我们仍未知道那天所分享的 Raft 的意思

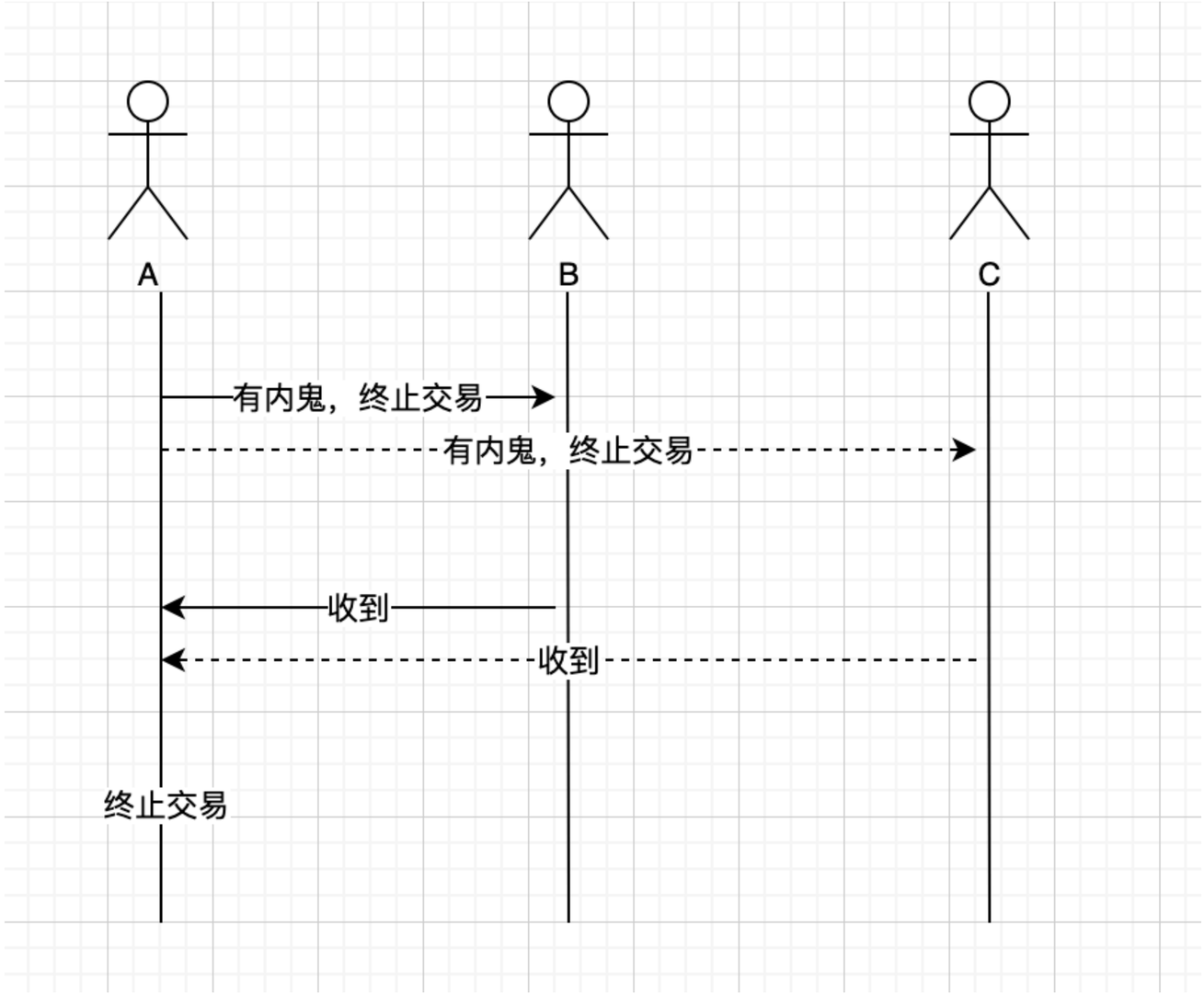
vizee

Re：从零开始分布式计算

Consensus

共识

- 多个参与者(Participant)
- 达成广泛一致(Agreement)



Consistency model

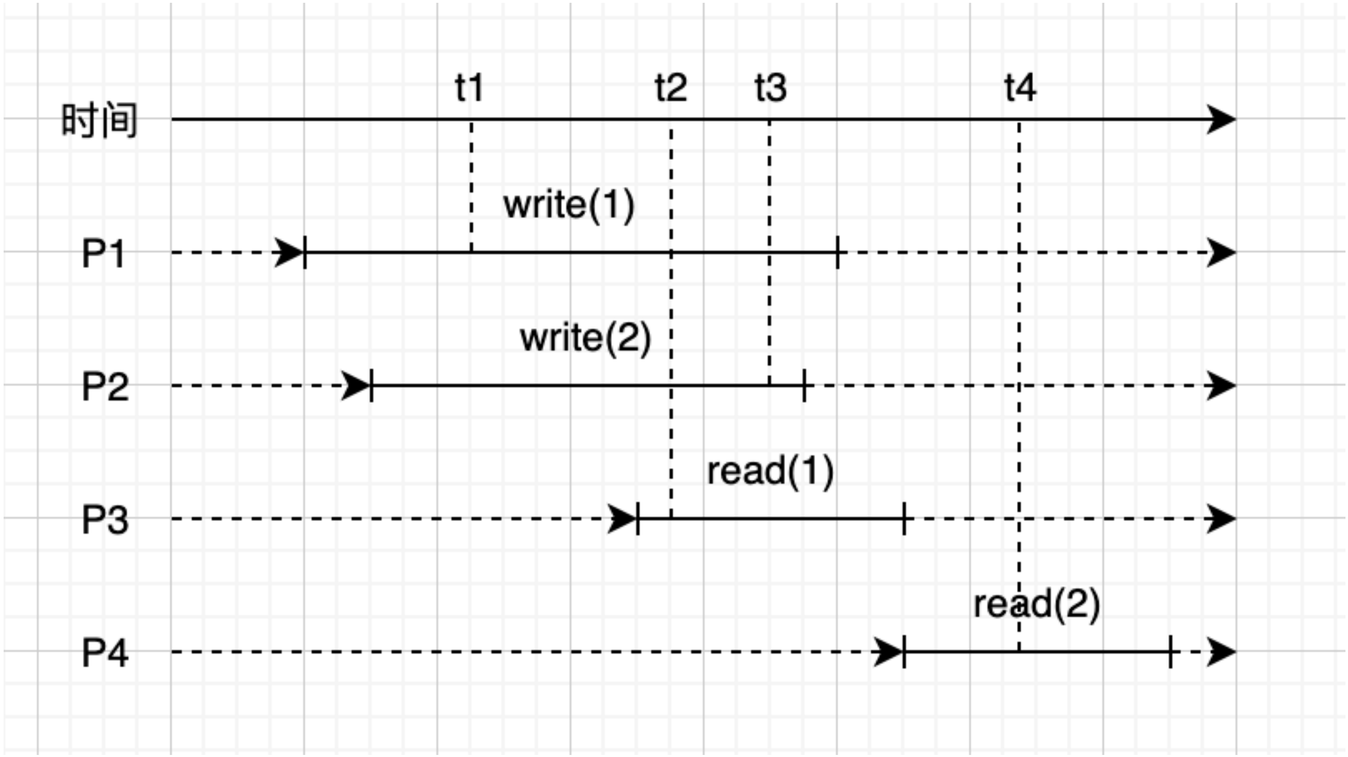
一致性模型

- 严格一致性(Strict consistency)
- 顺序一致性(Sequential consistency)
- 因果一致性(Causal consistency)
- 最终一致性(Eventual consistency)

