OceanBase 2.0 性能突破

颜然/韩富晟

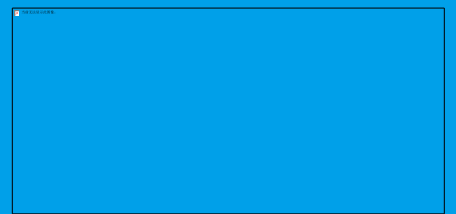
蚂蚁金服 ANT FINANCIAL

OceanBase

目录

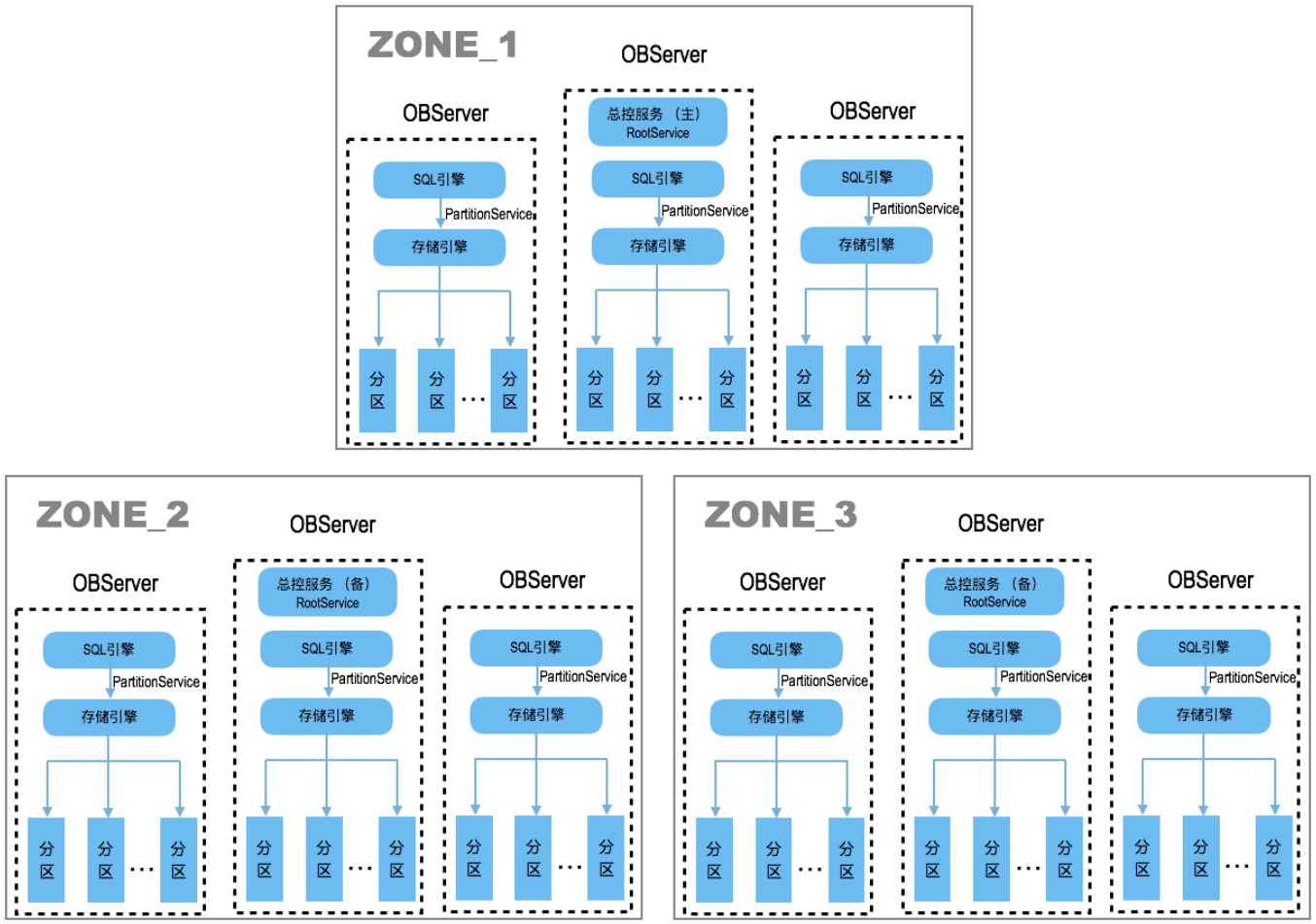
OceanBase

01 | OceanBase



OceanBase

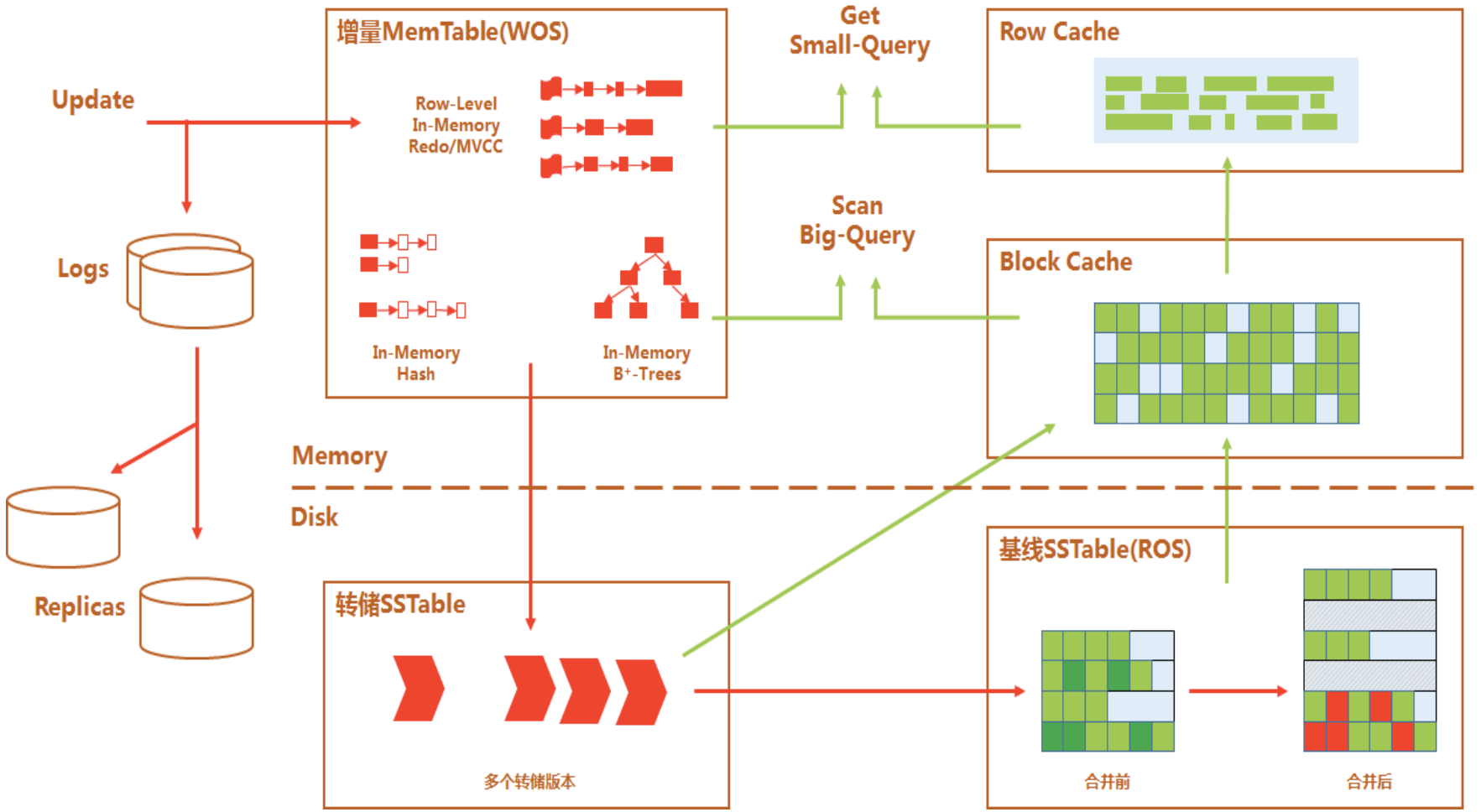
