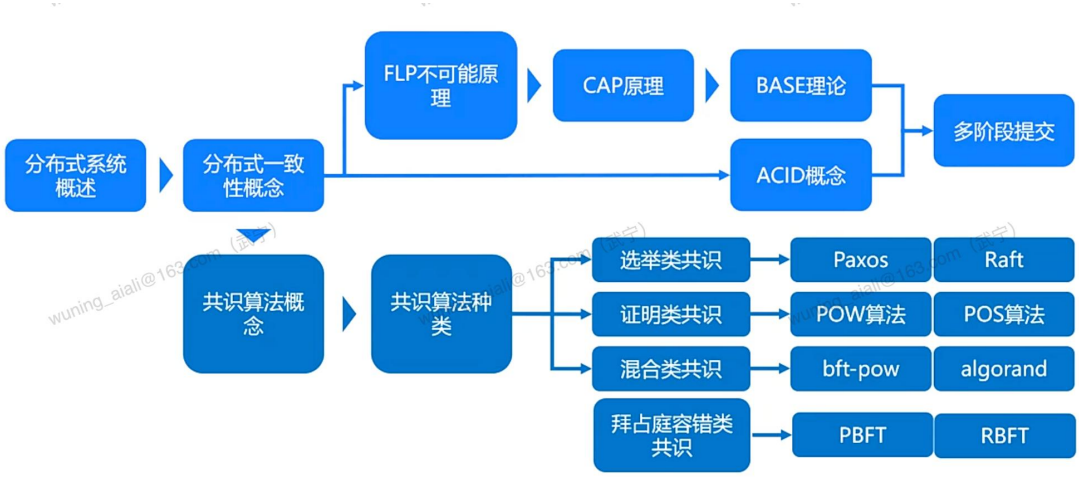
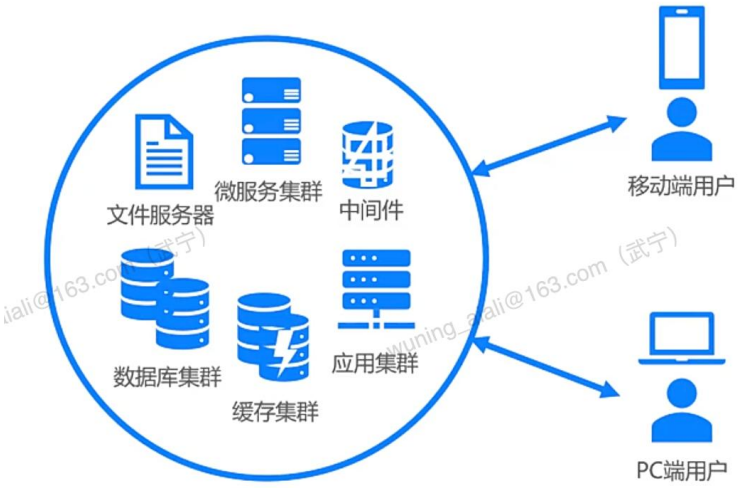
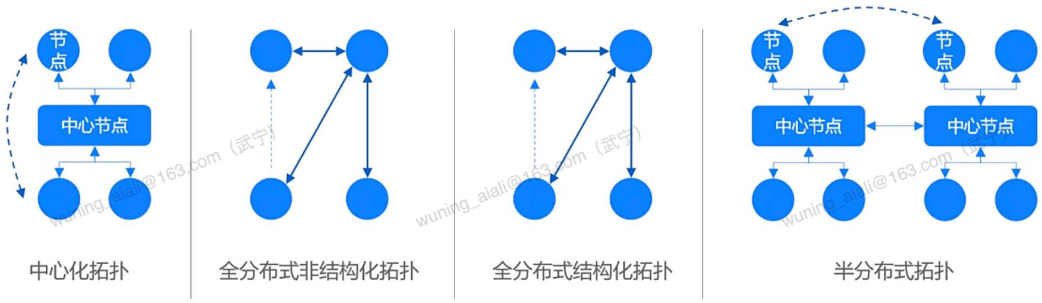
区块链学习笔记 -- 区块链分布式



分布式系统:



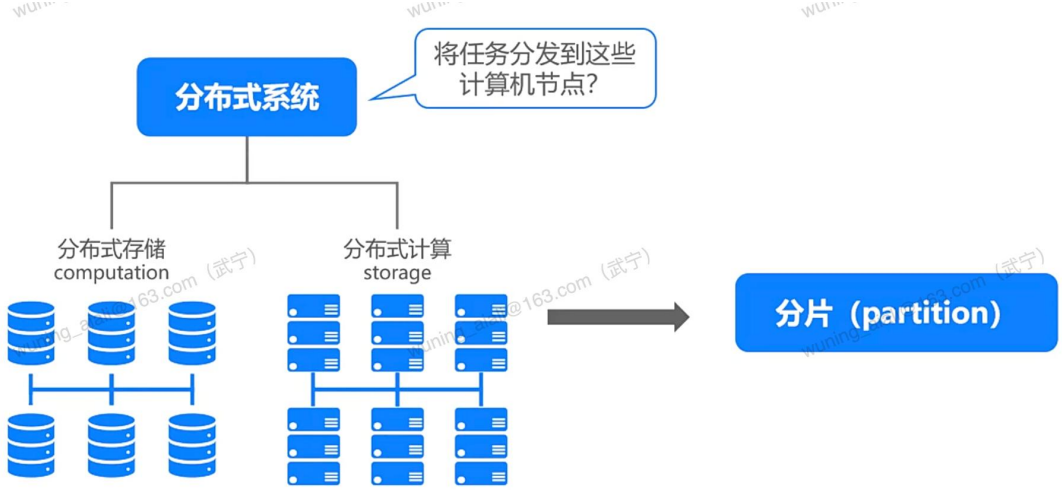
分布式拓扑结构:



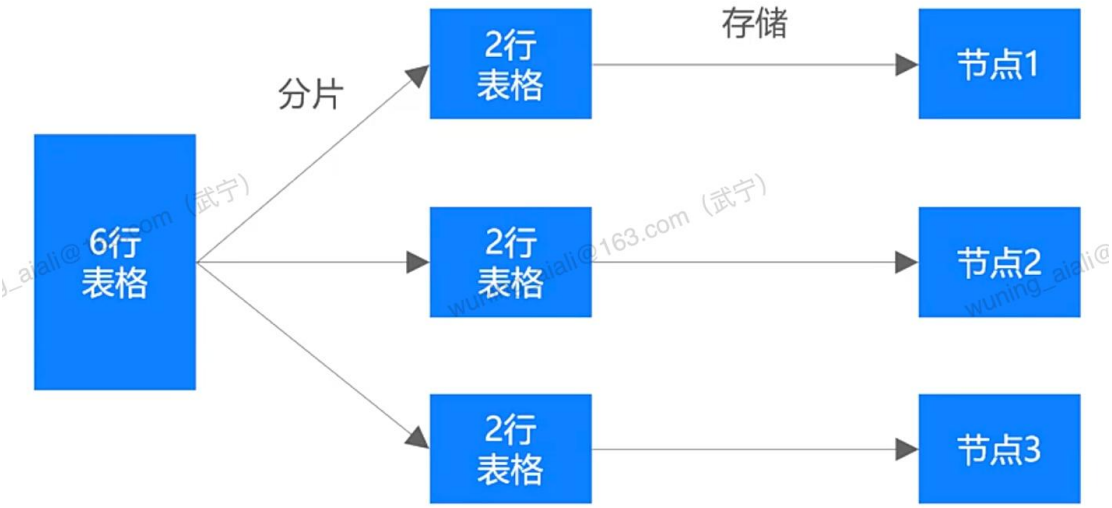
联盟链：采用中心化主从拓扑结构

公链：采用全分布式非结构化拓扑结构

分布式系统的基本机制：



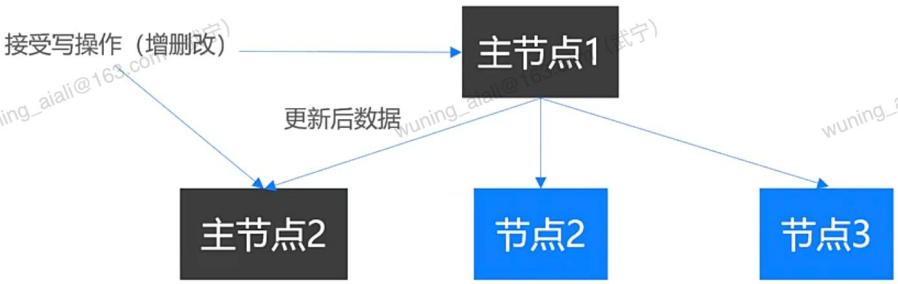
数据分片存储：将数据集划分为更小的不同的数据集。以避免存储空间不足，计算资源不足，网络带宽不够的问题