Linearizability

可线性化语义

- 线性化点(Linearization point)
- 事件排序
- 全序
- 原子性语义



CAP theorem

CAP 定理

- 分布式系统只能满足其中两个性质：
  - 一致性(Consistency)：每个进程都能访问到最新的数据
  - 可用性(Availability)：每个请求都能成功
  - 分区容忍性(Partition tolerance)：容忍进程之间通信出现分区

FLP Impossibility

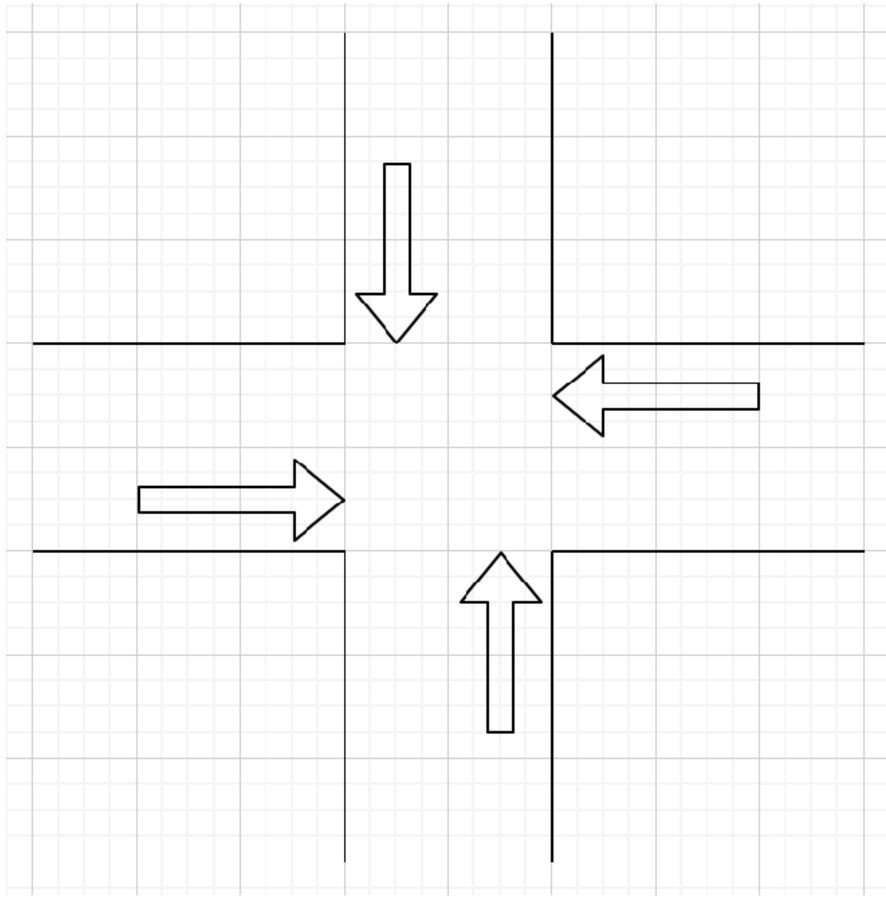
FLP 不可能性

- 共识算法必须满足：
  - 可终止性(Termination)： 每个正确的进程最终会决定一个值
  - 一致性(Agreement)： 所有进程必须同意同一个值
  - 有效性(Validity)：被决定的值必须由正确的进程提出
- FLP 不可能定理：No consensus protocol is totally correct in spite of one fault.

Correctness

正确性

- 安全性(Safety)：保证坏事不会发生
- 活性(Liveness)：最终好事一定会发生



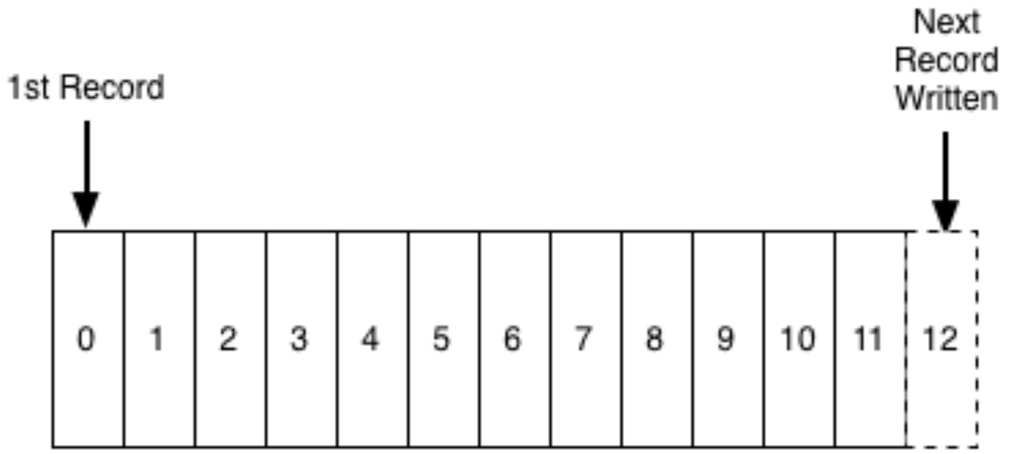
Quorum

法定人数

- 鴿巢原理
- Quorum Protocol
  - R + W > N
- 多数派(Majority)
  - W + W > N

Log
日志

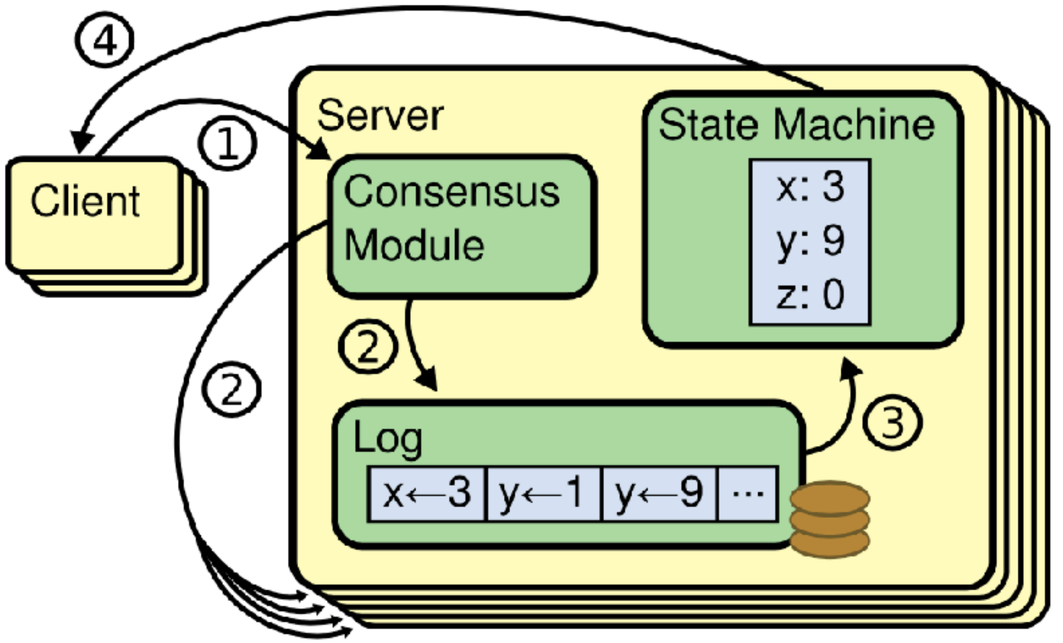
- 顺序(Ordering)
- 索引(Index)
- 仅追加(Append-only)原则