金融级分布式关系数据库



- 分布式集群线性扩展，不依赖共享存储
- 多副本高可用，金融级可靠性
- 使用通用硬件，不依赖高端设备
- 新存储和事务引擎，低成本高性能

OceanBase存储引擎

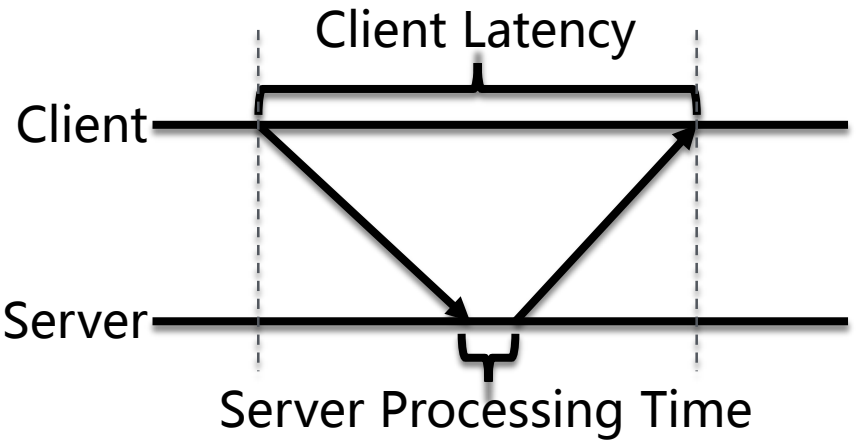
- 动态修改写内存
- 静态数据无修改
- 无随机写，SSD友好
- 批量写，高压缩支持
- 强数据校验

