05 | Paxos算法（一）：如何在多个节点间确定某变量的值?

2020-02-21 韩健

分布式协议与算法实战 进入课程 >

讲述：于航
时长 13:07　大小 10.52M

你好，我是韩健。

提到分布式算法，就不得不提 Paxos 算法，在过去几十年里，它基本上是分布式共识的代名词，因为当前最常用的一批共识算法都是基于它改进的。比如，Fast Paxos 算法、Cheap Paxos 算法、Raft 算法、ZAB 协议等等。而很多同学都会在准确和系统理解 Paxos 算法上踩坑，比如，只知道它可以用来达成共识，但不知道它是如何达成共识的。

这其实侧面说明了 Paxos 算法有一定的难度，可分布式算法本身就很复杂，Paxos 算然也不会例外，当然了，除了这一点，还跟兰伯特有关。

兰伯特提出的 Paxos 算法包含 2 个部分：

一个是 Basic Paxos 算法，描述的是多节点之间如何就某个值（提案 Value）达成共识；

另一个是 Multi-Paxos 思想，描述的是执行多个 Basic Paxos 实例，就一系列值达成共识。

可因为兰伯特提到的 Multi-Paxos 思想，缺少代码实现的必要细节（比如怎么选举领导者），所以在理解上比较难。

为了让你理解 Paxos 算法，接下来我会用 2 节课的时间，分别以 Basic Paxos 和 Multi-Paxos 为核心，带你了解 Basic Paxos 如何达成共识，以及针对 Basic Paxos 的局限性 Multi-Paxos 又是如何改进的。今天咱们先来聊聊 Basic Paxos。

在我看来，Basic Paxos 是 Multi-Paxos 思想的核心，说白了，Multi-Paxos 就是多执行几次 Basic Paxos。所以掌握它之后，你能更好地理解后几讲基于 Multi-Paxos 思想的共识算法（比如 Raft 算法），还能掌握分布式共识算法的最核心内容，当现在的算法不能满足业务需求，进行权衡折中，设计自己的算法。

来看一道思考题。

假设我们要实现一个分布式集群，这个集群是由节点 A、B、C 组成，提供只读 KV 存储服务。你应该知道，创建只读变量的时候，必须要对它进行赋值，而且这个值后续没办法修改。因此一个节点创建只读变量后就不能再修改它了，所以所有节点必须要先对只读变量的值达成共识，然后所有节点再一起创建这个只读变量。

那么，当有多个客户端（比如客户端 1、2）访问这个系统，试图创建同一个只读变量（比如 X），客户端 1 试图创建值为 3 的 X，客户端 2 试图创建值为 7 的 X，这样要如何达成共识，实现各节点上 X 值的一致呢？带着这个问题，我们进入今天的学习。

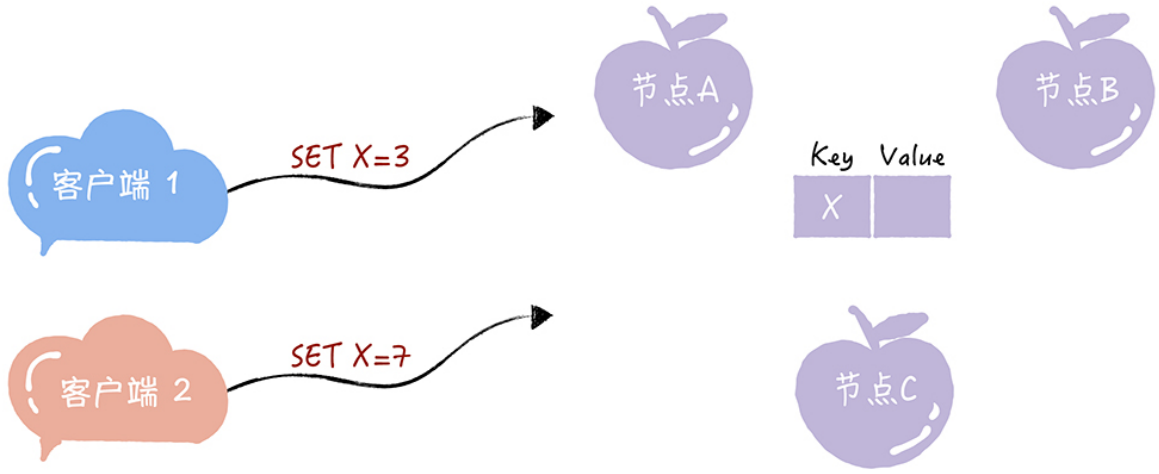

图1

在一些经典的算法中，你会看到一些既形象又独有的概念（比如二阶段提交协议中的协调者），Basic Paxos 算法也不例外。为了帮助人们更好地理解 Basic Paxos 算法，兰伯特在讲解时，也使用了一些独有而且比较重要的概念，提案、准备（Prepare）请求、接受（Accept）请求、角色等等，其中最重要的就是“角色”。因为角色是对 Basic Paxos 中最核心的三个功能的抽象，比如，由接受者（Acceptor）对提议的值进行投票，并存储接受的值。

