China · Shenzhen

分布式实现那些事儿

Pegasus背后的故事

孙伟杰

小米云存储工程师

China · Shenzhen

SPEAKER

INTRODUCE

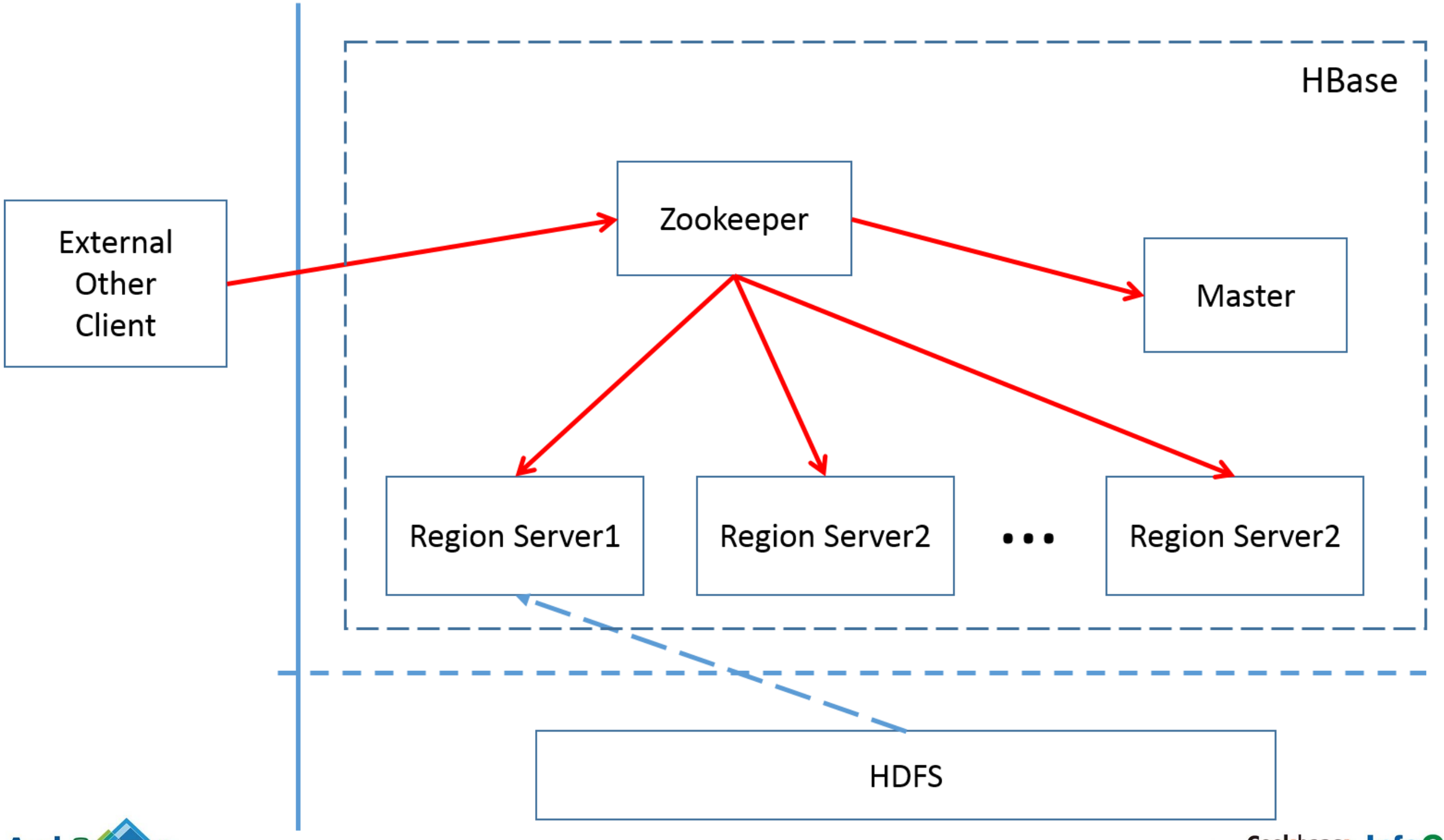
孙伟杰 云存储工程师

- 浙江大学硕士毕业。目前就职于小米，致力于分布式存储系统 Pegasus 的研发工作。热爱底层技术，热爱开源，是分布式系统框架 rDSN 的重要开发者。
- 博客：http://shengofsun.github.io/

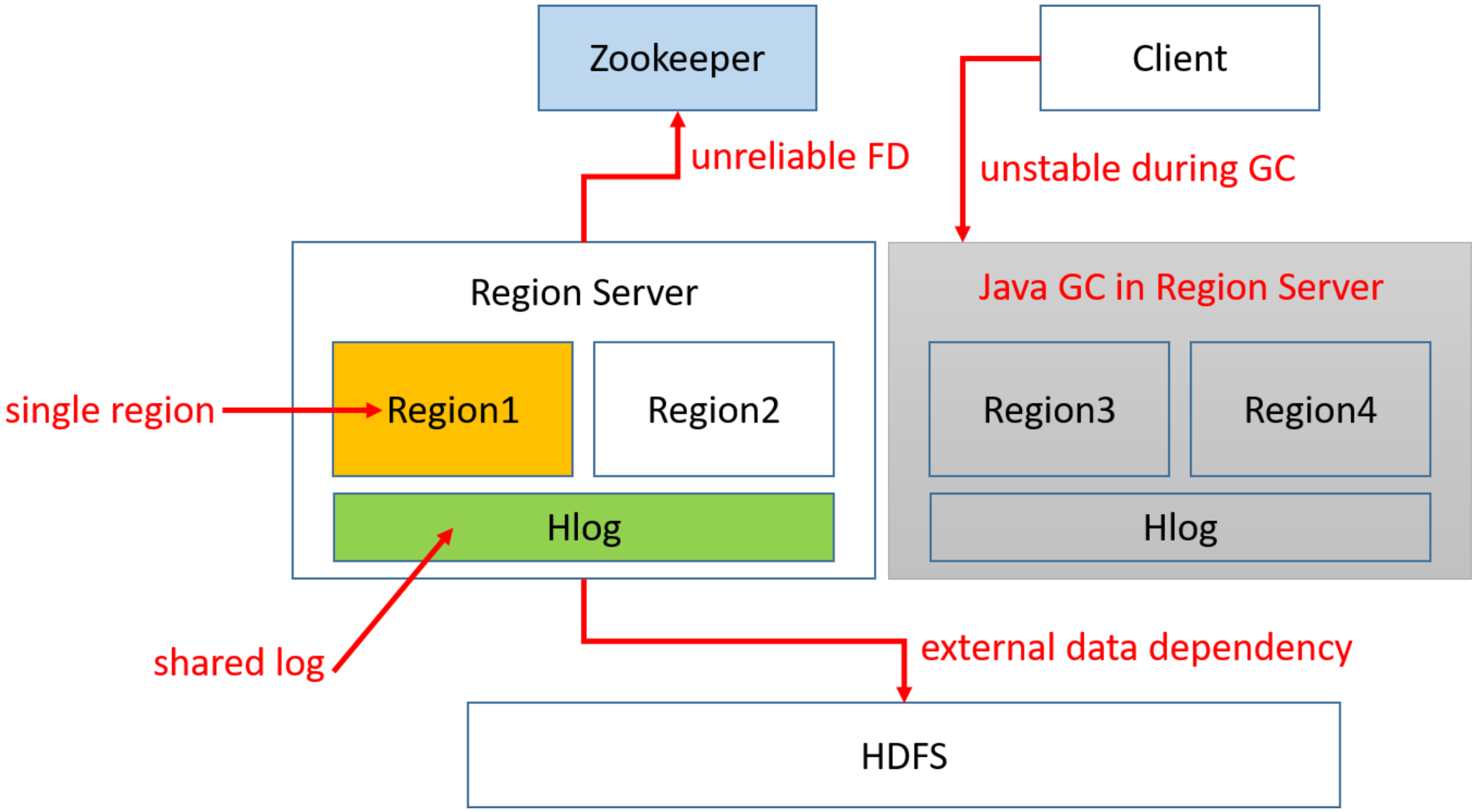
TABLE OF
CONTENTS 大纲

记一次HBase事故

- 外部客户端压力太大导致Zookeeper不可用
- Zookeeper不可用导致所有Region Server不可用



HBase的不足


Zookeeper
Client
unreliable FD
unstable during GC
Region Server
Java GC in Region Server
single region
Region1
Region2
Region3
Region4
Hlog
Hlog
shared log
external data dependency
HDFS