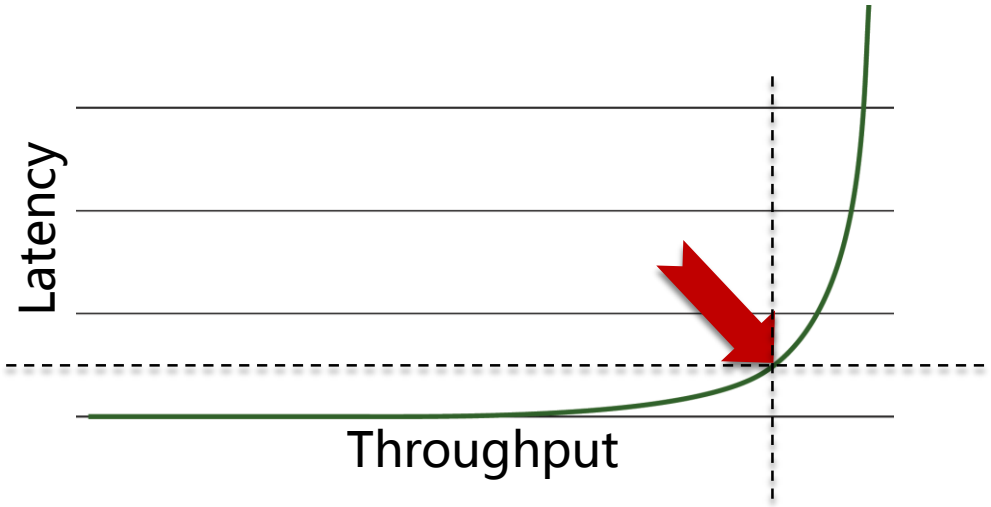


02 性能目标

性能目标

- 系统两个指标：**延迟**（Latency）和**吞吐量**（Throughput）
- 应用请求执行延迟需要控制在合理水平
- 尽可能增加系统吞吐量

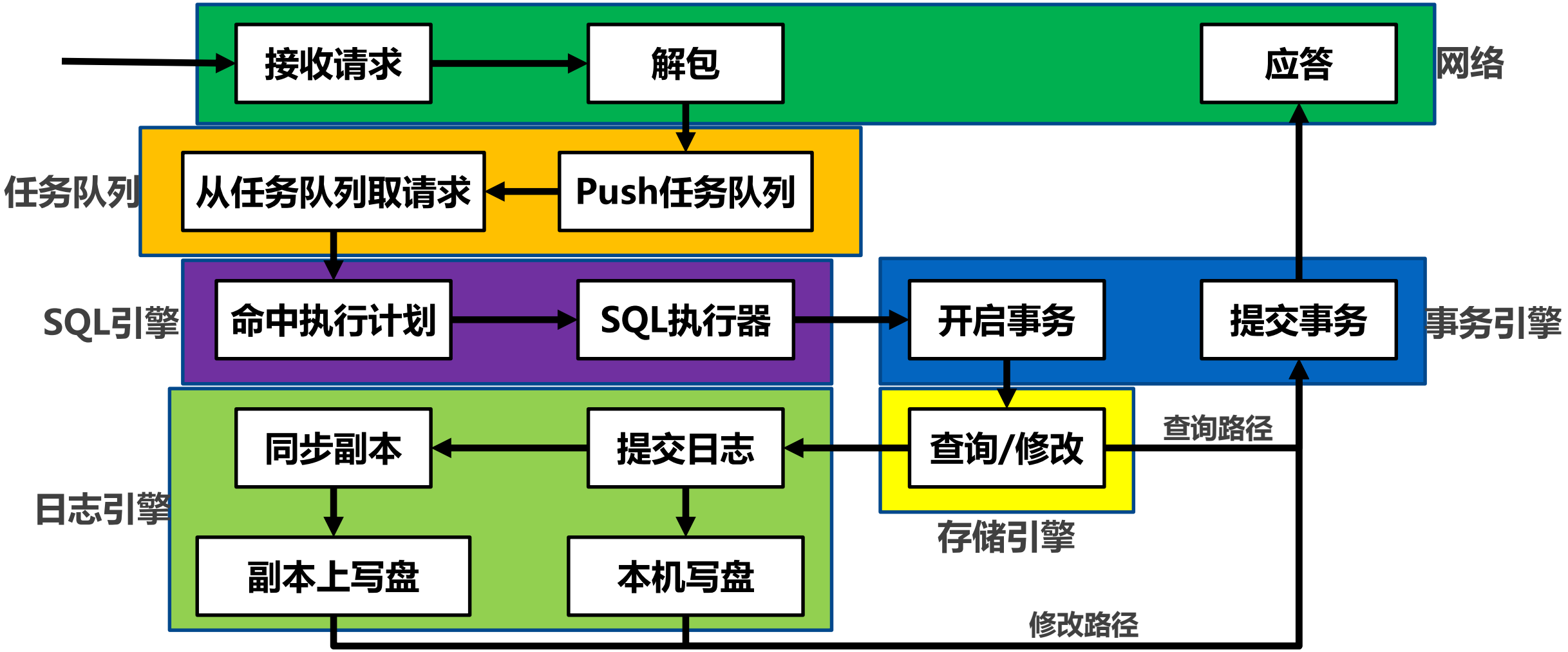




性能优化工作

- 双11场景下系统的瓶颈是 CPU 资源
- 两方面工作：
    - 优化语句执行消耗指令数 (Instructions / SQL)
    - 优化系统执行指令的效率 (Cycles / Instruction)