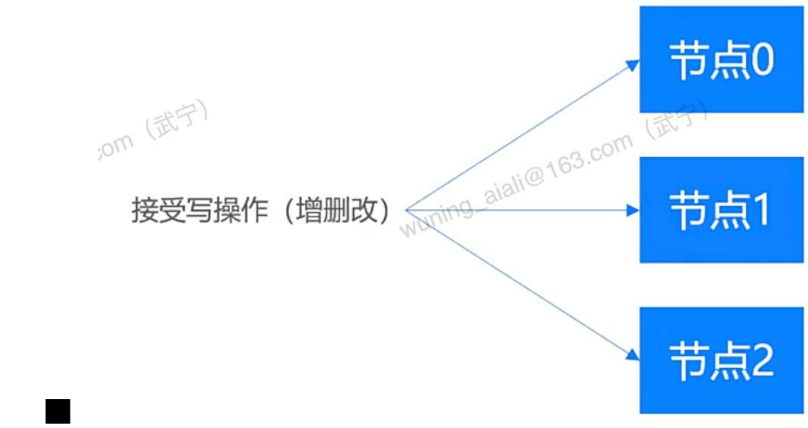
数据复制：在不同的节点上保存相同的节点副本，以提高数据的容错性，如节点 1 出现问题就可以从节点 4 获得备份数据。

数据一致性问题：

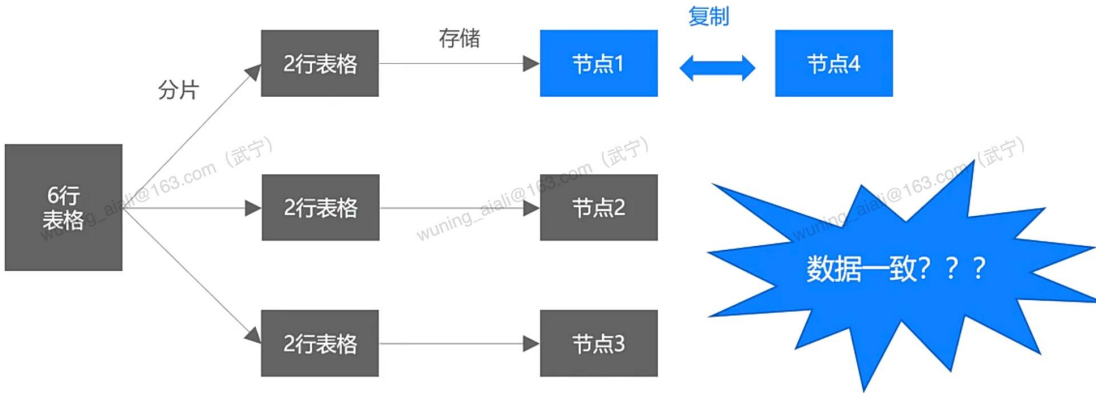
- 主从复制：由主节点接受写操作，然后将更新后的数据发送到所有从节点，读操作请求则在主从上都可以进行 ---中心化模式
- 多主节点复制：主节点都可以充当其他节点的从节点，相互备份



