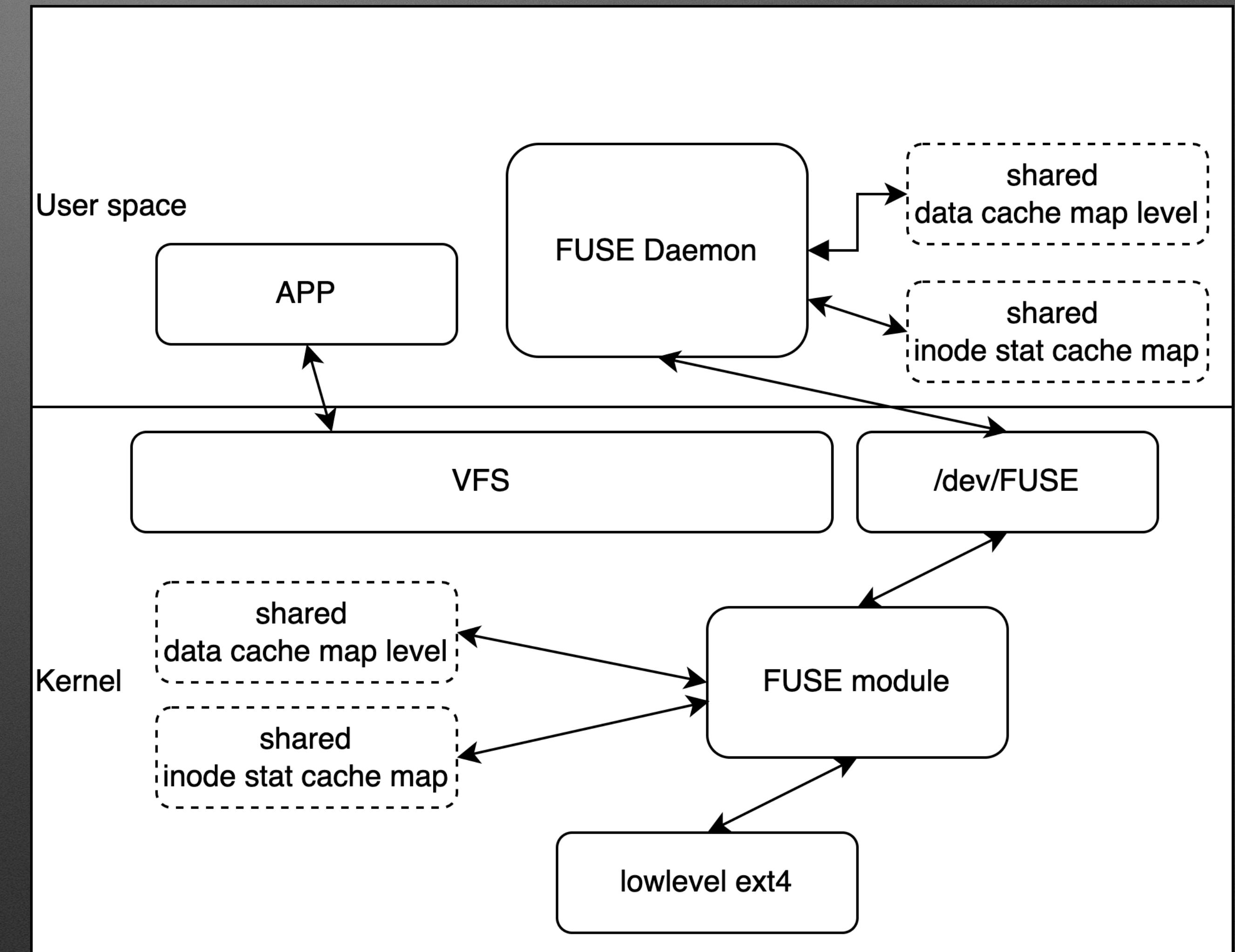
知乎 @超龄程序员

- others 开销 10us+
- fuse_ll_ops开销10us-

	元数据测试3次	32次写io测试	32次读io测试
ext4	64us	96us	37us
FUSE	156us	560us	118us
用户态的系统调用劫持	87us	312us	219us

