分布式一致性Paxos算法

在分布式系统中，总会发生诸如机器宕机或网络异常（包括消息延迟、丢失、重复、乱序及网络分区）等情况。Paxos算法解决上述分布式系统中异常发生时，快速且正确地在集群内部对某个数据的值达成一致，这些数据的值并不是狭义的某个数，可以是日志、甚至是命令等；这些算法保证不论发生以上任何异常，都不会破坏整个系统的一致性，从而解决分布式系统的可用性问题。

# 1.算法概述

Paxos算法是Leslie Lamport在1990年提出的一种基于消息传递的一致性算法，实现分布式系统中就某个值达成一致。可以通过Paxos算法实现多副本一致性、分布式锁、名字管理及序列化分配等。在一个分布式数据库系统中，如果各个节点的初始状态一致，每个节点执行相同的操作序列，那么他们最后能得到一个一致的状态。为保证每个节点执行相同的命令序列，需要在每一条指令上执行“一致性算法”以保证每个节点看到的指令一致。

对于Paxos算法，有以下概念：

* Proposer，提议发起者，处理客户端请求，将客户端请求发送到集群中，以便于决定这个值是否可以被批准
* Acceptor，提议批准者，负责处理接收到的提议，他们的回复就是一次投票，会存储一些状态来决定是否接收一个值
* Replica，节点或者副本，分布式系统中的一个Server，一台物理机或者虚拟机，同时承担paxos中的提议者和接受者角色
* ProposalId，每个提议都有一个编号，编号高的提议优先级高
* Paxos Instance，Poxos中用来在多个节点之间对同一个值达成一致的过程，比如同一个日志序列号
* LogIndex，不同的LogIndex属于不同的Paxos Instance
* AcceptedProposal，在一个Paxos Instance内，已经接受过的提议
* AcceptedValue，在一个Paxos内，已经接受过的提议对应值
* minProposal，在一个Paxos Instance内，当前接收的最小提议值，会不断更新

# 2.算法背景分析

在计算机通信理论中，有一个著名的两军问题(two-army problem)，讲述通信的双方通过ACK来达成共识，永远会有一个在途的ACK需要进行确认，因此无法达成共识。两军问题和Basic Paxos非常相似

1） 通信的各方需要达成共识；

2） 通信的各方仅需要达成一个共识；

3） 假设的前提是信道不稳定，有丢包、延迟或者重放，但消息不会被篡改。

为了配合Basic Paxos的多数派概念，把两军改为3军；同时假设了将军和参谋的角色。假设的3军问题：

1） 1支红军在山谷里扎营，在周围的山坡上驻扎着3支蓝军；

2）红军比任意1支蓝军都要强大；如果1支蓝军单独作战，红军胜；如果2支或以上蓝军同时进攻，蓝军胜；

3）三支蓝军需要同步他们的进攻时间；但他们惟一的通信媒介是派通信兵步行进入山谷，在那里他们可能被俘虏，从而将信息丢失；或者为了避免被俘虏，可能在山谷停留很长时间；

4）每支军队有1个参谋负责提议进攻时间；每支军队也有1个将军批准参谋提出的进攻时间；很明显，1个参谋提出的进攻时间需要获得至少2个将军的批准才有意义；

5）问题：是否存在一个协议，能够使得蓝军同步他们的进攻时间？

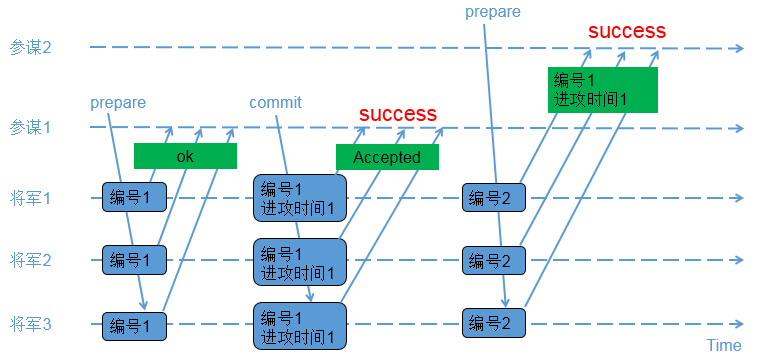
接下来以两个假设的场景来演绎BasicPaxos；参谋和将军需要遵循一些基本的规则