State Machine Replication

状态机复制

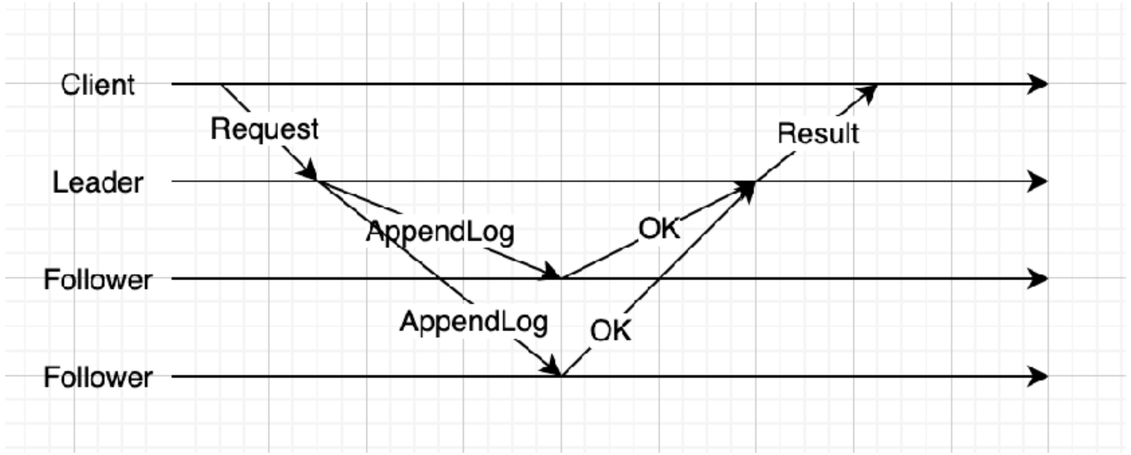
- 确定性(deterministic)状态机
- 起始状态相同，输入状态相同，执行后状态必定相同



某科学的 Raft 算法

Overview

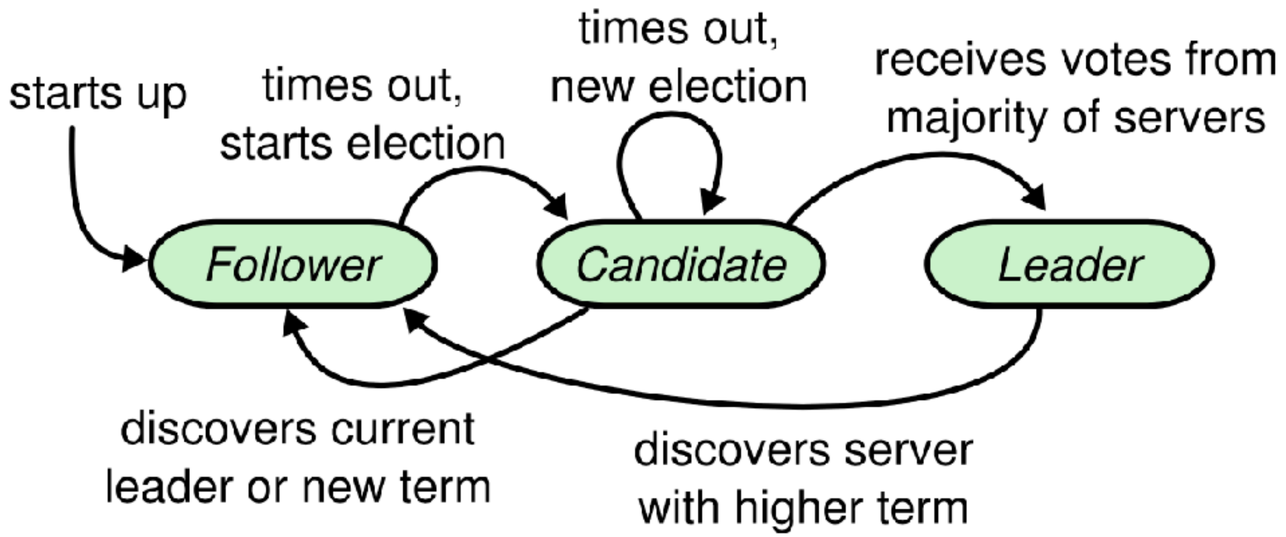
- 服务器状态(Server State)
- Leader 选举(Leader election)
- Log 复制(Log replication)



Server State

服务器状态

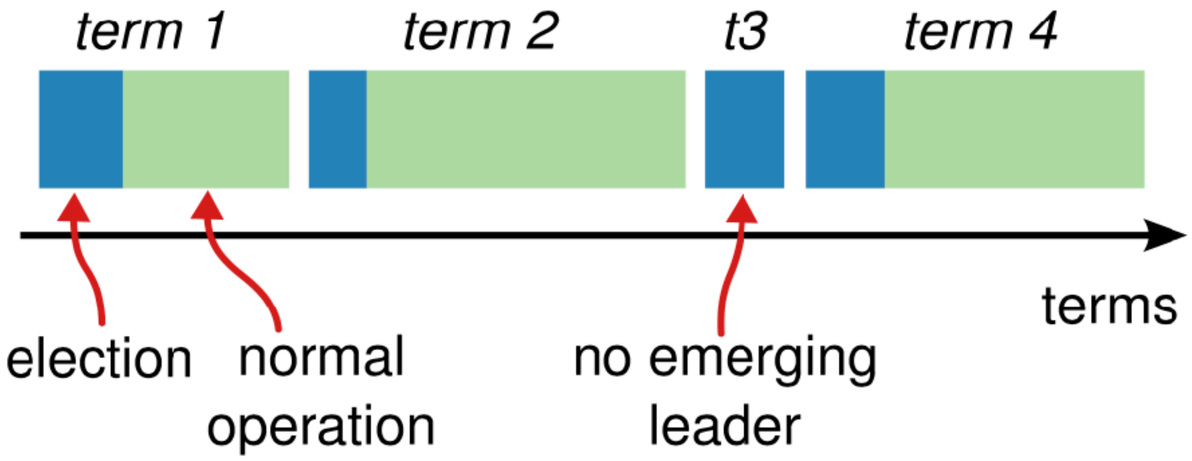
- Leader：复制 Log 到 Follower
- Candidate：选举 Leader
- Follower：接受 Leader Log



