3.3 共识协议

01 Paxos的提出

无论二阶段提交还是后面提出三阶段提交（三阶段提交是为解决两阶段提交协议的缺点而设计的）都无法很好的解决分布式的一致性问题。

直到Paxos算法的提出，**Paxos协议由Leslie Lamport最早在1990年提出，目前已经成为应用最广的分布式一致性算法。**

02 节点角色

Paxos 协议中，有三类节点:

1.Proposer:提案者

Proposer 可以有多个，Proposer 提出议案(value)。所谓 **value，在工程中可以是任何操作**，例如“修改某个变量的值为某个值”、“设置当前 primary 为某个节点”等等。Paxos 协议中统一将这些操作抽象为 value。**不同的 Proposer 可以提出不同的甚至矛盾的 value**，例如某个 Proposer 提议“将变量 X 设置为 1”，另一个 Proposer 提议“将变量 X 设置为 2”，**但对同一轮 Paxos 过程，最多只有一个 value 被批准。**

03 节点角色

Paxos 协议中，有三类节点:

2. Acceptor:批准者

- Acceptor 有 N 个，Proposer 提出的 value 必须获得**超过半数(N/2+1)的Acceptor 批准**后才能通过。
- Acceptor 之间完全对等独立。

04 节点角色

Paxos 协议中，有三类节点:

3.Learner: 学习者

Learner 学习被批准的 value。所谓学习就是通过读取各个 Proposer 对 value 的选择结果，如果某个 value 被超过半数 Proposer 通过，则 Learner 学习到了这个 value。这里类似 Quorum 议会机制，某个 value 需要获得 W=N/2 + 1 的 Acceptor 批准，Learner 需要至少读取 N/2+1 个 Accpetor，至多读取 N 个 Acceptor 的结果后，能学习到一个通过的 value。

Paxos中 **proposer 和 acceptor 是算法的核心角色**，paxos 描述的就是在一个由多个 proposer 和多个 acceptor 构成的系统中，**如何让多个 acceptor 针对 proposer 提出的多种提案达成一致的过程**，而 learner 只是“学习”最终被批准的提案。

05 约束条件

Paxos协议流程还需要满足如下约束条件：

1、Acceptor必须接受它收到的第一个提案；

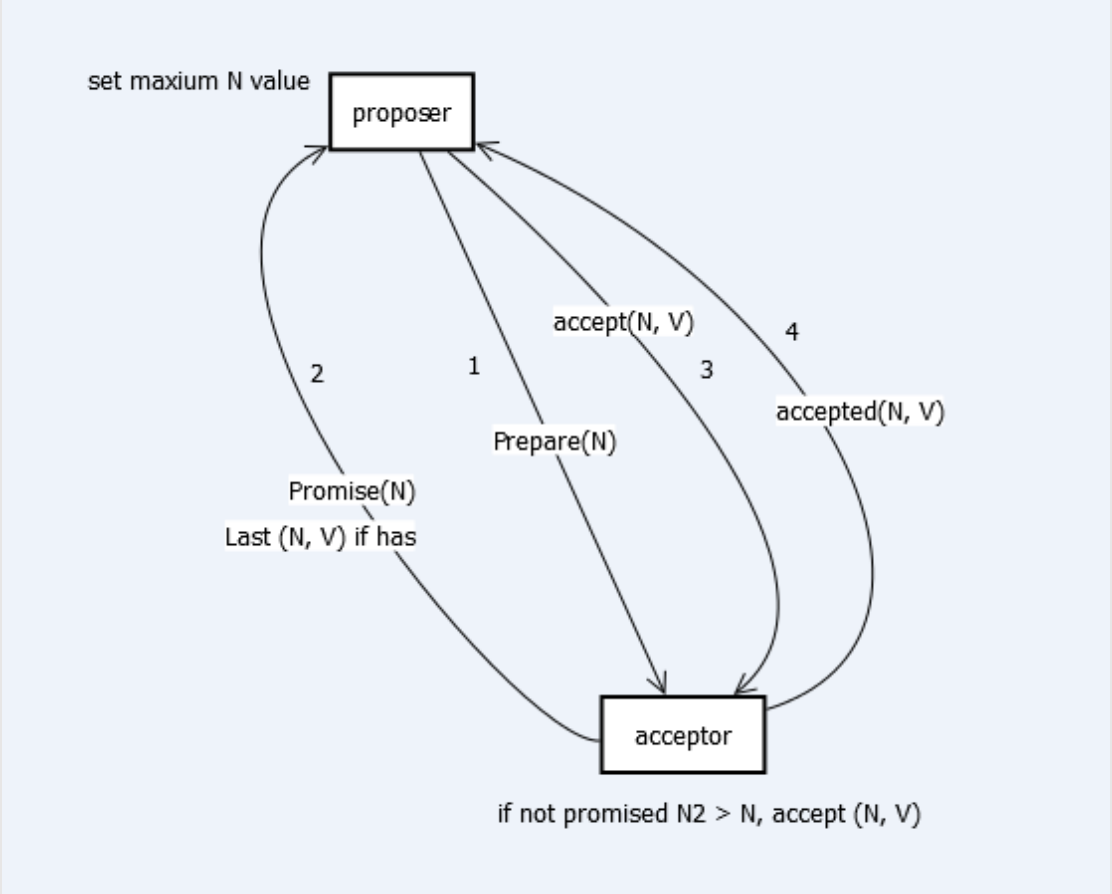
2、如果一个提案的v值被大多数Acceptor接受过，那后续的所有被接受的提案中也必须包含v值（v值可以理解为提案的内容，提案由一个或多个v和提案编号组成）；