Pegasus的定位

强一致性视图

动态伸缩

→ 取HBase所长

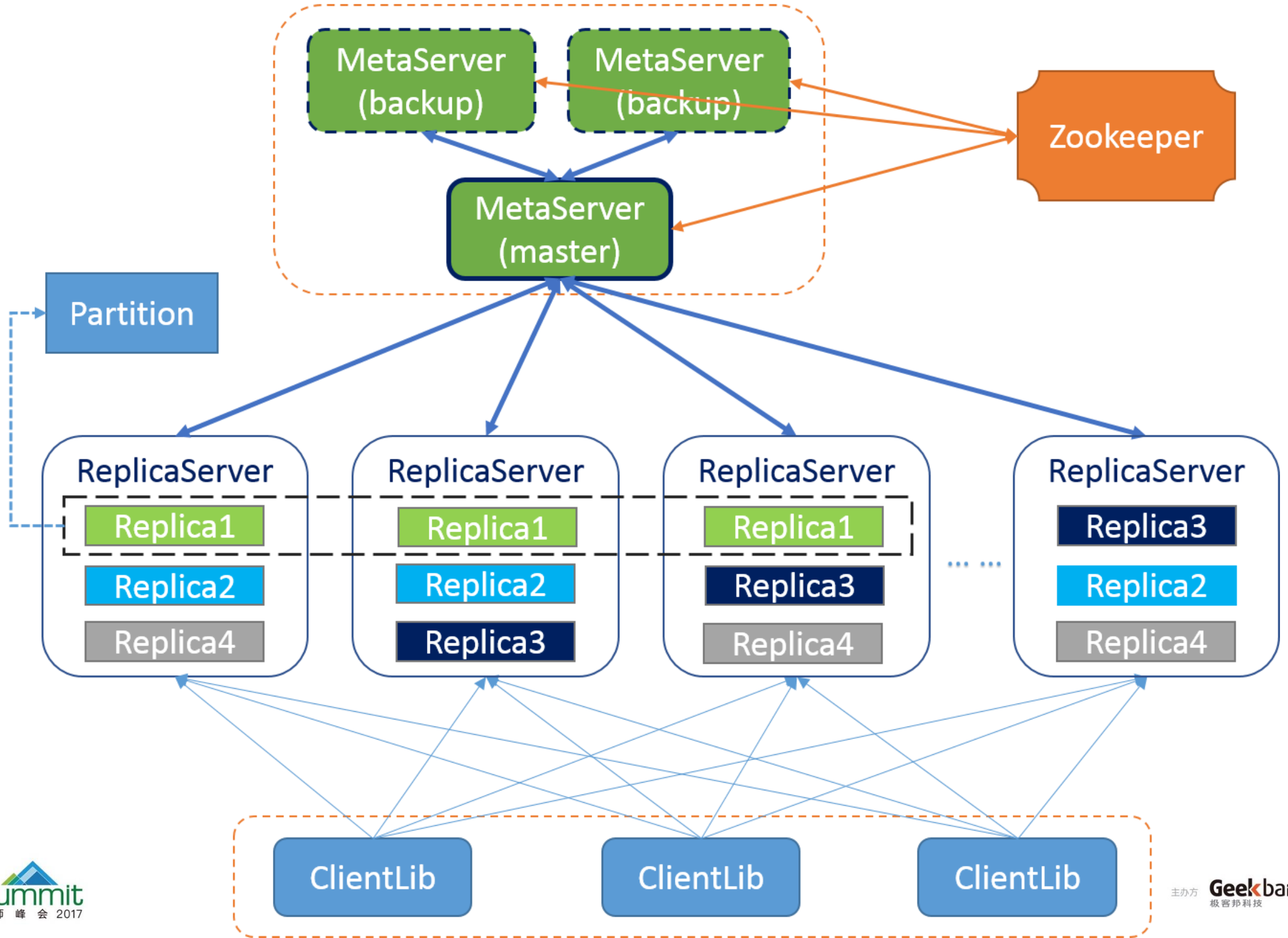
稳定的性能延时

快速failover
好的可用性

→ 补HBase所短

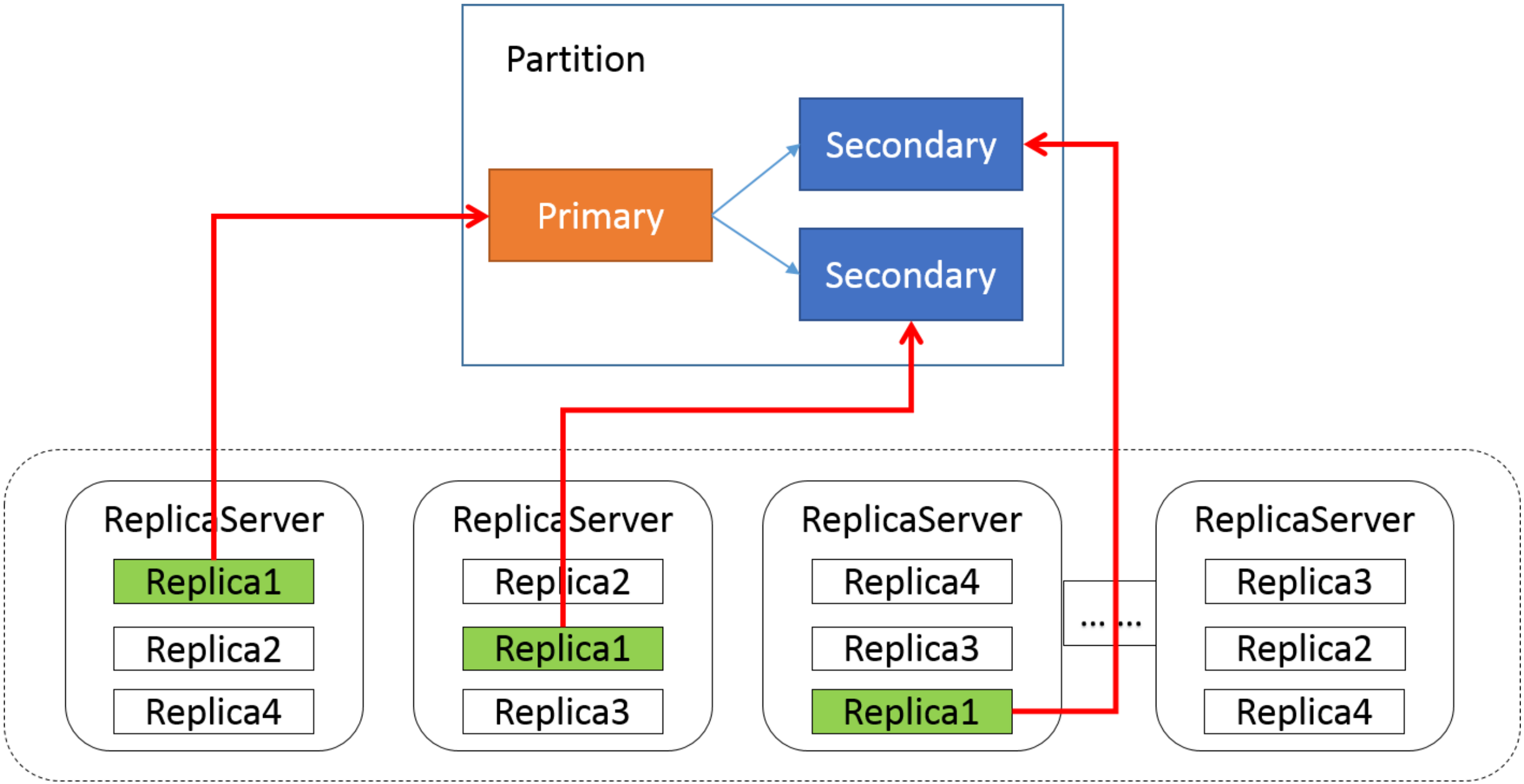
Pegasus架构一览

- 中心化、分Partition
- 心跳不依赖Zookeeper、不依赖DFS、多副本

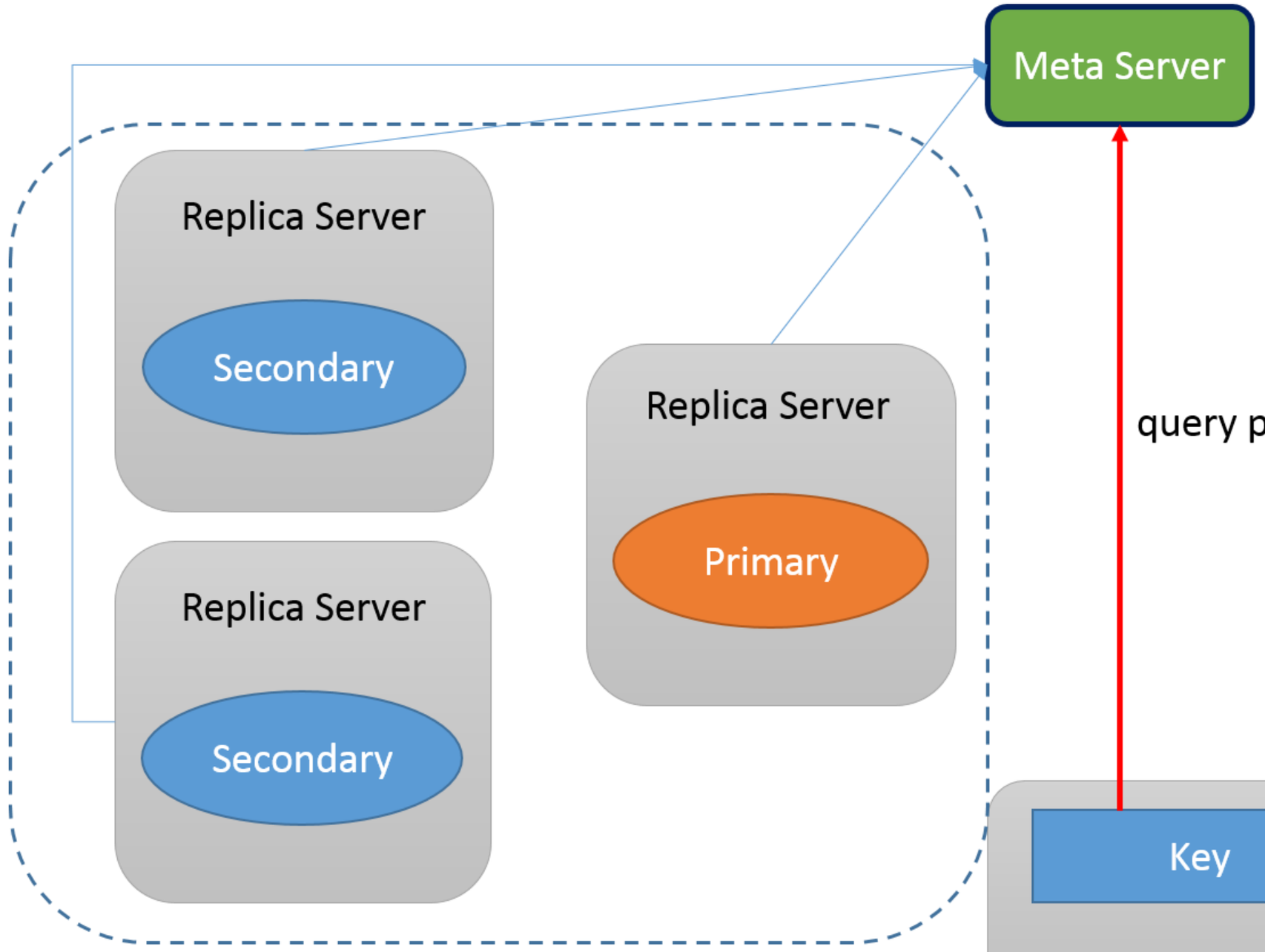


多副本间的一致性协议