Leader Election

Leader 选举

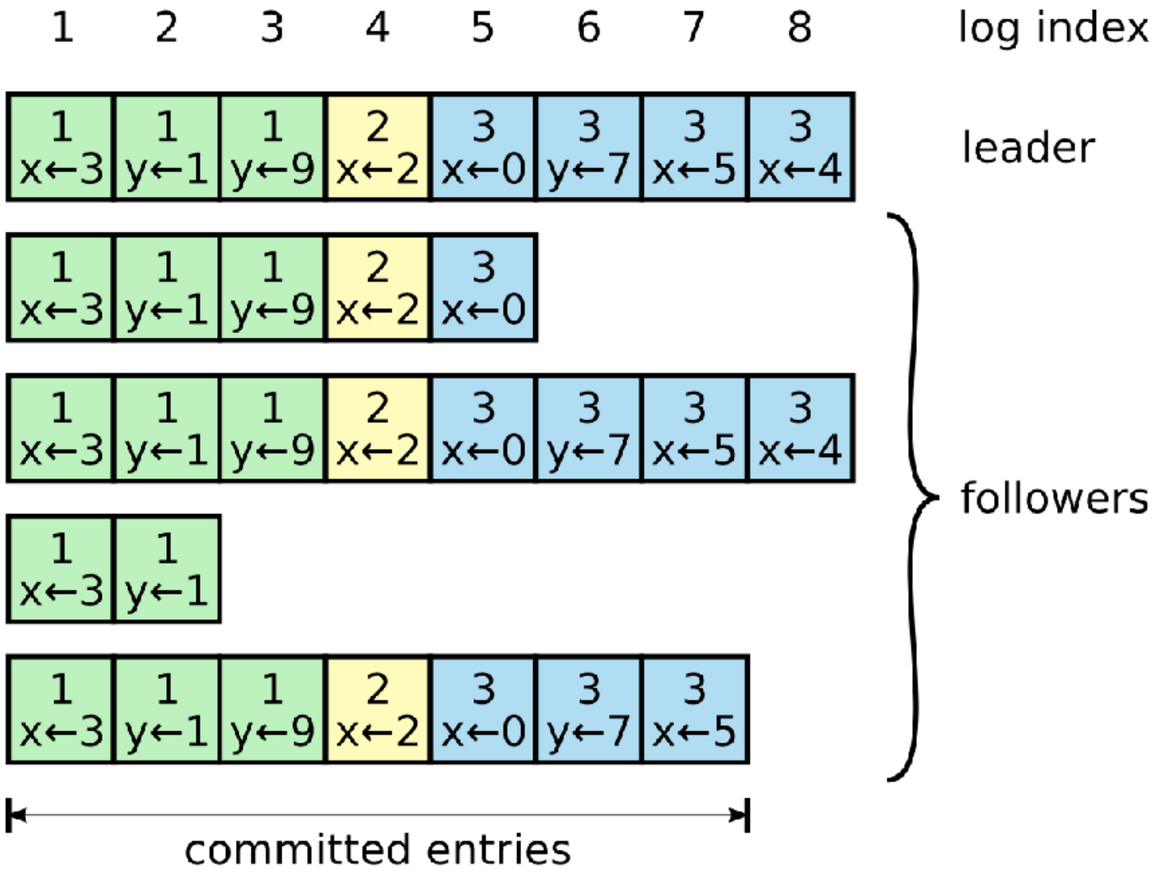
- 任期(Term)
- 投票
- 选举约束



Log Replication

日志复制

- 复制(Replication)
- 提交(Committing)



Raft Properties

Raft 属性

- 选举安全性(Election Safety)：每个 Term 至多选举出一个 Leader。
- Leader 日志仅追加(Leader Append-Only)：Leader 只会追加日志，不会是覆盖或删除日志。
- 日志匹配(Log Matching)：如果两个副本存在日志 Term 和 Index 相同，那么可以认为这两条日志前所有日志条目都相同。
- Leader 完整性(Leader Completeness)：如果一条日志成功提交，那么这条日志的 Term 为当前 Leader 最高 Term。
- 状态机安全性(State Machine Safety)：如果服务器状态机在给定 Index 执行了一条日志，那么其他服务器不会在这个 Index 执行不同的日志。

路人 Raft 的养成方法

Server Side

服务端实现

RPC

- VoteRequest
- AppendEntries
  - 加速 Log 匹配
- 幂等性(Idempotence)

AppendEntries RPC

Invoked by leader to replicate log entries (§3.5); also used as heartbeat (§3.4).

Arguments:

term	leader's term
leaderId	so follower can redirect clients
prevLogIndex	index of log entry immediately preceding new ones
prevLogTerm	term of prevLogIndex entry
entries[]	log entries to store (empty for heartbeat; may send more than one for efficiency)
leaderCommit	leader's commitIndex

Results:

term	currentTerm, for leader to update itself
success	true if follower contained entry matching prevLogIndex and prevLogTerm

Receiver implementation:

1. Reply false if term < currentTerm (§3.3)
2. Reply false if log doesn't contain an entry at prevLogIndex whose term matches prevLogTerm (§3.5)
3. If an existing entry conflicts with a new one (same index but different terms), delete the existing entry and all that follow it (§3.5)
4. Append any new entries not already in the log
5. If leaderCommit > commitIndex, set commitIndex = min(leaderCommit, index of last new entry)

RequestVote RPC

Invoked by candidates to gather votes (§3.4).

Arguments:

term	candidate's term
candidateId	candidate requesting vote
lastLogIndex	index of candidate's last log entry (§3.6)
lastLogTerm	term of candidate's last log entry (§3.6)

Results:

term	currentTerm, for candidate to update itself
voteGranted	true means candidate received vote

Receiver implementation:

1. Reply false if term < currentTerm (§3.3)
2. If votedFor is null or candidateId, and candidate's log is at least as up-to-date as receiver's log, grant vote (§3.4, §3.6)

State Changing

状态变更

- 成为 Leader
  - 重置 matchIndex
  - 立即广播心跳
- 成为 Candidate
  - Term + 1
  - 投票给自己
- 成为 Follower
  - Term 可能变化
  - 清空投票状态
  - 重置心跳超时

Follower Log Matching

Follower 日志匹配

- 对于每个 Follower，nextIndex 初始化为 Leader 最后 Log Index + 1
- 如果 Follower 无法接受 Leader 的 Log
  - Leader 减小 nextIndex 后重试
  - Follower 可以返回最后 Log 的元信息加速确定 nextIndex
- 如果 Follower 接受 Log
  - nextIndex 继续更新为发生成功的 Log Index + 1

Log Committing

日志提交

- 每当成功向 Follower 发送 Log
  - 更新 Follower 的 matchIndex
  - 如果所发送 Log 最后 Index 高于 commitIndex，且超过半数 Follower 的 matchIndex 大于等于所发送的最后 Index，更新 commitIndex 为最后发送的 Index
  - 向 Follower 广播 commitIndex

Persistence

持久化

- 持久化状态：
  - Term：防止处理过期消息
  - VoteFor: 结合 Term 防止重复投票
  - Log{Term, Index, Command} 序列：防止已经提交的消息丢失
- 可恢复的状态：
  - commitIndex：通过 Leader 心跳快速恢复，etcd 实现持久化
  - appliedIndex：状态机相关状态，通过启动后执行 Log 恢复

Timing
时机

broadcastTime << electionTimeout << MTBF

- broadcastTime：服务器广播消息且收到回复的平均值，0.5ms-20ms
- electionTimeout: 选举/心跳超时，10ms-500ms
  - 在某些系统里希望尽快发现 Leader 故障，心跳超时单独设置：**broadcastTime << heartbeatTimeout <= electionTimeout**
- MTBF：平均故障间隔时间，不稳定的系统必然使超时无效，通常几个月左右

Client Side

ClientRequest

- 全局唯一客户端 ID 和顺序号
- 如果服务器不是 Leader，返回 Leader 地址，客户端重试（而不是 Follower 代理）

ClientRequest RPC

Invoked by clients to modify the replicated state.