1）参谋以两阶段提交（prepare/commit）的方式来发起提议，在prepare阶段需要给出一个编号；

2）在prepare阶段产生冲突，将军以编号大小来裁决，编号大的参谋胜出；

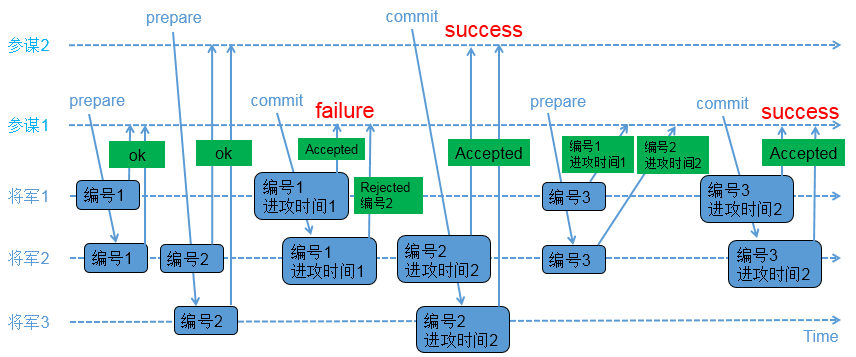
3）参谋在prepare阶段如果收到了将军返回的已接受进攻时间，在commit阶段必须使用这个返回的进攻时间；

## 2.1 两个参谋先后提议



1. 参谋1发起提议，派通讯兵带信给3个将军，内容为（编号1）
2. 3个将军收到参谋1的提议，由于之前还没有保存任何编号，因此把（编号1）保存下来，避免遗忘；同时让通讯兵带信回去，内容为（OK）
3. 参数1收到至少2个将军的回复，再次派通讯兵带信给3个将军，内容为（编号1，进攻时间1）
4. 3个将军收到参谋1的时间，把（编号1，进攻时间1）保存下来，避免遗忘；同时让通信兵带信回去，内容为Accepted
5. 参数1收到至少2个将军的（Accepted）内容，确认进攻时间已被大家接收；
6. 参谋2发起提议，派通信兵带信给3个将军，内容为（编号2）；
7. 3个将军收到参谋2的提议，由于（编号2）比（编号1）大，因此把（编号2）保存下来，避免遗忘；又由于之前已经接受参谋1的提议，因此让通信兵带信回去，内容为（编号1，进攻时间1）；
8. 参谋2收到至少2个将军的回复，由于回复中带来了已接受的参谋1的提议内容，参谋2因此不再提出新的进攻时间，接受参谋1提出的时间；

## 2.2 两个参谋交叉提议



1. 参谋1发起提议，派通讯兵带信给3个将军，内容为（编号1）
2. 3个将军的情况如下
3. 将军1和将军2收到参谋1的提议，将编号1记录下来，如果有其他参数提出更小的编号，将被拒绝；同时让通信兵带信回去，内容为（OK）
4. 负责通知将军3的通讯兵被抓，因此将军3没收到参谋1的提议
5. 参谋2在同一时间也发起了提议，派通信兵带信给3个将军，内容为（编号2）
6. 3个将军的情况如下

a) 将军2和将军3收到参谋2的提议，将军2和将军3把（编号2）记录下来，如果有其他参谋提出更小的编号，将被拒绝；同时让通信兵带信回去，内容为（ok）；

b) 负责通知将军1的通信兵被抓，因此将军1没收到参谋2的提议；

1. 参谋1收到至少2个将军的回复，再次派通信兵带信给有答复的2个将军，内容为（编号1，进攻时间1）；
2. 2个将军的情况如下

a) 将军1收到了（编号1，进攻时间1），和自己保存的编号相同，因此把（编号1，进攻时间1）保存下来；同时让通信兵带信回去，内容为（Accepted）；

b) 将军2收到了（编号1，进攻时间1），由于（编号1）小于已经保存的（编号2），因此让通信兵带信回去，内容为（Rejected，编号2）；

1. 参谋2收到至少2个将军的回复，再次派通信兵带信给有答复的2个将军，内容为（编号2，进攻时间2）；
2. 将军2和将军3收到了（编号2，进攻时间2），和自己保存的编号相同，因此把（编号2，进攻时间2）保存下来，同时让通信兵带信回去，内容为（Accepted）；
3. 参谋2收到至少2个将军的（Accepted）内容，确认进攻时间已经被多数派接受；

