

NoAxiom OS：设计与实现

刘文涛 陈润澎

2025.8.21



前言

- 从零**自主设计**操作系统内核
- **41229** 行 Rust 代码, **1116** 次commit记录
- 初赛阶段基础测例**满分**
- 现场赛物理上板**综合排名第 1**



前言

一年开发周期： 2024/9/1 - 2024/8/21



2024/9
龙芯杯结束
开始学习OS



2025/3
设计重构
硬件抽象层



2025/5
实现基础功能
通过初赛测例



2025/8
性能优化
与答辩



2025/7
实现板级驱动
重构文件系统



2025/6
实现ltp启动



CSCC 全国大学生
计算机系统能力大赛



杭州电子科技大学
HANGZHOU DIANZI UNIVERSITY

Part.01

系统架构

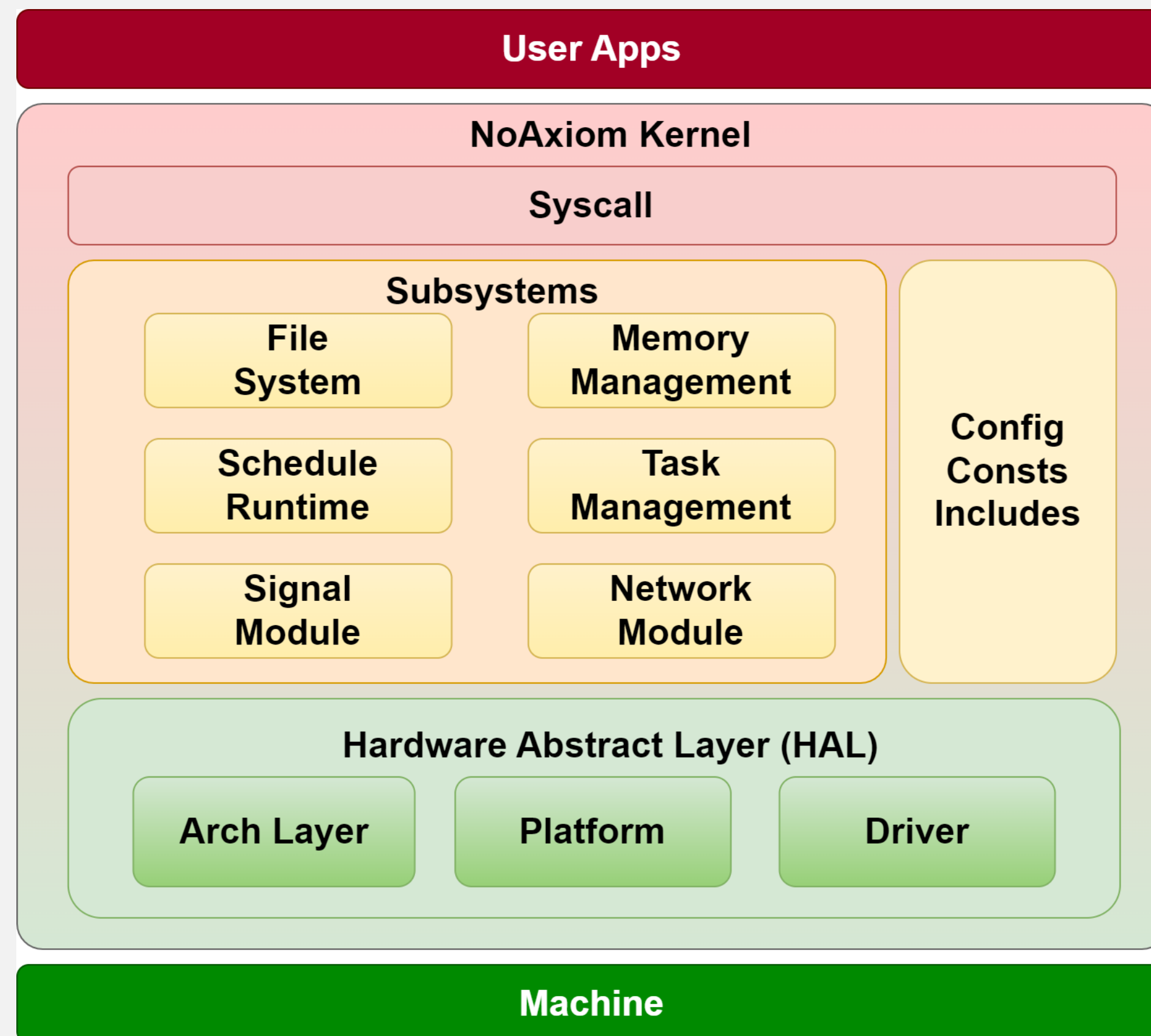


用户层

内核层

硬件抽象层

机器层



Part.02

硬件抽象层

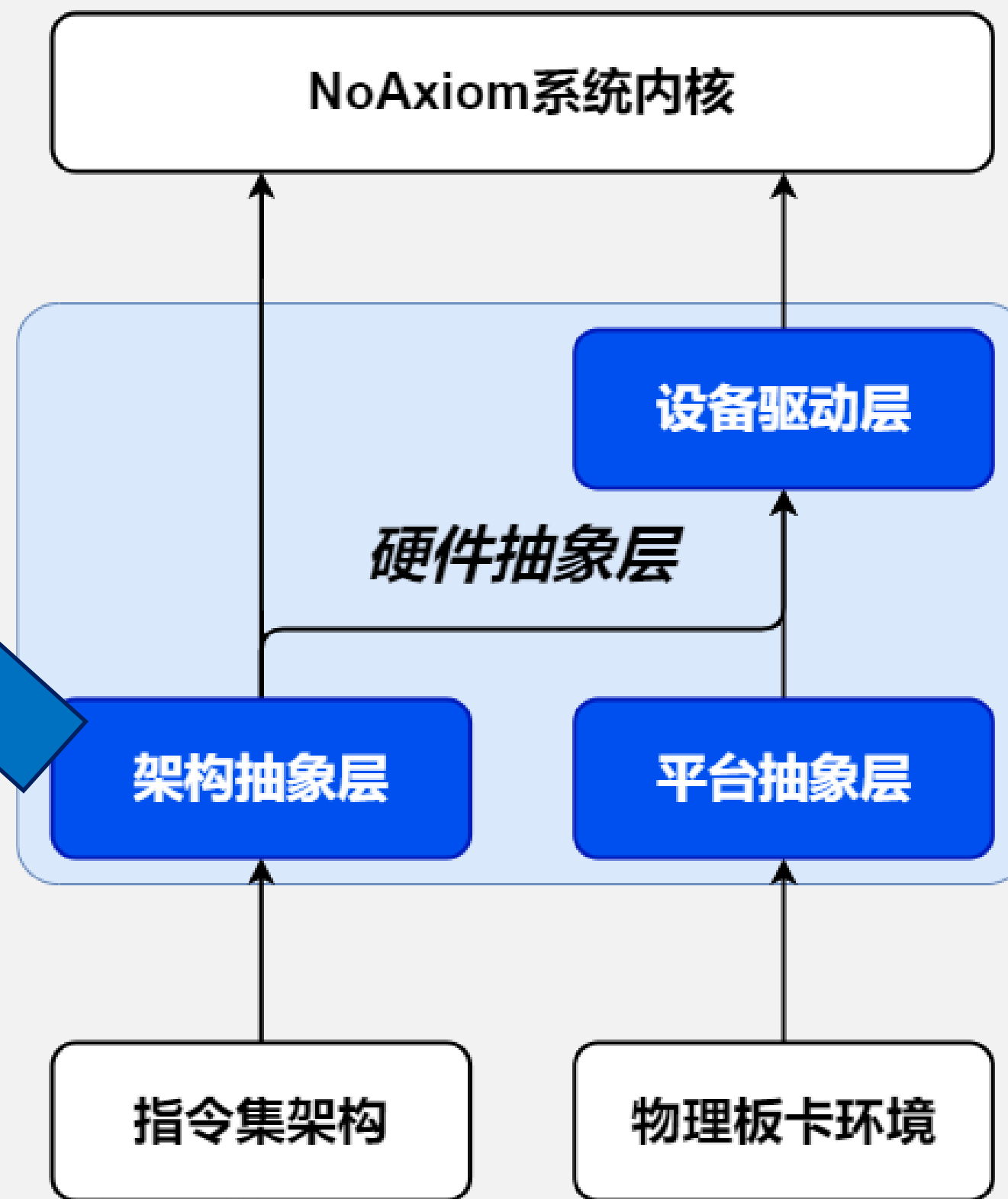


为了解耦：硬件抽象层

指令集架构抽象层：
最小化硬件接口

软件
高抽象层次

硬件
低抽象层次

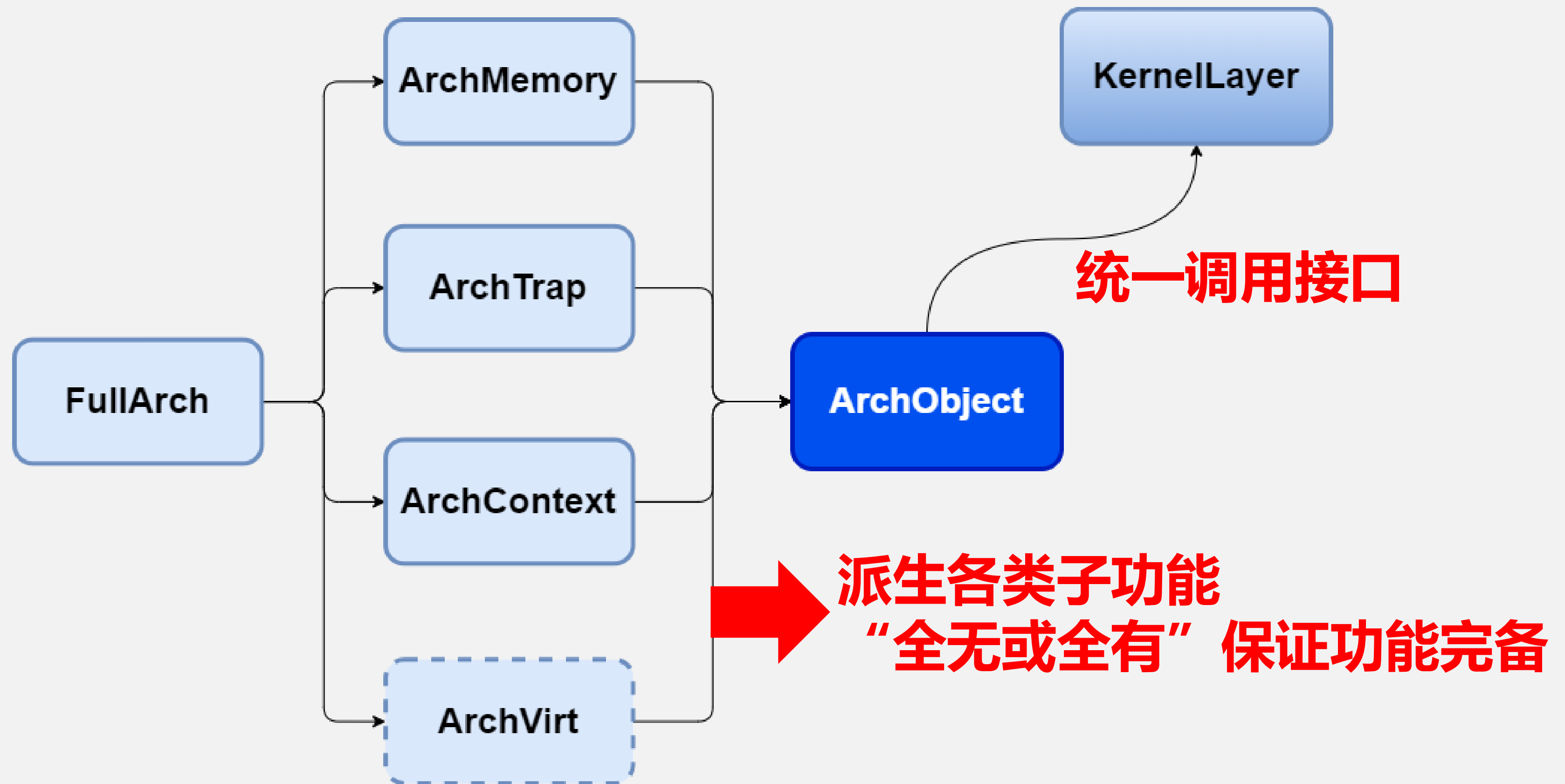


内核视角：
高度解耦

硬件视角：
消除架构差异

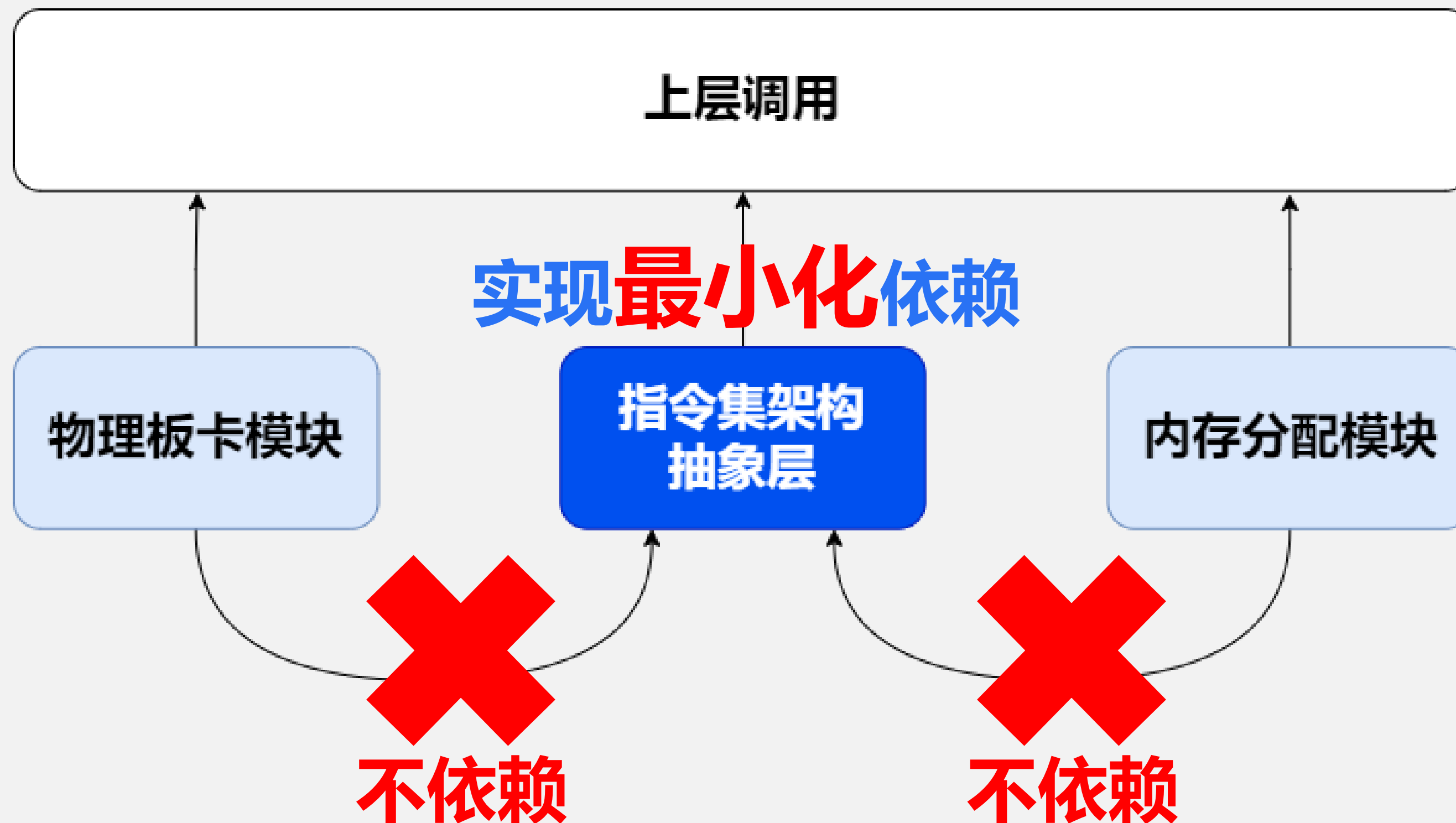


为了解耦：指令集架构抽象层



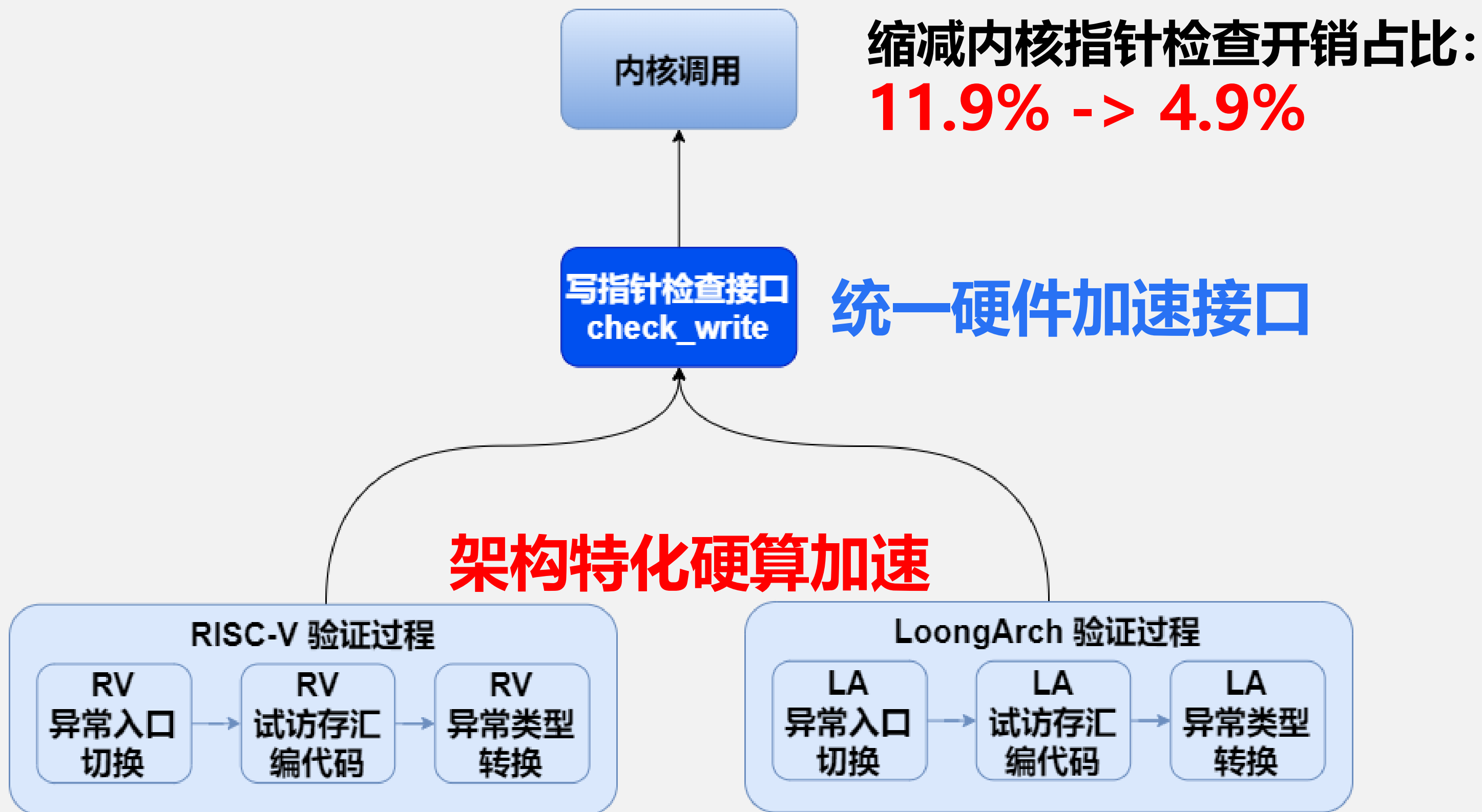


为了精简：最小化模块依赖





为了高效：自定义硬件加速功能扩展






最终呈现效果


特判架构行为：
逻辑复杂，维护困难

```
unsafe {  
    #[cfg(target_arch = "riscv64")]  
    copy_nonoverlapping(&value as *const u32, to, from);  
    #[cfg(target_arch = "loongarch64")]  
    copy_to_user(to: opt_addr, from: &value, len: len);  
    *opt_len = len as u32;  
};
```



统一HAL接口：
逻辑**精简**，易于维护

```