Arguments:

clientId	client invoking request (§6.3)
sequenceNum	to eliminate duplicates (§6.4)
command	request for state machine, may affect state

Results:

status	OK if state machine applied command
response	state machine output, if successful
leaderHint	address of recent leader, if known (§6.2)

Receiver implementation:

1. Reply NOT_LEADER if not leader, providing hint when available (§6.2)
2. Append command to log, replicate and commit it
3. Reply SESSION_EXPIRED if no record of clientId or if response for client's sequenceNum already discarded (§6.3)
4. If sequenceNum already processed from client, reply OK with stored response (§6.3)
5. Apply command in log order
6. Save state machine output with sequenceNum for client, discard any prior response for client (§6.3)
7. Reply OK with state machine output

Linearizable semantics

线性化语义

- 至少一次(at-least-once)和精确一次(exactly-once)
- 读操作有效性
  - 服务端读方式
  - 维护最后访问 Index

RegisterClient RPC

Invoked by new clients to open new session, used to eliminate duplicate requests. §6.3

No arguments

Results:

status	OK if state machine registered client
clientId	unique identifier for client session
leaderHint	address of recent leader, if known

Receiver implementation:

1. Reply NOT_LEADER if not leader, providing hint when available (§6.2)
2. Append register command to log, replicate and commit it
3. Apply command in log order, allocating session for new client
4. Reply OK with unique client identifier (the log index of this register command can be used)

进击的 Raft

Leader Transfer

Leader 主动变更

1. 前 Leader 停止接受请求，但和 Follower 保持心跳
2. 前 Leader 向目标服务器同步 Log
3. 前 Leader 向目标服务器发送 TimeoutNow，目标服务器立即开始选举
4. 如果目标服务器无法成为 Leader，前 Leader 继续接受请求

Configuration Changing

配置变更/成员关系(membership)变更

- 单服务器(Single Server)
- 联合共识(Joint Consensus)

Single Server

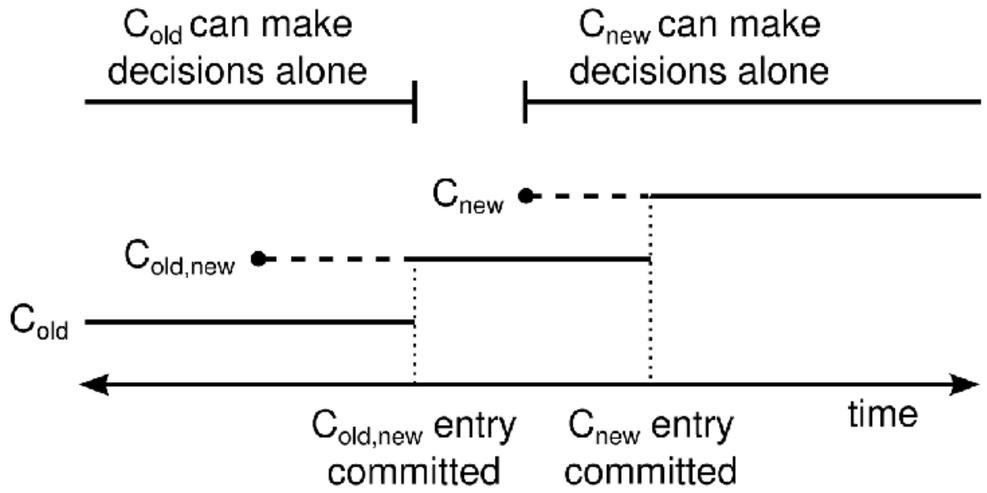
每次变更一个节点

- 一次只添加或删除一台服务器
- 配置通过 Raft 机制同步
- 服务器收到配置 Log 后立即生效(etcd 实现中为 Log 执行时生效)
- 先添加再删除原则
- 两成员集群问题

Joint Consensus

联合共识

- 任意配置变更
- 配置通过 Raft 机制同步
- 配置 Log 在 Commit 时生效
  - 旧配置 Leader 不在新配置中，新配置 Commit 后退出
- Leader 要同时成为联合配置的 Leader，Log 同时在联合配置中提交



Pre-Vote

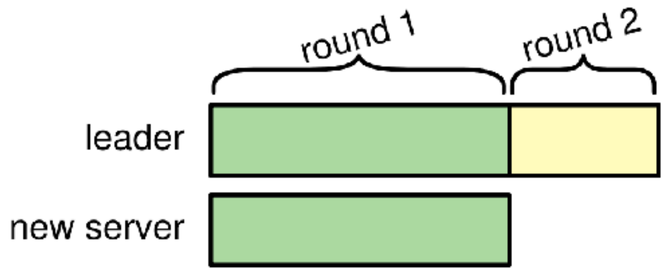
预选举

- 服务器选举前广播 PreVote 请求，Term 不变
- 如果大多数 Follower 响应请求，转为 Candidate，增加 Term

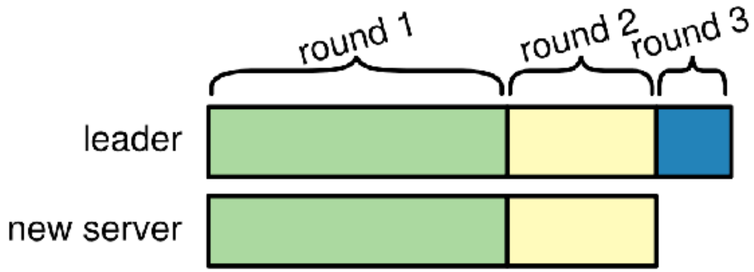
Log Catching up

新服务器追赶日志

- 加入时不参与投票
- 多轮同步
- 在日志追上 Leader 后，Leader 提交配置变更



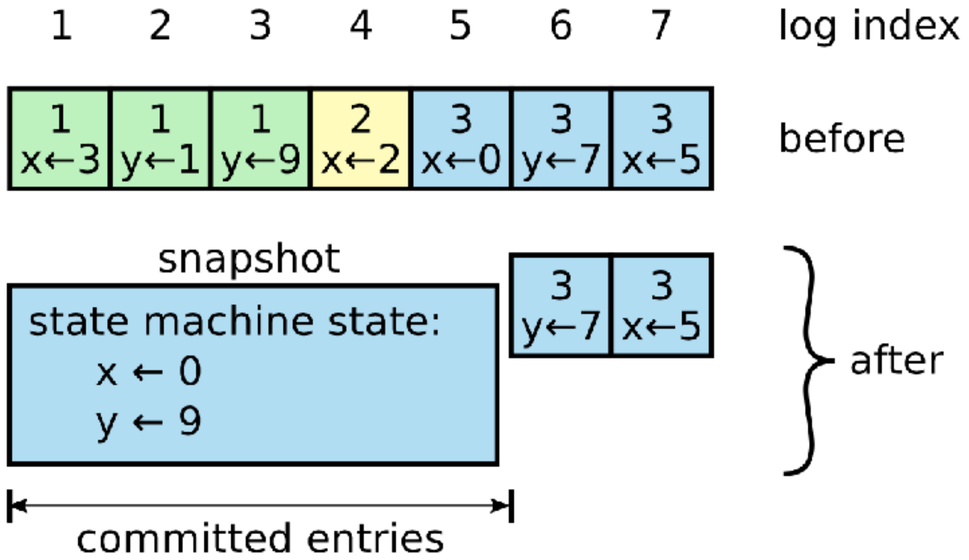
(a) Start of round 2.