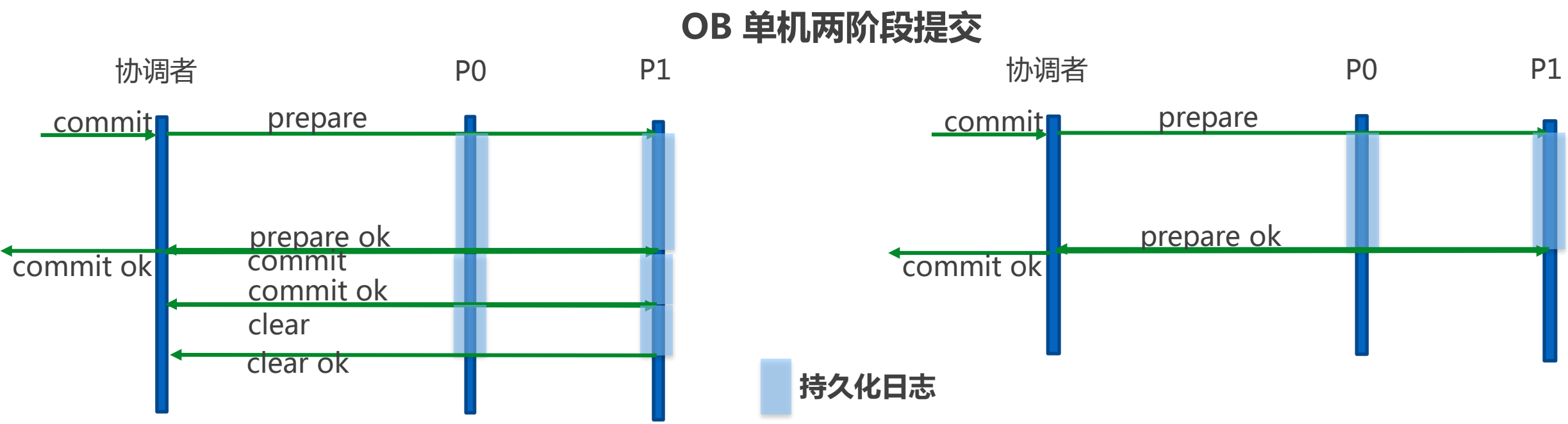
语句执行流程


接收请求
解包
应答
网络
任务队列
从任务队列取请求
Push任务队列
SQL引擎
命中执行计划
SQL执行器
开启事务
提交事务
事务引擎
同步副本
提交日志
查询/修改
查询路径
日志引擎
存储引擎
副本上写盘
本机写盘
修改路径