- 无主从复制
  - 所有节点都支持直接写操作请求



■



分布式系统的优点：

1. 资源共享：多节点通过网络彼此互联

2. 加快计算速度：将一组计算任务拆分成多个字任务，分派给不同的节点运行；负载均衡，如果单一节点任务过重，可对节点任务进行迁移。

3. 可靠性：多节点作业，具有高鲁棒性和容错性

4. 灵活性：易安装、易调试

缺点：架构复杂、维护成本高、故障排查难度大

1. 系统复杂度增加

2. 管理成本高

3. 维护难度增加

挑战：

1. 异构的机器与网络：多节点相互链接通信增加了节点维护成本和网络传输摩擦成本

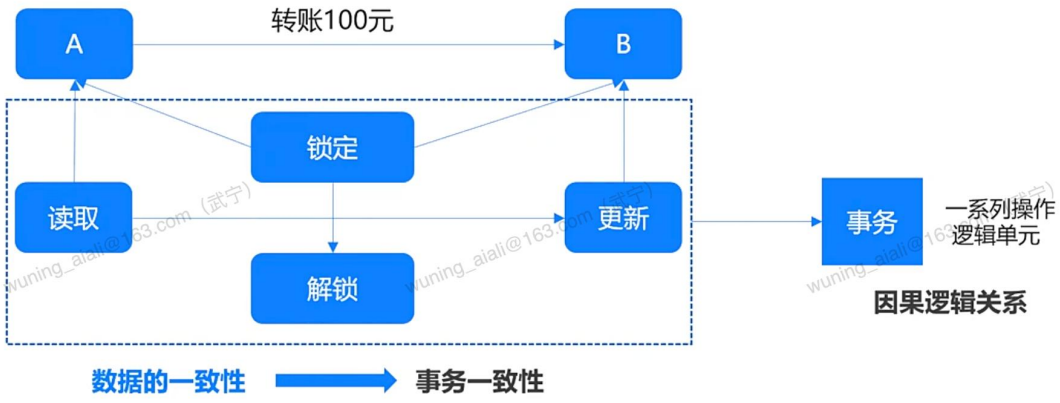
2. 节点故障容灾功能，需要监控节点的运行状态

3. 不可靠网络，发送方和接收方网络信息的确认

分布式一致性：

1. 数据一致性：多节点对某一变量的取值达成一致，一旦达成一致，则变量的本次取值便达成一致。

2. 事务（逻辑）一致性：



一致性类别：

1. 强一致性（原子一致性/线性一致性）：数据更新后，任何时刻所有用户或进程查询结果都是最新一次更次的数据。一个集群需要提供强一致性，集群内部某一节点数据发生改变就需要等待集群内其他节点同步更新数据后，才可以再次提供给服务。保证强一致性则需要牺牲高可用性。