3、如果某一轮 Paxos 协议批准了某个 value，则以后各轮 Paxos 只能批准这个value；

每轮 Paxos 协议分为准备阶段和批准阶段，在这两个阶段 Proposer 和 Acceptor 有各自的处理流程。

06 约束条件

Proposer与Acceptor之间的交互主要有4类消息通信，如下图：



07 约束条件

这4类消息对应于paxos算法的两个阶段4个过程：

Phase 1准备阶段

a) proposer向网络内超过半数的acceptor发送prepare消息

b) acceptor正常情况下回复promise消息

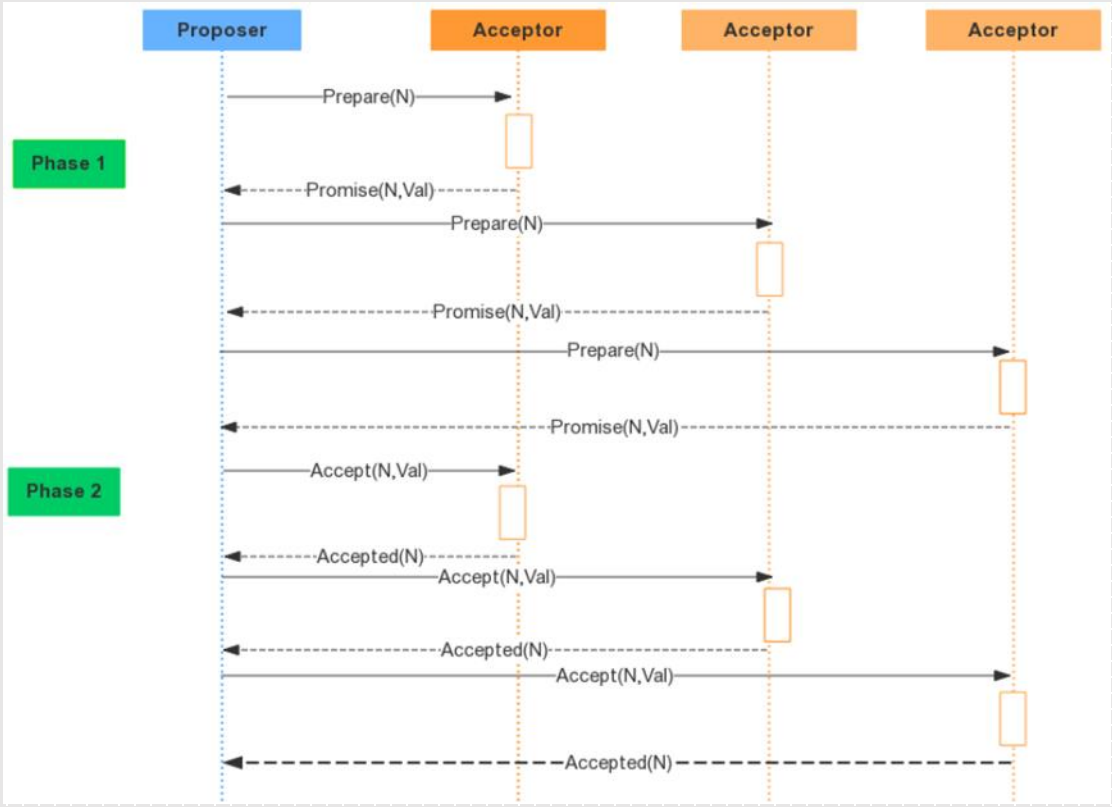
Phase 2批准阶段

a) 在有足够多acceptor回复promise消息时，proposer发送accept消息

b) 正常情况下acceptor回复accepted消息

08 选举过程

选举分为两个阶段，如下图所示。



09 选举过程

Phase1 准备阶段

P1a：Proposer 发送 Prepare请求

Proposer 生成全局唯一且递增的ProposalID，向 Paxos 集群的所有机器发送 Prepare请求，这里不携带value，只携带 ProposalID 。

P1b：Acceptor 应答 Prepare

Acceptor 收到 Prepare请求后，判断：收到的ProposalID 是否比之前已响应的所有提案的ProposalID大，**如果是，则**：

(1) 在本地持久化 ProposalID，可记为Max_ProposalID。

(2) 回复请求，并带上 已Accept的提案中 ProposalID 最大的 value（若此时还没有已Accept的提案，则返回value为空）。

(3) 做出承诺：不会Accept 任何 小于 Max_ProposalID的提案。

如果否：不回复或者回复Error

10 选举过程

Phase2 选举阶段

P2a：Proposer 发送 Accept

经过一段时间后，Proposer 收集到一些 Prepare 回复，有下列几种情况：

(1) 回复数量 > 一半的Acceptor数量，且所有的回复的value都为空，则Porposer发出accept请求，并带上自己指定的value。

