

硕 士 研 究 生 读 书 报 告



题目 分布式共识算法——Paxos

作者姓名 吴鑫康

作者学号 22051407

指导教师 李启雷

学科专业 软件工程

所在学院 软件学院

提交日期 2020 年 1 月

Distributed Consensus Algorithm——Paxos

A Dissertation Submitted to

Zhejiang University

in partial fulfillment of the requirements for

the degree of

Master of Engineering

Major Subject: Software Engineering

Advisor: Li Qilei

By

Wu Xinkang

Zhejiang University, P.R. China

2020

摘要

本文主要探讨分布式系统中的共识算法Paxos。从分布式系统中存在的问题，引出共识这一基本问题，接着论述Paxos算法要解决的问题，以及是如何解决分布式系统中的共识问题的。本文从Paxos算法中三种角色的职责，共识所需要满足的约束条件，以及共识的两个阶段来论述系统是如何选择一个值的，并论述了学习者如何学习被选择的值，以及如何保证算法的进展性。

**关键词**：分布式系统，共识算法，Paxos

Abstract

The paper discusses the consensus algorithm Paxos in distributed systems. This paper derives the basic problem of consensus from the problems in distributed systems, and then discusses the problems to be solved by the Paxos algorithm and how to solve the consensus problem in distributed systems by it. This paper discusses how the system choose a value by discussing the responsibilities of the three roles in the Paxos algorithm, the condition that the consensus needs, and the two phases of the consensus, and discusses how the learner learns the chosen value and how to ensure the progress of the algorithm.

**Keywords：**Distributed System, Consensus Algorithm, Paxos

1引言

在基于消息传递的分布式系统中，可能会出现各种各样的问题：进程可能会因为各种原因变慢、终止、重启，消息也可能会延迟、丢失、重复等。Paxos 算法可以使得分布式系统在发生上述问题时仍然可以对某个提议达成一致，保证不论发生以上任何问题，都不会破坏决议的共识。

Paxos是一种用于在通过异步网络进行通信的一组分布式计算机之间达成共识的算法。一个或多个客户端向Paxos提出了一个提议，当大多数运行Paxos的系统都同意其中的一个提议时，系统达成共识。Paxos被广泛使用，并且在计算机科学领域具有传奇色彩，因为它是第一个被严格证明为正确的共识算法。

**2共识**

共识是可容错分布式系统中的一个基本问题。共识涉及多个服务器就某个值达成一致。一旦他们对该值做出决定，该决定就是最终决定。当大多数服务器可用时，典型的共识算法才能取得进展。例如，在包含5台服务器的集群中即使2台出现故障，也可以继续运行。如果更多服务器发生故障，它们将停止取得进展，但不会返回错误的结果。

共识通常出现在复制状态机的背景下，复制状态机是构建可容错系统的通用方法。每个服务器都有一个状态机和一个日志。状态机是我们要容错的组件，例如哈希表。对于客户端来说，即使集群中的少数服务器出现故障，它们也会与一个可靠的状态机进行交互。每个状态机都从其日志中获取输入命令。日志将包含将x设置为某个值之类的命令。使用共识算法来同意服务器日志中的命令。共识算法必须确保，如果有任何状态机将x设置为某个值作为第n条命令，则其他任何状态机都不会应用不同的第n条命令。因此，每个状态机能处理一系列相同的命令，产生一系列相同的结果并到达一系列相同的状态。

**3 Paxos算法**

**3.1 解决的问题**

假设有一个可以提出提议的进程集合。共识算法可确保在提议的值中选择一个。如果未提出任何值，则不应选择任何值。如果选择了一个值，则这些进程应该能够学习被选择的值。达成共识的安全性要求是：