2. 弱一致性：数据更新后，可能只能访问部分更新后的数据或访问不到更新后的数据

3. 最终一致性：是弱一致性的一种特殊形式，一段时间后，数据会达到一致状态

4. 因果一致性：事务执行的前后逻辑保持一致

一致性出现的背景：

1. 节点间的网络通讯是不可靠的

2. 节点本身的逻辑可能是错误的

3. 同步调用会让系统变得不具备可扩展性

为解决一致性的问题分布式系统的要求：

1. 可终止性：一致的结果在有限的时间内可以完成同步

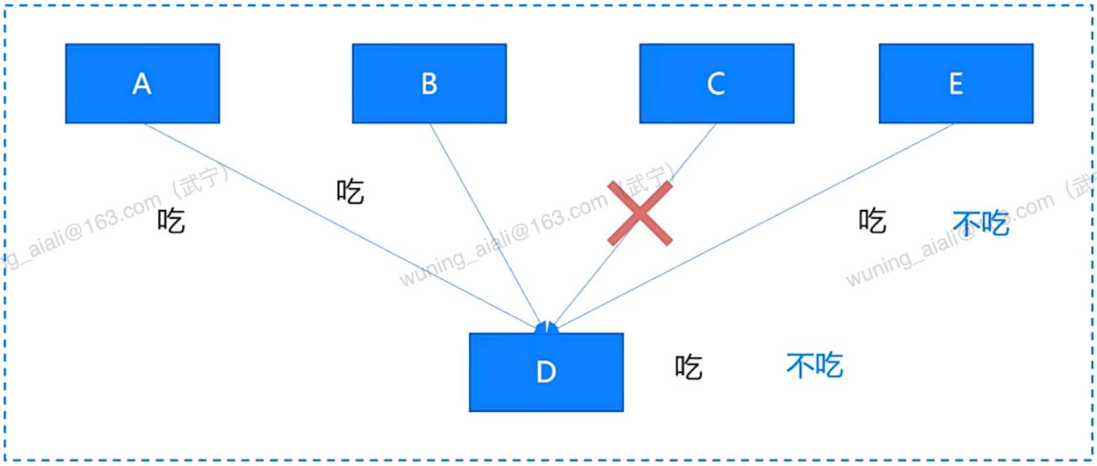
2. 共识性：不同的节点最终完成决策的结果要求相同

3. 合法性：决策的结果必须是其他进程提出的提案

FLP 不可能原理：

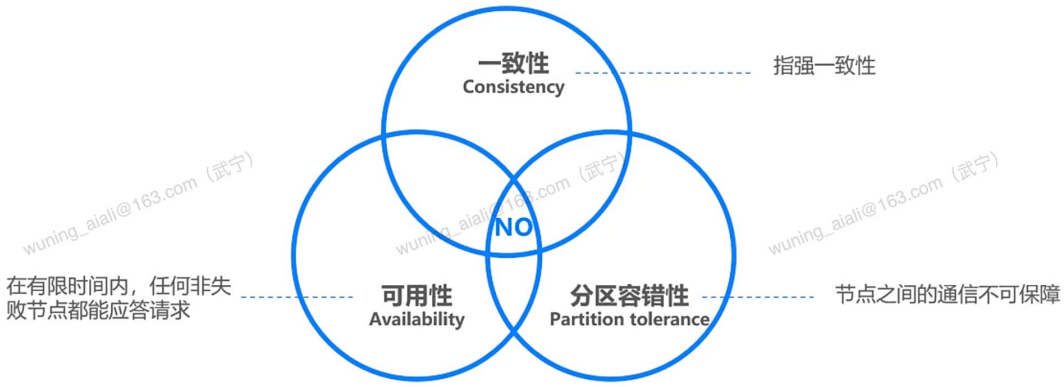
概述：在网络可靠，但允许节点失效（即便是只有一个节点失效）的最小化异步模型系统中，不存在一个可以解决一致性问题的确定性共识算法。 -- 《一个故障过程导致分布式共识无效》