03 优化CPU开销

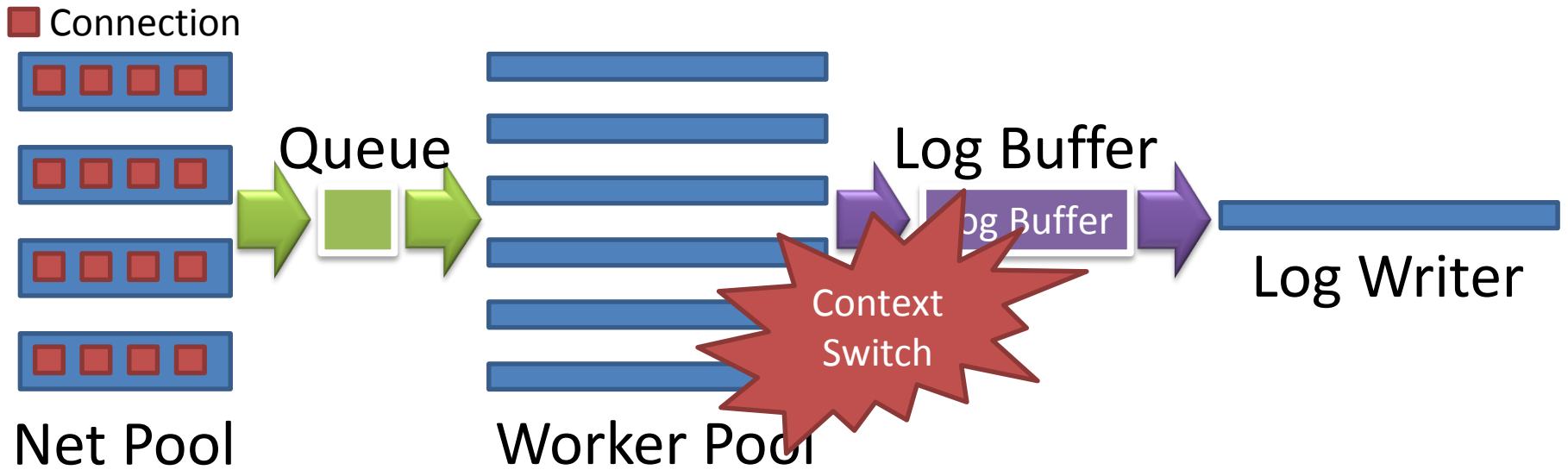
两阶段提交协议优化

- 结合事务与日志，事务 PREPARE 成功后内存不用持久保存状态，按需从日志中查询
- COMMIT 状态持久化转换成后台批量完成



Commit 异步化

- 事务提交等待日志持久化成功会导致大量 Context Switch
- 异步化后 Worker 不等待继续执行队列中下一个请求，日志持久化成功后会异步回调



04 优化系统扩展性

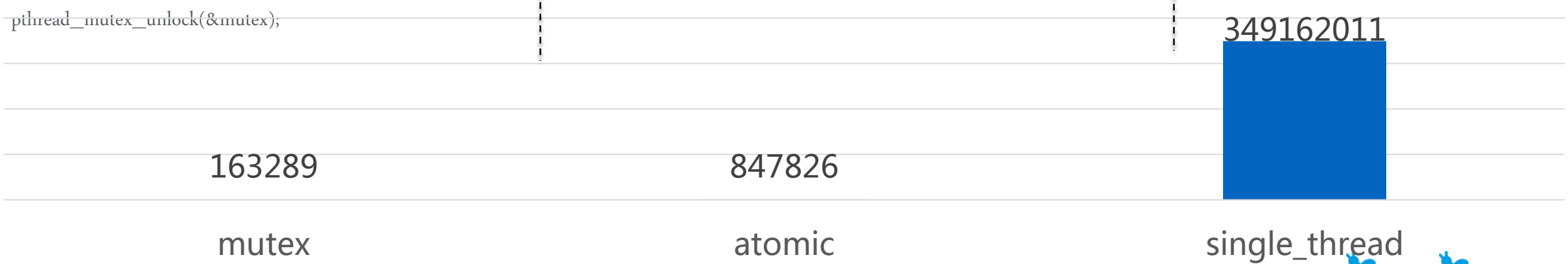
扩展性问题

在 64 核机器上
64 个线程同时给一个整型变量加一 vs. 1 个线程做加一操作

```
volatile int64_t count;