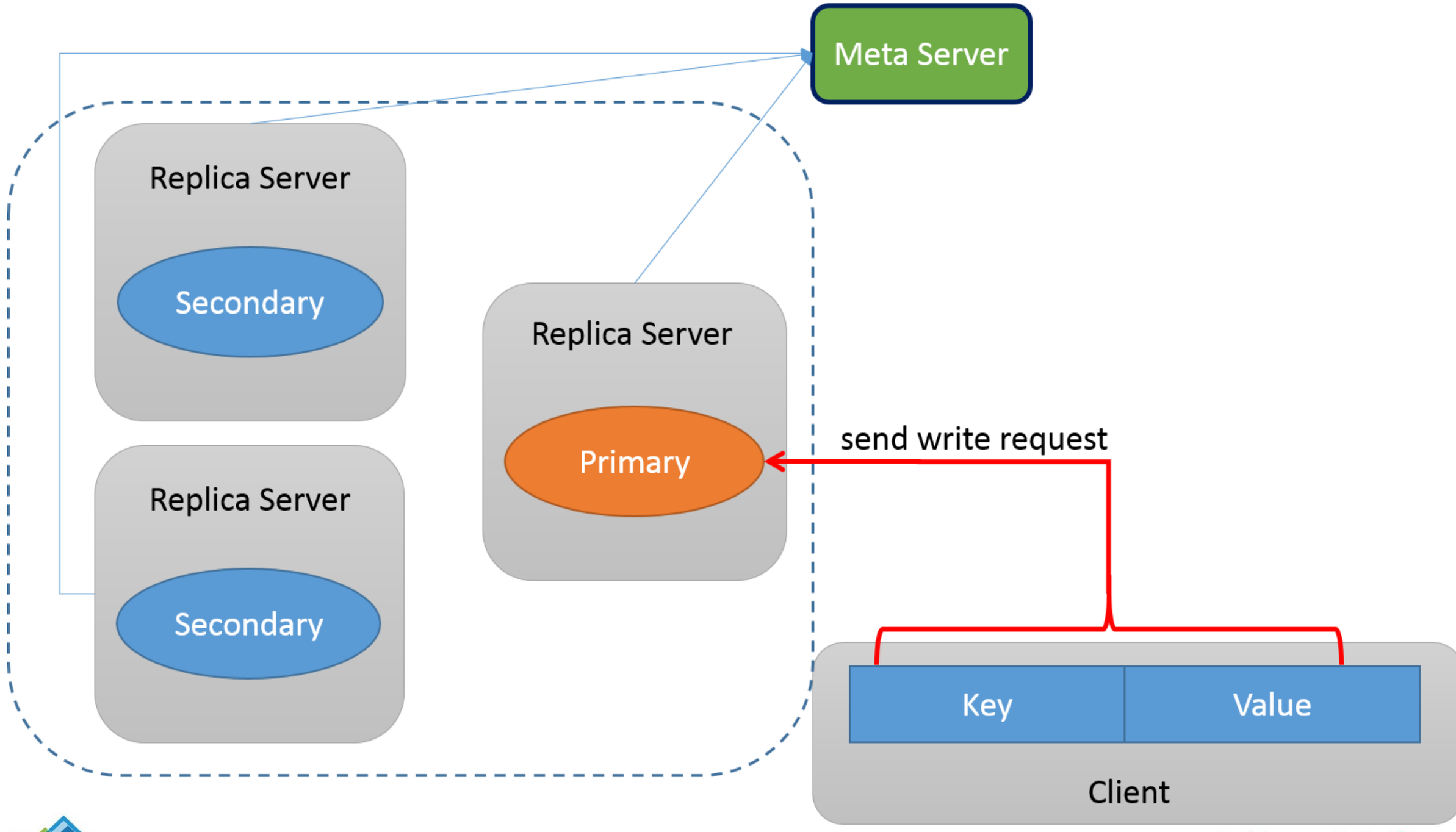
- PacificA一致性协议



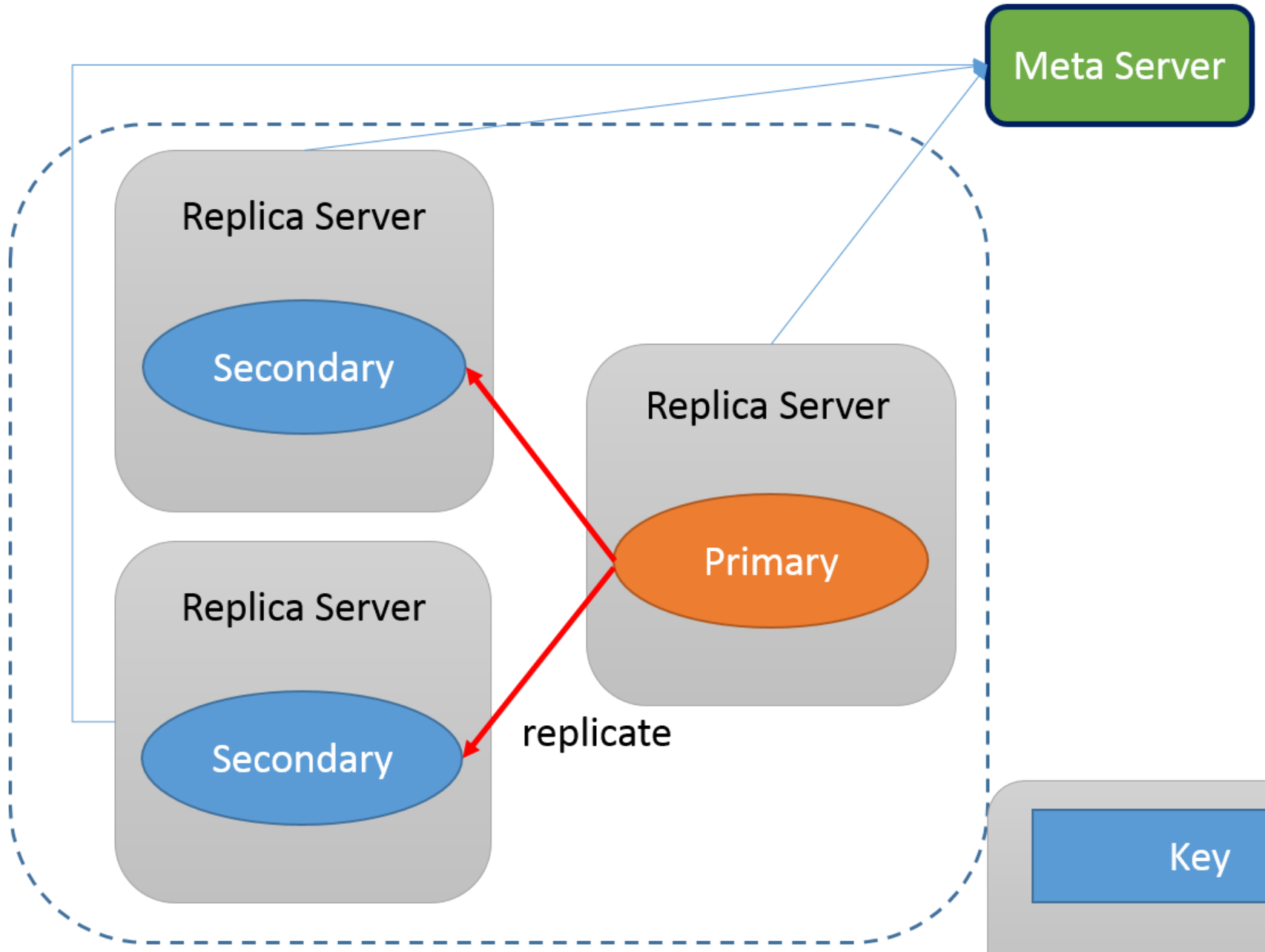
Pegasus写请求流程



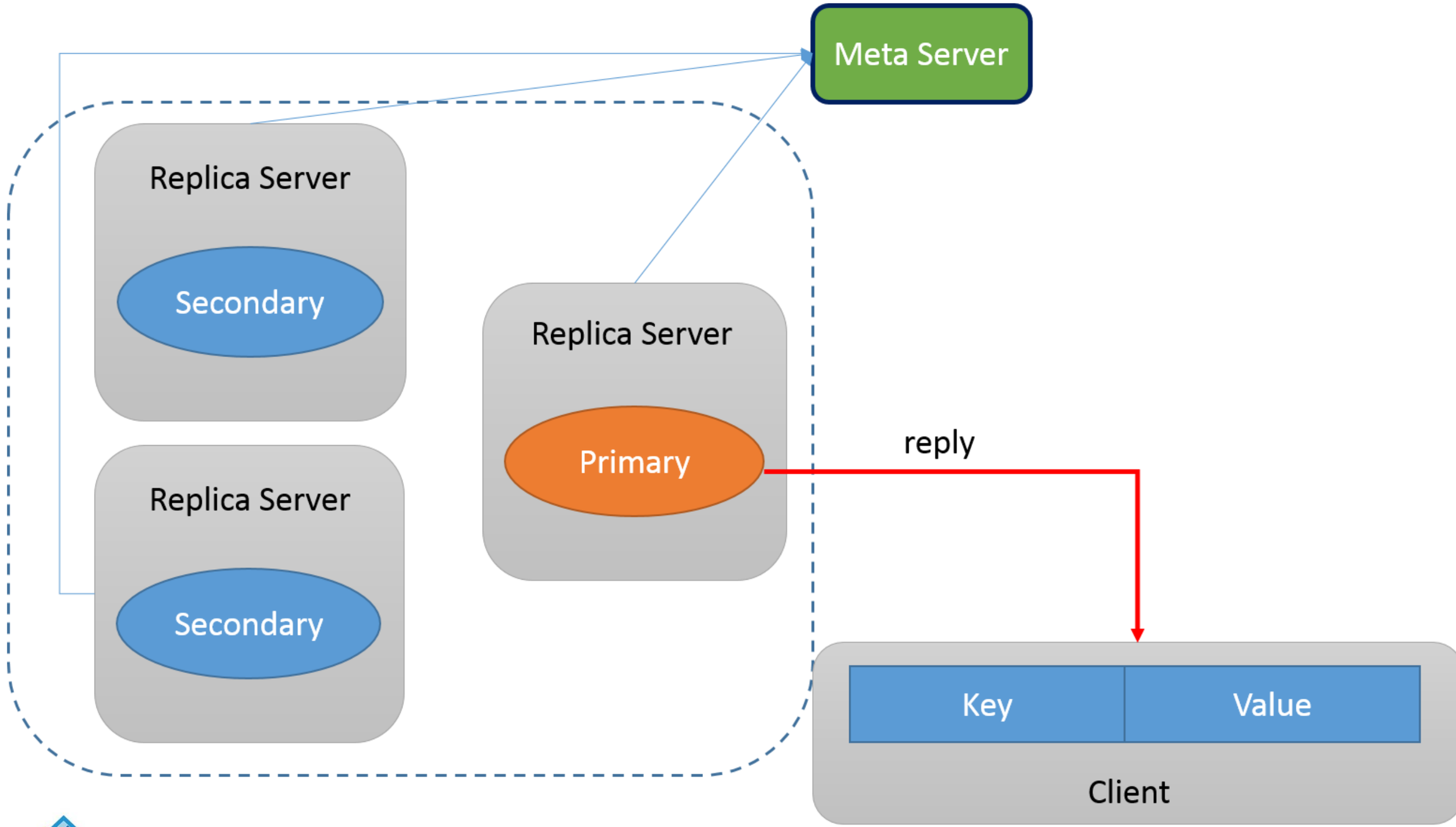
Pegasus写请求流程



Pegasus写请求流程



Pegasus写请求流程



Pegasus的读请求流程图

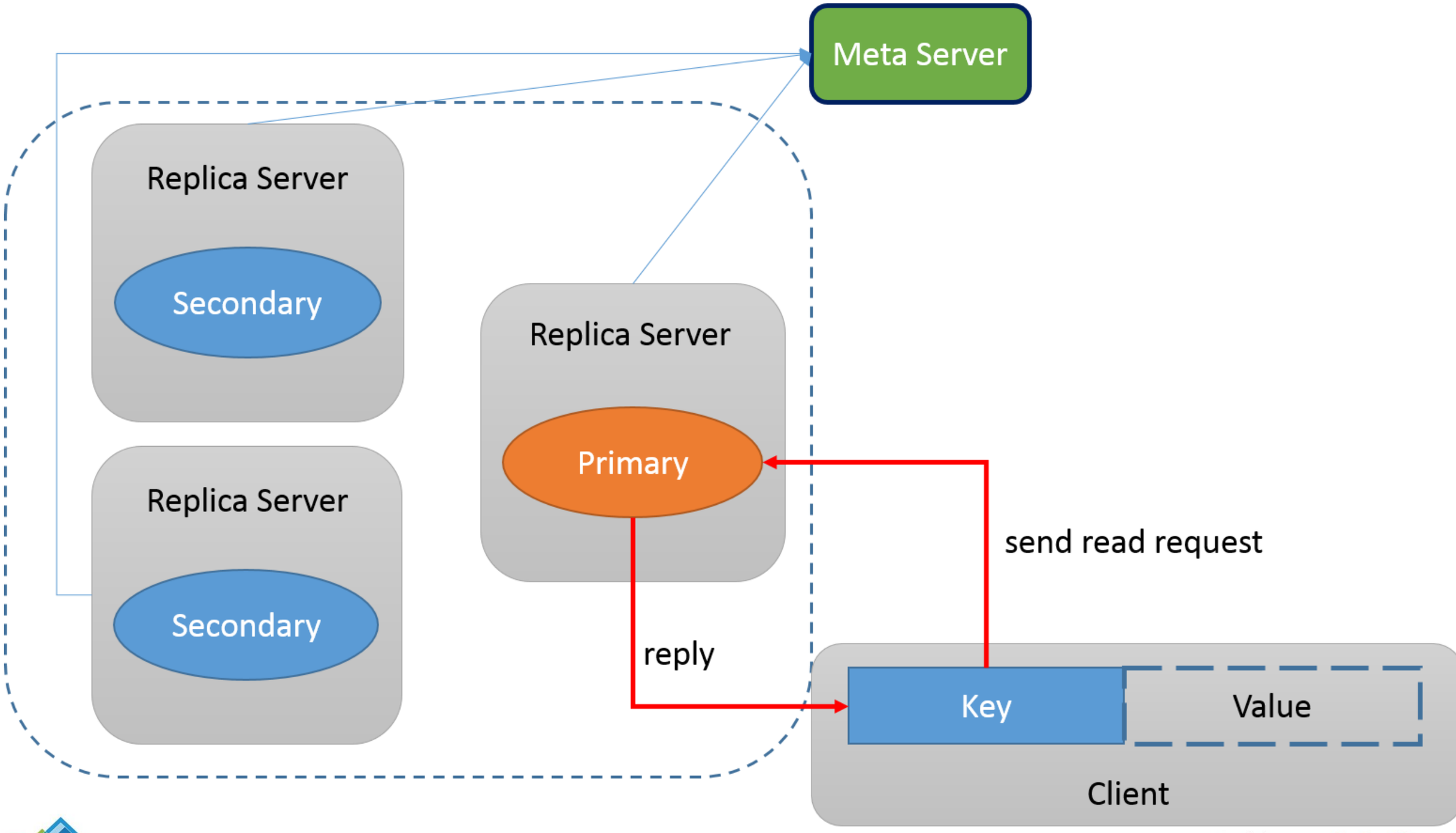
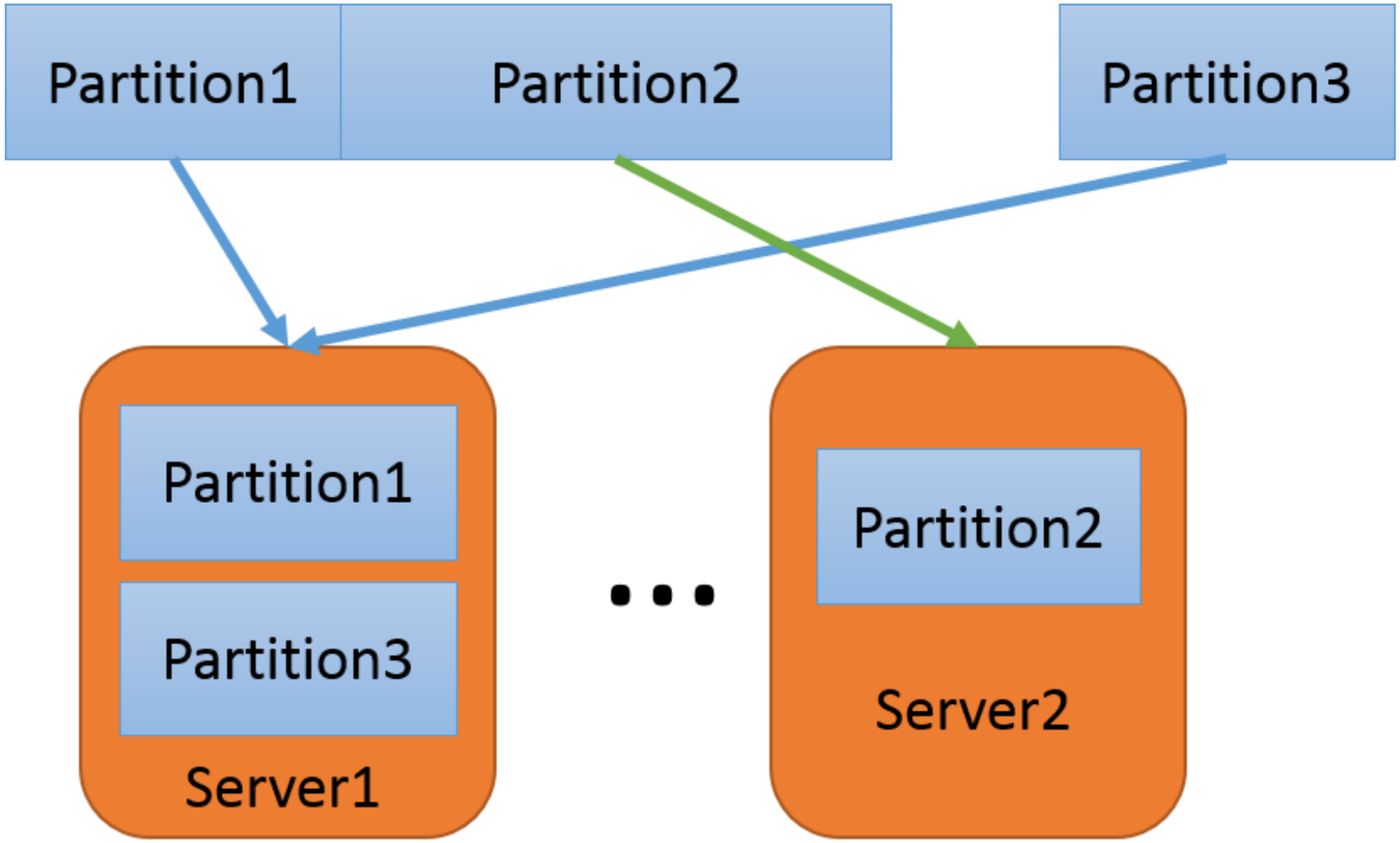