你需要了解的三种角色

在 Basic Paxos 中，有提议者（Proposer）、接受者（Acceptor）、学习者（Learner）三种角色，他们之间的关系如下：

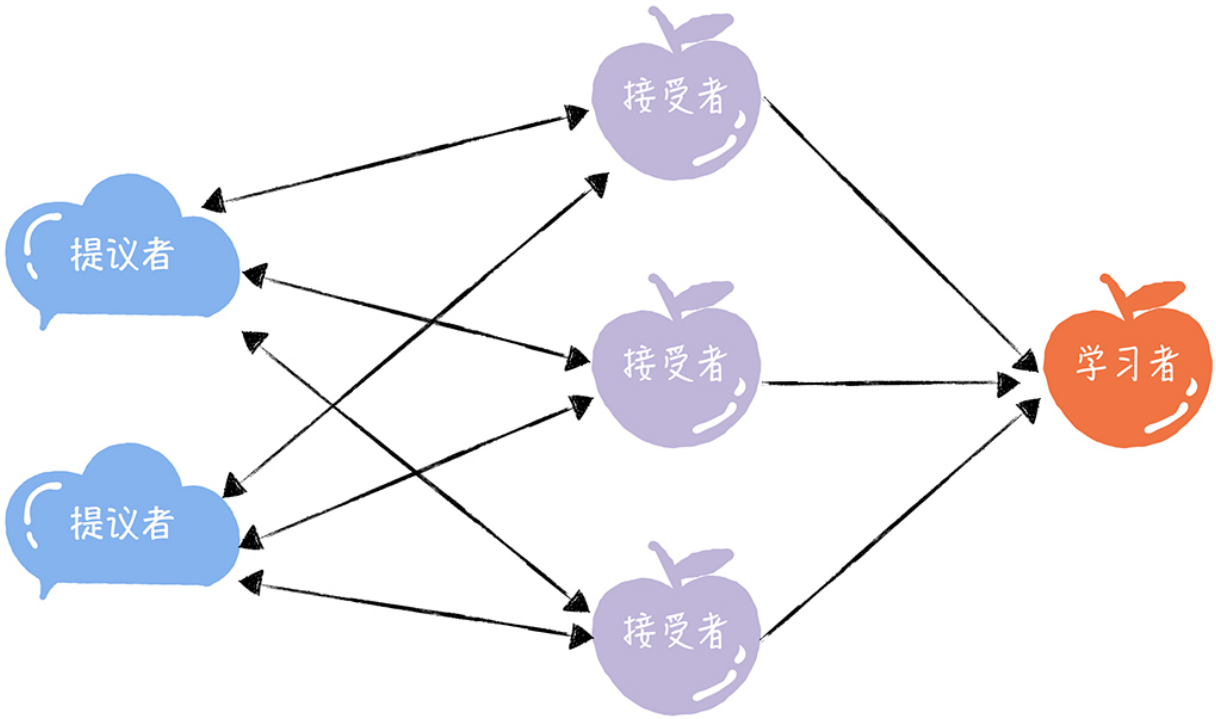

图2

看着是不是有些复杂，其实并不难理解：

提议者（Proposer）：提议一个值，用于投票表决。为了方便演示，你可以把图 1 中的客户端 1 和 2 看作是提议者。但在绝大多数场景中，集群中收到客户端请求的节点，才是提议者（图 1 这个架构，是为了方便演示算法原理）。这样做的好处是，对业务代码没有入侵性，也就是说，我们不需要在业务代码中实现算法逻辑，就可以像使用数据库一样访问后端的数据。

接受者（Acceptor）：对每个提议的值进行投票，并存储接受的值，比如 A、B、C 三个节点。 一般来说，集群中的所有节点都在扮演接受者的角色，参与共识协商，并接受和存储数据。

讲到这儿，你可能会有疑惑：前面不是说接收客户端请求的节点是提议者吗？这里怎么又是接受者呢？这是因为一个节点（或进程）可以身兼多个角色。想象一下，一个 3 节点的集群，1 个节点收到了请求，那么该节点将作为提议者发起二阶段提交，然后这个节点和另外 2 个节点一起作为接受者进行共识协商，就像下图的样子：

