共识 | 拜占庭容错的代表 PBFT

高亦民 溪塔科技 2019-12-13

本文是共识系列文章的第一篇。在共识系列文章中，我将会向大家介绍常见的共识算法。在文章开头，我会用一定篇幅介绍一致性问题的基础知识。一致性问题是分布式系统中最基础也是最重要的问题，而共识算法就是用来解决分布式系统一致性的。之后，我会介绍一个非常经典的拜占庭容错算法 PBFT。

如果您对技术感兴趣，欢迎关注我们开源代码库（点个 Star 呗）：
GitHub：https://github.com/citahub

预备

一致性

一致性（Consistency），早期也叫（Agreement），是指在分布式系统领域中，对于多个服务节点，给定一系列操作，在约定协议的保障下，使得它们对处理结果达成“某种程度”的协同。分布式系统达成一致的过程应满足：

- 可终止性（Termination）：一致的结果在有限时间内能完成；
- 约同性（Agreement）：不同节点最终完成决策的结果是相同的；

- 合法性（Validity）：决策的结果必须是某个节点提出的提案。

共识

要保障系统不同程度的一致性，就需要共识算法来完成。共识算法解决的是分布式系统对某个提案（Proposal），所有诚实节点达成一致意见的过程。那么共识需要解决的问题可以做如下的抽象：

- 如何提出一个待共识的提案?
- 如何让多个节点对该提案达成共识?

CFT & BFT

对于分布式系统来说，如果节点间通信十分顺畅，各个节点都能瞬间响应，那么只需要简单广播投票和应答就可以解决一致性问题。然后现实并不是这样。节点往往会遇到网络中断、节点故障，甚至是被非法入侵伪造消息的问题。节点遇到的问题可以进行如下的分类：

- 节点出现故障但不会伪造信息的情况称为“非拜占庭错误”；
- 节点会伪造信息恶意响应的情况称为“拜占庭错误”，伪造信息的节点称为拜占庭节点。

相对应的，共识算法也可以分为 CFT 和 BFT 两类：

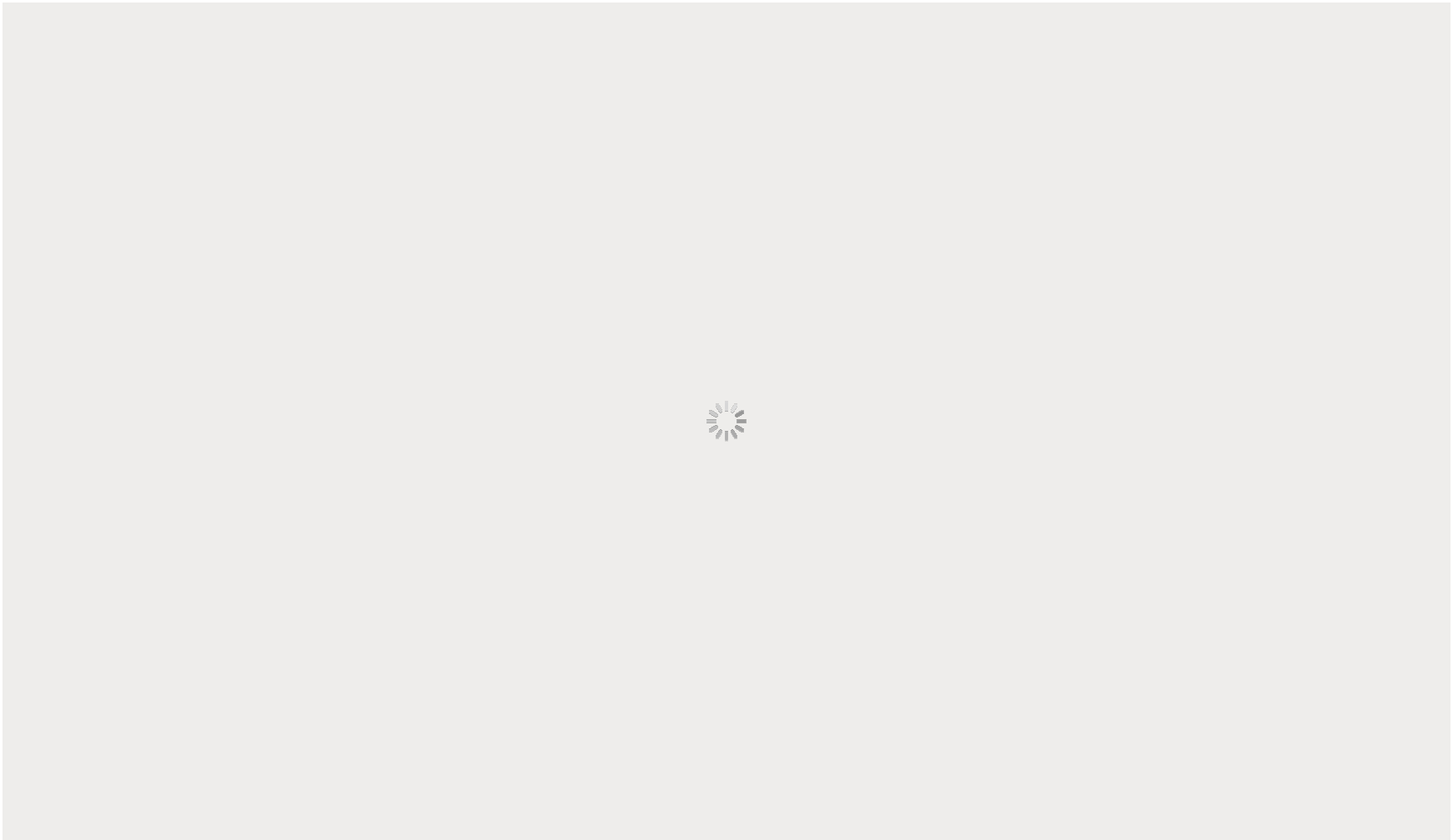
- CFT（Crash Fault Tolerance）：只容忍节点故障，不容忍节点作恶；
- BFT（Byzantine Fault Tolerance）：容忍节点故障与作恶。