if self.is_null() {  
    warn!("[write] write null pointer");  
    return Err(errno::EFAULT);  
}  
match Arch::check_write(self.addr()) {  
    Ok(()) => Ok(()),  
    Err(trap_type: TrapType) => {  
        match trap_type {  
            ...  
        }  
    }  
}
```






开源HAL库发布

已发布于Github

期待开源为社区带来新的活力

 **NoAxiom-HAL** Public

Unpin

Watch 0

Fork 0


Star 0


master









Go to file

+

Code

 **crpboy** initial commit ...

e256b72 · 3 hours ago 

| | | |
|---|----------------|-------------|
|  arch | initial commit | 3 hours ago |
|  config | initial commit | 3 hours ago |
|  platform | initial commit | 3 hours ago |
|  .gitignore | initial commit | 3 hours ago |
|  Cargo.lock | initial commit | 3 hours ago |
|  Cargo.toml | initial commit | 3 hours ago |
|  Makefile | initial commit | 3 hours ago |
|  README.md | initial commit | 3 hours ago |

About

HAL layer from NoAxiom-OS

[github.com/NoAxiom/NoAxiom...](#)

rust

operating-system

hal

riscv64

loongarch64

Readme

Activity

0 stars

0 watching

0 forks

Releases

No releases published

[Create a new release](#)