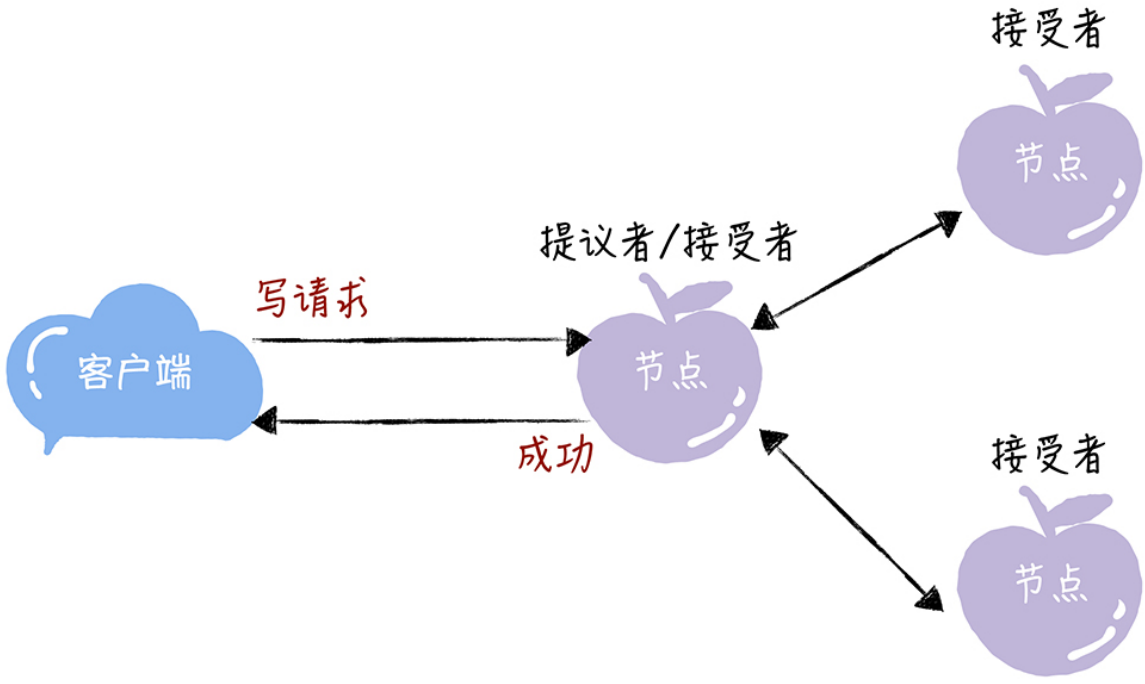

图3

学习者（Learner）：被告知投票的结果，接受达成共识的值，存储保存，不参与投票的过程。一般来说，学习者是数据备份节点，比如“Master-Slave”模型中的 Slave，被动地接受数据，容灾备份。

其实，这三种角色，在本质上代表的是三种功能：

提议者代表的是接入和协调功能，收到客户端请求后，发起二阶段提交，进行共识协商；

接受者代表投票协商和存储数据，对提议的值进行投票，并接受达成共识的值，存储保存；

学习者代表存储数据，不参与共识协商，只接受达成共识的值，存储保存。

因为一个完整的算法过程是由这三种角色对应的功能组成的，所以理解这三种角色，是你理解 Basic Paxos 如何就提议的值达成共识的基础。那么接下来，咱们看看如何使用 Basic Paxos 达成共识，解决开篇提到的那道思考题。

如何达成共识?

想象这样一个场景，现在疫情这么严重，每个村的路都封得差不多了，就你的村委会不作为，迟迟没有什么防疫的措施。你决定给村委会提交个提案，提一些防疫的建议，除了建议之外，为了和其他村民的提案做区分，你的提案还得包含一个提案编号，来起到唯一标识的作用。

与你的做法类似，在 Basic Paxos 中，兰伯特也使用提案代表一个提议。不过在提案中，除了提案编号，还包含了提议值。为了方便演示，我使用[n, v]表示一个提案，其中 n 为提案编号，v 为提议值。

我想强调一下，整个共识协商是分 2 个阶段进行的（也就是我在 03 讲提到的二阶段提交）。那么具体要如何协商呢?

我们假设客户端 1 的提案编号为 1，客户端 2 的提案编号为 5，并假设节点 A、B 先收到来自客户端 1 的准备请求，节点 C 先收到来自客户端 2 的准备请求。

准备（Prepare）阶段

先来看第一个阶段，首先客户端 1、2 作为提议者，分别向所有接受者发送包含提案编号的准备请求:

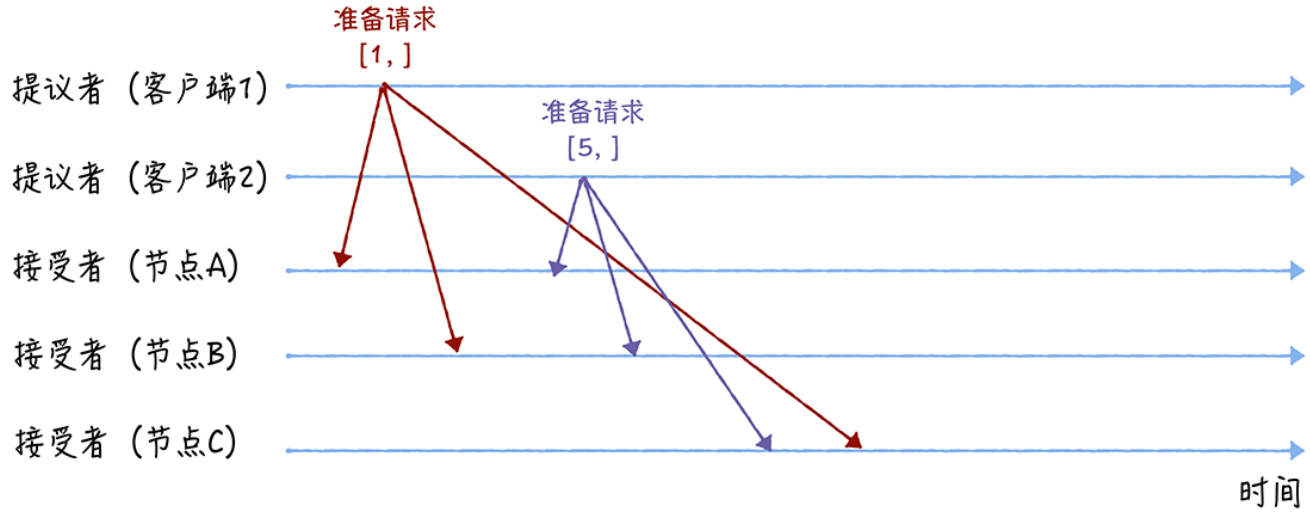

图4

你要注意，在准备请求中是不需要指定提议的值的，只需要携带提案编号就可以了，这是很多同学容易产生误解的地方。

接着，当节点 A、B 收到提案编号为 1 的准备请求，节点 C 收到提案编号为 5 的准备请求后，将进行这样的处理：

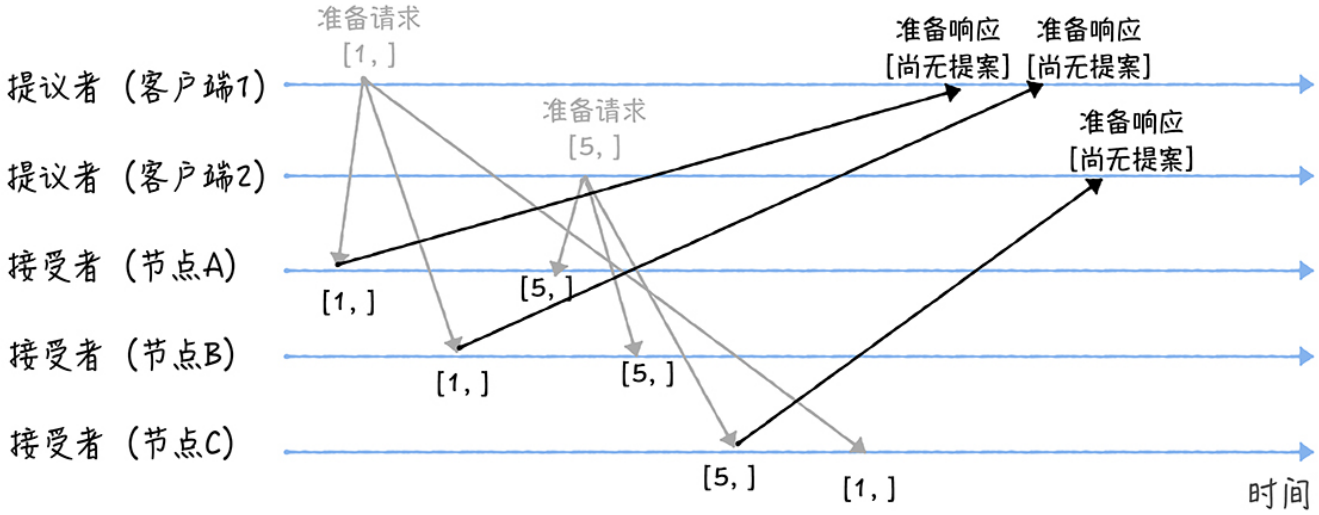

图5

由于之前没有通过任何提案，所以节点 A、B 将返回一个 “尚无提案”的响应。也就是说节点 A 和 B 在告诉提议者，我之前没有通过任何提案呢，并承诺以后不再响应提案编号小于等于 1 的准备请求，不会通过编号小于 1 的提案。