(b) End of round 2.

Log Compaction
日志压缩

- 日志过长恢复状态机困难
- 快照和服务器相关



Read-Only Request

只读请求

Log Read
日志读

1. Leader 处理读请求
2. Leader 提交一个 read-op Log
3. 等待状态机执行 read-op
4. 返回客户端状态机执行结果

Read-Index Read

读索引读

- 最新 commitIndex
- 选择 readIndex
- Leader 权威
- 当 appliedIndex >= readIndex，Leader 直接提供读操作

Read-Index Read 达成时：commitIndex >= appliedIndex >= readIndex

新 Leader 需要 no-op Log 成功 Commit 后才能提供 Read-Index Read

Follower Read

Follower 读

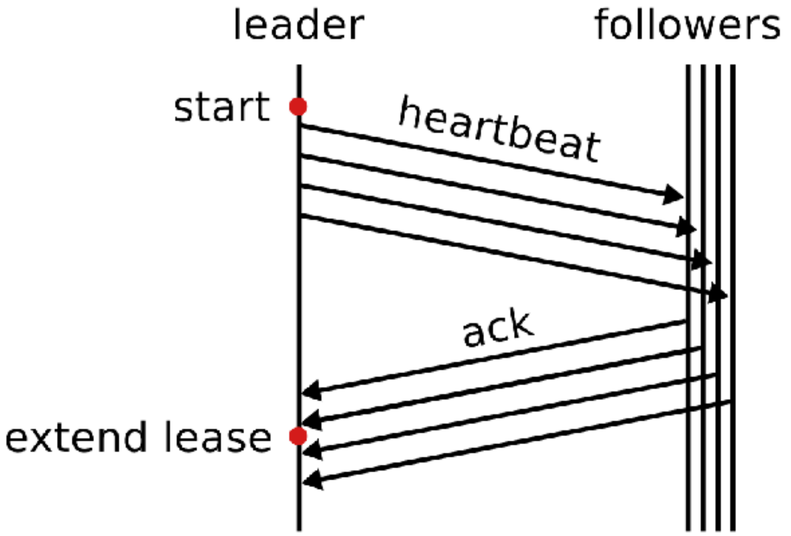
1. 从 Leader 获取 readIndex
2. Leader 确认权威后返回 readIndex
3. Follower 等待 appliedIndex 满足条件后提供读操作

Read-Index Read 每次读都要 Leader 通过心跳确认（维持）权威

Lease Read

租约读

1. Leader 心跳时带上本地时间 start
2. 如果大多数 Follower 回复成功，则认为本地时间在 $[start, start + \frac{election\ timeout}{clock\ drift\ bound})$内时，Leader 可以直接回复读操作
3. Leader 定期续租



Lease Read Corner Case

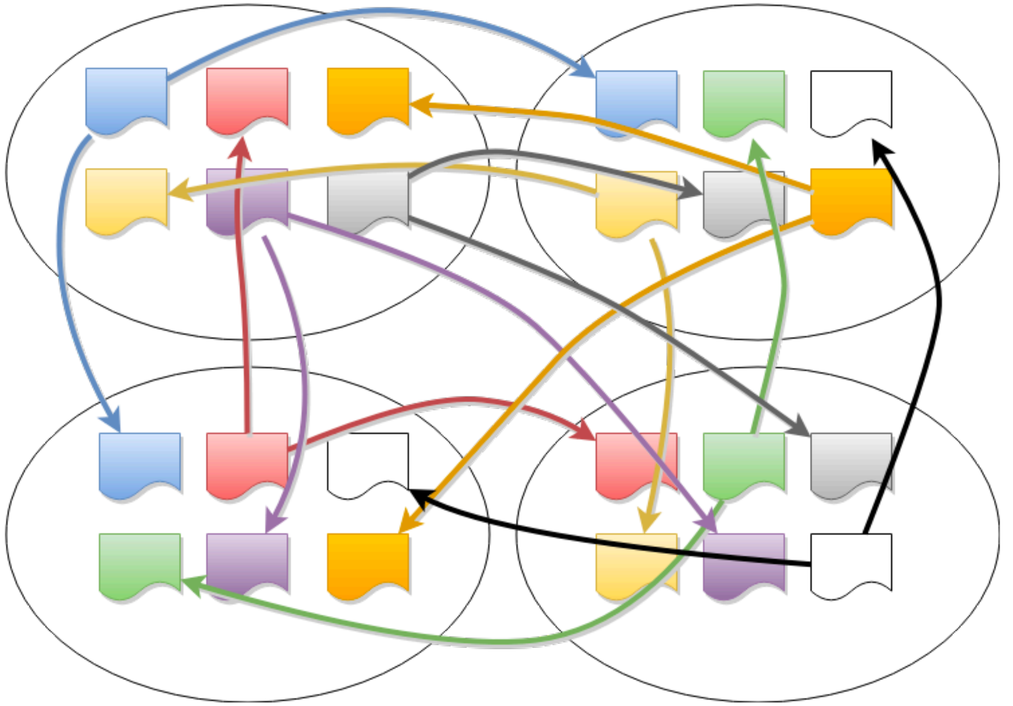
租约读边角情况

- 本地时钟漂移(clock drift)限制
- 单调时钟和 NTP 时钟回拨
- Leadership Transfer 机制导致 Leader 提前变更

Raft Group

Raft 组/Multi-Raft/Scaling Raft

- 每个 Server 会同时服务于多个 Raft 组
- 每个 Raft 组都有自己的 Leader，通常不会在相同 Server
- Log 按一定规则分配给不同的 Raft 组，通常基于 KV 分片(Sharding)