10）参谋1只收到了1个将军的（Accepted）内容，同时收到一个（Rejected，编号2）；参谋1重新发起提议，派通信兵带信给3个将军，内容为（编号3）；

11）3个将军的情况如下

a) 将军1收到参谋1的提议，由于（编号3）大于之前保存的（编号1），因此把（编号3）保存下来；由于将军1已经接受参谋1前一次的提议，因此让通信兵带信回去，内容为（编号1，进攻时间1）；

b) 将军2收到参谋1的提议，（编号3）大于之前保存的（编号2），因此把（编号3）保存下来；由于将军2已经接受参谋2的提议，因此让通信兵带信回去，内容为（编号2，进攻时间2）；

c) 负责通知将军3的通信兵被抓，因此将军3没收到参谋1的提议；

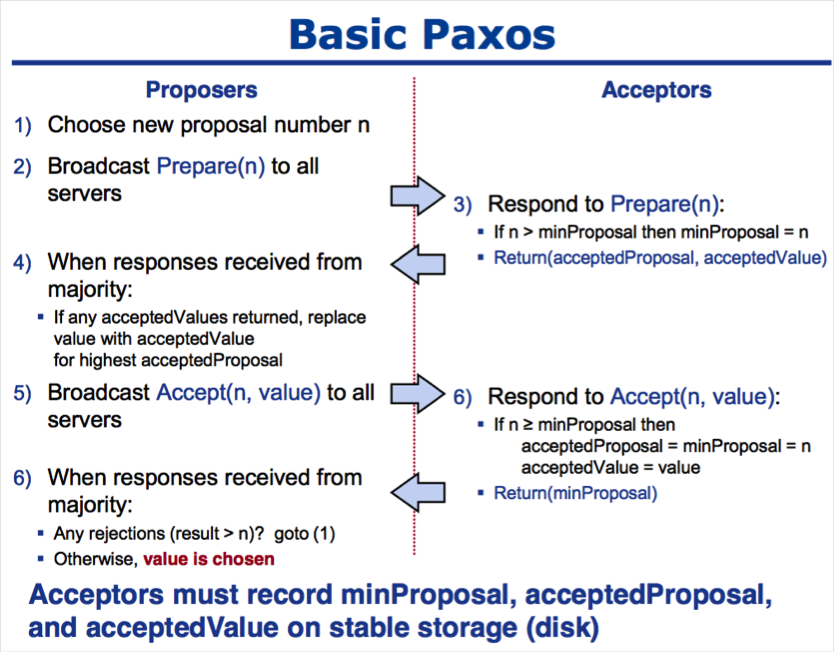
12） 参谋1收到了至少2个将军的回复，比较两个回复的编号大小，选择大编号对应的进攻时间作为最新的提议；参谋1再次派通信兵带信给有答复的2个将军，内容为（编号3，进攻时间2）；

13） 将军1和将军2收到了（编号3，进攻时间2），和自己保存的编号相同，因此保存（编号3，进攻时间2），同时让通信兵带信回去，内容为（Accepted）；

14） 参谋1收到了至少2个将军的（accepted）内容，确认进攻时间已经被多数派接受；

# 3. Paxos算法抽象分析

基于Paxos协议构建的系统，只需要系统中超过半数节点在线且相互通信正常即可正常对外提供服务，其核心实现Paxos Instance主要包括两个阶段：准备阶段（prepare phase）和提议阶段（Accept Phase），如下图所示：



1. 获取一个ProposalId，可以采用时间戳+serverID方式生成来保证ProposalId递增
2. 提议者向所有节点广播prepare(n)请求
3. 接受者比较n和minProposal，如果n>minProposal，表示有更新的提议，minProposal=n；否则将(acceptedProposal,acceptedValue)返回
4. 提议者接收过半数请求后，如果发现有acceptedValue返回，表示有更新的提议，保存acceptedValue到本地，然后跳转1，生成一个更高的提议
5. 到这里表示在当前paxos instance内，没有优先级更高的提议，可以进入第二阶段，广播accept(n,value)
6. 接收者比较n和minProposal，如果n>=minProposal，则acceptedProposal=minProposal=n，acceptedValue=value，本地持久化后返回；否则返回minProposal
7. 提议者接收到半数请求后，如果发现有返回这>n，表示有更新的提议，跳转1;否则value一致

从上述流程上看，并发情况下，可能出现第4步和第7步频繁重试的情况下，导致性能低下，更严重可能导致永久无法达成一致的情况，就是所谓的活锁。

http://iunknown.iteye.com/blog/2246484?from=message&isappinstalled=0

https://www.cnblogs.com/cchust/p/5617989.html

https://yq.aliyun.com/articles/156281