节点 C 也是如此，它将返回一个 “尚无提案”的响应，并承诺以后不再响应提案编号小于等于 5 的准备请求，不会通过编号小于 5 的提案。

另外，当节点 A、B 收到提案编号为 5 的准备请求，和节点 C 收到提案编号为 1 的准备请求的时候，将进行这样的处理过程：

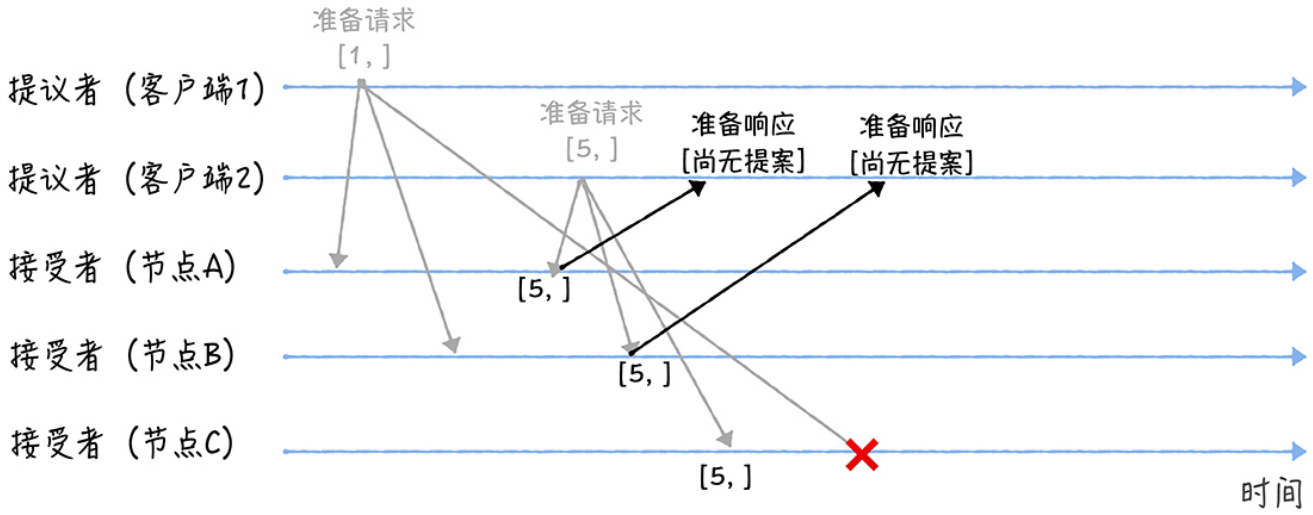

图6

当节点 A、B 收到提案编号为 5 的准备请求的时候，因为提案编号 5 大于它们之前响应的准备请求的提案编号 1，而且两个节点都没有通过任何提案，所以它将返回一个 “尚无提案”的响应，并承诺以后不再响应提案编号小于等于 5 的准备请求，不会通过编号小于 5 的提案。

当节点 C 收到提案编号为 1 的准备请求的时候，由于提案编号 1 小于它之前响应的准备请求的提案编号 5，所以丢弃该准备请求，不做响应。

接受（Accept）阶段

第二个阶段也就是接受阶段，首先客户端 1、2 在收到大多数节点的准备响应之后，会分别发送接受请求：

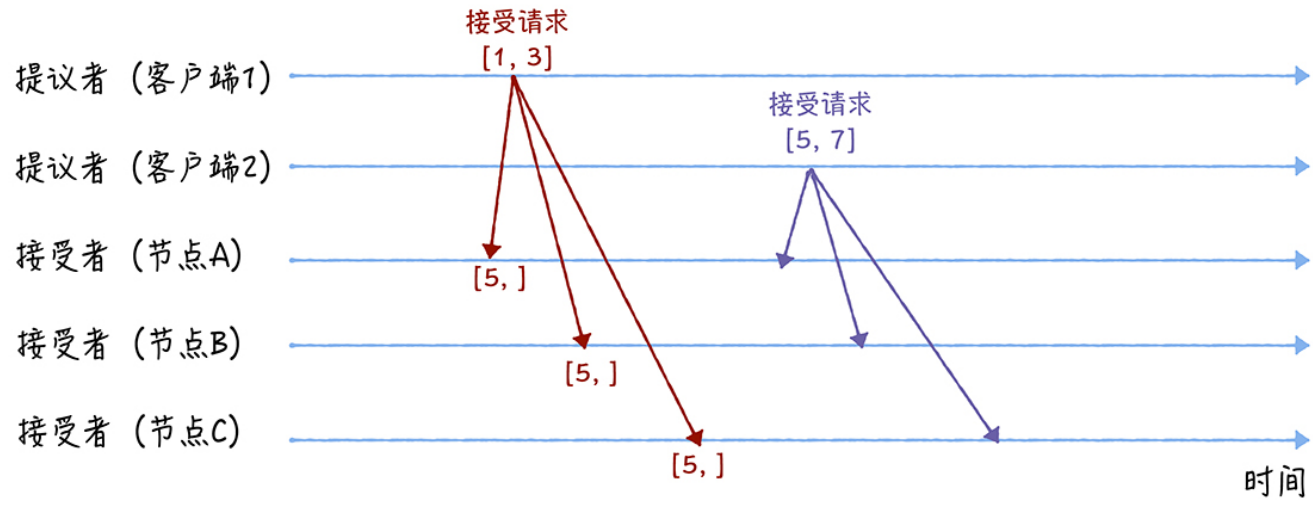

图7

当客户端 1 收到大多数的接受者（节点 A、B）的准备响应后，根据响应中提案编号最大的提案的值，设置接受请求中的值。因为该值在来自节点 A、B 的准备响应中都为空（也就是图 5 中的“尚无提案”），所以就把自己的提议值 3 作为提案的值，发送接受请求[1, 3]。