TABLE OF
CONTENTS 大纲

扩展性

- Partition Schema

- Load Balance

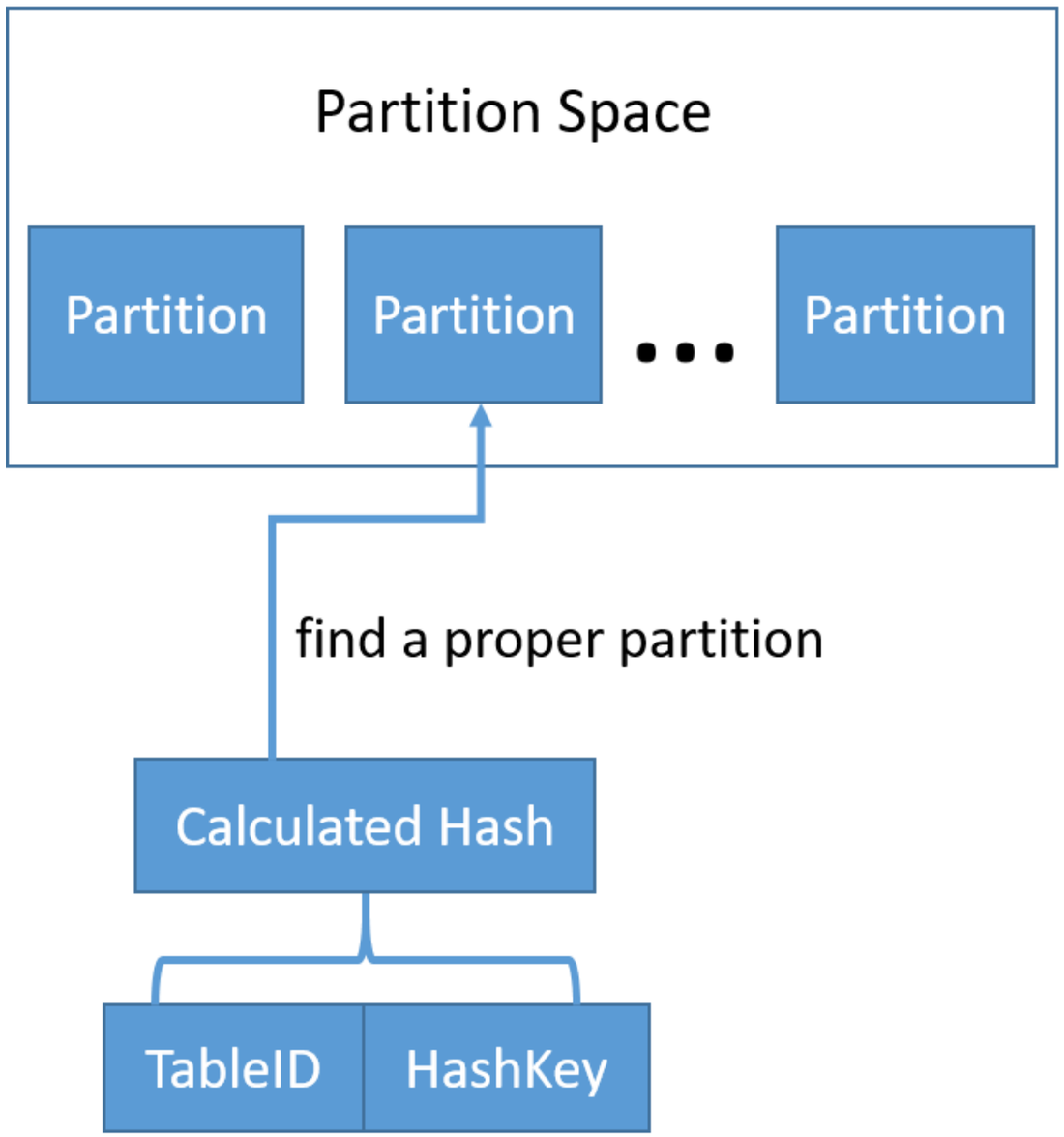


Partition方案

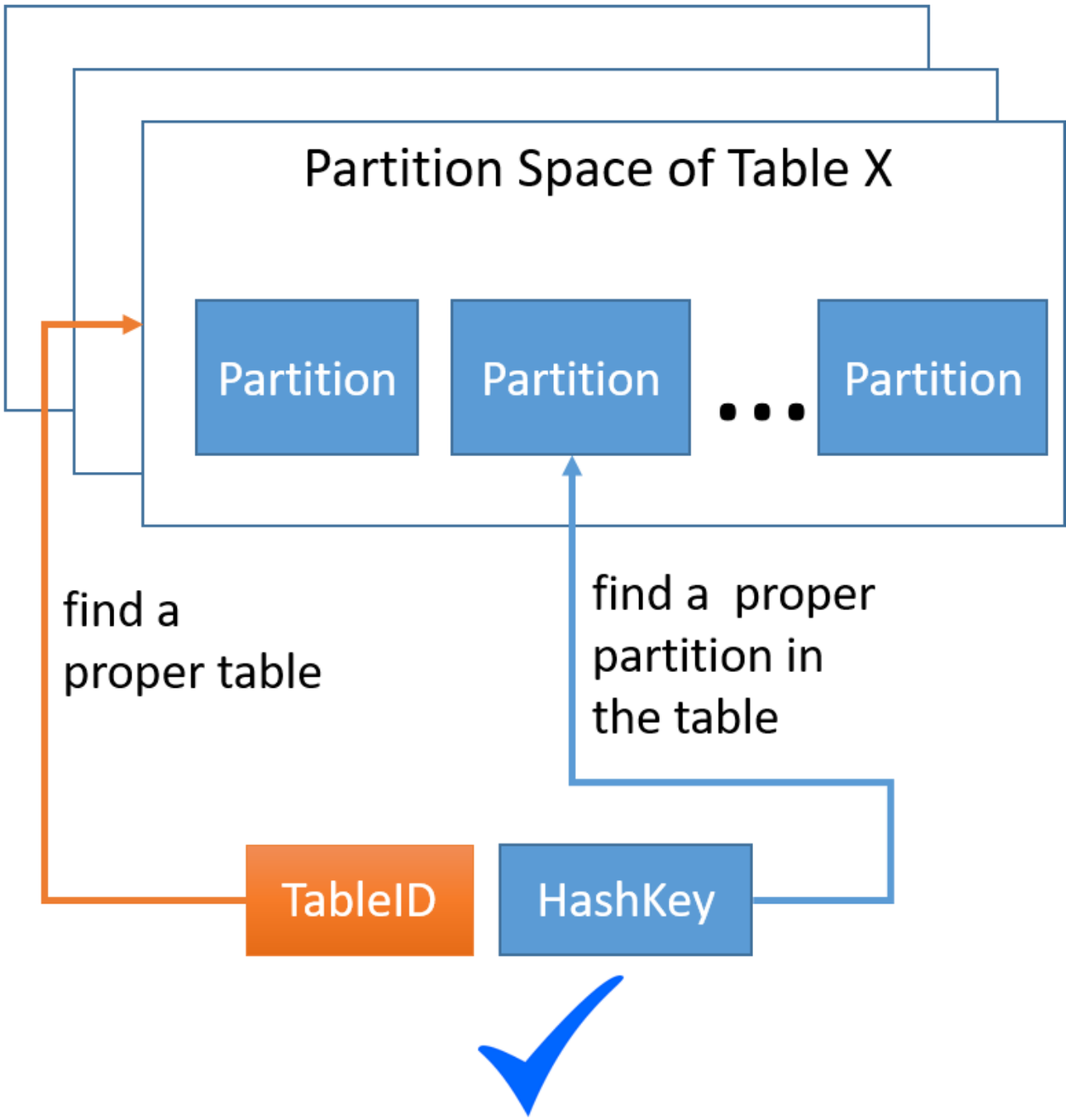
	Hash方案	排序方案
全局有序	无	有
实现难易程度	容易	复杂
热点问题	无	有