1. 只有被提议的值才能被选择；
2. 只有一个值被选择；
3. 除非真正选择了某个值，否则一个进程永远不会得知它已被选择。

目标是确保某些提议的值能被最终选择，并且如果选择了某个值，进程最终可以学习该值。

共识算法中的三种角色由三种执行者执行：提议者，接受者和学习者。单个进程可以充当多个执行者，但这里无需关心从执行者到进程映射的实现。

假设执行者可以通过发送消息相互通信。我们使用惯用的非拜占庭式异步模型，其中：

1. 执行者以任意速度运行，可能因停止而失败，并可能重新启动。由于在选择一个值然后重新启动之后，所有执行者都可能会失败，因此这是不可行的，除非失败并重新启动的执行者可以记录某些信息。
2. 消息可能要花很长时间才能传递完成，可能重复并且可能丢失，但它们不会被修改。

**3.2 选择一个值**

选择值的最简单方式是只有一个接受者。提议者将提议发送给接受者，接受者选择它收到的第一个提议值。虽然简单，但是该解决方案并不可行，因为接受者的失败会使取得进一步的进展变得不可能。

因此，需要尝试另一种选择值的方式，使用多个接收者，而不是单个接收者。提议者将提议的值发送给一组接受者，接受者可以接受提议的值。当一组数量足够大的接受者接受它时，才选择该值。为了确保只选择一个值，我们可以让这组数量足够大的接受者包含任意大部分的执行者。因为任意两组大部分的执行者集合都至少有一个共同的接受者，因此如果一个接受者最多可以接受一个值，则此方式是可行的。

在没有失败或消息丢失的情况下，当一个值被单个提议者提出时，我们希望有一个值被选择。这表明要求：

**P1** 接受者必须接受它收到的第一个提议。

但是这个要求引出了一个问题。不同的提议者可能会在大约同一时间提出多个值，导致每个接受者都接受一个值，但没有一个值能被大多数接受者接受。即使只有两个提议的值，如果每个值都被大约一半的接受者接受，则单个接受者的失败也可能导致无法确定选择了哪个值。

P1和仅在大多数接受者接受的情况下才选择值的要求意味着必须允许一个接受者接受多个提议。我们通过为每个提议分配一个编号来跟踪接受者可能接受的不同提议，提议由提议编号和值组成。为避免混淆，我们要求不同的提议具有不同的编号。

当大多数接受者接受具有某个值的一个提案时，将选择该值，也就代表选择了该提案（及其值）。

我们可以允许选择多个提议，但是我们必须保证所有选择的提议都具有相同的值。通过对提案编号的归纳，需要保证：

­­­­­­­**P2** 如果选择了值为v的提议，则每个被选的编号更高的提议都具有值v。

由于编号是完全有序的，因此条件P2保证了仅选择一个值这一关键的安全性要求。

要被选择，提议必须至少由一个接受者接受。因此，我们可以通过满足以下条件来满足P2：

**P2a** 如果选择了值为v的提议，则任何接受者接受的每个编号更高的提议都具有值v。

我们仍然维持P1以确保选择一些提议。因为通信是异步的，所以一个提议可以被一个从未接受过任何提议的接受者c选择。假设有一个新的提议者“醒来”，并发布了一个具有不同值的编号更高的提议。P1要求c接受此提议，但这违反了P2a。要同时维持P1和P2a就要求将P2a加强为：

**P2b** 如果选择了值为v的提议，则任何提议者发出的每个编号更高的提议都具有值v。

由于提案必须先由提案者发布，才能被接受者接受，因此P2b代表了P2a，而后者又代表了P2。

下面是如何证明P2b成立。假设选择了一个编号为m且值为v的提案，并且任何编号为n（n>m）的提案也都具有值v。我们可以简单地通过对n归纳来证明，在每个编号为m～n-1的提议的值都为v的附加假设下，编号为n的提案具有值v。为了选择编号为m的提议，必须有一些由大多数接受者组成的集合C，以便C中的每个接受者都接受它。将其与归纳中的假设相结合，选择提议m的假设意味着：