pthread_t mutex;

// 64 threads
while (true) {
    pthread_mutex_lock(&mutex);
    ++count;
    pthread_mutex_unlock(&mutex);
}
```

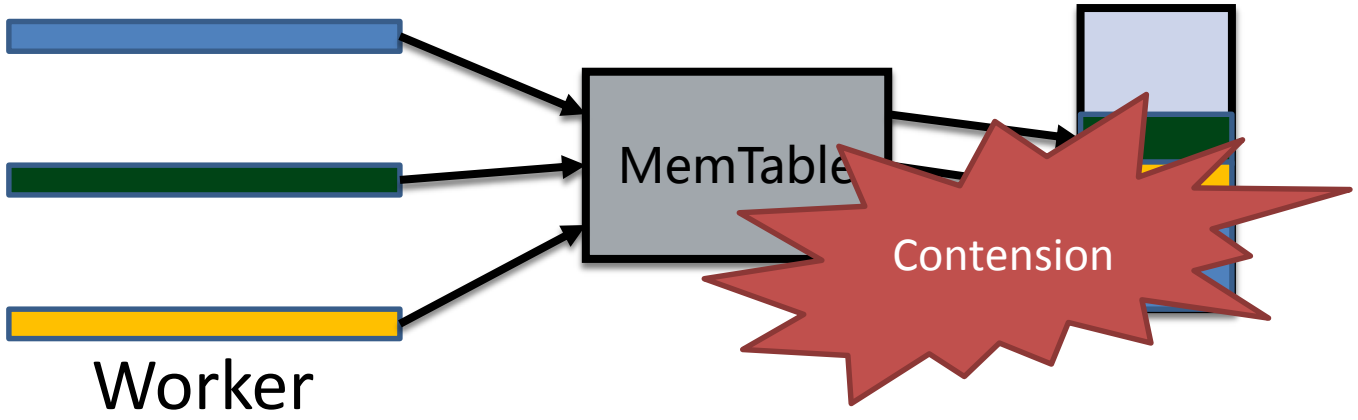
```
volatile int64_t count;

// 64 threads
while (true) {
    ___sync_add_and_fetch(&count, 1);
}
```

```
volatile int64_t count;

while (true) {
    ++count;
}
```