FLP 不可能原理

计算机科学家证明了：在网络可靠，但允许节点失效的最小化异步系统中，不存在一个可以解决一致性问题的确定性共识算法。这似乎意味着去设计一个共识算法是徒劳的，然而科学告诉你什么是不可能的；工程则告诉你，付出一些代价，可以把它变成可行。也就是说在付出多大的代价的情况下，能够达到共识。

两军问题

白军驻扎在沟渠里，蓝军和红军分别驻扎在沟渠两边。白军比蓝军和红军中任何一支军队都更为强大，但是蓝军和红军若能同时合力进攻则能够打败白军。蓝军和红军不能够越过沟渠远程地沟通，只能派遣通信兵穿过沟渠去通知对方协商进攻时间。但是通信兵可能会迷路或者被敌军截获，消息被篡改。



根据 已有证明这个问题无通用解，然而这一问题在通信领域又必须解决。基于成本可控的考虑，现在使用 TCP 协议的三次握手来（不彻底的）解决这一问题。

拜占庭将军问题

一组拜占庭将军分别各率领一支军队共同围困一座城市。为了简化问题，将各支军队的行动策略限定为进攻或撤离两种。因为部分军队进攻部分军队撤离可能会造成灾难性后果，因此各位将军必须通过投票来达成一致策略，即所有军队一起进攻或所有军队一起撤离。因为各位将军分处城市不同方向，他们只能通过信使互相联系。在投票过程中每位将军都将自己投票给进攻还是撤退的信息通过信使分别通知其他所有将军，这样一来每位将军根据自己的投票和其他所有将军送来的信息就可以知道共同的投票结果而决定行动策略。

上述的故事映射到分布式系统里，将军便成了共识节点，叛徒就是拜占庭节点。与两军问题不同的是，拜占庭将军问题中并不去考虑通信兵是否会被截获或无法传达信息等问题，也就是说在假定信道没有问题的情况下去讨论一致性与容错性。