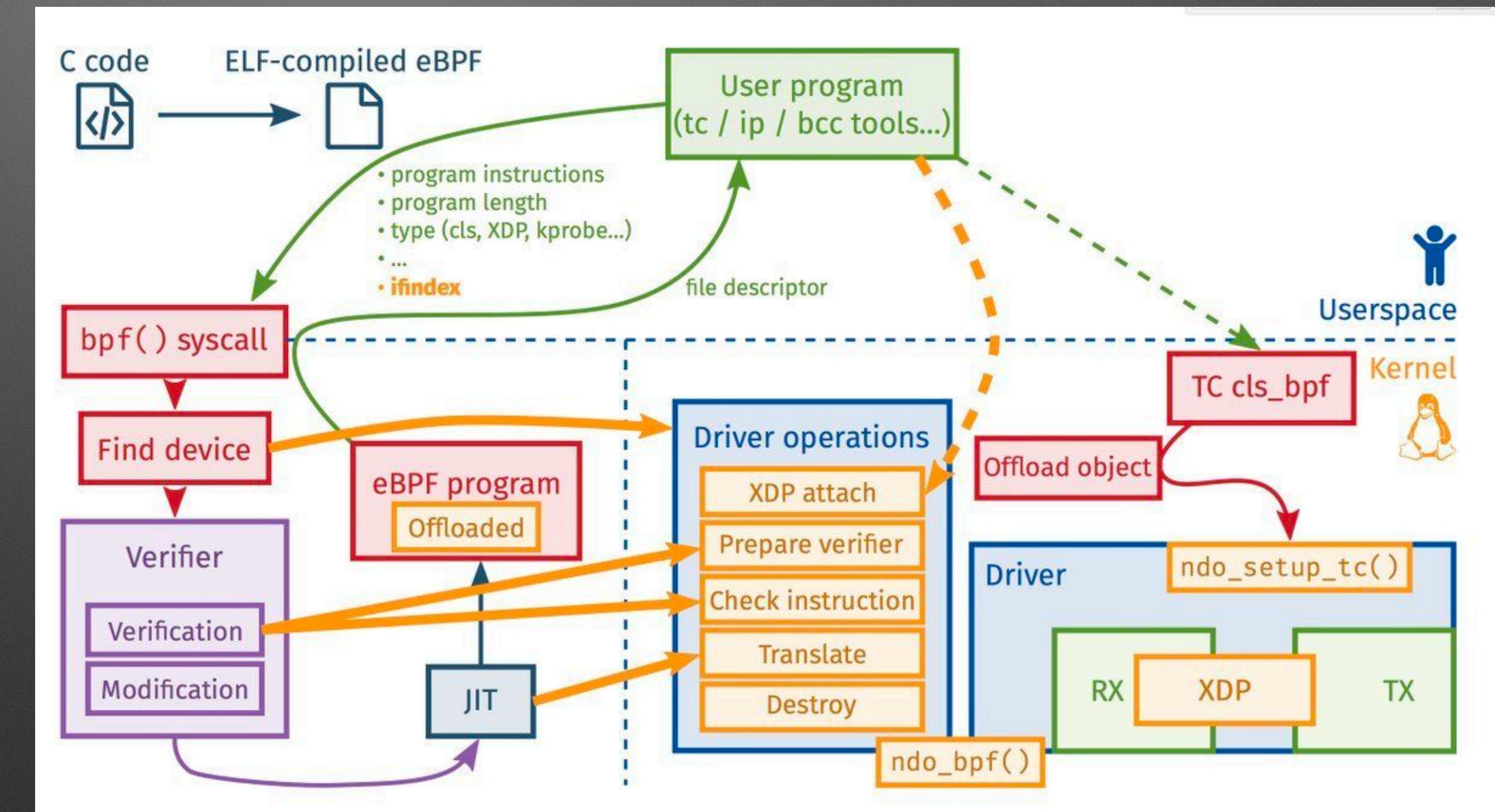
基于FUSE的优化框架

- 框架优化的要点
 - 共享inode cache
 - 共享data cache的映射
- GETATTR流程
- 文件读取流程
- 相关工作
 - extFUSE
 - google android12 passthrough