当客户端 2 收到大多数的接受者的准备响应后（节点 A、B 和节点 C），根据响应中提案编号最大的提案的值，来设置接受请求中的值。因为该值在来自节点 A、B、C 的准备响应中都为空（也就是图 5 和图 6 中的“尚无提案”），所以就把自己的提议值 7 作为提案的值，发送接受请求[5, 7]。

当三个节点收到 2 个客户端的接受请求时，会进行这样的处理：

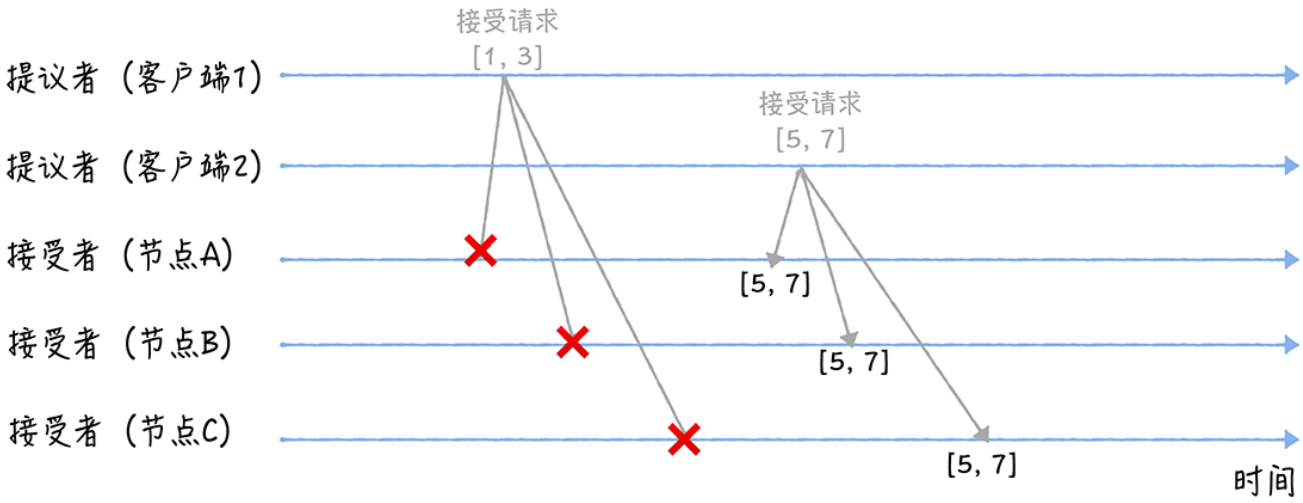

图8

当节点 A、B、C 收到接受请求[1, 3]的时候，由于提案的提案编号 1 小于三个节点承诺能通过的提案的最小提案编号 5，所以提案[1, 3]将被拒绝。

当节点 A、B、C 收到接受请求[5, 7]的时候，由于提案的提案编号 5 不小于三个节点承诺能通过的提案的最小提案编号 5，所以就通过提案[5, 7]，也就是接受了值 7，三个节点就 X 值为 7 达成了共识。

讲到这儿我想补充一下，如果集群中有学习者，当接受者通过了一个提案时，就通知给所有的学习者。当学习者发现大多数的接受者都通过了某个提案，那么它也通过该提案，接受该提案的值。

通过上面的演示过程，你可以看到，最终各节点就 X 的值达成了共识。那么在这里我还想强调一下，Basic Paxos 的容错能力，源自“大多数”的约定，你可以这么理解：当少于一