Hash Schema实现方式

多表混存



各表分开存



两种方案对比

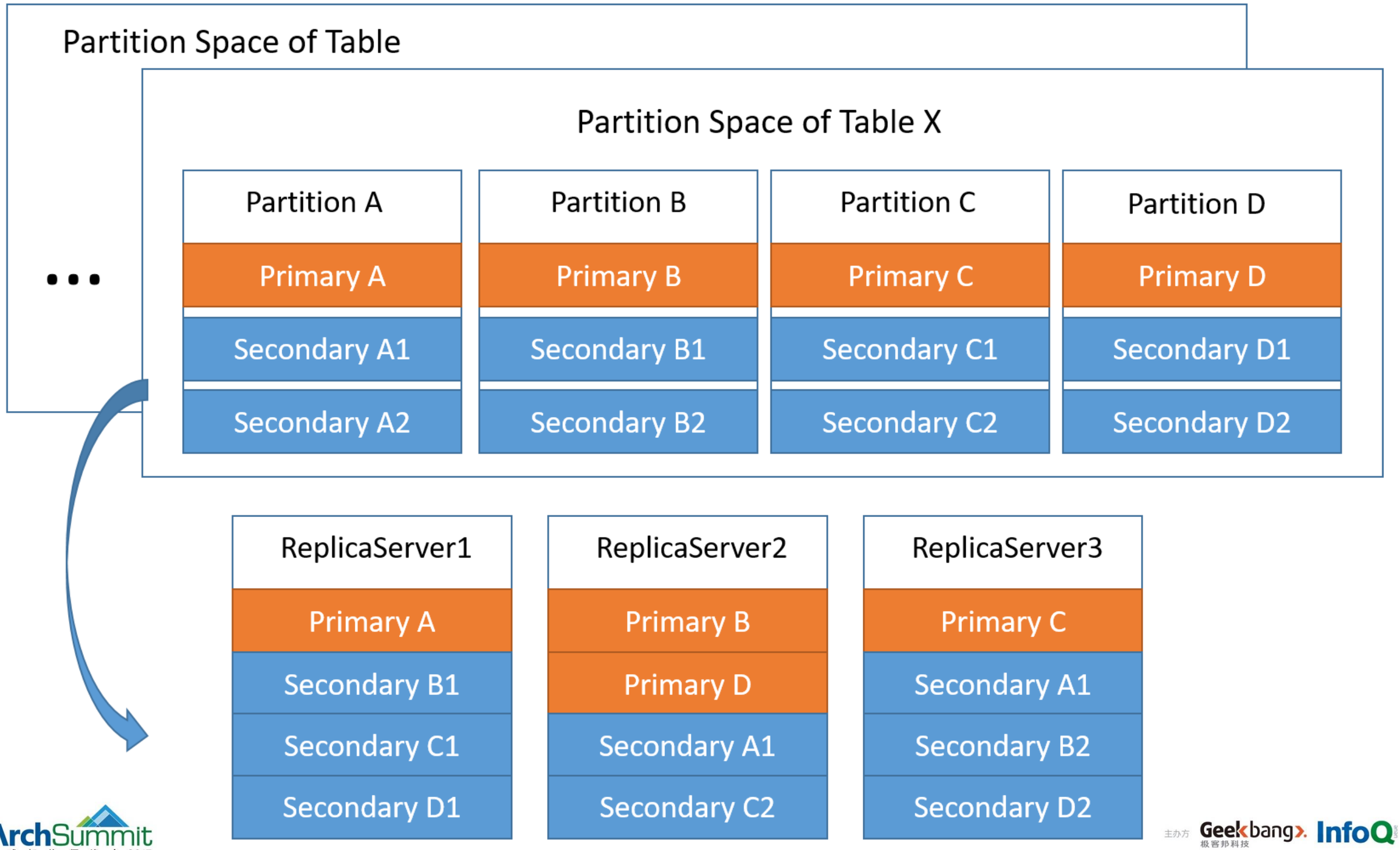
	多表混存	各表分开存
元数据管理	容易	不容易
负载均衡	容易	不容易
表间资源限制(quota)	不容易	容易
表级别的监控	不容易	容易
删表操作	不容易	容易
误操作带来的影响	高概率影响多个业务	低概率影响多个业务

Hash Schema的负载均衡

- 可能的考虑因素：
  - 单个Key请求过热(难以解决)
  - 单个Partition请求过热(无需考虑)
  - Partition容量分布不均(无需考虑)
  - Partition个数分布不均(需要处理)