什么是eBPF

- ebpf是不同环境下内核配置，调试，监控工具
- map映射
- 验证器
- Hook
- Helper api



配置TCP Initial RTO

- 场景 内核4.12之前 initial RTO是一个常数1s
- 应用类型BPF_PROG_TYPE_SOCK_OPS
- HOOK BPF_SOCK_OPS_TIMEOUT_INIT
- 内核中调用栈
 - tcp_timeout_init
 - tcp_call_bpf(BPF_SOCK_OPS_TIMEOUT_INIT)
 - bpf_cgroup_run_sock_ops
 - ...
 - set_initial_rto

```
$ sudo tcpdump -nn -i enp0s3 host 9.9.9.9 and port 9999
21:26:43.834860 IP 192.168.1.5.53844 > 9.9.9.9.9999: Flags [S], seq 281070166, ... length 0 # +0s
21:26:44.859801 IP 192.168.1.5.53844 > 9.9.9.9.9999: Flags [S], seq 281070166, ... length 0 # +1s
21:26:46.876328 IP 192.168.1.5.53844 > 9.9.9.9.9999: Flags [S], seq 281070166, ... length 0 # +2s
21:26:51.068268 IP 192.168.1.5.53844 > 9.9.9.9.9999: Flags [S], seq 281070166, ... length 0 # +4s
21:26:59.259304 IP 192.168.1.5.53844 > 9.9.9.9.9999: Flags [S], seq 281070166, ... length 0 # +8s
21:27:15.389522 IP 192.168.1.5.53844 > 9.9.9.9.9999: Flags [S], seq 281070166, ... length 0 # +16s
...
```

```
__section("sockops")
int set_initial_rto(struct bpf_sock_ops *skops)
{
    int timeout = 3;
    int hz = 250; // grep 'CONFIG_HZ=' /boot/config-$(uname -r), HZ of my machine

    int op = (int) skops->op;
    if (op == BPF_SOCK_OPS_TIMEOUT_INIT) {
        skops->reply = hz * timeout; // 3 seconds
    }

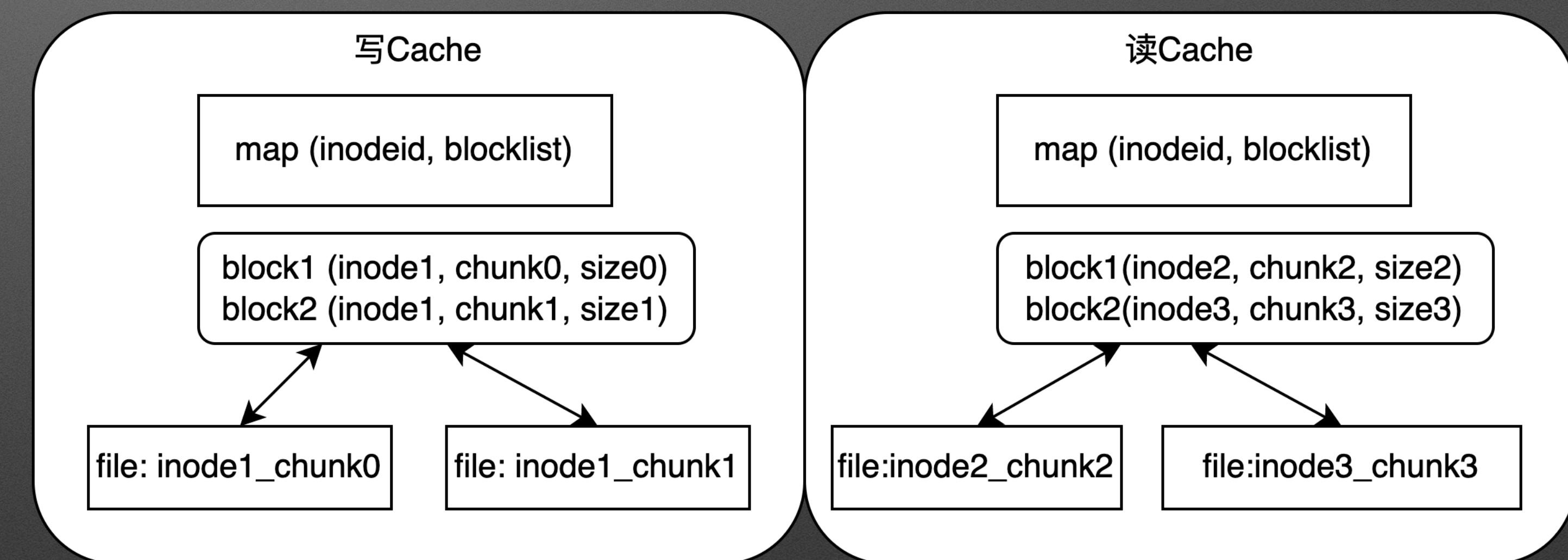
    return 1;
}

char _license[] __section("license") = "GPL";
```

```
$ sudo tcpdump -nn -i enp0s3 host 9.9.9.9 and port 9999
21:37:46.357686 IP 192.168.1.5.53866 > 9.9.9.9.9999: Flags [S], seq 3392061475, .. length 0 # +0s
21:37:49.372053 IP 192.168.1.5.53866 > 9.9.9.9.9999: Flags [S], seq 3392061475, .. length 0 # +3s
21:37:55.515914 IP 192.168.1.5.53866 > 9.9.9.9.9999: Flags [S], seq 3392061475, .. length 0 # +6s
21:38:07.547362 IP 192.168.1.5.53866 > 9.9.9.9.9999: Flags [S], seq 3392061475, .. length 0 # +12s
21:38:32.635499 IP 192.168.1.5.53866 > 9.9.9.9.9999: Flags [S], seq 3392061475, .. length 0 # +24s
...
```