Part.03

进程调度

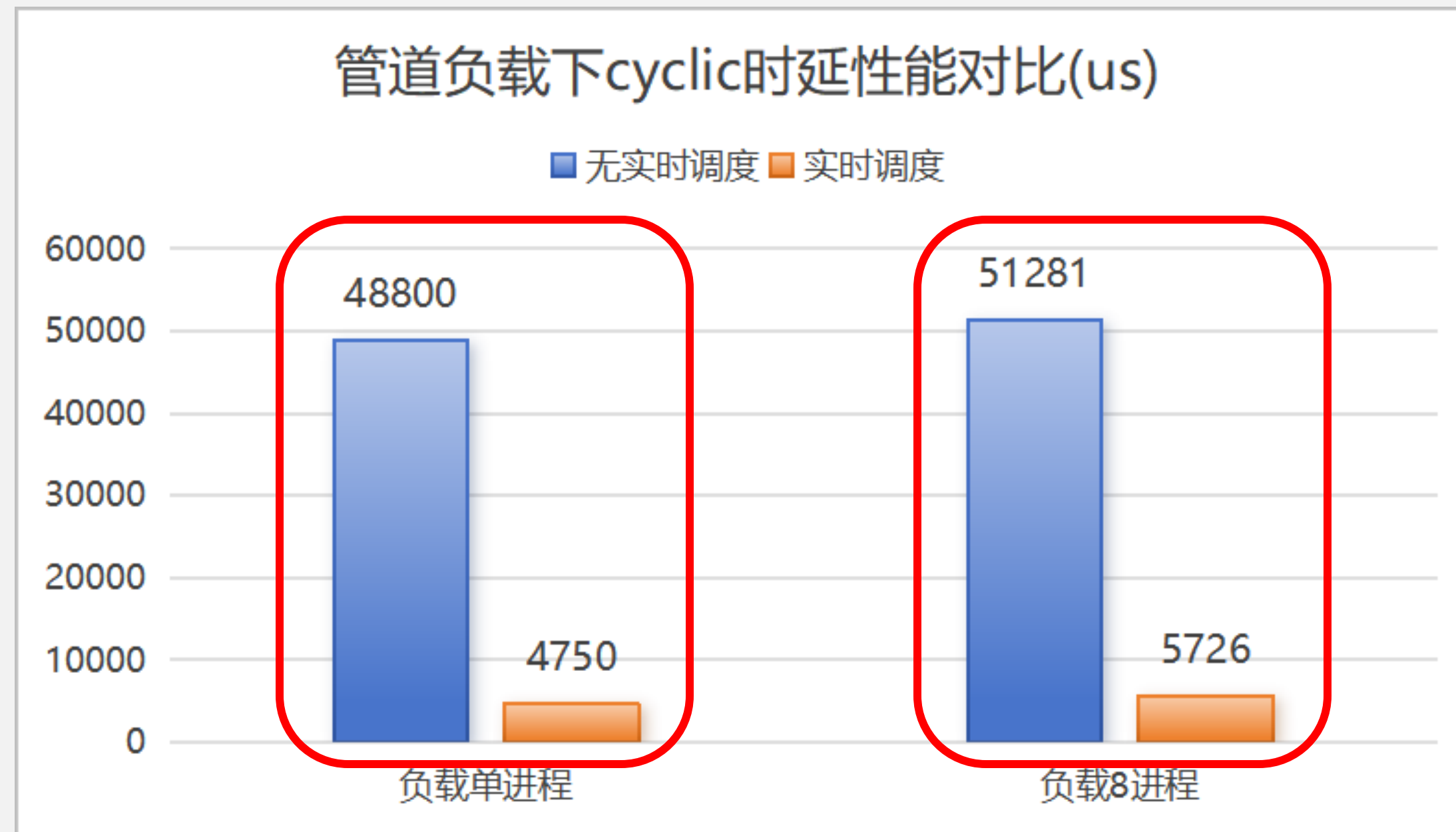


为了性能：多级实时性调度





为了性能：多级实时性调度



单线程响应速度提升
10.27倍！

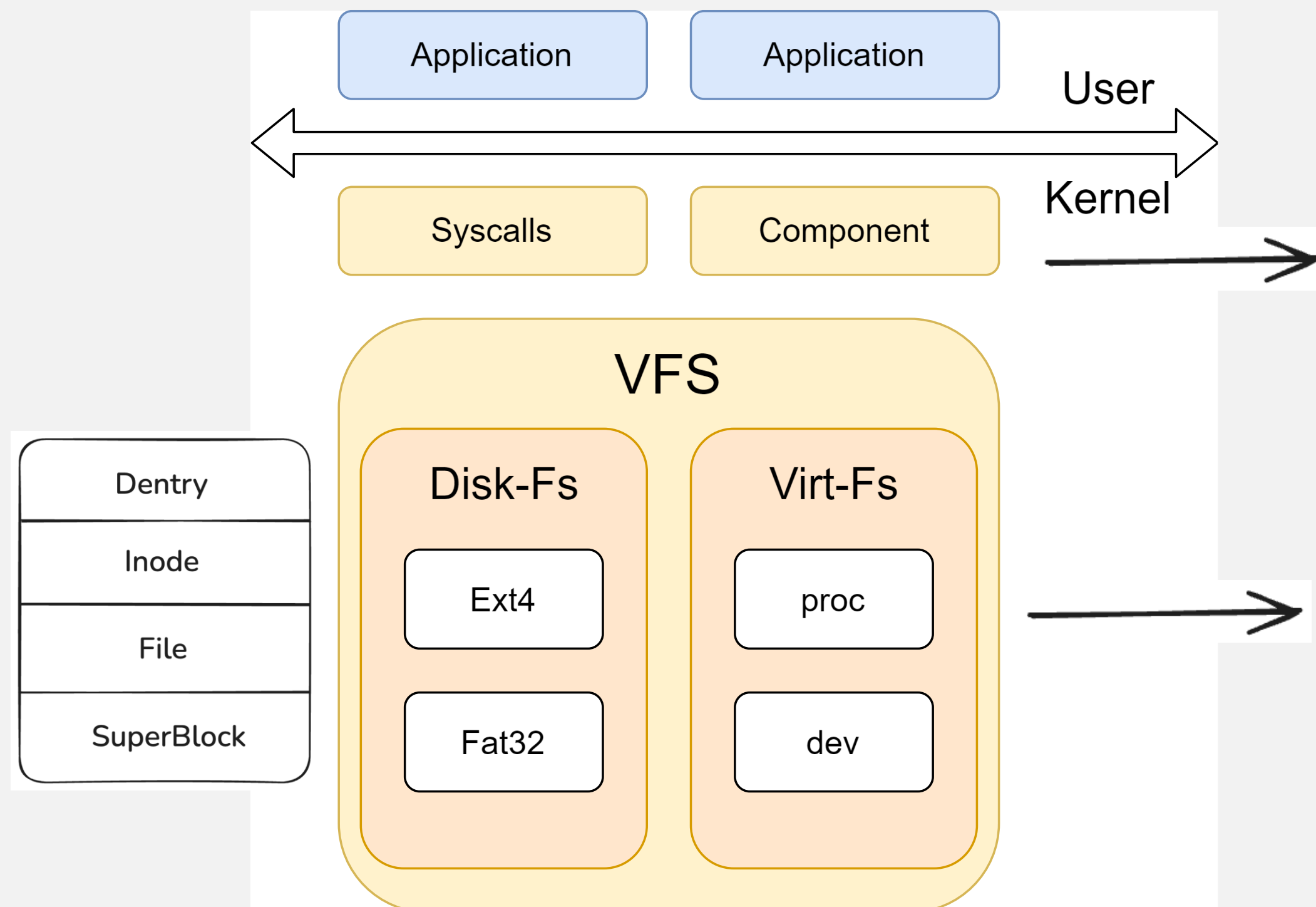
多线程响应速度提升
8.96倍！

Part.04

文件系统与驱动



4.1 文件系统框架

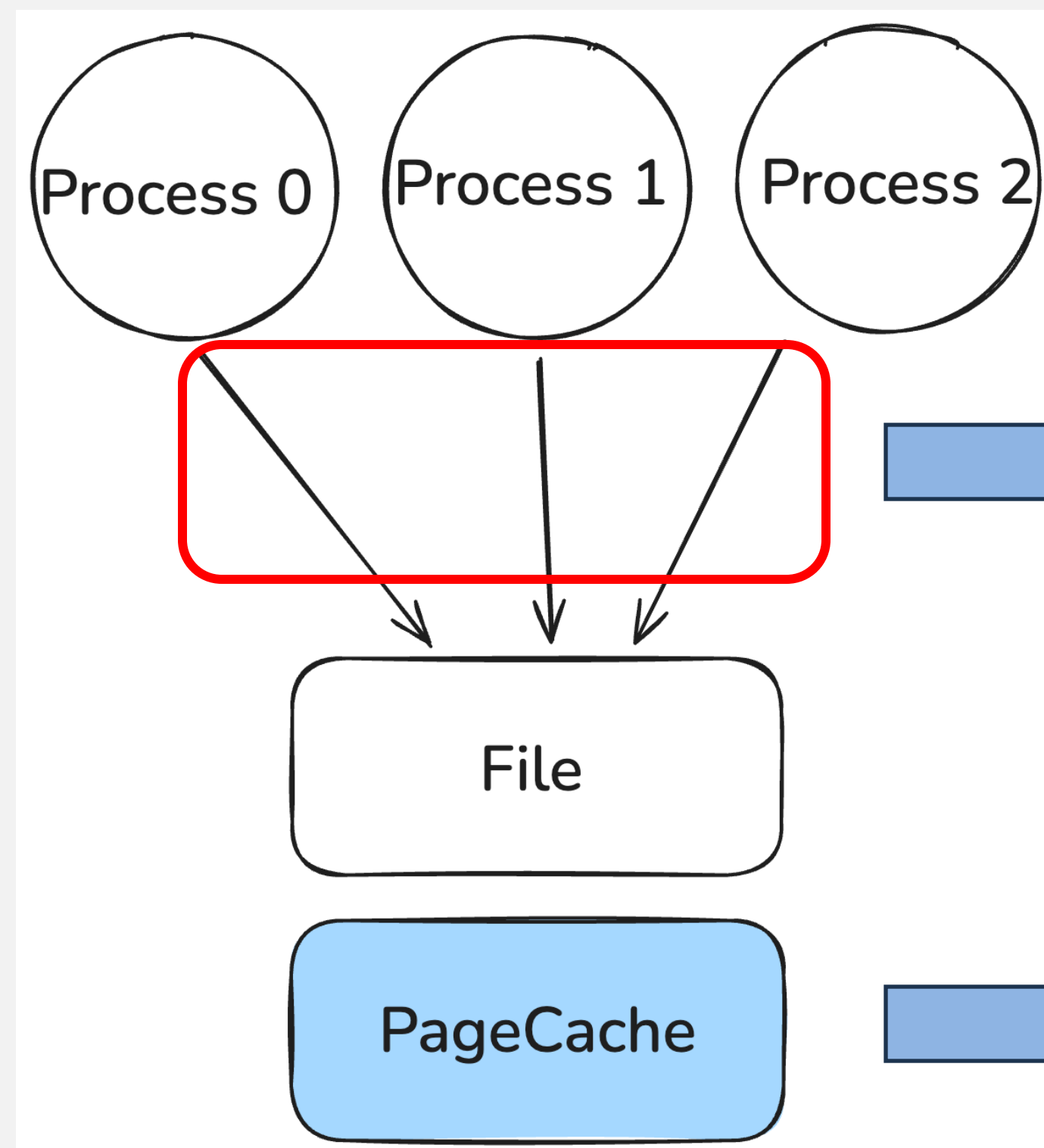


向上提供**完全一致**的接口

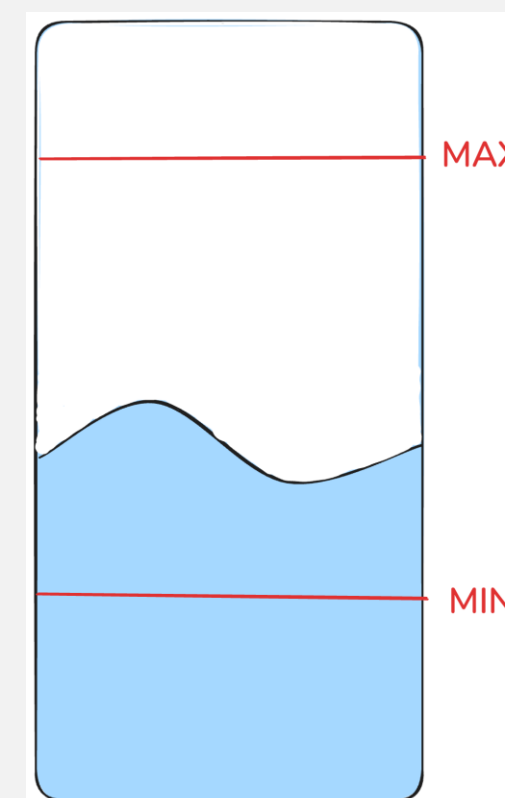
已支持 **5** 种文件系统!