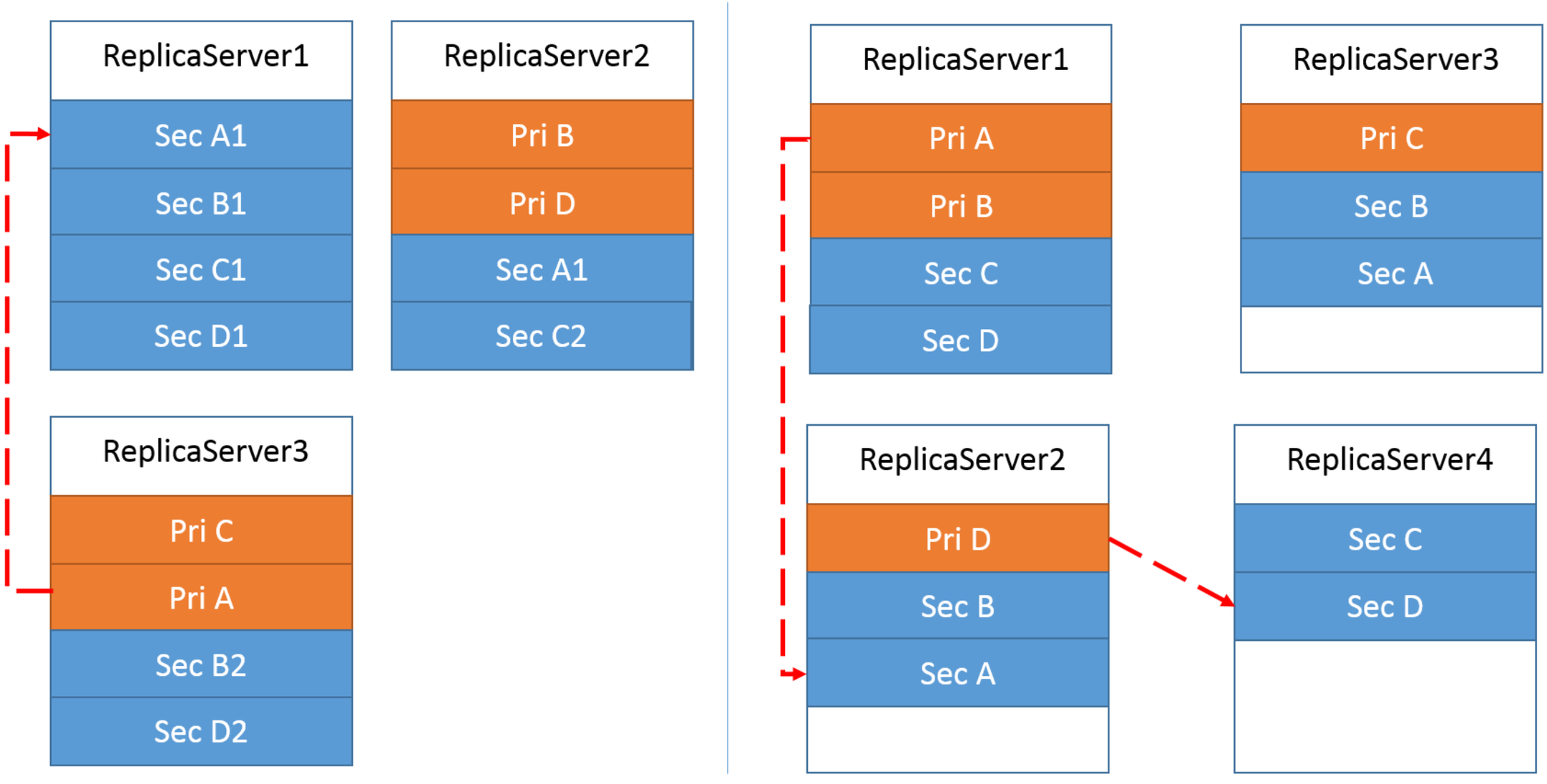
负载均衡-目标

- Primary、Secondary不共享物理机
- 对于每个表：Primary、Secondary都平均分配



负载均衡-算法

- 角色切换优于数据拷贝
- 根据Primary的可能流向建立有向图
- Ford-Fulkerson方法：最短路 + 迭代

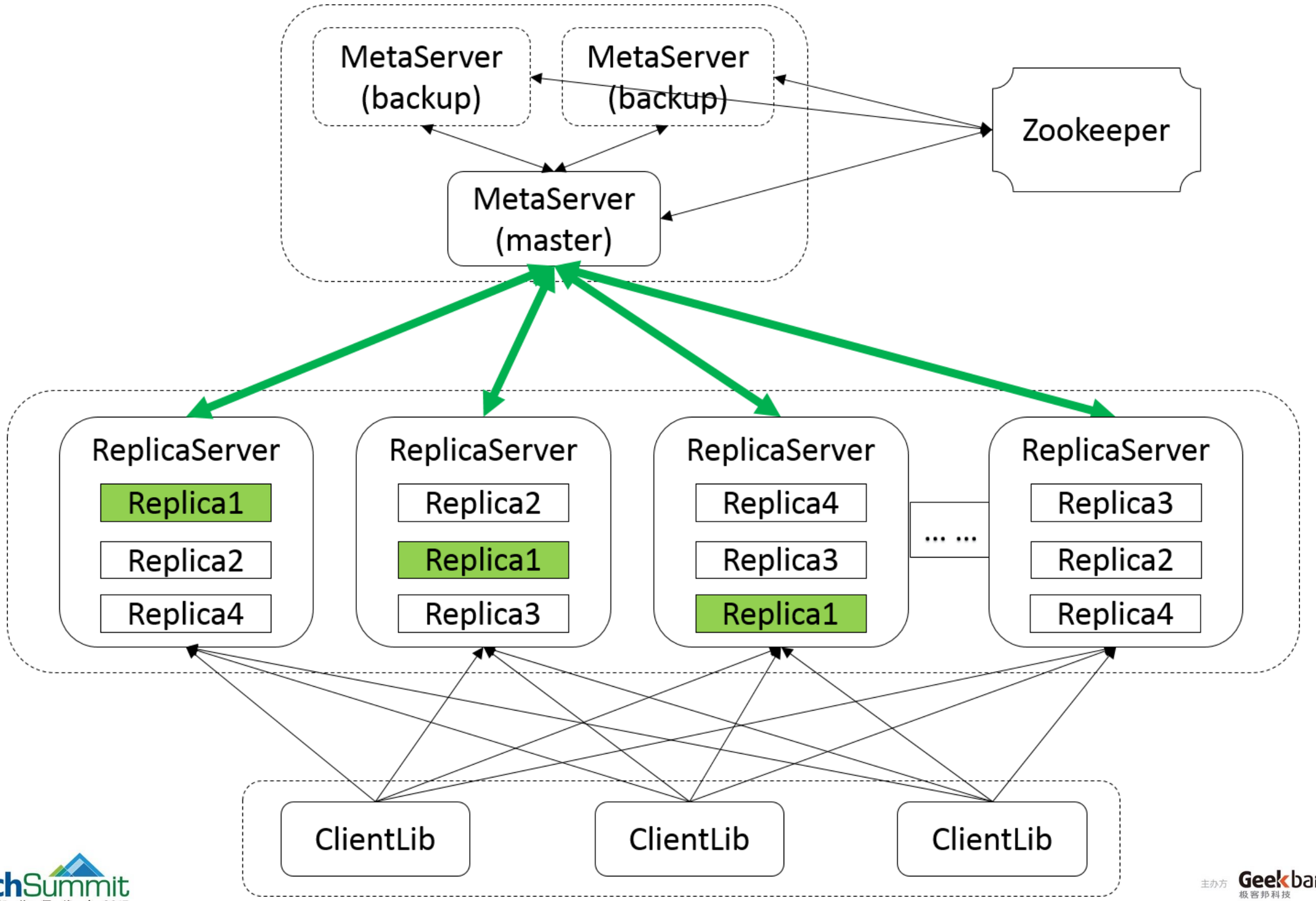


真实的负载均衡

- 考虑机架位等信息
- 节点需要有权重
- 拷贝数据时限流

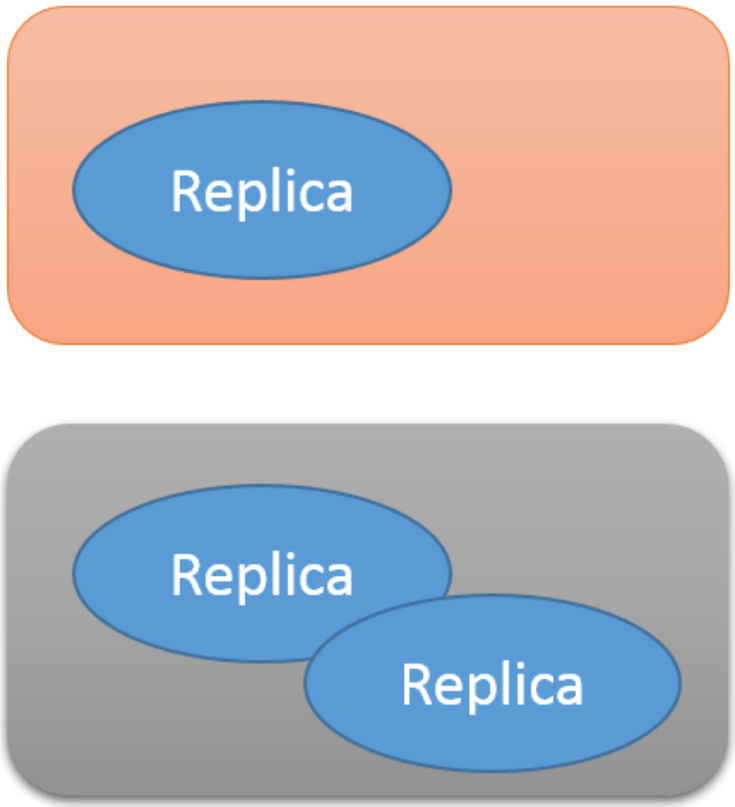
一致性和可用性

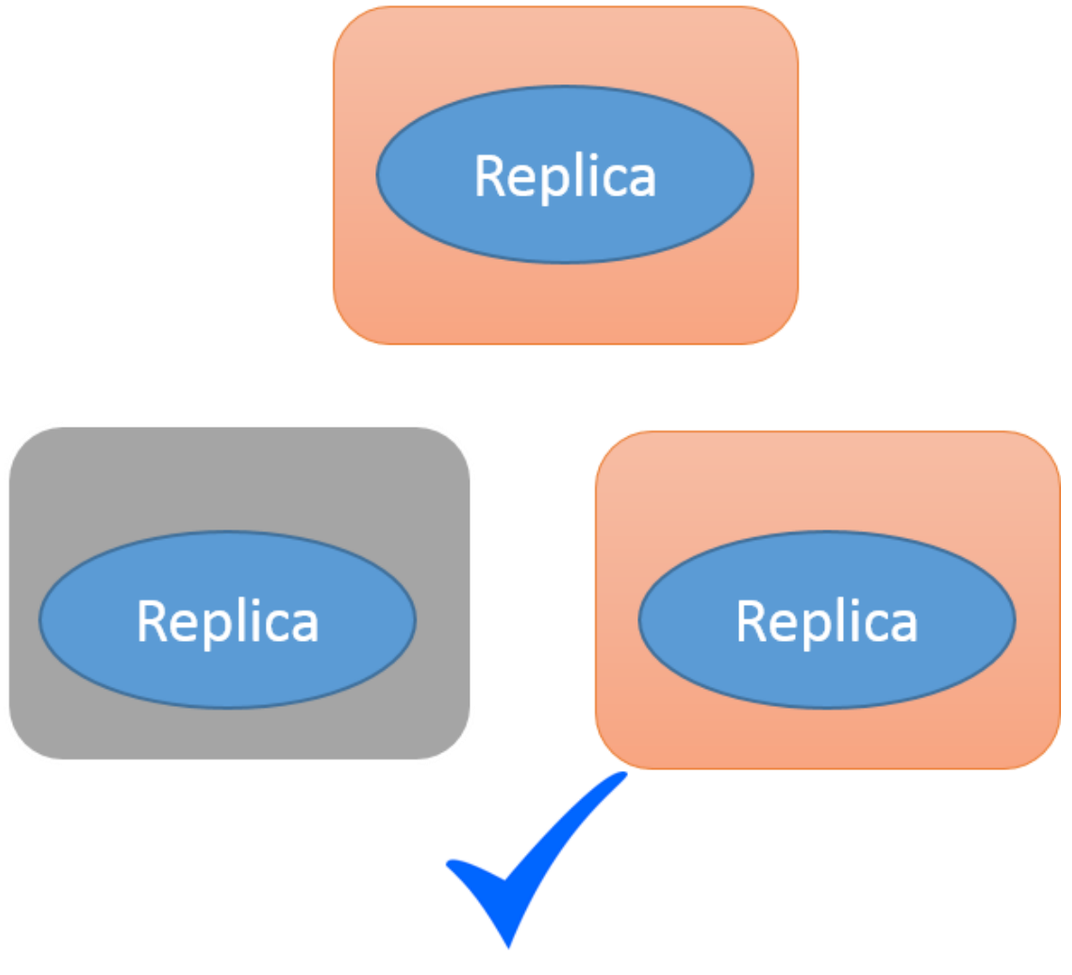
- 心跳不依赖zookeeper
- 数据不依赖DFS
- PacificA算法
- 多副本



“你们有双机房热备吗？”

- 在强一致下，跨机房的可用性至少涉及三机房
- 三机房replication性能保障不够好

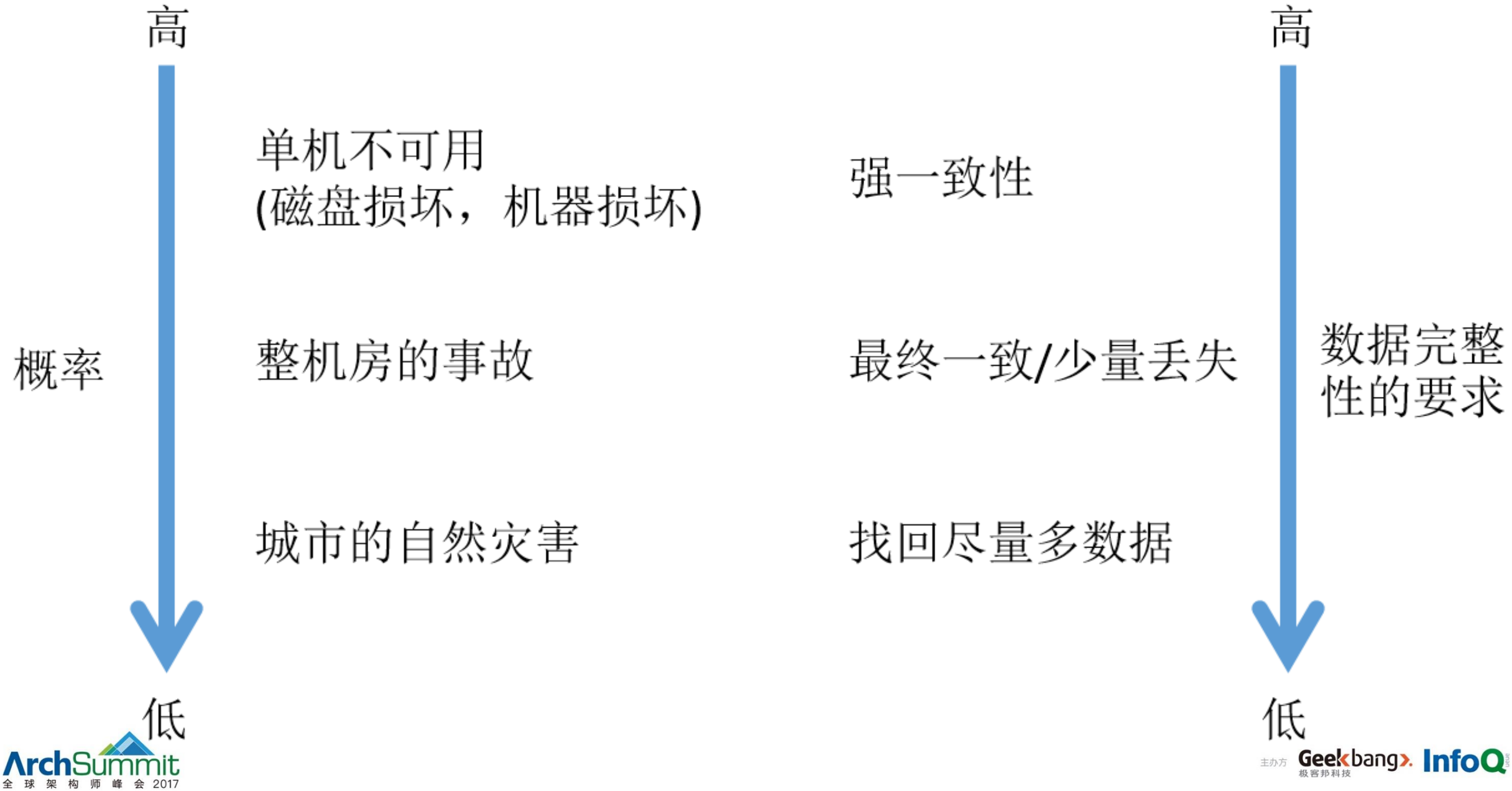





可以双机房热备吗？
臣妾做不到啊!