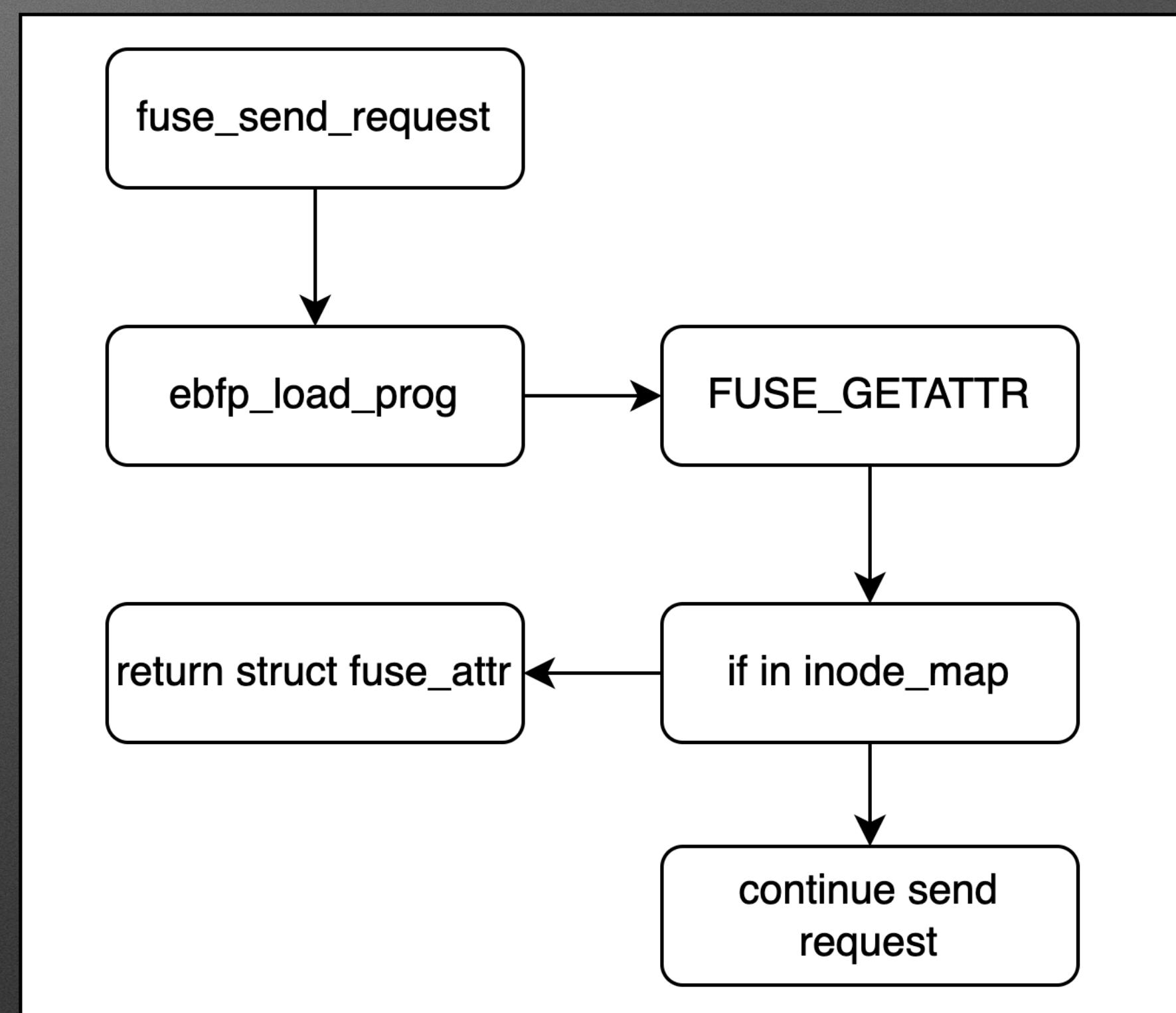
Curve的Cache模块

- 底层ext4文件系统作为cache
- cache分为写cache与读cache, 读/写cache独立配置
- 与底层文件关系 (filename, offset, len) = func (inodeid, offset, len)
- 读cache流程
- 写cache流程