我的 Raft 不可能这么优化

Batching

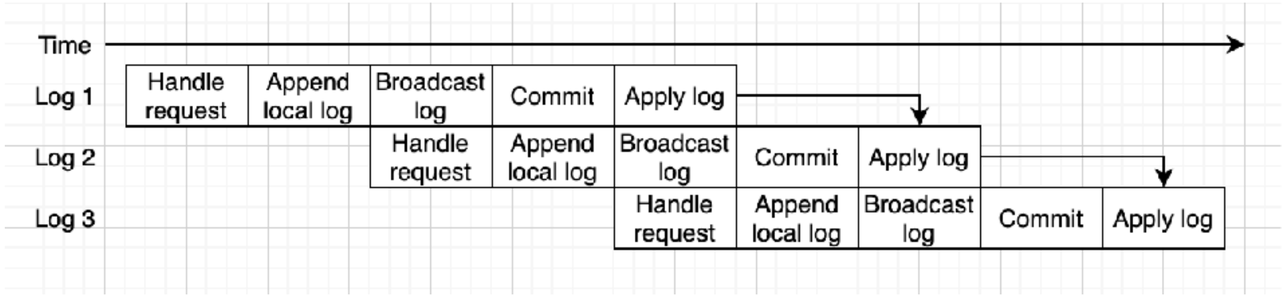
批量提交

- Leader 批量发送 Log
- Follower 只需确认目前最后成功的 Log Index

Pipelining

流水线

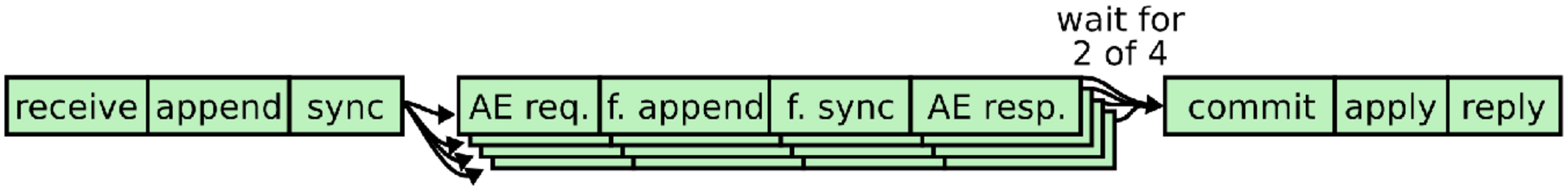
1. Leader 收到请求
2. 追加到本地 Log
3. 开始广播 Log 给 Follower，并立即开始处理下一个请求
4. 收到大多数 Follower 确认后更新 commitIndex，等待前一条 Log 执行成功
5. 开始执行 Log
6. 回复客户端请求



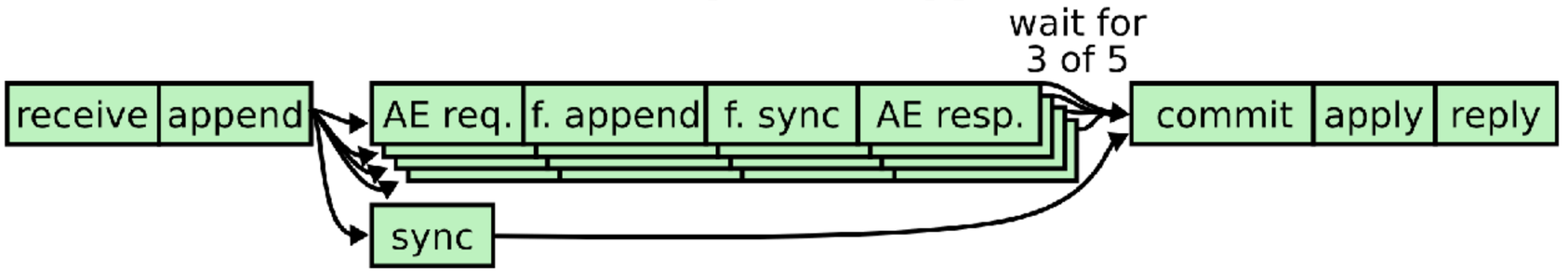
Writing Disk on Parallel

并发写盘

- 避免 Leader 追加本地 Log 时 IO 同步等待
- 立即广播并同时执行持久化



(a) Unoptimized Raft pipeline.



(b) Optimized Raft pipeline.