有了这些预备知识可以更好的理解共识算法。下面我们进入今天的正题——PBFT。

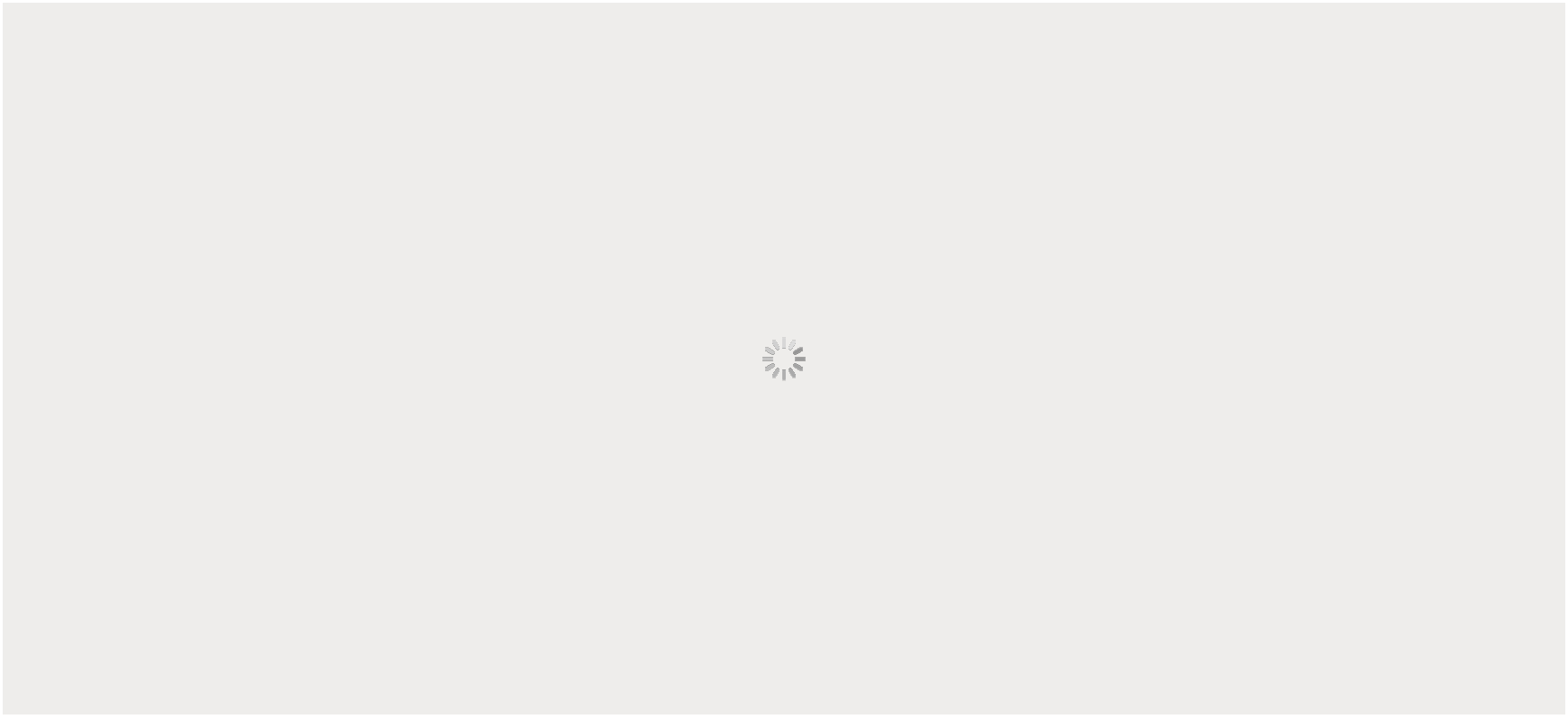
PBFT 共识算法

PBFT 是 Practical Byzantine Fault Tolerance 的缩写，意为实用拜占庭容错算法。该算法首次将拜占庭容错算法复杂度从指数级降低到了多项式级，其可以在恶意节点不高于总数 1/3 的情况下同时保证安全性（Safety）和活性（Liveness）。我们假设所有节点的总数为 R ，拜

占庭节点数量为 f，下面给出安全性证明：

设想 f 个叛变者和 k 个忠诚者，叛变者故意使坏，可以给出错误的结果，也可以不响应。某个时候 F 个叛变者都不响应，则 k 个忠诚者取多数既能得到正确结果。当 f 个叛变者都给出一个恶意的提案，并且 k 个忠诚者中有 f 个离线时，剩下的 k - f 个忠诚者此时无法分别是否混入了叛变者，仍然要确保取多数能得到正确结果，因此，k - f > f，即 k > 2f 或 R - f > 2f，所以系统整体规模 R 要大于 3f。所以为了能确保达成共识的拜占庭系统节点数至少为 4，此时最多允许出现 1 个拜占庭节点。