4.2 页缓存



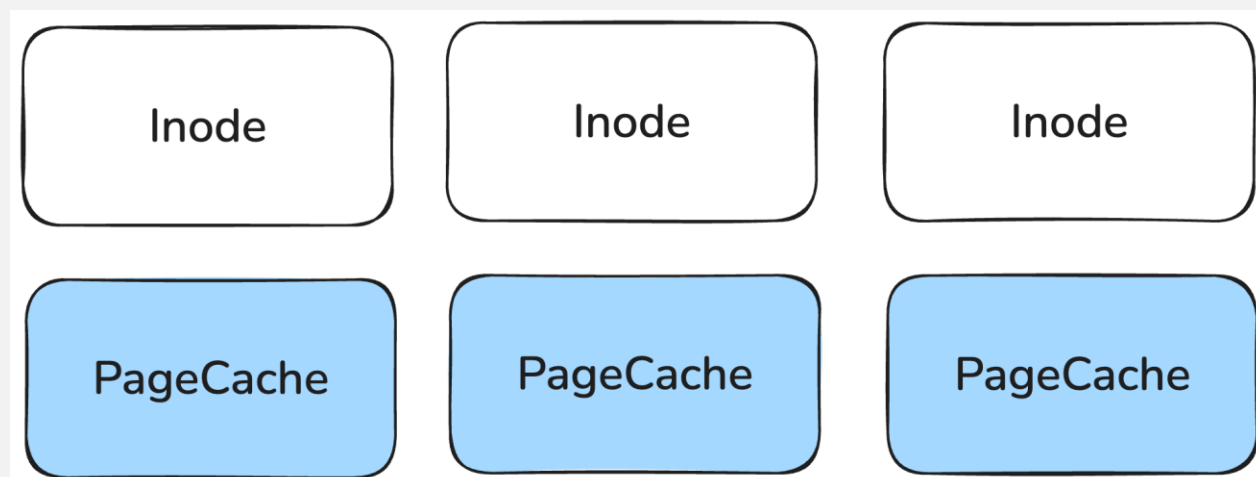
使用**读写锁**



原创**潮汐页缓存**
动态维护总容量

4.2 内存卫士：页缓存

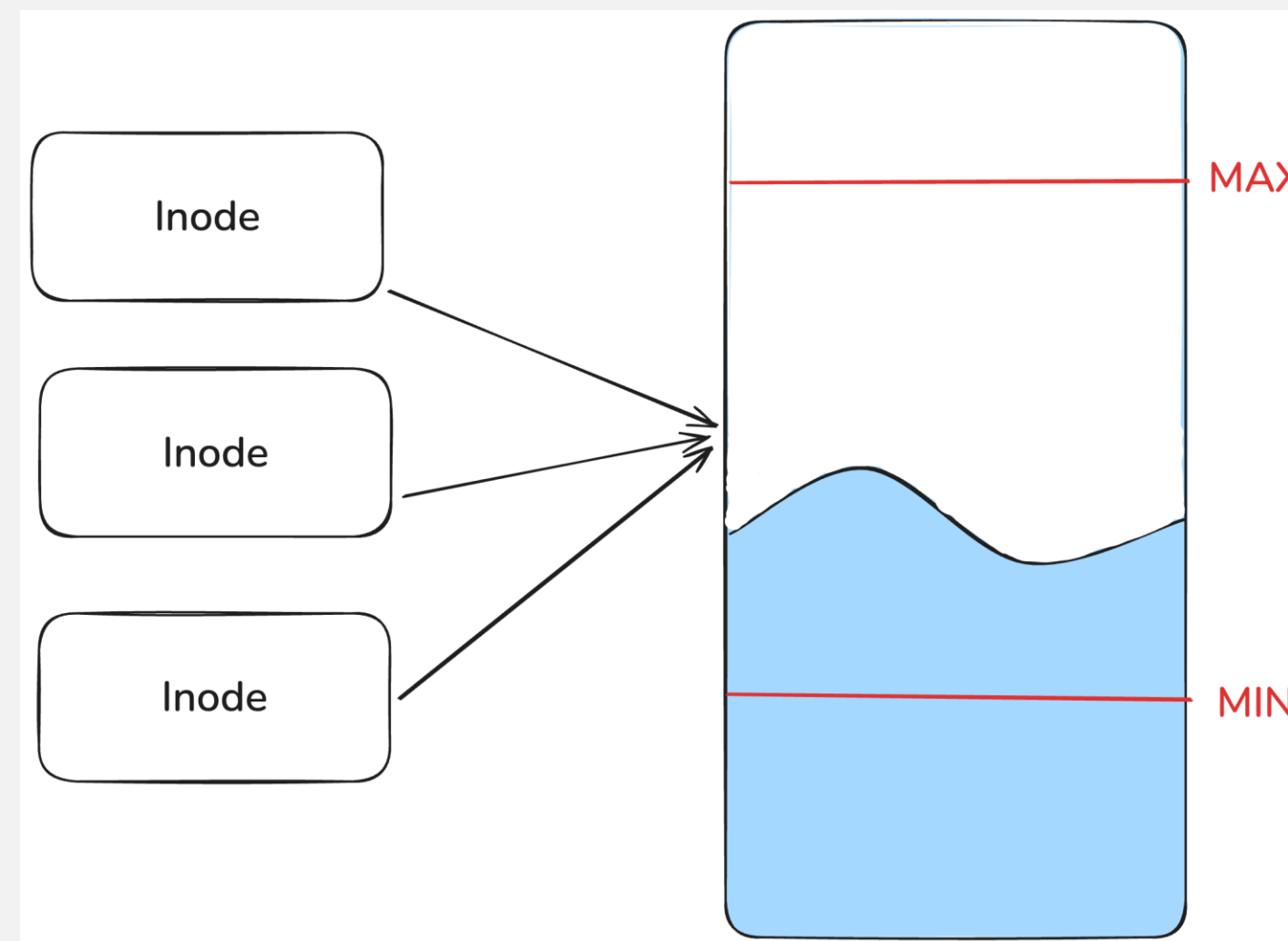
传统设计：
2024年一等奖Phoenix-OS



若文件一旦存在于VFS树中，
Pagecache将**永久占用**内存空间
直到用户删除



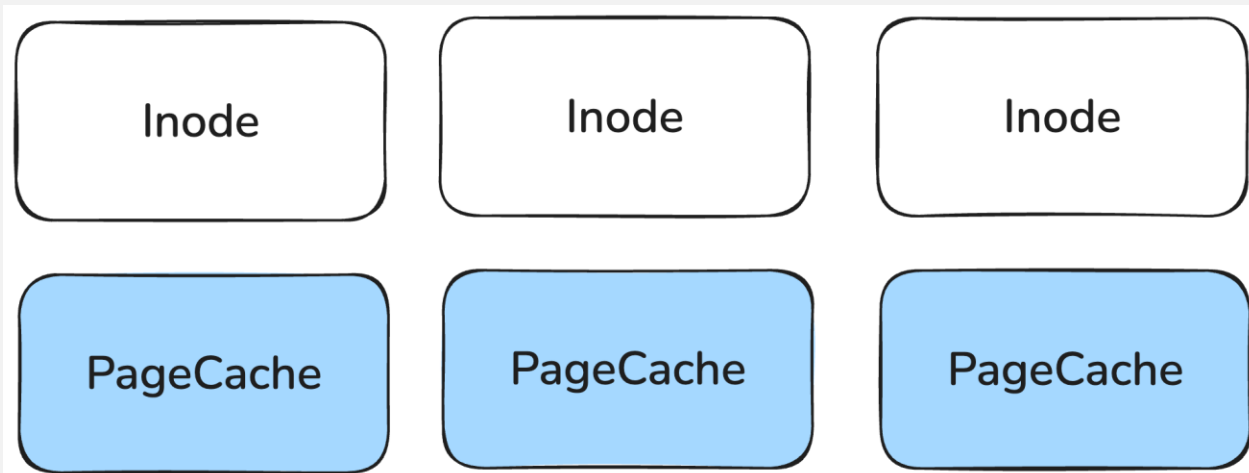
潮汐页缓存设计：
2025年NoAxiom-OS



永远保证页缓存内存占用在**阈值之内**

4.2 内存卫士：页缓存

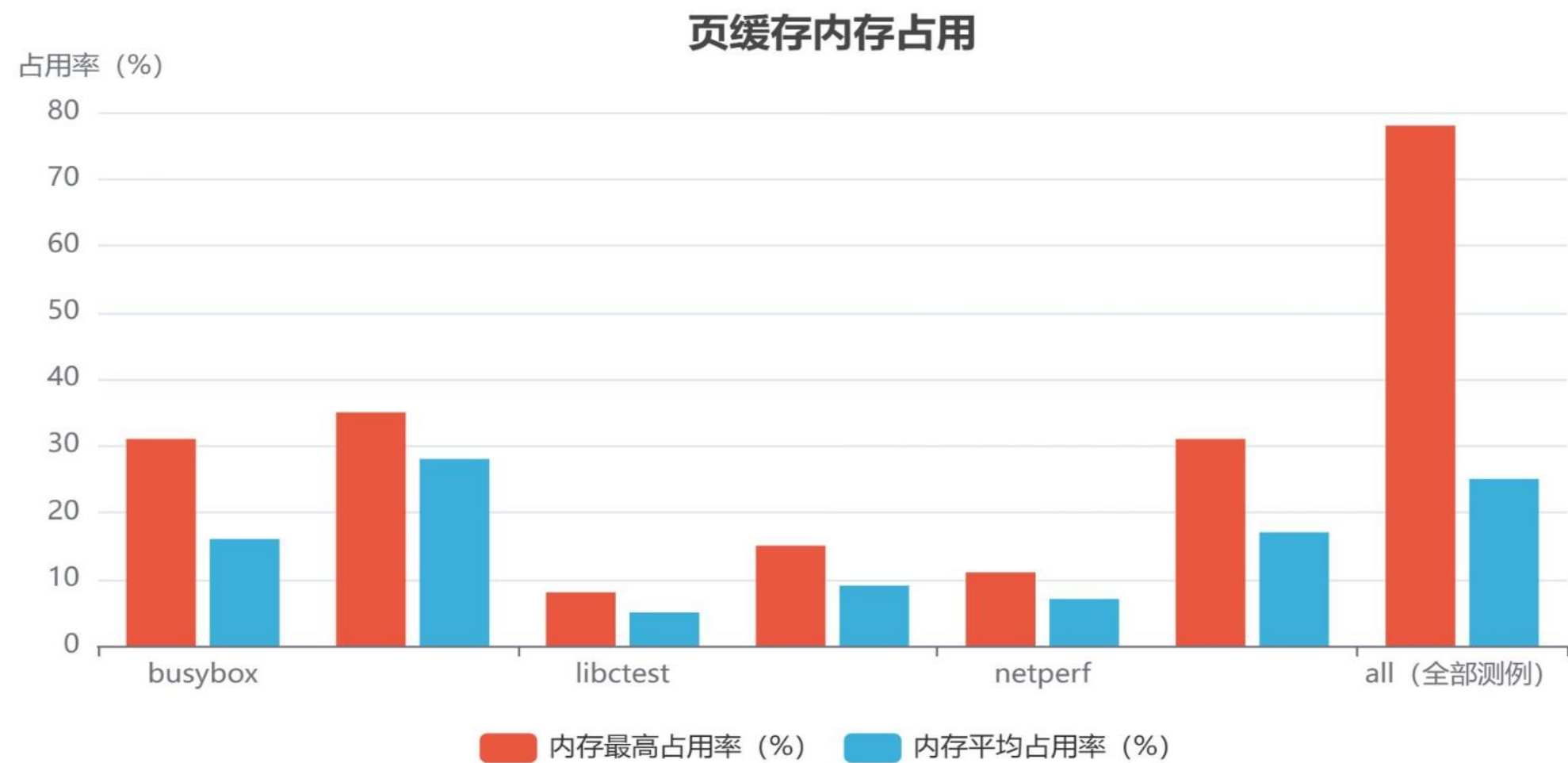
传统设计：
2024年特等奖Phoenix-OS



若文件一旦存在于VFS树中，
Pagecache将**永久**占用内存空间
直到用户删除



潮汐页缓存设计：
2025年NoAxiom-OS



永远保证页缓存内存占用在**阈值之内**，
平均内存占用全程不超过**30%**！

4.2 性能怪兽：页缓存

Qemu平台本地测试结果

```
Children see throughput for 4 initial writers = 3980.07 kB/sec
Parent sees throughput for 4 initial writers = 3736.22 kB/sec
Min throughput per process = 944.54 kB/sec
Max throughput per process = 1044.70 kB/sec
Avg throughput per process = 995.02 kB/sec
Min xfer = 917.00 kB
```

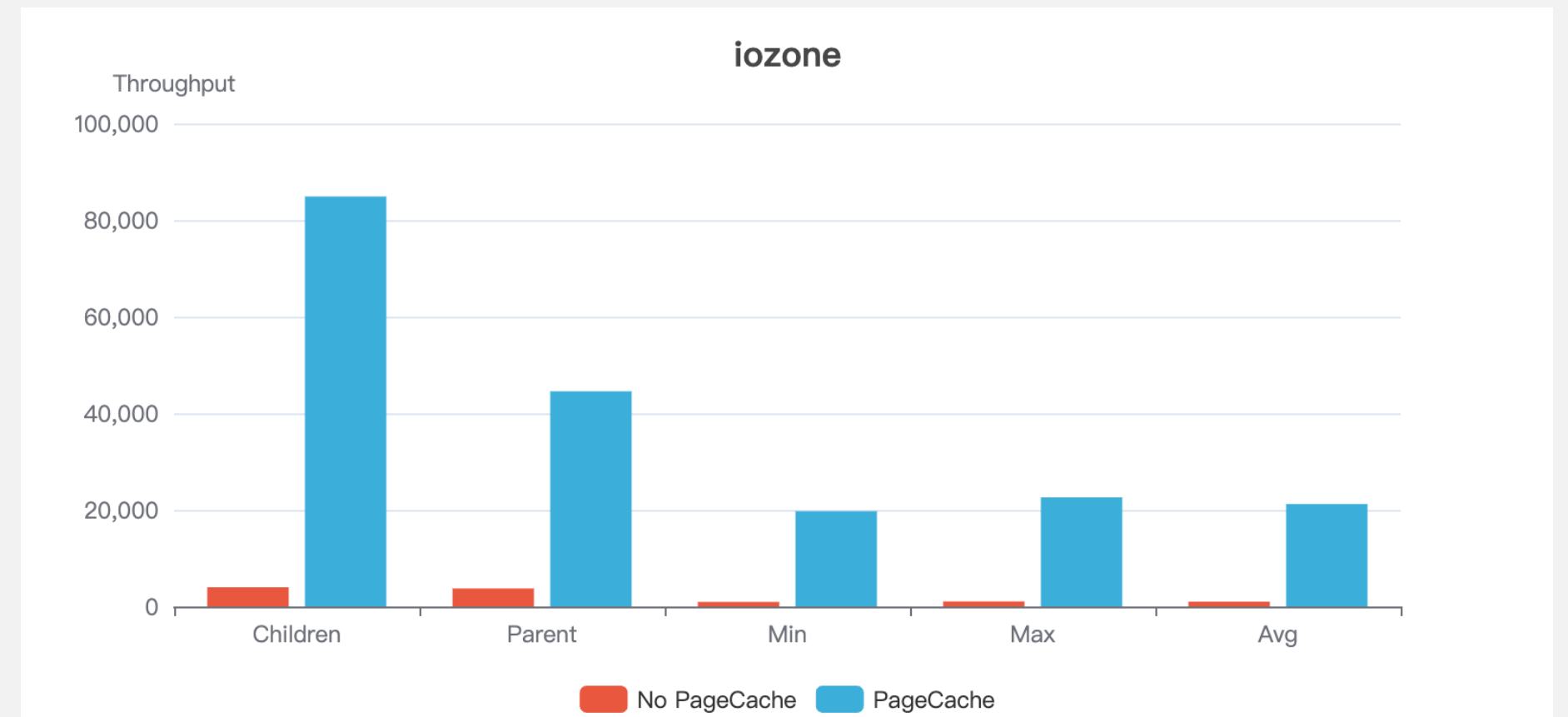
```
Children see throughput for 4 rewriters = 9272.78 kB/sec
Parent sees throughput for 4 rewriters = 8719.70 kB/sec
Min throughput per process = 1459.18 kB/sec
Max throughput per process = 2760.32 kB/sec
Avg throughput per process = 2318.19 kB/sec
Min xfer = 513.00 kB
```

无页缓存

```
Children see throughput for 4 initial writers = 84879.03 kB/sec
Parent sees throughput for 4 initial writers = 44541.83 kB/sec
Min throughput per process = 19725.98 kB/sec
Max throughput per process = 22593.38 kB/sec
Avg throughput per process = 21219.76 kB/sec
Min xfer = 661.00 kB
```

```
Children see throughput for 4 rewriters = 209664.22 kB/sec