C中的每个接受者都接受了一个编号为m～n-1的提议，并且每个接受者接受的每个m～n-1编号的提议都具有值v。

由于由多数接受者组成的任何集合S都包含C的至少一个成员，因此我们可以通过确保以下条件来断定编号为n的提案具有值v：

**P2c** 对于任何v和n，如果发布了具有值v和编号n的提议，则存在一个由多数接受者组成的集合S，使得（a）S中的任何接受者都不接受任何小于n的提案，或者（b）v是在S中接受者接受的编号小于n的所有提议中编号最大的提议的值。

因此，我们可以通过保持P2c的不变性来满足P2b。

为了保持P2c的不变性，想要发布编号为n的提议的提议者必须学习编号小于n（如果有的话）的最高编号的提议，该提议已被或将要被大多数接受者接受。要学习已经被接受的提议非常容易，而预测未来的接受很难。提议者要求接受者不接受任何其他编号小于n的提议。这引出了以下用于发出提议的算法：

1. 提议者选择一个新的提议编号n并向某组接受者的每个成员发送一个请求，要求其回应：

（a）承诺不再接受少于n的提案，并且

（b）已经接受的编号小于n且最大的提案（如果有）。

我们将这种请求称为准备请求，编号为n。

2. 如果提议者从大多数接受者那里收到请求的回应，那么它可以发出编号为n且值为v的提议，其中v是回应中编号最大的提议的值，或者是由提议者选择的任何值（如果响应者未回应任何提议）。

提议者通过向一组接受者发送接受该提议的请求来发布提议。 （不必是响应初始请求的同一组接受者）我们称其为接受请求。

以上是提议者的算法。对于接受者，则可以接收来自提议者的两种请求：准备请求和接受请求。接受者可以忽略任何请求而不会影响安全性。因此，只需说明何时才允许响应请求。它们总是可以响应准备请求，而只要它没有承诺不接受，它就可以响应接受请求，接受提议。也就是说：

**P1a** 接受者可以接受编号为n的提议，前提是它没有响应大于编号n的准备请求。

现在，我们就有了一个完整的算法来选择满足所需安全性要求的值（假设使用唯一的提议编号）。最终的算法则是经过一个小优化后获得的。

假设接受者收到编号为n的准备请求，但是它已经响应了编号大于n的准备请求，因此承诺不接受任何编号为n的新提议。这样，接受者就没有理由响应新的准备请求，因为它不会接受提议者要发布的编号为n的提议。因此，我们让接受者忽略这样的准备请求，也可以忽略已经被接受的提案的准备请求。

通过这种优化，接受者仅需要记住它曾经接受的编号最高的提议以及它已响应的编号最高的准备请求的编号。因为无论是否失败，P2c都必须保持不变，所以即使失败并重启，接受者也必须记录这些信息。而提议者可以放弃某个提议而忘记有关它的一切，只要它不尝试发布具有相同编号的其他提议。

将提议者和接受者的操作放在一起，可以看出该算法在以下两个阶段中运行。

**阶段1**：

（a）提议者选择提议编号n，并向大多数接受者发送提议编号n的准备请求。

（b）如果接受方收到的准备请求的编号n大于已响应的准备请求的编号，则接受方以承诺不接受任何其他编号小于n的提议，和已接受提议中的最大编号作为响应。

**阶段2**：

（a）如果提议者从大多数接受者那里收到对其编号为n的准备请求的响应，则向每个接受者发送对编号为n，值为v的提议的接受请求，其中v为响应中编号最大的提议的值；如果响应中没有提议编号，则为任意值。

（b）如果接受者收到对编号为n的提议的接受请求，它将接受该提案，除非该它已响应编号大于n的准备请求。

一个提议者可以提出多个提议，只要它遵循每个提议的算法即可。它可以随时在协议中间放弃提议（即使提议的请求或响应可能在提议被放弃后很长时间才到达目的地，也能保持正确性）。如果某些提议者已经开始尝试发布一个编号更大的提议，则放弃提议能是一个好主意。因此，如果接受者因为已经收到了编号较大的准备请求而忽略了准备或接受请求，则它应该通知提议者，然后由其放弃提议。这是一种不影响正确性的性能优化。