不存在一种模型能够适用于分布式中的任何场景。开发者不必追求完美的解决方案，只要能够满足问题的实际要求即可



CAP 原理：

分布式技术系统不可能同时保持三种特性：一致性、可用性、分区容忍性

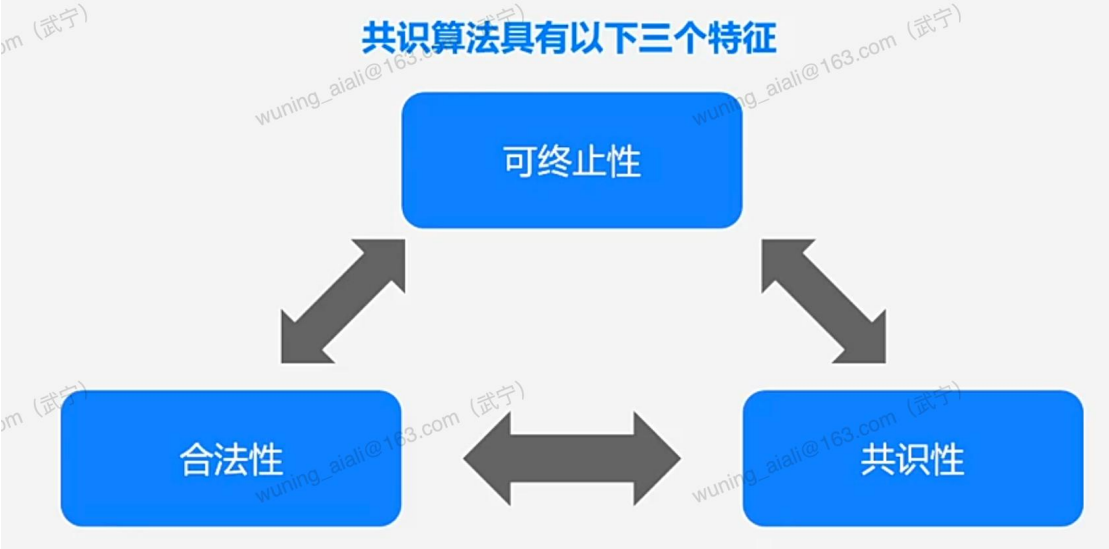


分布式系统的两个原则与多阶段提交

ACID 原则：是关系型数据库事务要遵守的原则

原子性：要求事务的所有操作都必须做完，如果有一个没有做完，则视为所有的操作都失败

共识算法：



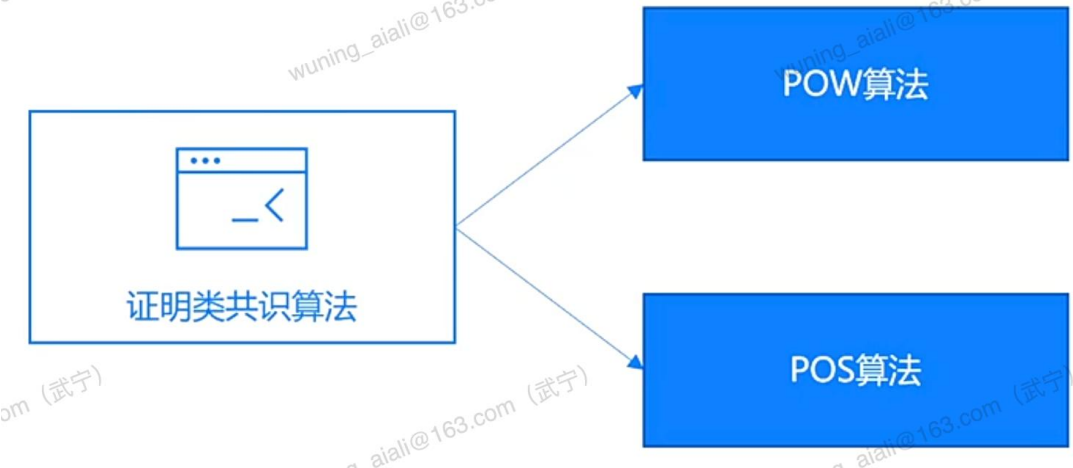
共识性：不同节点完成决策的结果应该是相同的

可终止性：一致性结果在有限的时间内可以完成

合法性：　学习的结果必须是所有进程中提出的议案

共识算法：本质都是为了实现一致性目标。

共识算法举例：PoW、PoS、DPos、Raft



PoW：按劳分配原则，将算力作为记账。基于算力实现记账权。

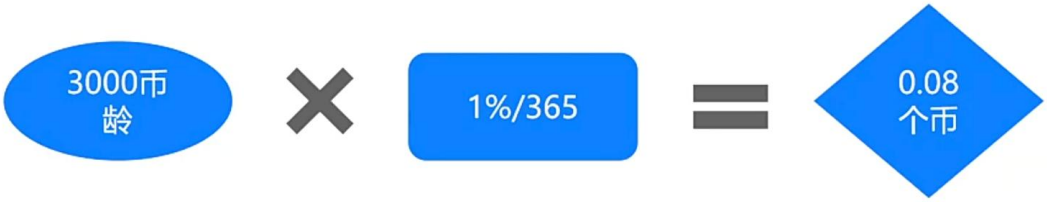
- 首先，客户端产生新的交易，向全网广播
- 第二，每个节点收到请求，将交易暂存于存储池中
- 第三，每个节点进行pow工作量证明
- 第四，当某个节点找到了证明，向全网广播
- 第五，当且仅当该区块的交易是有效的且在之前中未存在的，其他节点才认同该区块的有效性
- 第六，接受该区块且在该区块的末尾制造新的区块

PoS：大户拥有记账权，持有数字货币数量越多，越可能获得记账权。基于权益实现记账权。

币龄，即持有货币的时间。这是POS机制难度值确定的核心

币龄计算举例：

利息，即在发现区块之后会根据一定的利率给付数字货币。



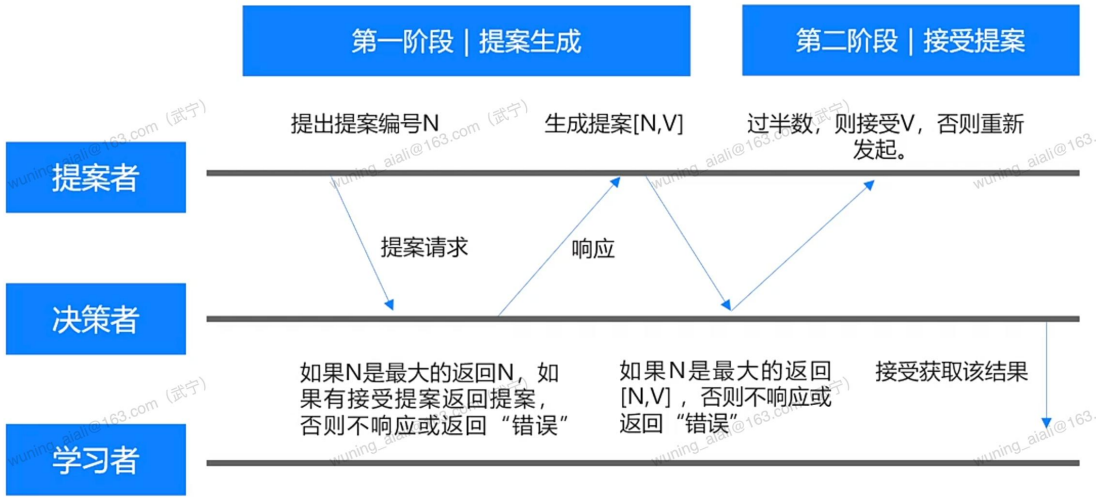
PoW vs PoS:

比较项	POW	POS
记账权获得	以算力竞争记账权利	以权益竞争记账权利
篡改难度	低	高
头部玩家	未能解决	未能解决

Paxos 算法：一种选举类算法，采用少数服从多数原则选取结果。是解决网络故障中达成一致性的先驱算法。

- **Paxos节点被划分为不同角色：**

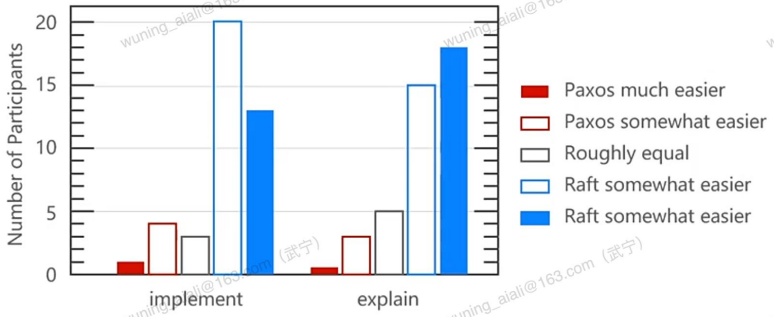
Paxos共识算法过程包括提案生成过程和接受提案过程。



Raft 算法机制：

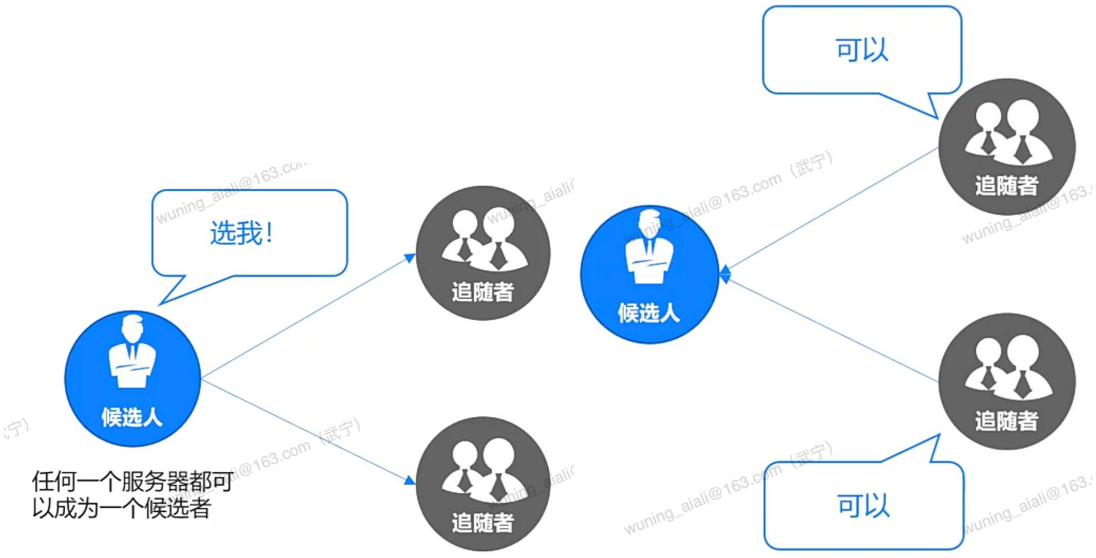
Paxos和Raft都是为了实现一致性这个目标。

后来斯坦福大学的两位博士基于Paxos思想在《In Search of an Understandable Consensus Algorithm》一文提出了Raft算法。