Parent sees throughput for 4 rewriters = 76162.14 kB/sec
Min throughput per process = 0.00 kB/sec
Max throughput per process = 209664.22 kB/sec
Avg throughput per process = 52416.05 kB/sec
Min xfer = 0.00 kB
```

有页缓存



I/O-Zone测试中，吞吐量提升至原先的**21**倍！

在决赛排行榜中，该测例NoAxiom位列**前二**



4.3 性能基石：块缓存

```
Children see throughput for 4 initial writers = 723.74 kB/sec
Parent sees throughput for 4 initial writers = 714.74 kB/sec
Min throughput per process = 179.09 kB/sec
Max throughput per process = 182.96 kB/sec
Avg throughput per process = 180.94 kB/sec
Min xfer = 1001.00 kB
```

```
Children see throughput for 4 rewriters = 1258.79 kB/sec
Parent sees throughput for 4 rewriters = 1247.26 kB/sec
Min throughput per process = 309.40 kB/sec
Max throughput per process = 319.09 kB/sec
Avg throughput per process = 314.70 kB/sec
Min xfer = 994.00 kB
```

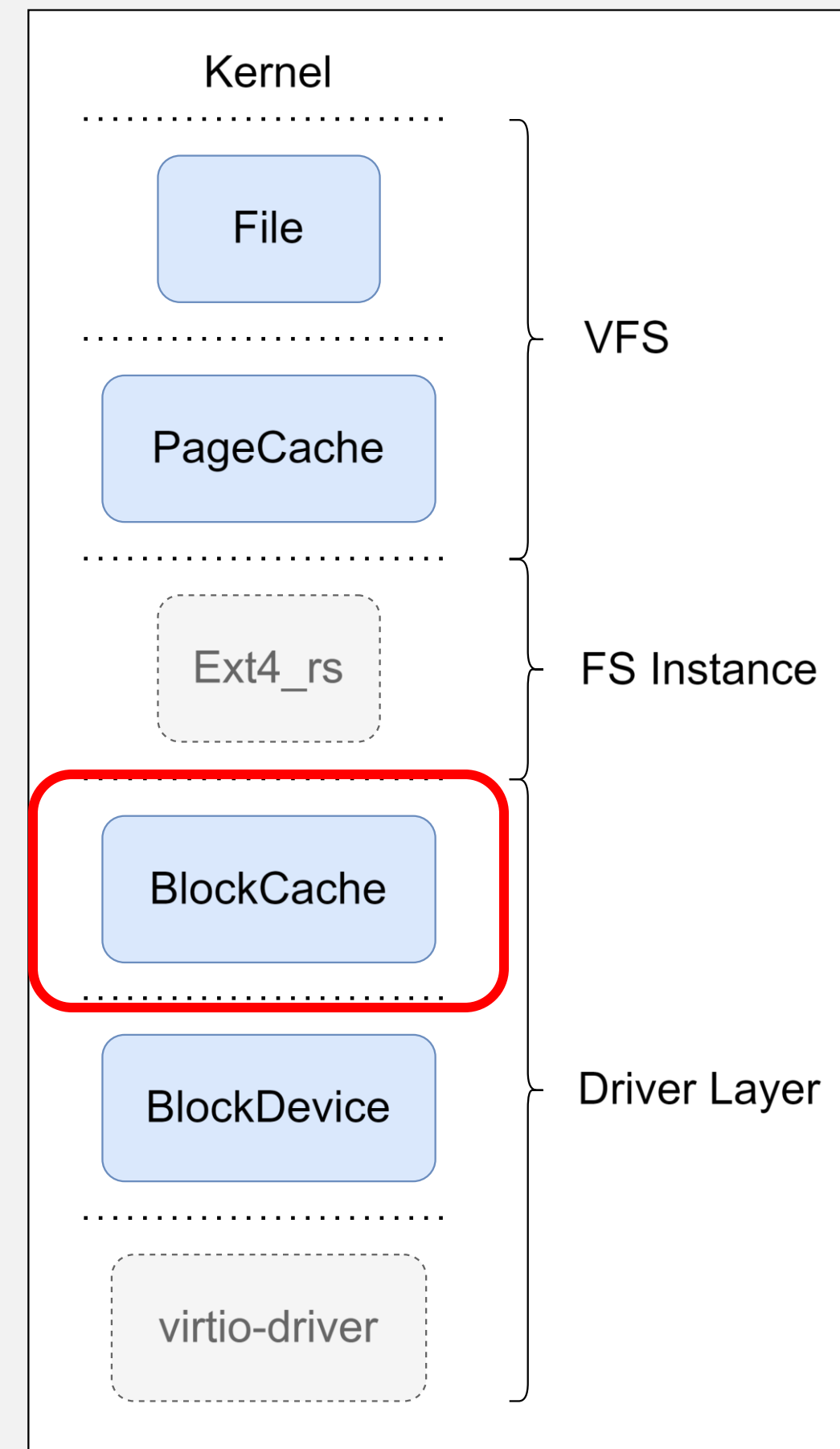
无块缓存

```
Children see throughput for 4 initial writers = 3980.07 kB/sec
Parent sees throughput for 4 initial writers = 3736.22 kB/sec
Min throughput per process = 944.54 kB/sec
Max throughput per process = 1044.70 kB/sec
Avg throughput per process = 995.02 kB/sec
Min xfer = 917.00 kB
```

```
Children see throughput for 4 rewriters = 9272.78 kB/sec
Parent sees throughput for 4 rewriters = 8719.70 kB/sec
Min throughput per process = 1459.18 kB/sec
Max throughput per process = 2760.32 kB/sec
Avg throughput per process = 2318.19 kB/sec
Min xfer = 513.00 kB
```