要搞明白业务的需求

- 长得好看不重要，能过日子最重要



Pegasus的多级冗余策略

- 满足多种业务需求
- 表级别灵活控制，避免备份太多

一致性协议：
单机房部署

跨机房同城热备：
双机房均可以写、异步复制
基于时间戳的最终一致性

Snapshot定期冷备份：
跨地域

一致性和可用性：权衡

Consistency

Availability

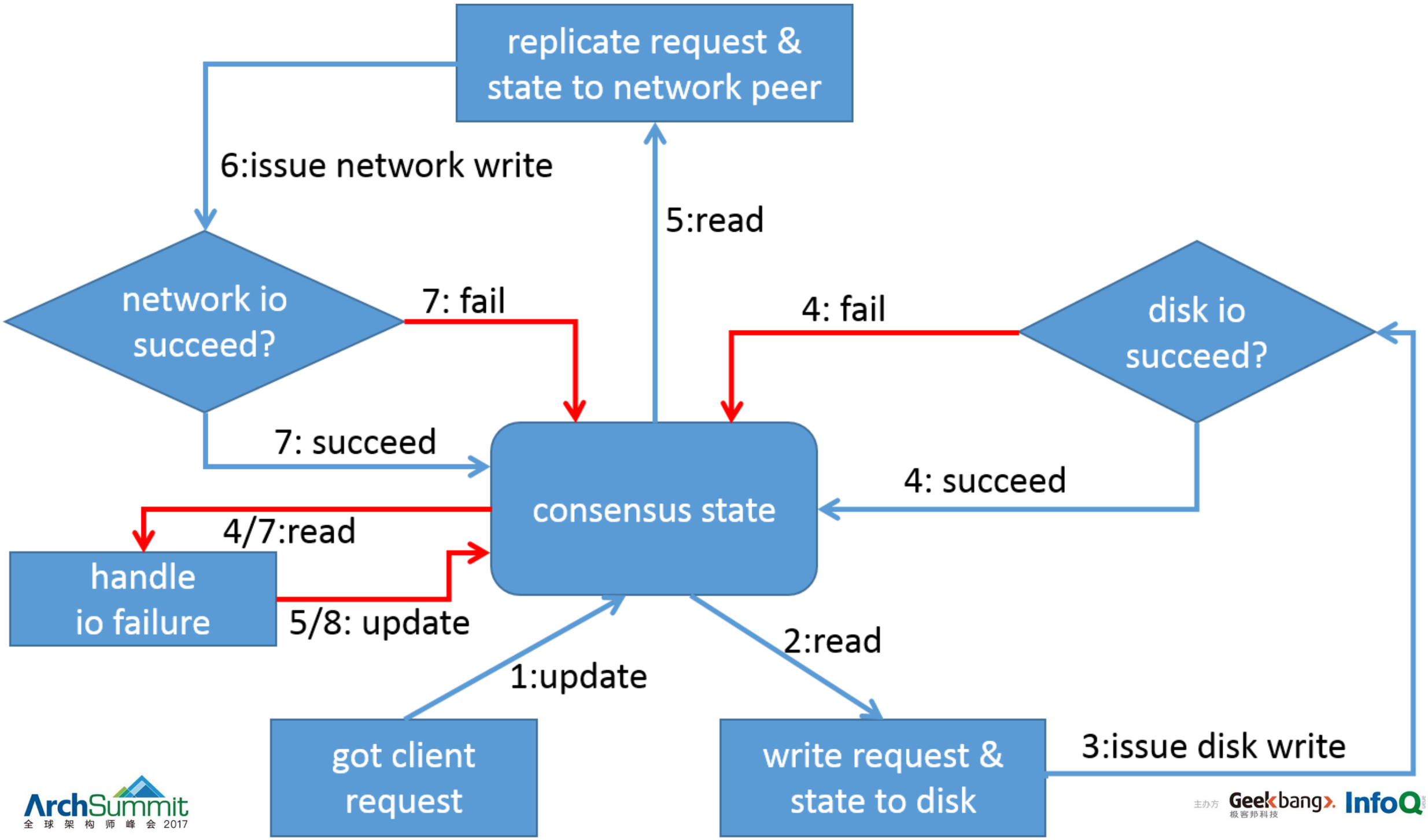
Partition Tolerance

延时保证

- 实现语言：C++
- 产品可用的一致性协议的实现

一致性协议很难实现

- 要求：正确、高效、易维护、可测试
- 难点：多阶段、涉及IO→并发→细粒度加锁



争抢临界区→无锁串行化排队

实现要点

- 事件驱动、纯异步、无锁串行化
- 额外优势：方便监控、追踪

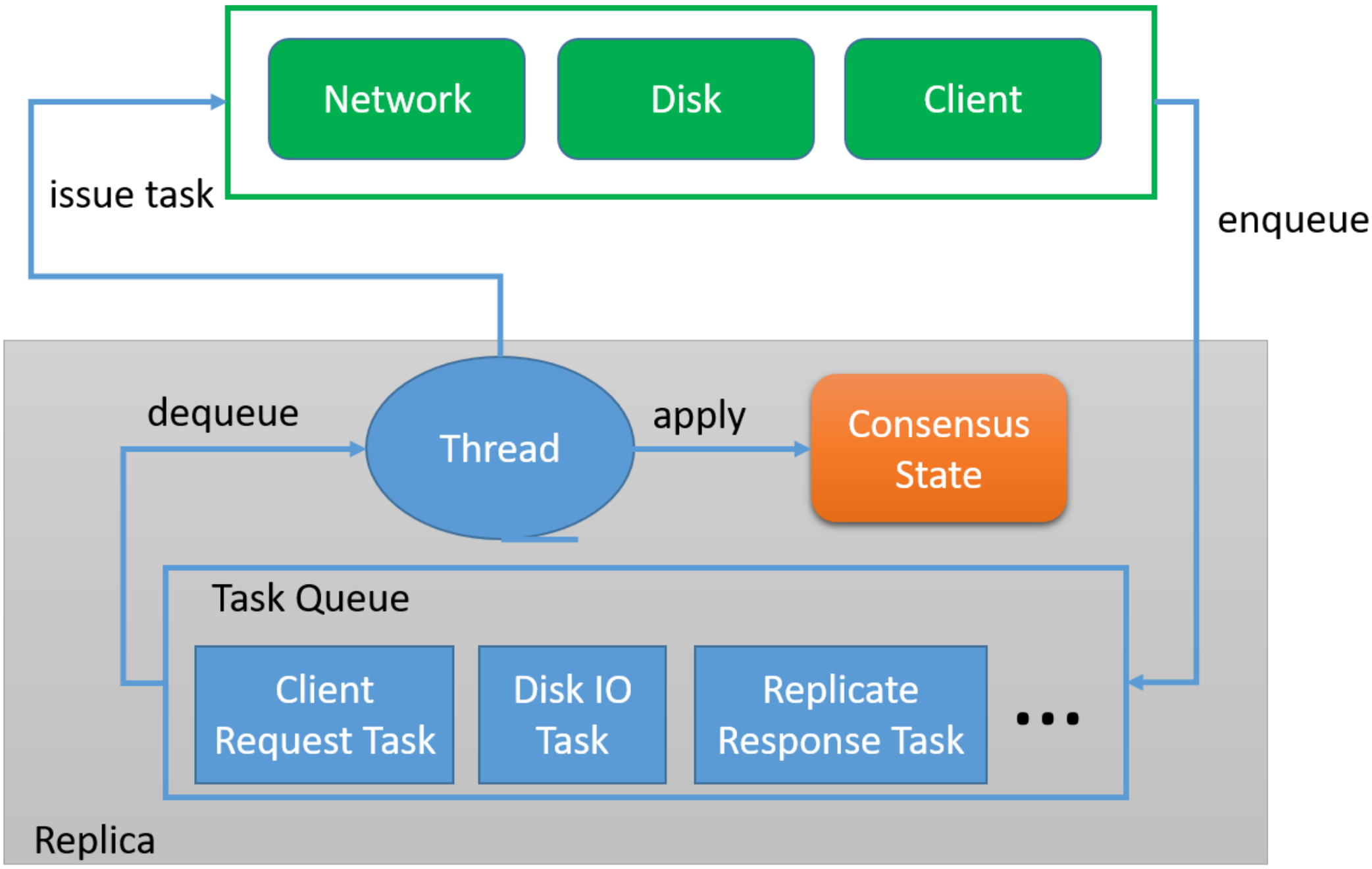


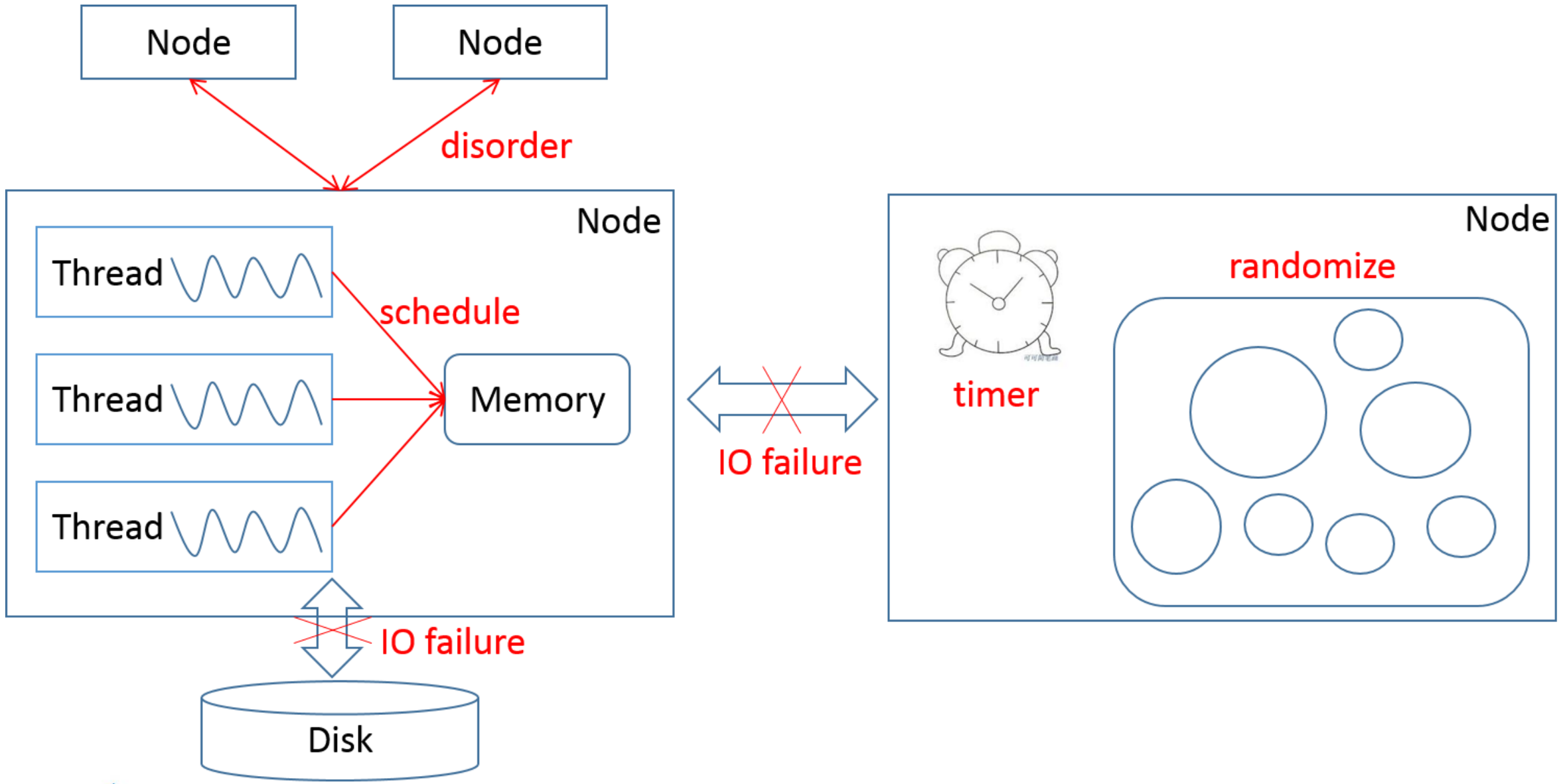
TABLE OF
CONTENTS 大纲

分布式系统做稳定很难

- 难以测试：难以通过有效的测试手段发现bug
- 难以复现：就算触发到了bug，也难以定位、复现和调试
- 难以回归：bug是不是被正确的解决了

根源：不确定性

- 程序自身的随机：调度、定时器、随机数、多节点并行
- 外部IO的错误：失败、超时、丢包、乱序

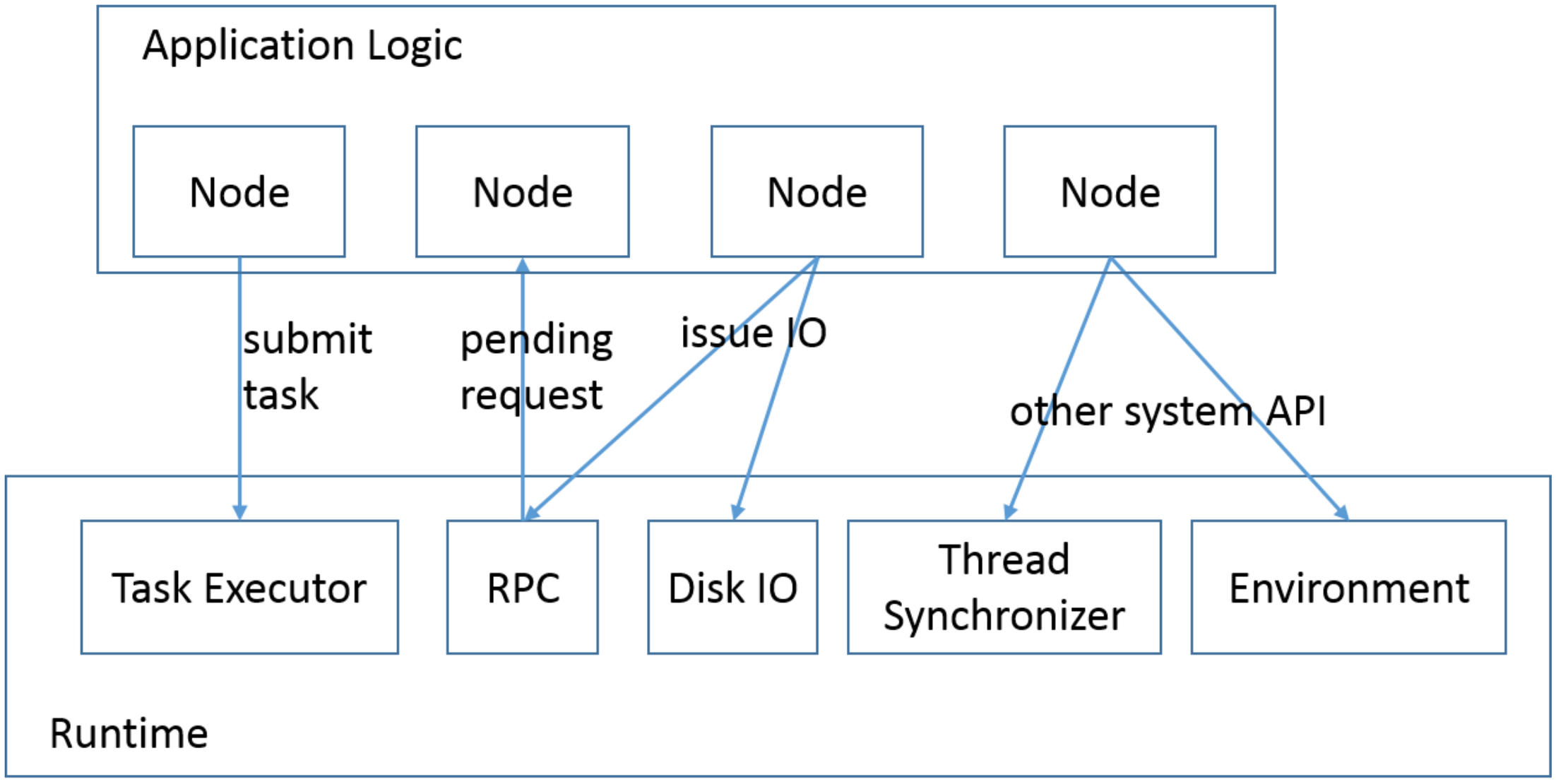


不确定性的问题

- 小概率IO错误 + 随机执行路径 = 不易复现异常状态
- 能不能模拟？
  - 模拟IO错误
  - 控制程序的执行顺序

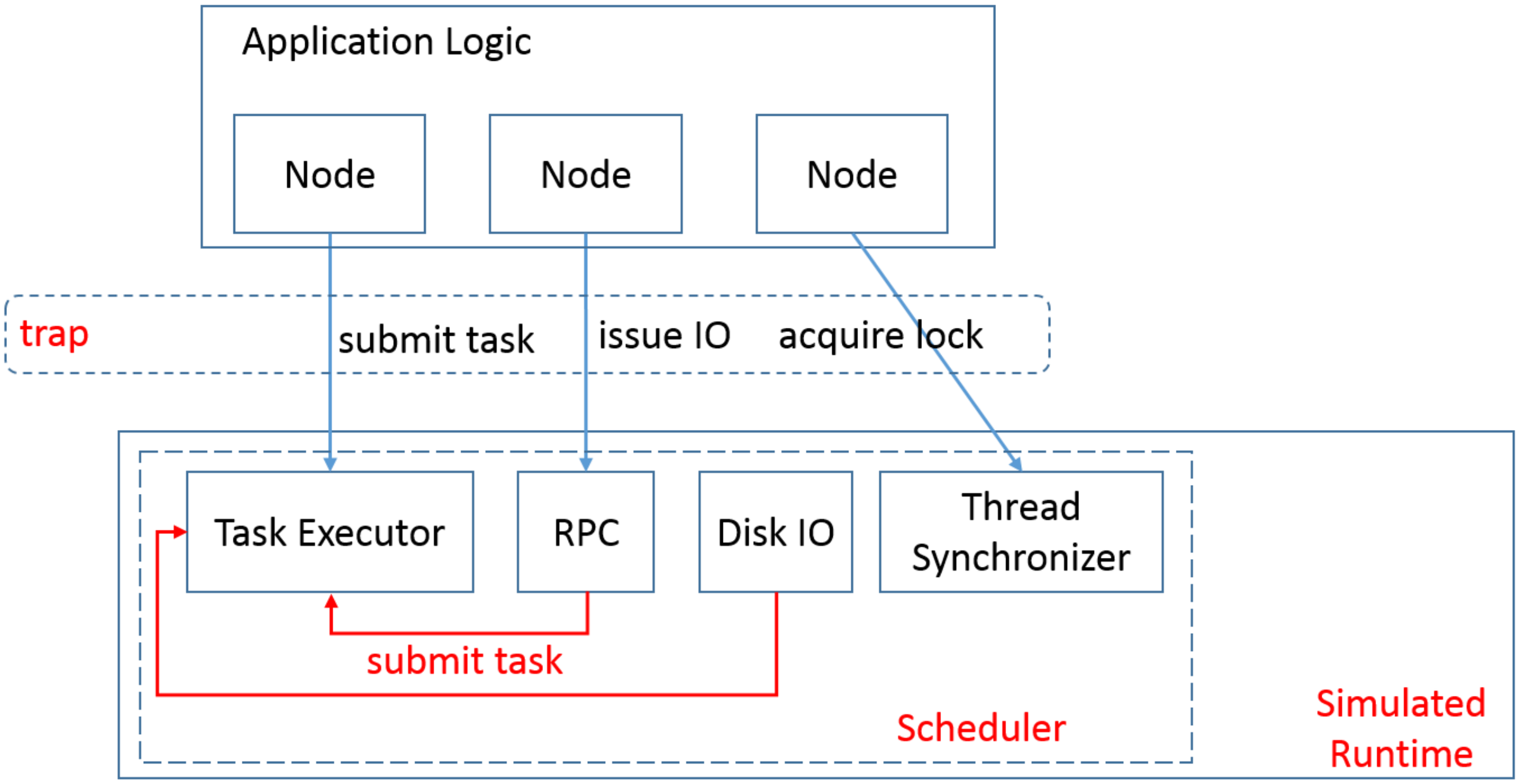
大方向：提供统一的编程模型

- 将不确定性因素做接口封装
- 单进程模拟多个网络节点



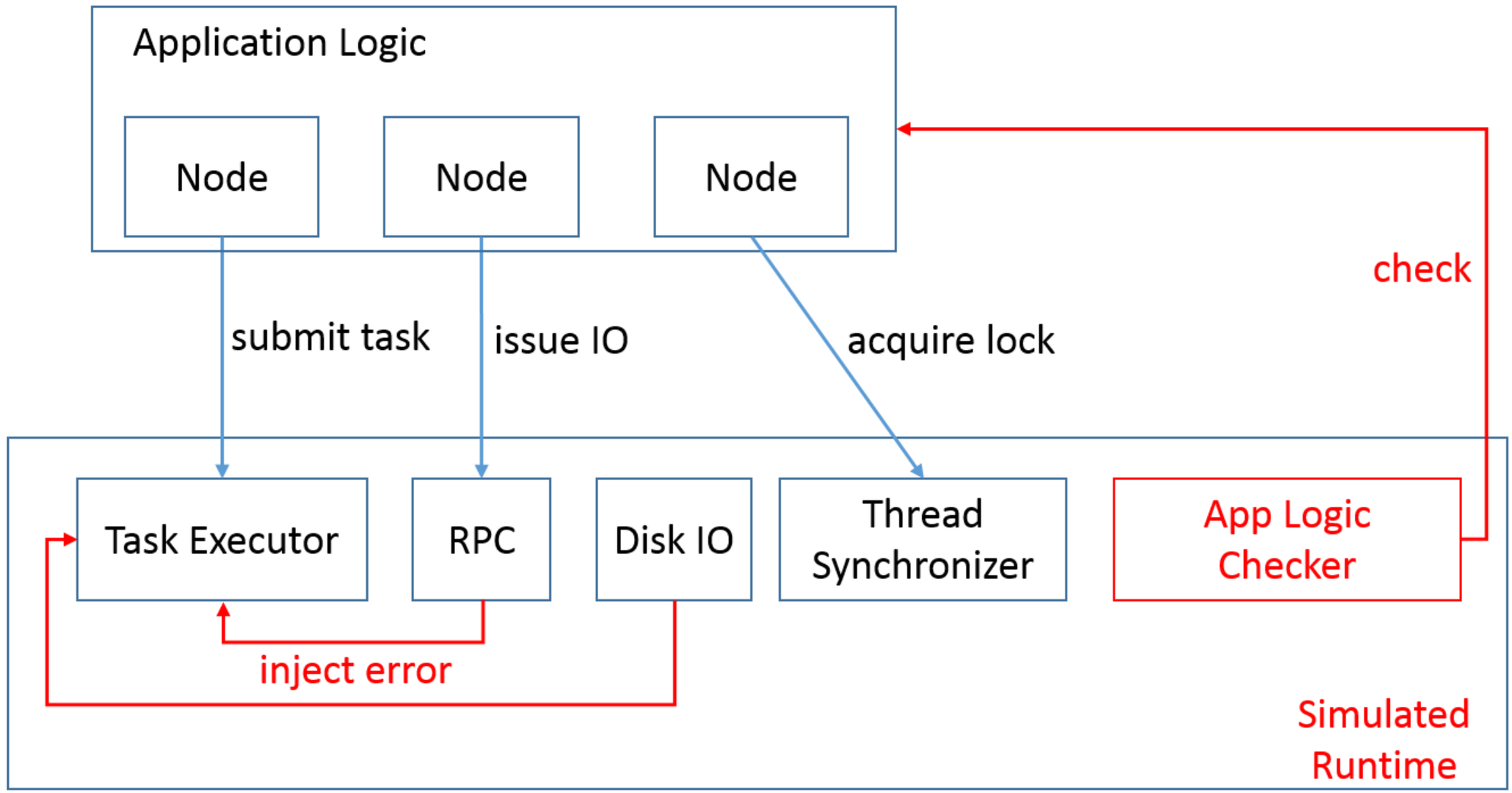