**3.3 学习被选择的值**

要学习已经被选择的值，学习者必须得知一个提议已被大多数接受者接受。 显而易见的算法是让每个接受者在接受提议时对所有学习者做出响应，向他们发送该提议。这使学习者能够尽快发现被选择的值，但是它要求每个接受者对每个学习者做出响应，响应的数量相当于接受者数量与学习者数量的乘积。

非拜占庭式失败的假设使得一个学习者很容易从另一个学习者那里发现一个值已经被接受。我们可以让接受者们将已被接受的值响应给某个特定的学习者，它将通知其他学习者所选择的值。这种方法需要所有学习者进行额外的一轮通知才能发现所选的值。它也不太可靠，因为特定的学习者可能会失败。但是，它需要的响应数量仅等于接受者数量和学习者数量之和。

接受者可以将所接受的值向某个学习者集合响应，那么每一个学习者都可以通知所有其他学习者已被选择的值。使用更大的学习者集合可以提供更高的可靠性，但需要承受更大的通信复杂性。

由于消息丢失，可能会在没有学习者得知的情况下选择一个值。学习者可以向接受者们询问他们接受了哪些提议，但是接受者的失败可能使得其无法知道大多数接受者是否接受了该提议。在这种情况下，学习者只有在一个新提议被选择时才能得知选择了什么值。如果学习者需要知道是否选择了一个值，则可以使用上述算法让提议者发布提议。

**3.4 进展性**

很容易可以构造这样一个场景，即两个提议者各自不断发布编号越来越大的提议序列，而它们都未被选择。提议者p完成了提议编号n1的阶段1。然后，另一个提议者q完成提议编号为n2> n1的阶段1。提议者p在阶段2中对n1提议的接受请求被忽略，因为所有接受者都承诺不接受任何编号小于n2的新提议。 然后，提议者p就开始并完成了一个编号n3> n2的新提议的阶段1，导致提议者q在阶段2中所发出的接受请求也被忽略。

为了保证进展，必须选择特定的提议者作为尝试发布提议的唯一提议者。如果特定的提议者可以与大多数接受者成功通信，并且使用的提议编号大于已被使用的提议编号，那么它将成功发布被接受的提议。通过放弃一个提议，并在得知某个提议具有更大编号的提议请求时进行再次尝试，该特定的提案者最终将选择一个足够大的提案编号。

如果系统基本上正常工作（提议者，接受者和通信网络），那么可以通过选举一个特定的提议者来实现系统的活动。Fischer，Lynch和Patterson的著名结论表明，一种可靠的选举提议者的算法必须使用随机性或实时性，例如使用超时。但无论选举的成功与否，安全性都能被确保。

**4 小结**

Paxos算法的作者通过划分三种角色并定义每种角色的指责，精准地定义了共识所需要满足的条件，并通过分析现有条件下存在的问题，不断加强和完善现有条件，从而推导出新的条件，以作为算法的基础理论。并将算法分为两个阶段，使其实现思路更为明确。接着分析实现这一条件可能存在的问题，包括学习者如何学习被选值，以及共识如何得以继续等。

在论述Paxos算法的过程中，我不仅对Paxos这一算法的推导及过程有了比较清晰的认识，而且学到了一个算法是如何逐步划分为大问题，并解决小问题的。这种“分而治之”并不断完善的过程很值得我去学习，并在今后的学习和工作生活中学以致用。

参考文献

[1] Leslie Lamport. The part-time parliament. ACM Transactions on Computer Systems, 16(2):133–169, May 1998.

[2] Leslie Lamport. 2001. Paxos Made Simple. ACM SIGACT News 32, 4 (2001), 51–58.