Applying Log Asynchronous

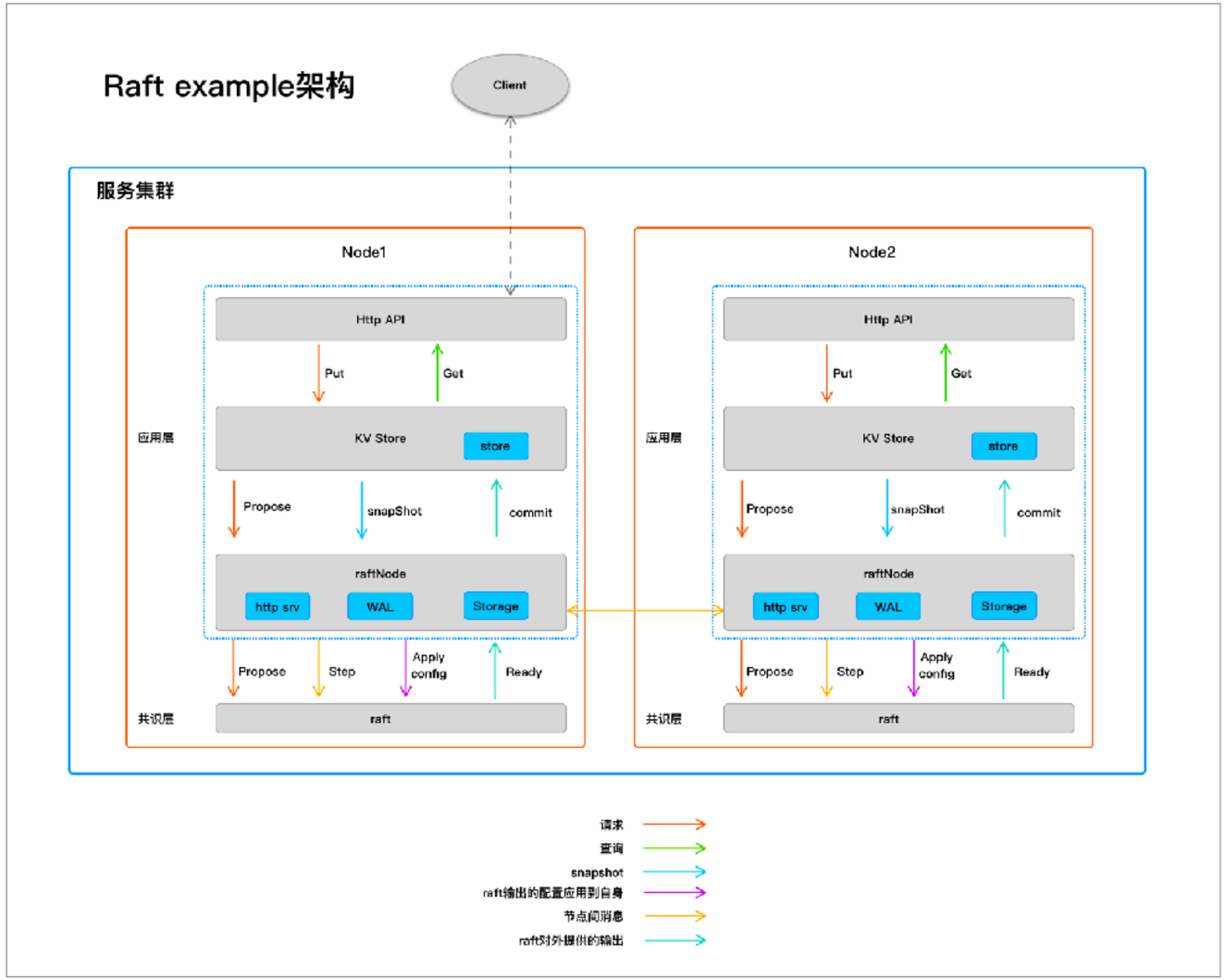
异步执行日志

- 执行 Log 不占用 Raft 流水线
- 负责 Log 执行的线程回复客户端请求

Raft 的彼方

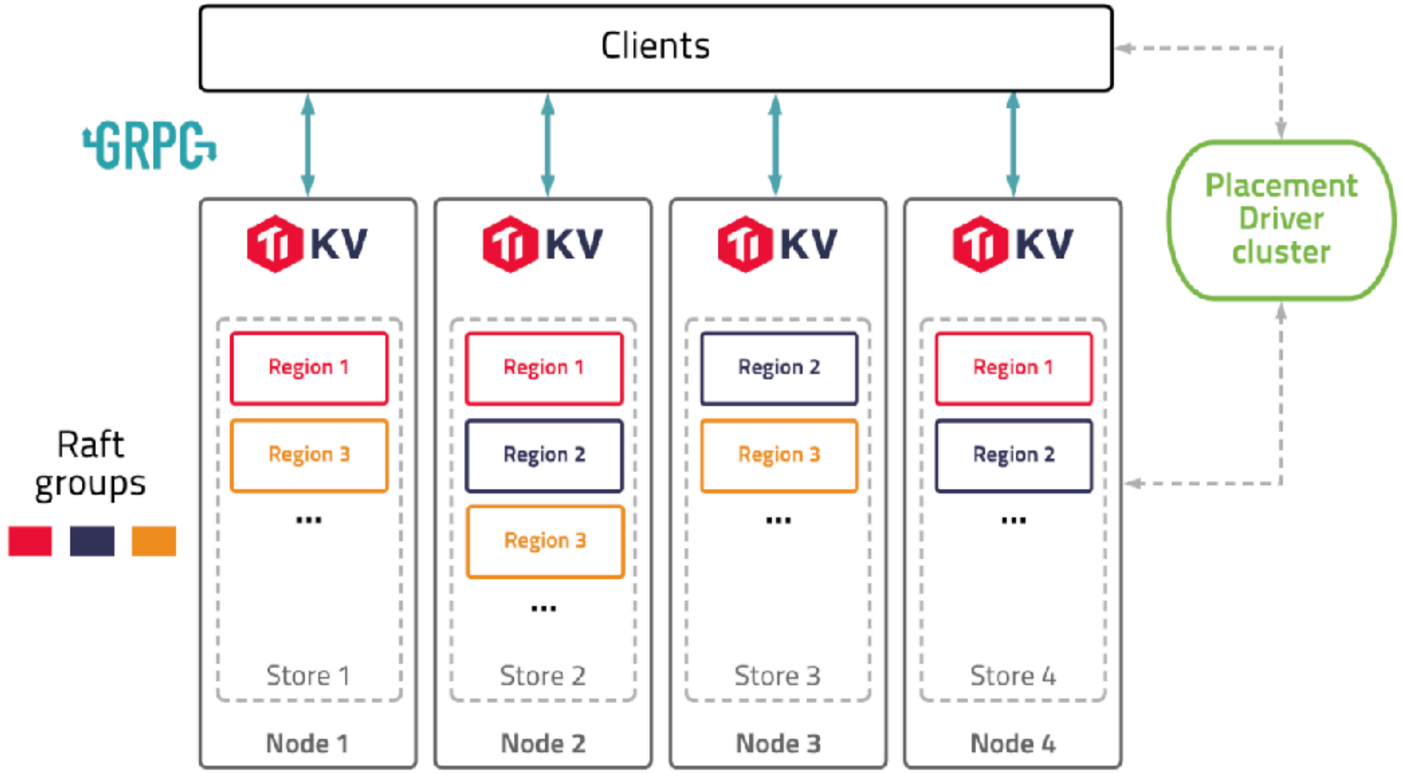
etcd

A distributed, reliable key-value store for the most critical data of a distributed system



TiKV

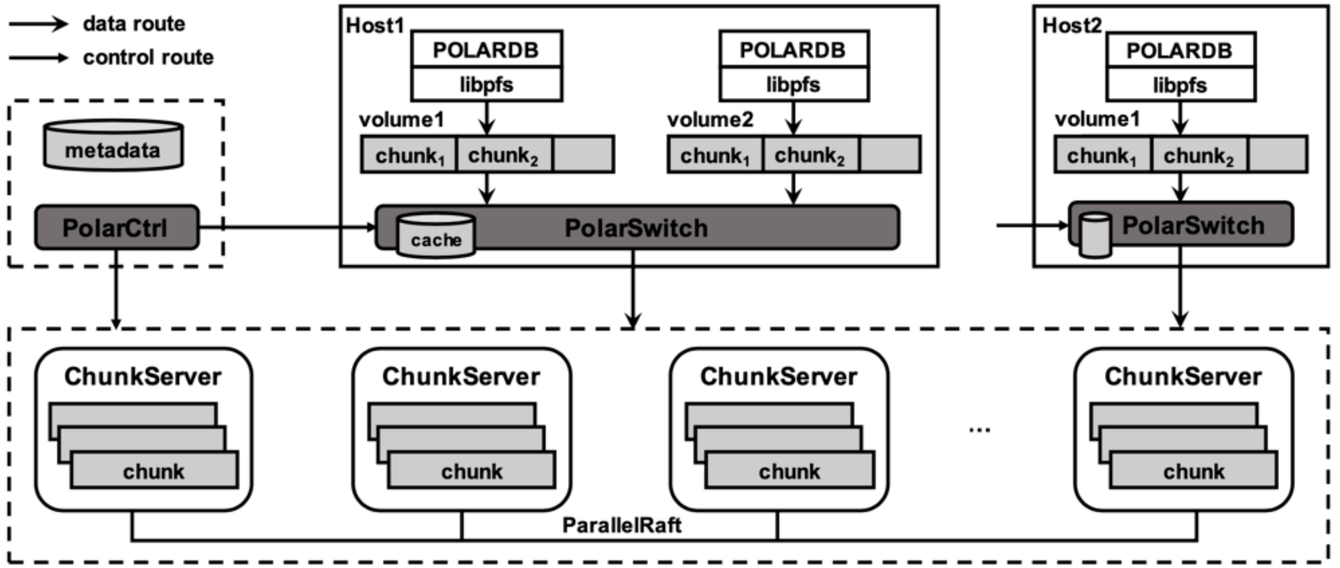
A distributed transactional key-value database



PolarFS

An Ultra-low Latency and Failure Resilient Distributed File System for Shared Storage Cloud Database

- ParallelRaft
  - 乱序复制
  - 乱序应答
  - 乱序提交
  - 空洞日志(Hole log)
  - Look Behind Buffer



终物语

共识

- 共识算法属性
- 可线性化
- Quorum
- SMR

Raft

- 长期 Leader
- 容错
- 连续日志
- 可线性化语义
- 易于学习

完结撒花