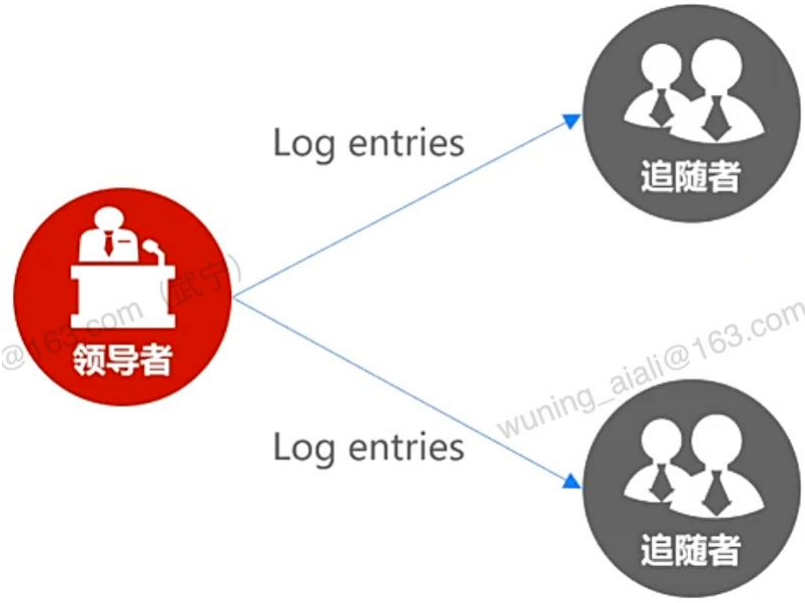
Raft同样包括三类角色。

Leader：用于实现日志复制的工作，是主从结构重的中心节点，用于复制和更新数据。



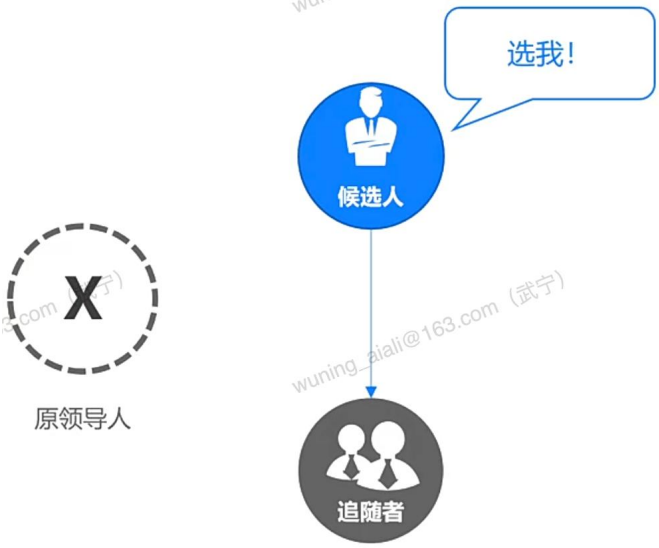
step1: 每个节点都是一个倒计时器，倒计时结束后，该节点切换为候选人状态。它想其他节点发出要求选举自己的请求。

step2: 其他服务器同意了，发出 ok 请求，只要达到 N/2+1 的大多数选票，候选人即可成为 Leader。



step3: 候选人成为领导人，领导人可以像选民发出指令。

step4: 心跳机制，通过“心跳”进行日志复制的通知。



step5: 如果 Leader 宕机，那么 Follower 中有一个成为候选者，发出邀请选举。

选举类共识算法：

■ **Paxos共识算法**

- 将节点划分为提议者 、决策者 、最终决策学习者
- 通过两个阶段实现共识，首先是提案生成阶段，随后是提案决策段

■ **Raft共识算法**

- 节点具有领导者、追随者、候选人三种状态，初始阶段皆为追随者状态
- 通过两个阶段实现共识，首先是领导者选举阶段，随后是数据复制阶段

PBFT 共识算法：拜占庭问题。是一种基于联盟链的共识算法，具有 3f+1 的容错性，可以容忍小于 1/3 个无效或者恶意节点，并同时保证一定的性能。少数服从多数原则。

- 节点可以通过轮流等方式确定
- 达成共识的过程包含预准备阶段、准备阶段、承诺阶段

	PBFT	POW机制
优点	• 允许33%的容错； • 可以快速结算和快速担保交易。	• 不需要知道参与网络的所有节点； • 任何节点都可以在任何时间点离开或加入；可以扩展到分布在全世界的大量节点和参与者。
缺点	• 无法扩展到1000个节点以上	• 处理速度非常慢； • 吞吐量非常有限； • 消耗了大量的能量。

蜜罐共识算法：基于 PBFT 的优化算法。 是一种异步共识算法，删除了需要响应时间的要求。包括整体算法分为：步骤随机选择交易打包、加密生成、交易广播、解密交易，生

成区块。

PBFT 对于响应时间有限制，如果未在规定时间内返回响应，则认为节点失活。

蜜罐算法则对响应时间无时间限制。只要能够收到信息就可以。

Avalanche 共识算法：

将传统分布式一致算法与经典区块链共识机制的设计思路结合。 通过朋友圈到朋友圈的扩散，达到整个网络形成共识的目的。

Algorand 共识算法：

利用作恶者找不到投票人，使投票人放心地快速的达成共识，向全网传播经过安全认证的区块。