PBFT 是一种基于状态机副本复制的算法，每个状态机的副本都保存了服务的状态，同时也实现了服务的操作。PBFT 中所有的副本都在视图（View）的轮换过程中运作，当主节点掉线的时候就启动视图更换过程保证算法的持续运行。



从上面的流程图可以看出，PBFT 算法的流程如下：

1. 客户端向主节点发送请求；
2. 主节点向其他副本广播请求；
3. 所有副本执行请求后，将结果返回给客户端；
4. 客户端需要等待 2f+1 个不同副本返回相同的结果，作为最终结果。

这里面暗含着的是所有节点都是确定性的和所有节点都从相同的状态开始执行这两个条件。

首先客户端发送了一个请求到主节点，之后经典的三阶段协议（three-phase protocol）就拉开了序幕。

预准备阶段

首先，主节点向所有副本节点发送预准备消息。这里面包含有消息序号，视图编号和消息的摘要。需要注意的是预准备消息是不包含请求的，这样做有两个好处，一是压缩消息大小提升传播效率，二是将请求排序与请求传输解耦。

接着副本节点会去验证消息的签名是否正确，视图编号是否一致和消息序号是否满足水线要求。这里就要引出 PBFT 中的水线机制（watermark）。对于消息的序号 n，要求其在水线 h 和 H 之间。水线存在的意义在于防止一个失效节点使用一个很大的序号消耗序号空间。

准备阶段

如果副本节点接受预准备消息，就进入了准备阶段。在准备阶段，每一个节点都向其他节点发送包含自己 ID 的准备消息，同时也接收其他节点的准备消息。对于收到准备消息同样进行合法性检查。验证通过则把这个准备消息写入自己的消息日志中。一个节点集齐至少 2f+1 个验证过的消息才进入准备状态。

提交阶段

在提交阶段，每个节点广播 commit 消息告诉其他节点自己已经进入准备状态。如果集齐至少 2f+1 的 commit 消息则说明提案通过。

在经过了三阶段协议之后，每个副本节点都想客户端发送回复，副本节点会把时间戳比已回复时间戳更小的请求丢弃，以保证请求只会被执行一次。

总结

PBFT 共识算法在 1999 年的时候由 Castro 和 Liskov 正式提出，它在设计时考虑的共识对象是一些相对不大的消息。为了对消息进行排序，PBFT 设计了水线机制，通过 checkpoint 机制移动水线，用以并发地处理多个消息的投票过程。同时， PBFT 只有当某个节点作恶或掉线才触发视图的切换，主节点的更换。这是因为视图的切换过程也是需要共识的，这一过程非常

耗时，因此 PBFT 不能接受频繁的视图变更。再加上为了配合水位机制，视图切换的消息都相对普通消息要大得多。因为以上原因，PBFT 的设计非常复杂，效率不高。

然而，随着技术的发展，区块链技术的诞生轻松的化解掉了 PBFT 在设计上的一些问题。在区块链中，每一个消息（区块）前后相继，用于并发处理的水位机制毫无用处，因此水线机制以及为此服务的 checkpoint 机制就没有存在的意义了。没有了水线机制和 checkpoint，阻碍视图切换的就只剩下视图切换的共识过程了，而这一点又被区块链本身作为共识账本的特点给简化掉了。如果节点的切换通过链上的数据来达成共识，那么原本需要经过在线共识的过程又省掉了。

因此大量的区块链项目都使用了改进的 PBFT 用作共识算法，作为拜占庭容错的代表的 PBFT 也在不断地优化的过程中焕发出了新的生机。

ref：

1.区块链技术指南 ： https://legacy.gitbook.com/book/yeasy/blockchain_guide/details

2.Castro M, Liskov B. Practical Byzantine fault tolerance[C]//OSDI. 1999, 99: 173-186.

推荐阅读

- *拜占庭容错的代表 PBFT*
- *BLS 签名和基于 BLS 签名的门限签名*
- *密码学入门系列（一）*
- *CITA架构与云计算服务*
- *白话零知识证明（一）*
- *白话零知识证明（二）*
- *Authenticated Dynamic Dictionaries in Ethereum*
- *区块链与共同知识库*
- *P2P 网络思考*
- *智能合约的一种设计结构*

如果您对技术感兴趣，欢迎关注我们开源代码库（点个 Star 呗）：
GitHub：https://github.com/citahub

阅读原文