控制程序的执行顺序：消除并行

- 单节点串行化：按序执行 + 异步IO事件伪随机delay
- 节点间串行化：parallel → concurrent，多队列伪随机调度

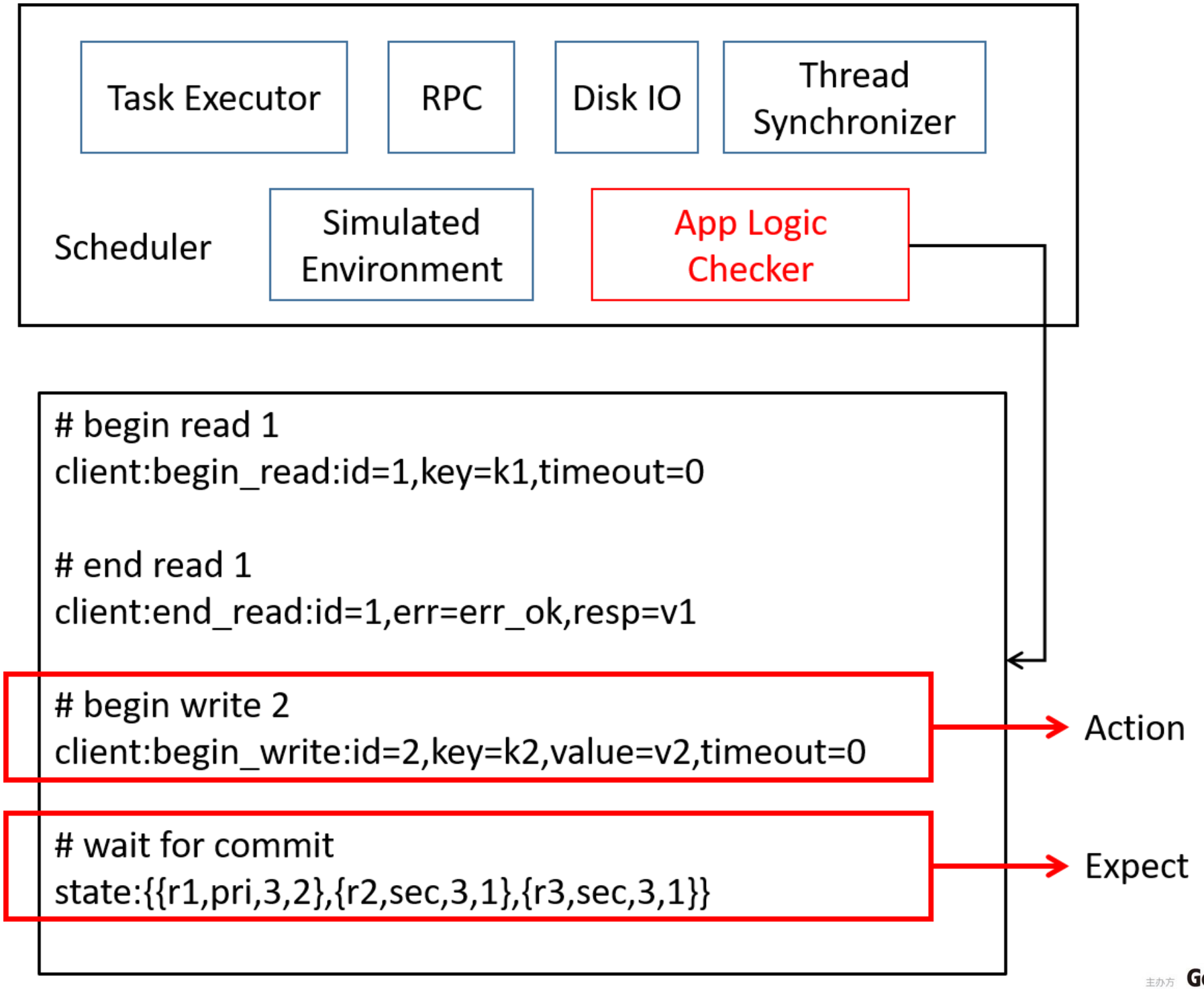


模拟IO的不确定性

- 以一定的概率注入错误
- 业务逻辑assert、添加全局状态检查模块
- 发生异常时：用相同的随机数种子重新运行



应用case：一致性协议的单元测试



分布式框架rDSN

- 异步编程模型
- Deterministic框架：模拟测试、 debug
- https://github.com/Microsoft/rDSN

TABLE OF
CONTENTS 大纲

现状

- 设计、接口、存储引擎、benchmark、对比HBase ：http://bj2016.archsummit.com/presentation/3023
- 公司内上线服务业务
- 开源：修改过的rDSN框架、带replication的kv数据库
- 待开源：跨机房复制、snapshot备份

计划

- 持续开源更多功能
- RESTful API
- 多租户支持
- 支持Schema、SQL、跨行Transaction

TABLE OF
CONTENTS 大纲

对于一个好的项目

- 关注业务
  - 明确业务需求

- 关注架构
  - 一致性、可用性、扩展性、性能

- 关注软件工程
  - 可维护性、测试、监控

THANKS!

附录

- PacificA一致性协议：https://www.microsoft.com/en-us/research/publication/pacifica-replication-in-log-based-distributed-storage-systems/
- 《ArchSummit 2016北京》关于Pegasus的技术分享：http://bj2016.archsummit.com/presentation/3023