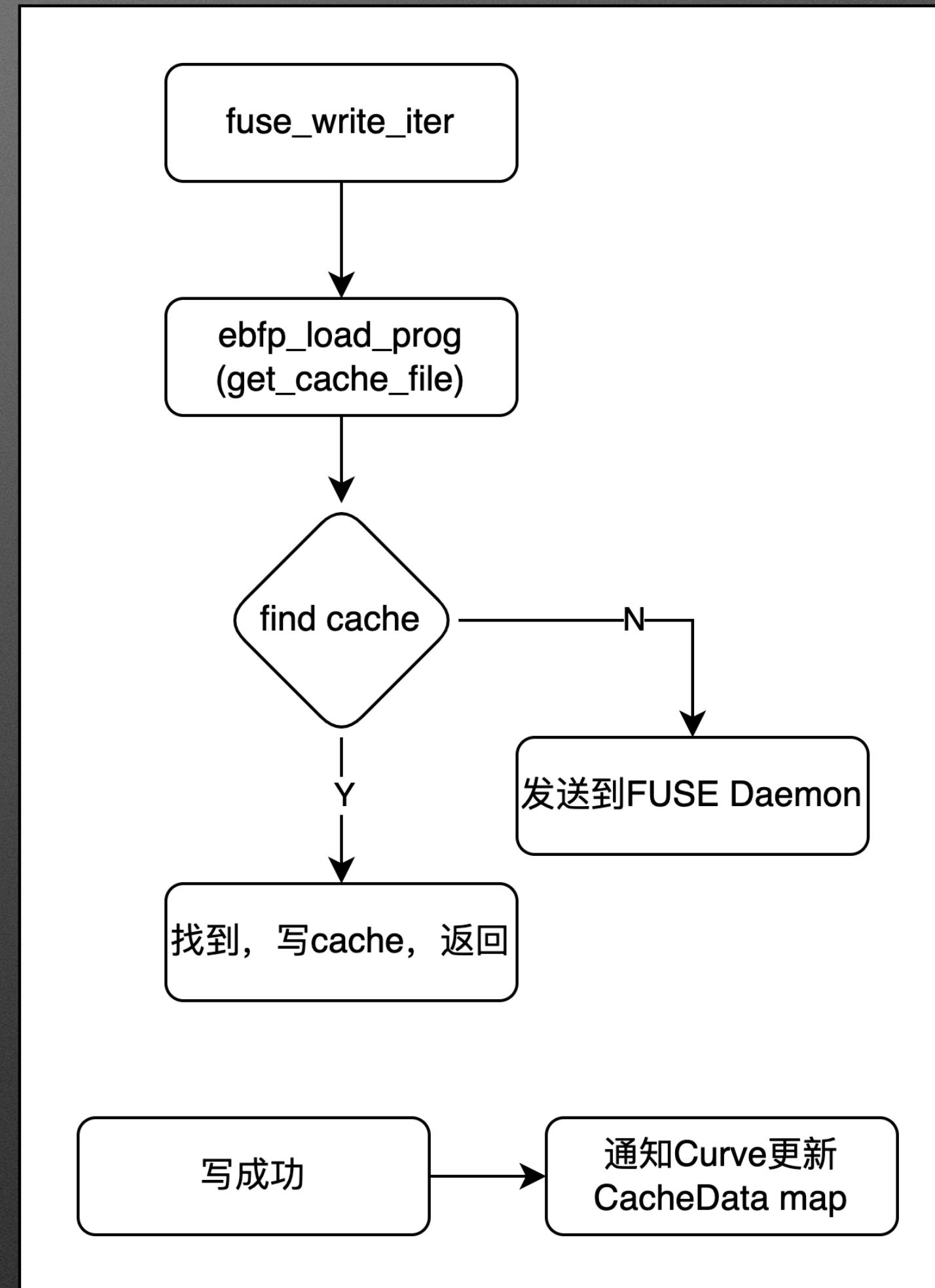
基于inode cache优化

- bpf程序类型 BPF_PROG_TYPE_EXTFUSE
- Hook点及方法
 - fuse_request_send
 - FUSE_LOOKUP / FUSE_GETATTR / FUSE_SETATTR /
- map 结构
 - dentry map BPF_MAP_TYPE_HASH
 - key (inode id, node name)
 - value inode id
 - inode map BPF_MAP_TYPE_HASH
 - key inode id
 - value fuse_attr (文件属性)



基于data cache部分

- bpf程序类型 BPF_PROG_TYPE_EXTFUSE
- Hook点及方法
 - fuse_file_read_iter, fuse_file_write_iter
 - FUSE_READ / FUSE_WRITE
- Map
 - WriteCache map BPF_MAP_TYPE_HASH
 - key inodeid
 - value CacheStatus, blocklist[] {fd, chunk, size, kernel file, ver}
 - ReadCache map BPF_MAP_TYPE_HASH
 - key inodeid
 - value CacheStatus, filemap[] {fd, chuk, size, kernel file, ver}
 - update list map BFP_MAP_TYPE_PERF_EVENT_ARRAY
 - updata cache map



Curve社区介绍

- Curve 的成长离不开大家的支持和参与。非常欢迎社区用户参与社区共建，可以通过贡献代码、丰富文档、提交issue、改进网站、交流分享等，提高自己专业能力的同时还可以提升个人影响力、扩展人脉。
- 项目<https://github.com/opencurve/curve>
- 版本发布周期：每半年一个大版本，1~2个月一个小版本
- 了解Curve进展：每隔2周的Curve周会说明Curve进展以及讨论相关问题
- 提交bug与建议：<http://github.com/opencurve/curve/issues>
- 参与Curve交流与讨论：微信群，右手边二维码



You can take away

- Curve文件系统采用cache来提升性能，对象存储来降低成本
- 目前面临Curve文件系统客户端延迟较大的问题
- 解决延迟有ld_preload以及ebpf的方式，各有优缺点
- 使用ebpf可以在内核中直接读取cache数据，并返回，降低了延迟。
- 介绍了ebpf基于FUSE的cache设计

THANKS

IAS 2022