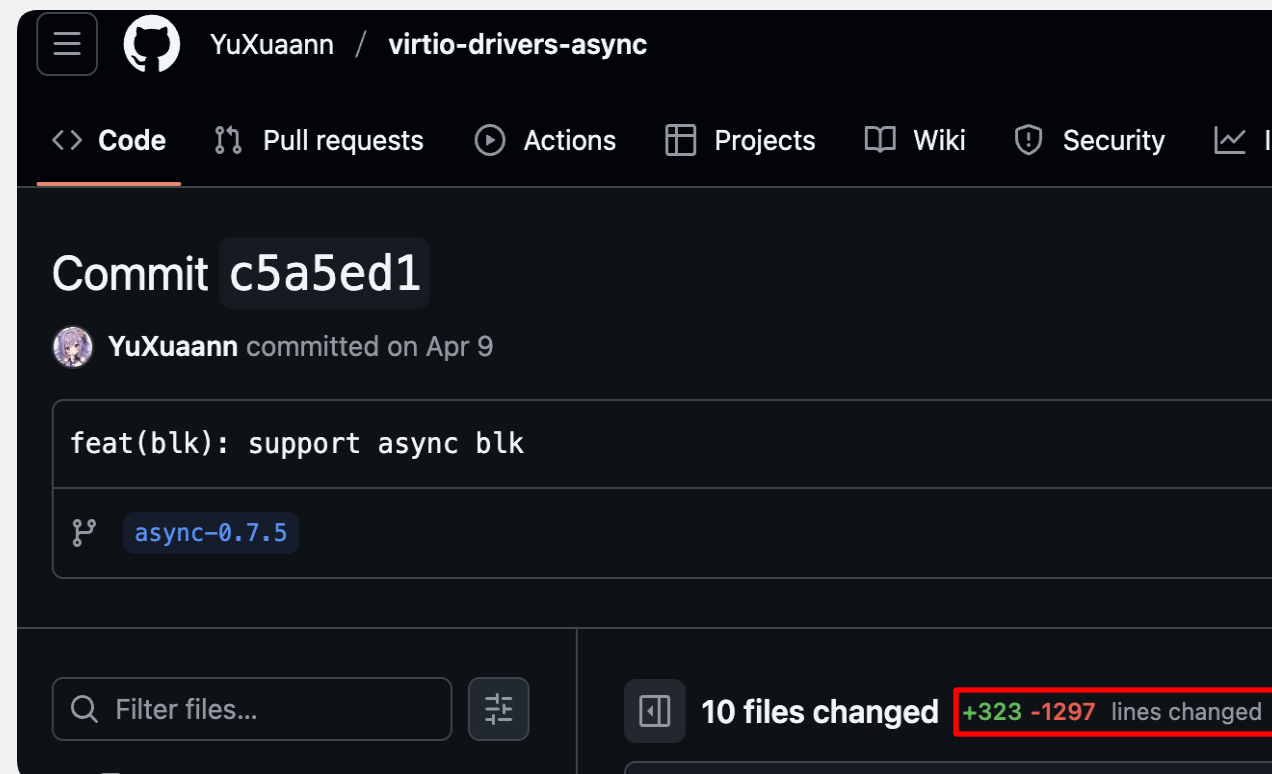
有块缓存

吞吐量
提升
730%



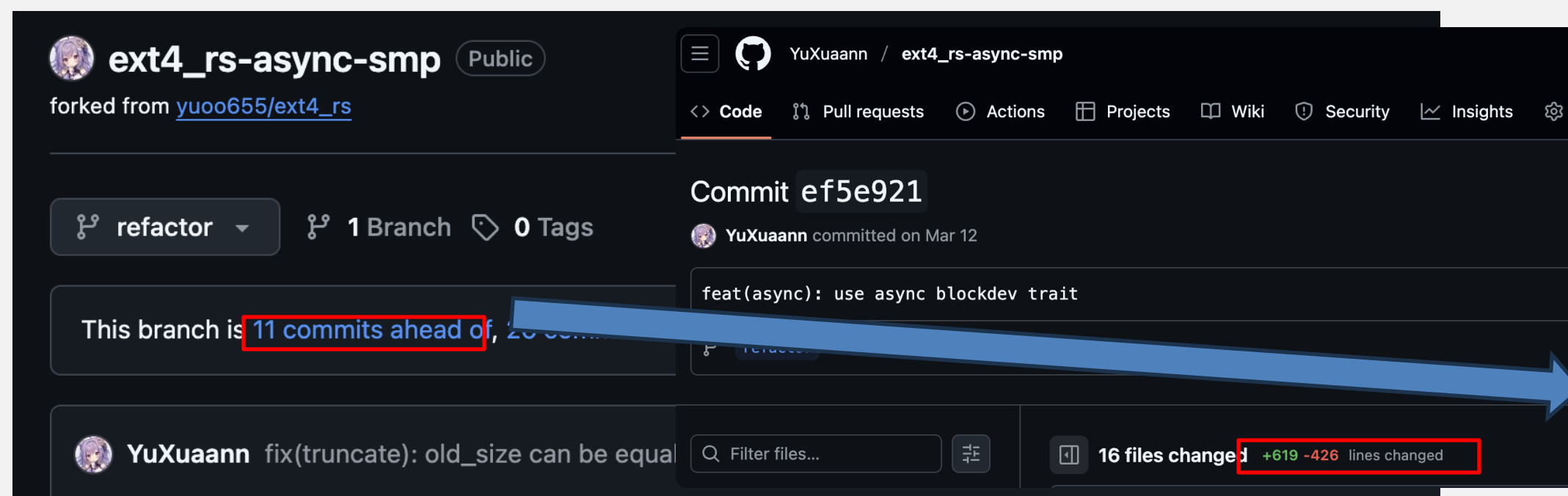


4.4 性能底座：异步驱动&异步EXT4文件系统



Fork开源virtio-driver仓库
实现**异步**特征块设备驱动
支持通过外部**中断**唤醒！

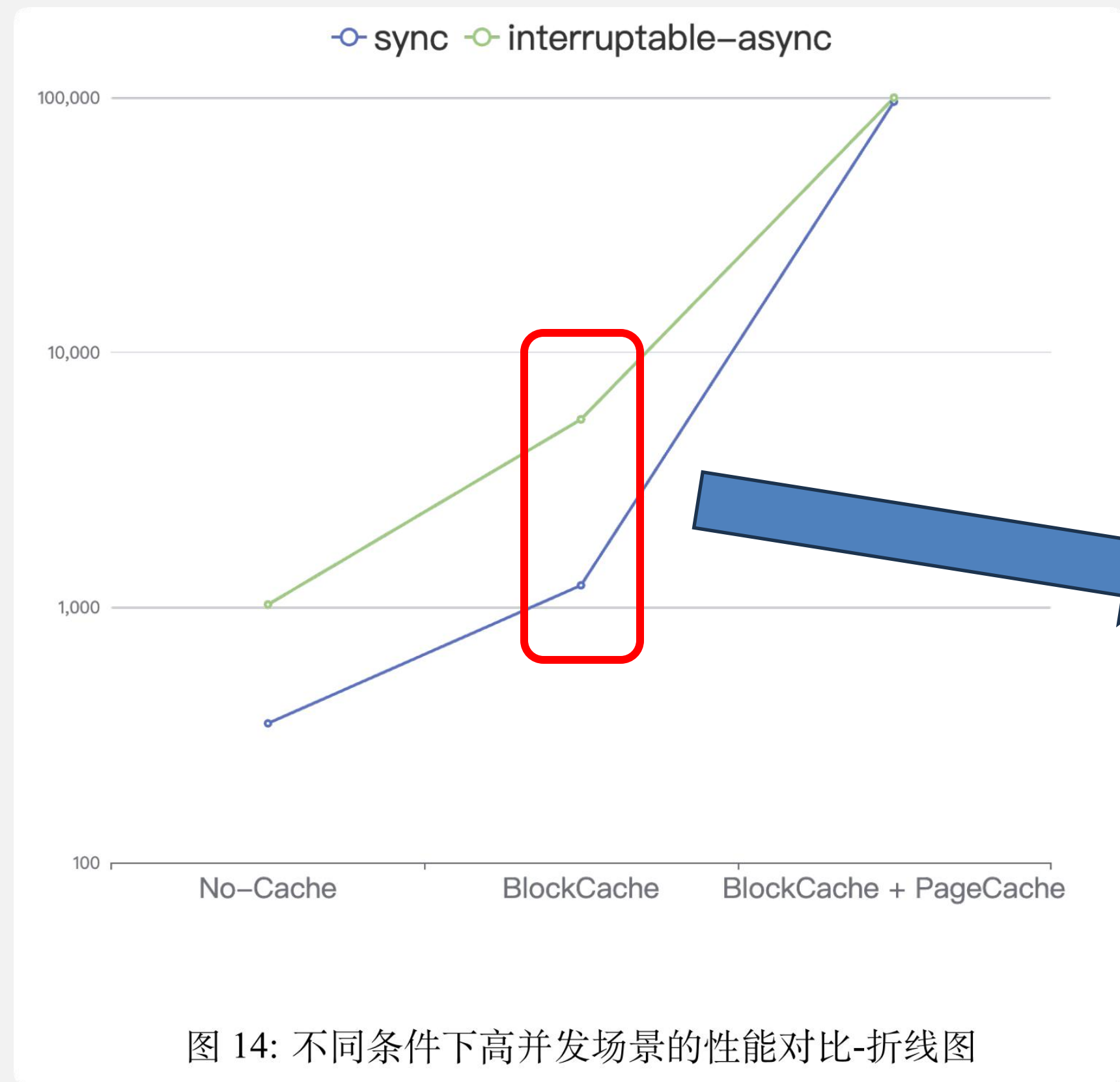
改动**1600+**行



Fork官方推荐开源仓库ext4_rs
实现**异步**特征EXT4文件系统

提交**11**次，累计改
动**1000+**行

4.4 性能底座：异步驱动&异步EXT4文件系统



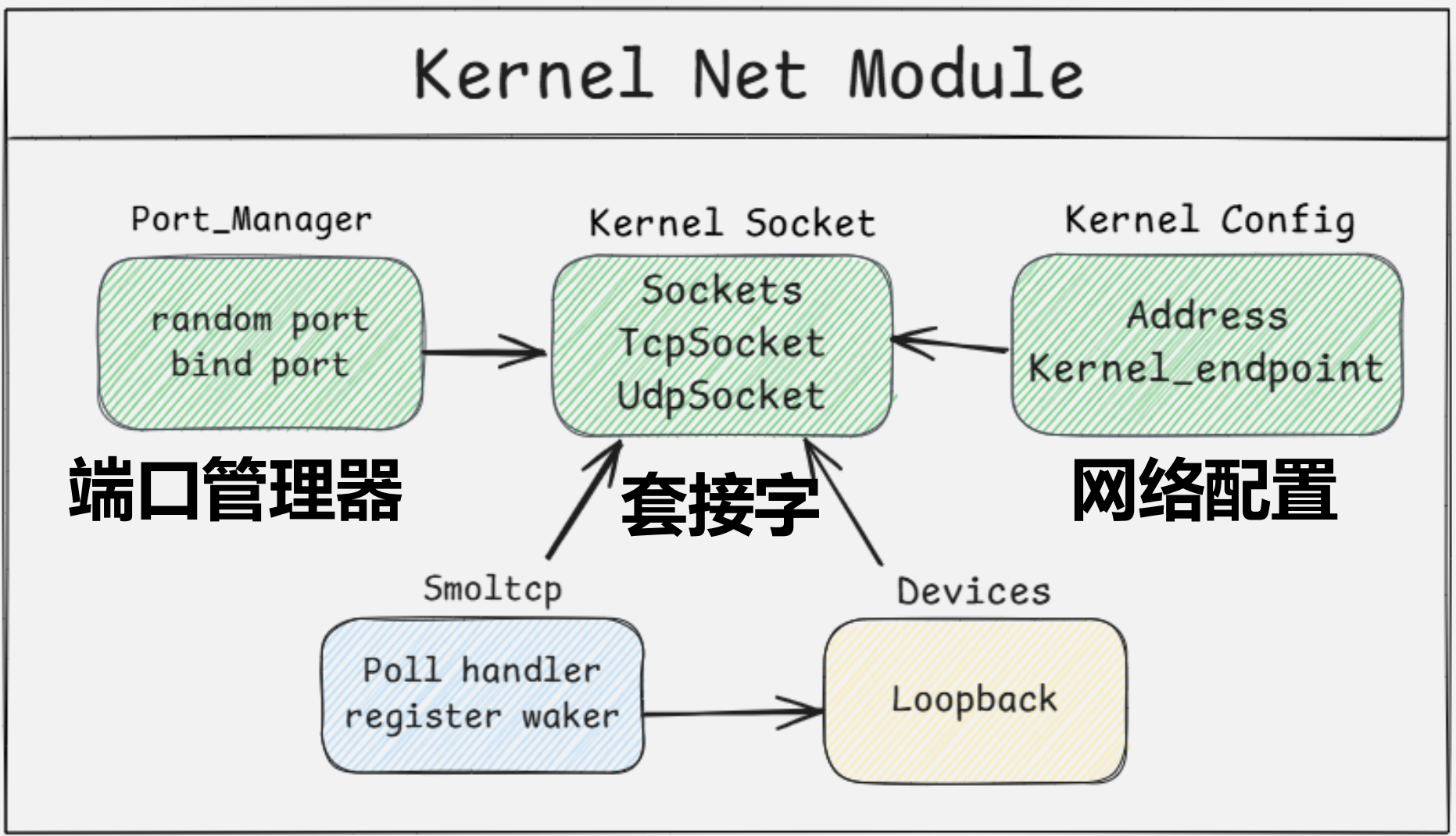
在初赛文档设计的高并发大文件传输场景下