(2) 回复数量 > 一半的Acceptor数量，且有的回复value不为空，则Porposer发出accept请求，并带上回复中ProposalID最大的value(作为自己的提案内容)。

(3) 回复数量 <= 一半的Acceptor数量，则尝试更新生成更大的ProposalID，再转P1a执行。

11 选举过程

Phase2 选举阶段

P2b：Acceptor应答accept

Accpetor 收到 Accpet请求 后，判断：

(1) 收到的N >= Max_N (一般情况下是 等于)，则回复提交成功，并持久化N和value。

(2) 收到的N < Max_N，则不回复或者回复提交失败。

12 选举过程

Phase2 选举阶段

P2c：Proposer 统计投票

经过一段时间后，Proposer 收集到一些 Accept 回复提交成功，有几种情况：

(1) 回复数量 > 一半的Acceptor数量，则表示提交value成功。此时，可以发一个广播给所有Proposer、Learner，通知它们已commit的value。

(2) 回复数量 <= 一半的Acceptor数量，则尝试更新生成更大的 ProposalID，再转P1a执行。

(3) 收到一条提交失败的回复，则尝试更新生成更大的 ProposalID，再转P1a执行。

13 Paxos算法的核心思想

(1) 引入了 多个Acceptor，避免单个Acceptor成为单点。

Proposer用更大ProposalID来抢占临时的访问权，避免其中一个Proposer崩溃宕机导致死锁。

(2) 保证一个ProposalID，只有一个Proposer能进行到第二阶段运行，Proposer按照ProposalID递增的顺序依次运行。

(3) 新ProposalID 的proposer 采用后者认同前者的思路运行。

在肯定旧ProposalID 还没有生成确定的value (Acceptor提交成功一个value)时，新ProposalID 会提交自己的value，不会冲突。

一旦旧ProposalID 生成了确定的value，新ProposalID 肯定可以获取到此值，并且认同此值。