半的节点出现故障的时候，共识协商仍然在正常工作。

内容小结

本节课我主要带你了解了 Basic Paxos 的原理和一些特点，我希望你明确这样几个重点。

1. 你可以看到，Basic Paxos 是通过二阶段提交的方式来达成共识的。二阶段提交是达成共识的常用方式，如果你需要设计新的共识算法的时候，也可以考虑这个方式。
2. 除了共识，Basic Paxos 还实现了容错，在少于一半的节点出现故障时，集群也能工作。它不像分布式事务算法那样，必须要所有节点都同意后才提交操作，因为“所有节点都同意”这个原则，在出现节点故障的时候会导致整个集群不可用。也就是说，“大多数节点都同意”的原则，赋予了 Basic Paxos 容错的能力，让它能够容忍少于一半的节点的故障。
3. 本质上而言，提案编号的大小代表着优先级，你可以这么理解，根据提案编号的大小，接受者保证**三个承诺**，具体来说：如果准备请求的提案编号，**小于等于**接受者已经响应的准备请求的提案编号，那么接受者将承诺不响应这个准备请求；如果接受请求中的提案的提案编号，**小于**接受者已经响应的准备请求的提案编号，那么接受者将承诺不通过这个提案；如果接受者之前有通过提案，那么接受者将承诺，会在准备请求的响应中，包含**已经通过的最大编号的提案信息**。

课堂思考

在示例中，如果节点 A、B 已经通过了提案[5, 7]，节点 C 未通过任何提案，那么当客户端 3 提案编号为 9 时，通过 Basic Paxos 执行“SET X = 6”，最终三个节点上 X 值是多少呢？为什么呢？欢迎在留言区分享你的看法，与我一同讨论。

最后，感谢你的阅读，如果这篇文章让你有所收获，也欢迎你将它分享给更多的朋友。

上一篇　04 | BASE理论：CAP的碱，追求可用性

下一篇　06 | Paxos算法（二）：Multi-Paxos不是一个算法，而是统称

精选留言 (25)

写留言

小晏子
2020-02-21

如果节点 A、B 已经通过了提案[5, 7]，节点 C 未通过任何提案，那么当客户端 3 提案编号为 9 ，通过 Basic Paxos 执行“SET X = 6”，最终节点值应该是[9,7], 过程如下：
1. 在准备阶段，节点C收到客户端3的准备请求[9,6], 因为节点C未收到任何提案，所以返回“尚无提案”的相应。这时如果节点C收到了之前客户端的准备请求[5, 7], 根据提案编号5小于它之前响应的准备请求的提案编号9，会丢弃该准备请求。...

展开

作者回复: 加一颗星:)

3　5

Jialin

2020-02-21

如果节点 A、B 已经通过了提案[5, 7]，节点 C 未通过任何提案，那么当客户端 3 提案编号为 9 时，通过 Basic Paxos 执行“SET X = 6”时
准备阶段：当节点 A、B 收到提案编号为 9 的准备请求的时候，因为提案编号 9 大于它们之前响应的准备请求的提案编号 5，但这两个节点之前通过了提案[5, 7]，接受者A、B会在准备请求的响应中，包含已经通过的最大编号的提案信息[5, 7]，并承诺以后不再响应提...
展开

作者回复: 加一颗星:)

3

kernel_distribution
2020-02-23

http://pages.cs.wisc.edu/~remzi/Classes/739/Fall2017/paxos.lecture.pdf 请问下老师，这个最后的Q2有什么问题么。
展开

1

Dovelol
2020-02-22

老师好，今天题目开始说到是一个只读的kv存储，那是不是只要最后所有节点数据一样就算成功了？如果某个节点挂了，重启了该怎么获取到最新值呢？如果是可读写的节点，该怎么取更新某个值呢？

作者回复: 大多数节点就某个值，达成共识后，值就不再变了，哪怕有新的提案，这个值，也能保证不再变了。节点挂了，属于系统实现层面，如何同步数据的问题，如果要处理的话，比如，可以这么做，实现数据同步机制或重新执行Basic Paxos。如果更新某个值，这是需要Multi-Paxos和状态机，我19、20讲，我会具体说说。
咱们要注意的一点，Basic Paxos只能就单值达成共识，属于一个基础算法，Multi-Paxos，才能在实际场景中落地。

1

Purson
2020-02-22

前提：假设我们要实现一个分布式集群，这个集群是由节点 A、B、C 组成，提供只读 KV 存储服务。你应该知道，创建只读变量的时候，必须要对它进行赋值，而且这个值后续没办法修改。

准备阶段

...

展开

作者回复: 答案是7，在准备阶段是可以发现之前通过的值。

1

ID171

2020-02-21

因为节点C没有通过提案，可能是在接受阶段没有接收到来自客户端2的信息。
当客户端3提出提案9，提议阶段节点A, B将会把已经通过的值7返回给C，在接受阶段，客户端3会通知节点C设置[9, 7] （这里有个问题，还需要同步信息给A, B吗，需要把提案编号9保存在A, B, C节点上吗？），然后三个节点 A, B ,C 的X值均为7

展开

作者回复: 加一颗星:)，从代码实现的角度，是需要的。

1

nestle

2020-02-25

图7的 [5,] 没太看明白是啥意思。

展开

作者回复: 这个是为了显式表示，之前接收到了提案编号为5的准备请求，承诺以后不再响应提案编号小于等于 5 的准备请求，不会通过编号小于 5 的提案，辅助理解。另外，如果咱们实现Basic Paxos代码时，也需要找个变量记录下这个值的。