内存分配器优化

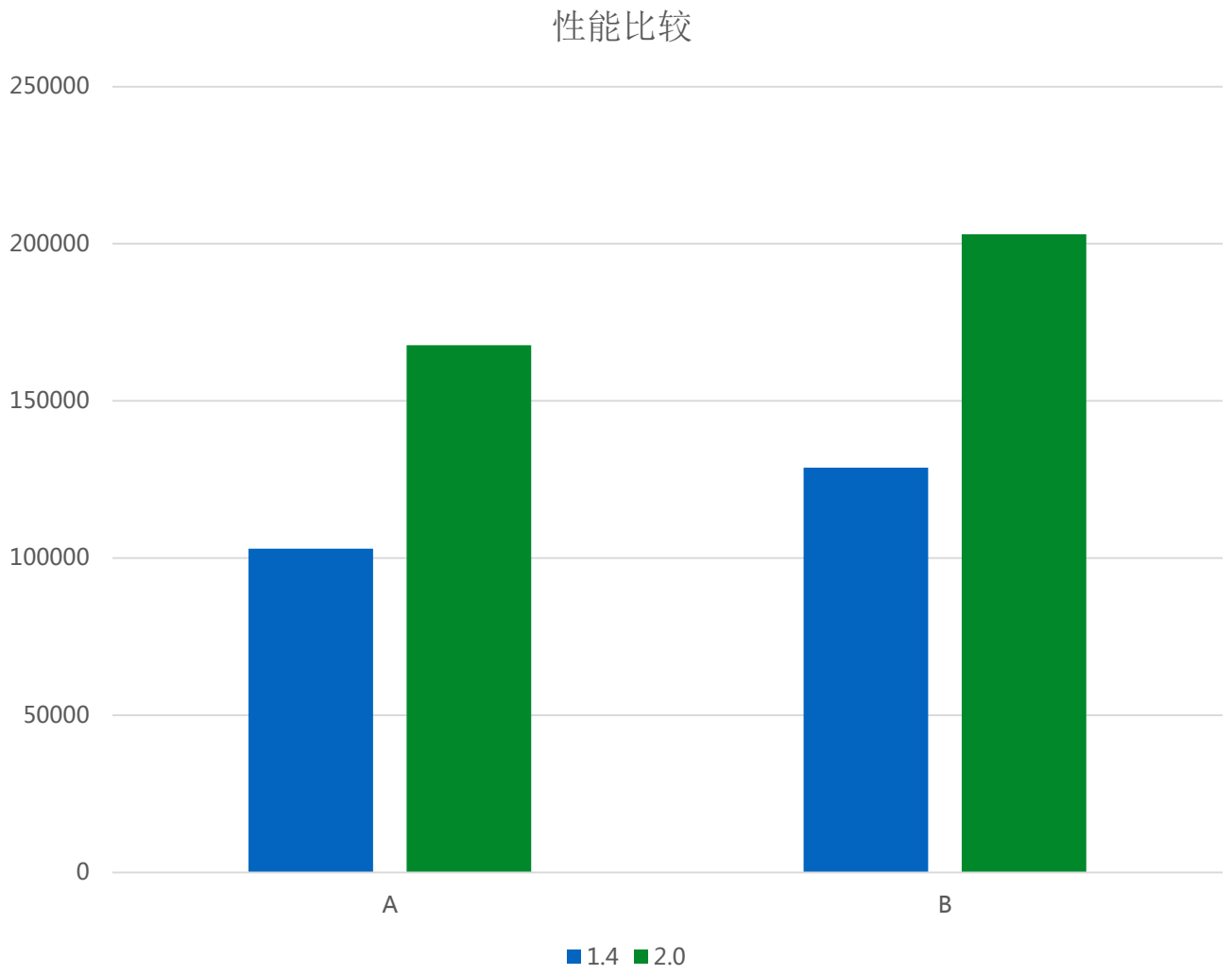
- 组织了两层映射关系，既要提升性能又要支持大量分区
- 第一层把分区映射到分区组上
- 第二层把分区组内使用的内存映射到不同的内存分配器分组



05 性能无止尽

性能比较

- A场景是下单场景，性能提升 63%
- B场景是支付场景，性能提升 58%



未来工作

- 面向全栈的优化
- 针对更多工作负载的优化
- 面向新硬件的优化

谢谢

THANK YOU

颜然/韩富晟

微信公众号
OceanBase

关注OceanBase微信公众号
回复关键词“1027”获取PPT

蚂蚁金服 ANT FINANCIAL

OceanBase