基于异步特征的FS在页缓存外具有**200%**以上I/O性能提升

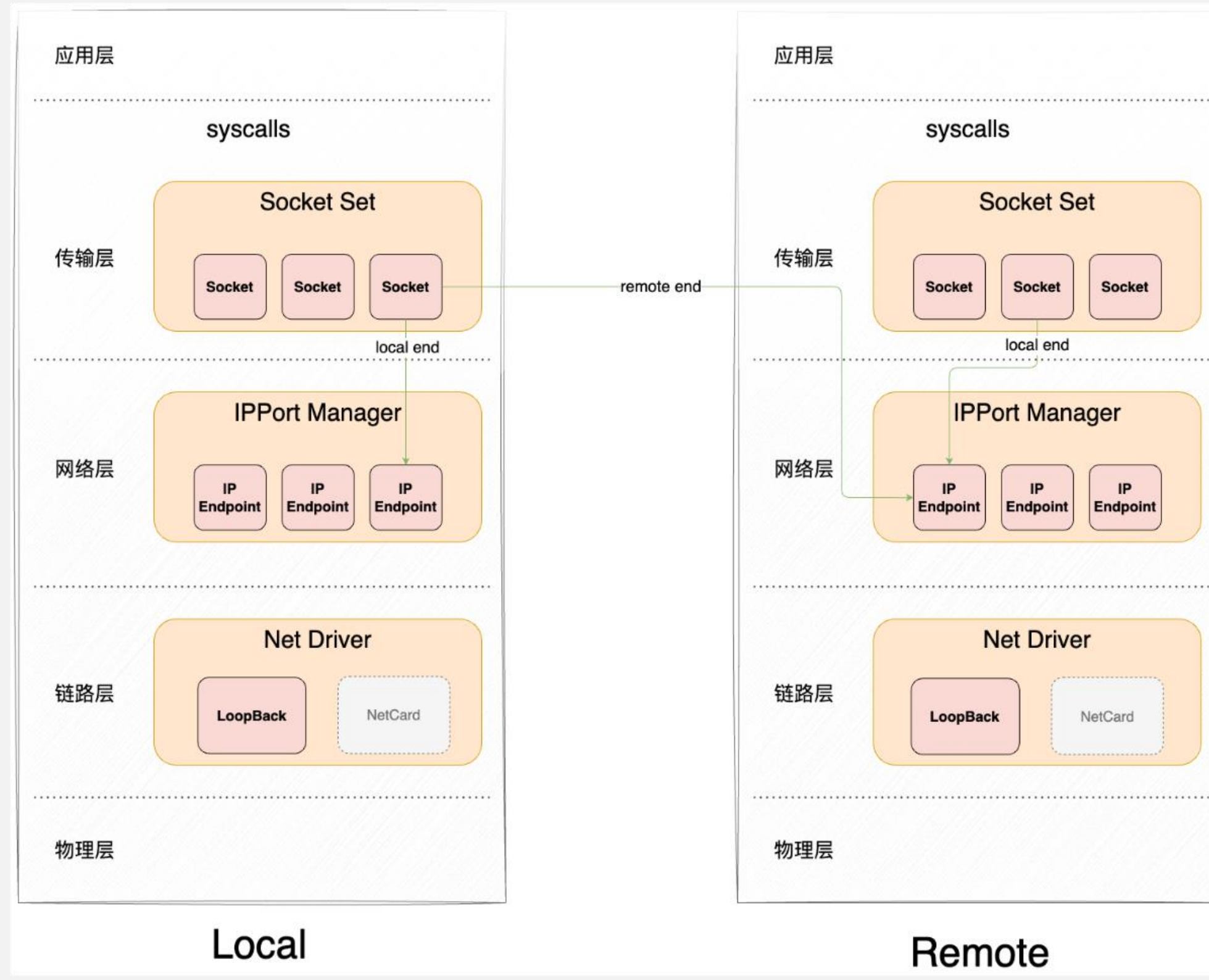
Part.05

网络模块

5.1 网络模块概述

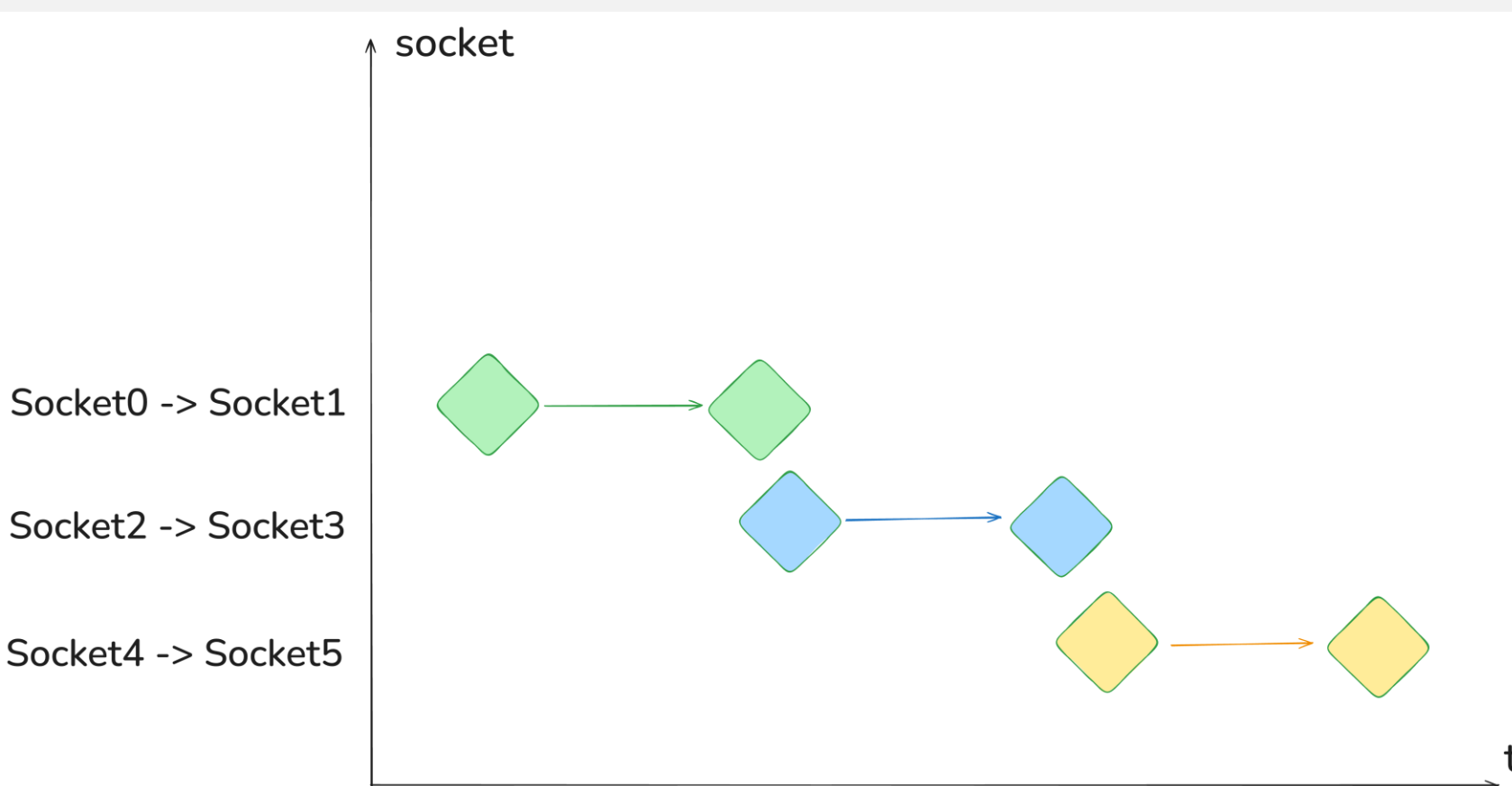


支持TCP、UDP套接字

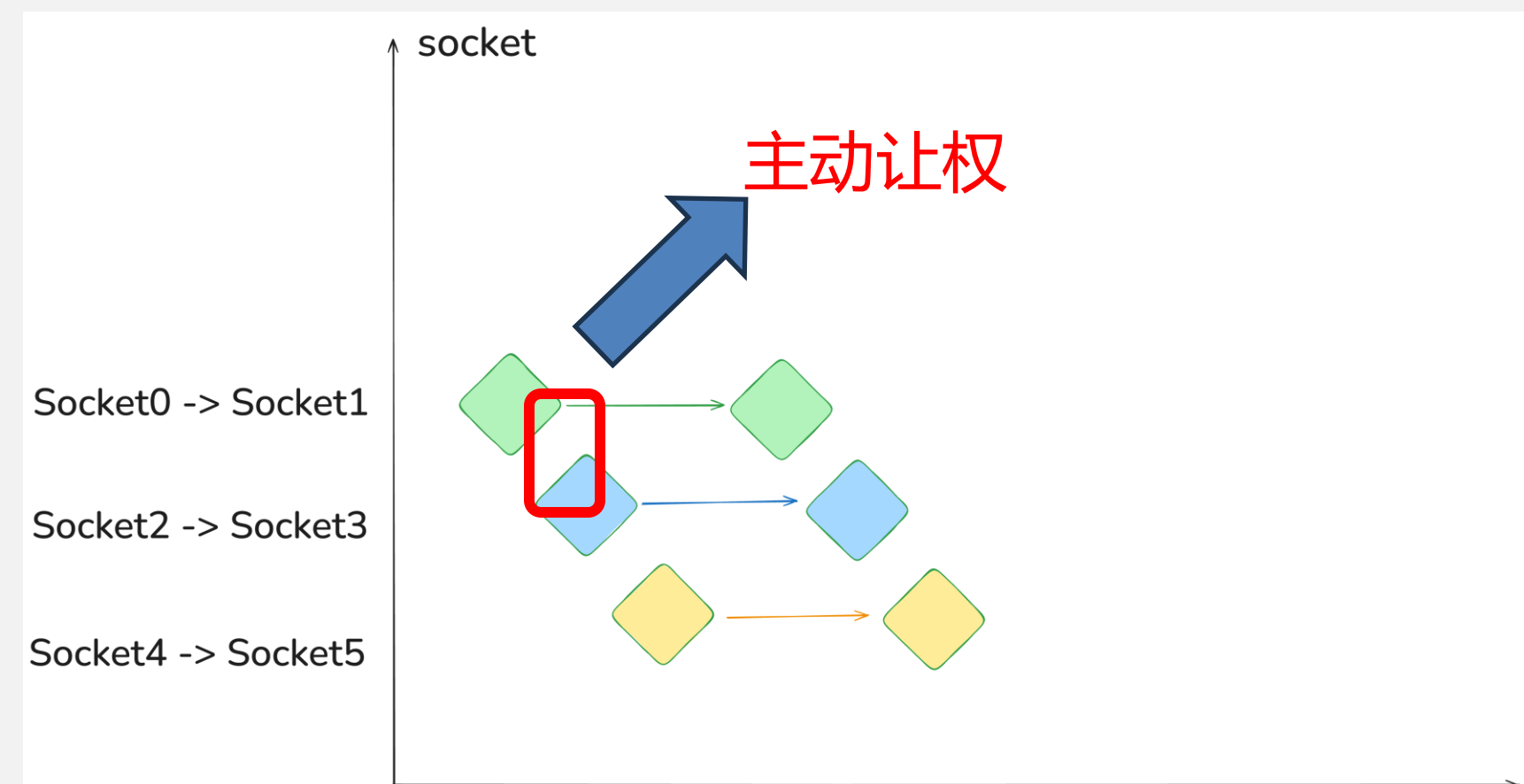


5.2 网络高并发示例

传统设计：
阻塞式等待连接

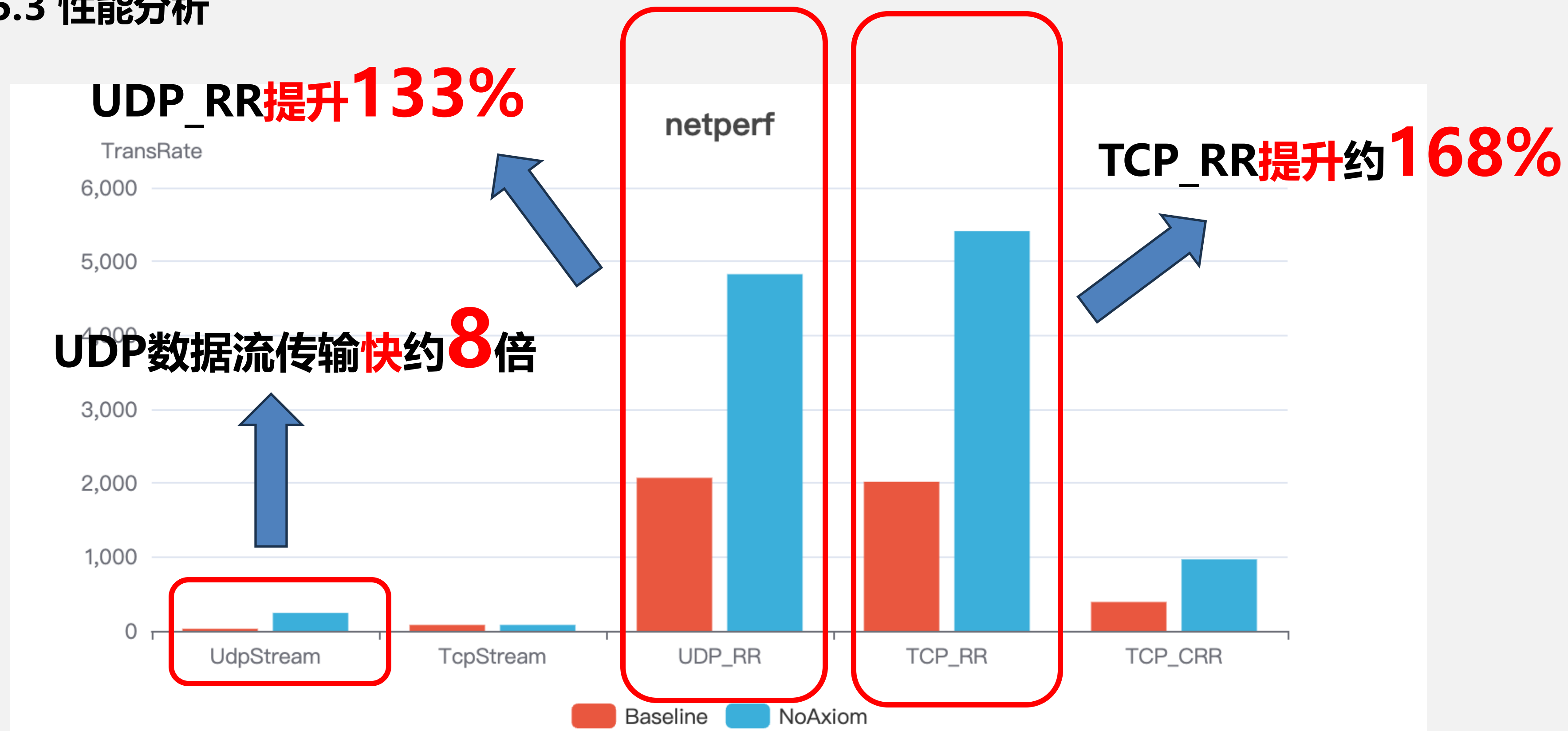


无栈协程异步调度：





5.3 性能分析



Part.06

总结

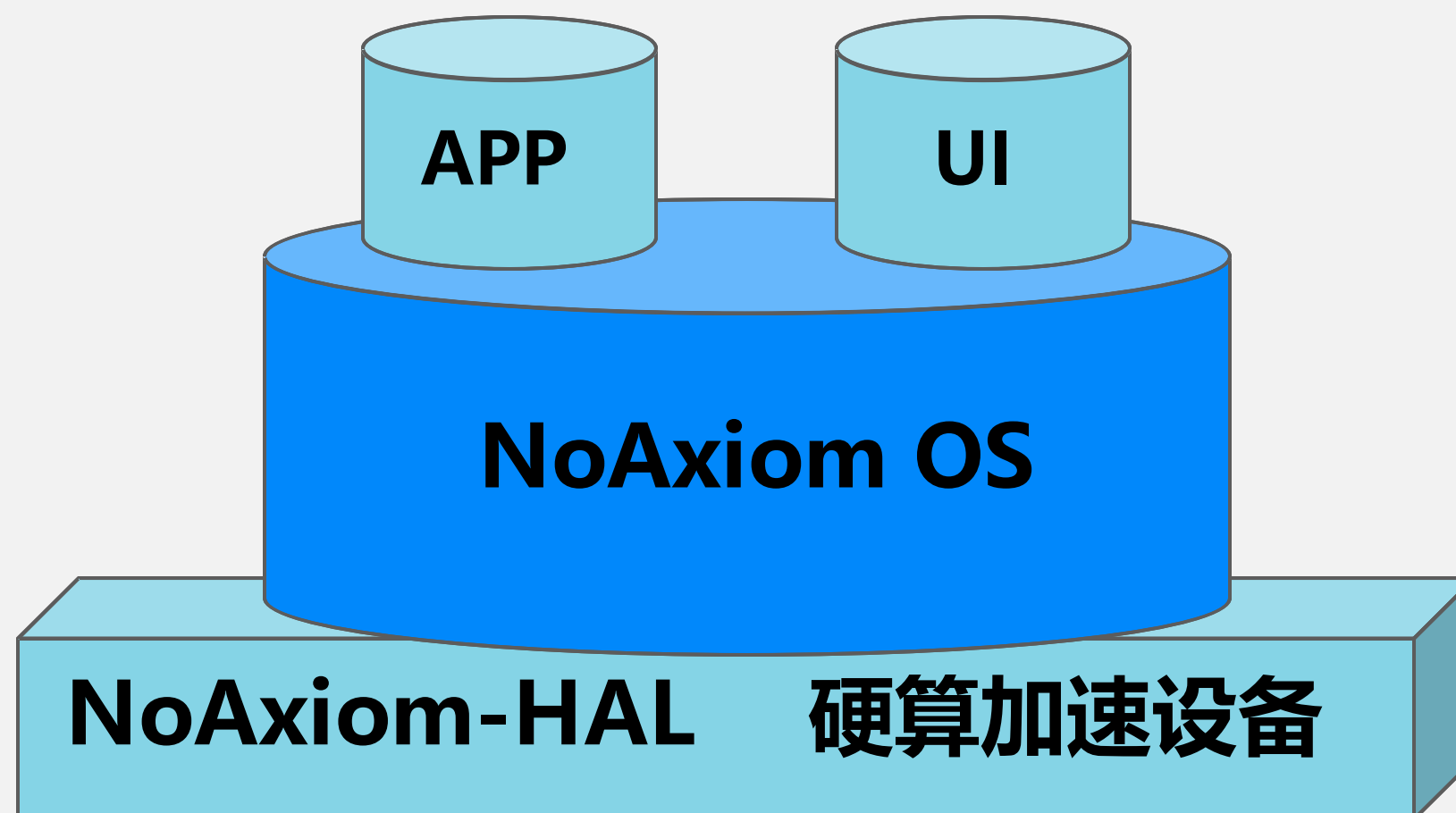


6.1 与参考作品对比和优势

| NoAxiom | Pantheon | 优势 |
|---------------------------------------|---------------|--|
| 硬件抽象层 支持双架构 | 单Riscv架构 | 令系统具备 跨架构 运行稳定性 |
| 多级实时 调度队列 | 朴素FIFO调度 | 提升 实时响应 速度 |
| 异步网络模块 全局端口管理器实现 端口复用 | 异步网络模块 | 更高的网络数据处理的 并发性能 更高的CPU的 有效使用率 |
| 全局定时器队列 IO多路复用 异步文件系统及驱动 | 同步文件系统 及驱动 | 更高的系统的 实时性能 高并发场景拥有 更高吞吐率 |

未来展望

- 完善NoAxiom-HAL库，引入KVM虚拟化功能
- 添加完整的图形化界面，提升用户交互体验
- 完善多核一致性维护，添加众核调度器支持
- 探索异构融合计算技术，实现软硬协同加速



谢谢!