Fs

2020-02-25

怎么感觉有一定误导性，还是我理解错了。如果不是只读值呢，是可修改的呢。
那么最终示例还是[9,7]吗？

展开

作者回复: 还是7，解这个问题的关键是，在准备阶段会发现之前已经通过的提案的值:)

1

EidLeung
2020-02-23

“作者回复: 大多数节点就某个值，达成共识后，值就不再变了，哪怕有新的提案，这个值，也能保证不再变了。”这个最后的值肯定是7，至于编号嘛，取最大的。

作者回复: 加一颗星:)

约书亚
2020-02-23

这里是只读KV，后面会分析可以修改的KV系统下的场景么？看到对有相同疑问的同学的回复里提到：
"需要反复修改时，之前的共识是不能修改的，可以在之前的共识的基础之上，实现新的指令，在提交给状态机后，就可以修改value了。"
我可以理解成这样嘛？：...
展开

作者回复: 会的，在19、20讲，会具体分析如何实现kv存储，比如，可以将“set key = value”作为指令，需要更改值的时候，提出一个新指令，例如“set key = value2”，在状态机执行后，就可以更key的值了，这时需要的是Multi-Paxos和状态机了。

起个名字真难
2020-02-22

老师：
当客户端A 在准备请求的阶段，没有接收者收到响应信息，那么A会增大请求编号，重新发送请求么？按照什么规则增大的呢？

姜川
2020-02-22

准备请求：就是发送一个提案编号给各个节点
准备响应：各节点返回已经通过的提案中，编号最大的那个<提案编号：提案结果>
接受请求：就是发送一个<提案编号：提案结果>给各个节点，特殊的是，提案结果是受准备响应中提案结果影响的

展开

姜川
2020-02-22

准备请求：就是发送一个提案编号

展开

羽翼1982
2020-02-21

为什么当少于一半的节点故障时，共识协商依然能够正常工作？这块老师能够展开讲讲吗？
上面描述的达成共识的过程中并没有解释这个问题
[我的猜想] 是不是提议者会受到接受者的回复，告诉他的提议是否被接受，如果收到超过一半的节点回复（成功 / 失败），他就认为他的提议已经完成，否则他还会不断重发消...

展开

作者回复: 达成的是“大多数”节点的共识，少于一半的节点故障，也就是少数的节点故障时，不影响“大多数”这个约定。

子钧
2020-02-21

1、如何实现提案编号的共识？
2、半数机器同意，里机器数也是一个共识数值，在有机器宕机时，这个数值如何达成共识？

展开

作者回复: 是大多数，不是半数哈，只要宕机数不过半，算法是能继续运行的。

沉淀的梦想
2020-02-21

如果此时有客户端要读数据，那怎么处理？

展开

作者回复: 读，算法没有约定，Basic Paxos解决的是如何就单值达成共识。读，可以按需实现。

益军
2020-02-21

客户端A编号1，客户端B编号5怎么来的，避免编号冲突本身也是共识。

作者回复: 如何实现提案编号，没有约定的，但提案编号，不影响共识协商，提案编号的大小是个优先级，终归是有个顺序的。

Geek_MYMSwen
2020-02-21

思考题应该是6吧。
原因：接受者的第三个承诺：在接受者之前通过提案，会在准备请求的响应中包含已经通过最大编号的提案信息。
在接到编号为9的准备请求时，接受者的状态相当于：已经响应了编号为5的准备请求。
然后接到了编号为9的准备请求。按照流程应该会将状态调整为已经响应编号为9的准备...
展开

作者回复: 是7哈，在准备阶段，可以发现之前通过的提案的值。这也是为什么大多数节点达成共识后，这个值就不再变了。

沉淀的梦想
2020-02-21

Basic Paxos 是只能用来解决这种只有一个 Key 的并且一次性赋值的只读 KV 存储的一致性问题吗？如果有多个 Key，或者要反复修改，就要用 Multi Paxos?
展开

作者回复: Basic Paxos解决的单值的共识的问题，如果需要就一系列值达成共识，就多执行几次Basic Paxos，也就是Multi Paxos。如果，多个key，需要反复修改时，之前的共识是不能修改的，可以在之前的共识的基础之上，实现新的指令，在提交给状态机后，就可以修改value了。

沉淀的梦想

2020-02-21

提案编号需要唯一且递增吗？如果需要的话，这在算法中又要如何保证呢？

> 作者回复: 实现细节，兰伯特是没有提的，需要自己实现的， 比如，在Raft中，可以将任期编号理解为提案编号，跟随者发起选举时，增加自己的任期编号；发现自己的任期编号比其他节点小，那么它会更新自己的编号到较大的编号值；还通过随机超时时间，避免相同任期编号的跟随者同时发起选举。