

微信PaxosStore：深入浅出Paxos算法协议

原创 郑建军 InfoQ 2016-11-15



作者 | 郑建军

编辑 | Cindy

“

与其预测未来，不如限制未来”，这应该是Paxos协议的核心思想。Paxos协议本身是比较简单的，如何将Paxos协议工程化，才是真正的难题。这是来自微信工程师的经验，以供参考。

引言

早在1990年，Leslie Lamport（即 LaTeX 中的"La"，微软研究院科学家，获得2013年图灵奖）向ACM Transactions on Computer Systems (TOCS)提交了关于Paxos算法的论文The Part-Time Parliament。几位审阅人表示，虽然论文没什么特别的用处，但还是有点意思，只是要把Paxos相关的故事背景全部删掉。Leslie Lamport心高气傲，觉得审阅人没有丝毫的幽默感，于是撤回文章不再发表。

直到1998年，用户开始支持Paxos，Leslie Lamport重新发表文章，但相比1990年的版本，文章没有太大的修改，所以还是不好理解。于是在2001年，为了通俗性，Leslie Lamport简化文章发表了Paxos Made Simple，这次文中没有一个公式。

但事实如何？大家不妨读一读Paxos Made Simple。Leslie Lamport在文中渐进式地、从零开始推导出了Paxos协议，中间用数学归纳法进行了证明。可能是因为表述顺序的问题，导致这篇文章似乎还是不好理解。

于是，基于此背景，本文根据Paxos Made Simple，重新描述Paxos协议，提供两种证明方法，给出常见的理解误区。期望读者通过阅读本文，再结合Paxos Made Simple，就可以深入理解基本的Paxos协议理论。

基本概念

- Proposal Value：提议的值；
- Proposal Number：提议编号，要求提议编号不能冲突；
- Proposal：提议 = 提议的值 + 提议编号；
- Proposer：提议发起者；
- Acceptor：提议接受者；
- Learner：提议学习者。

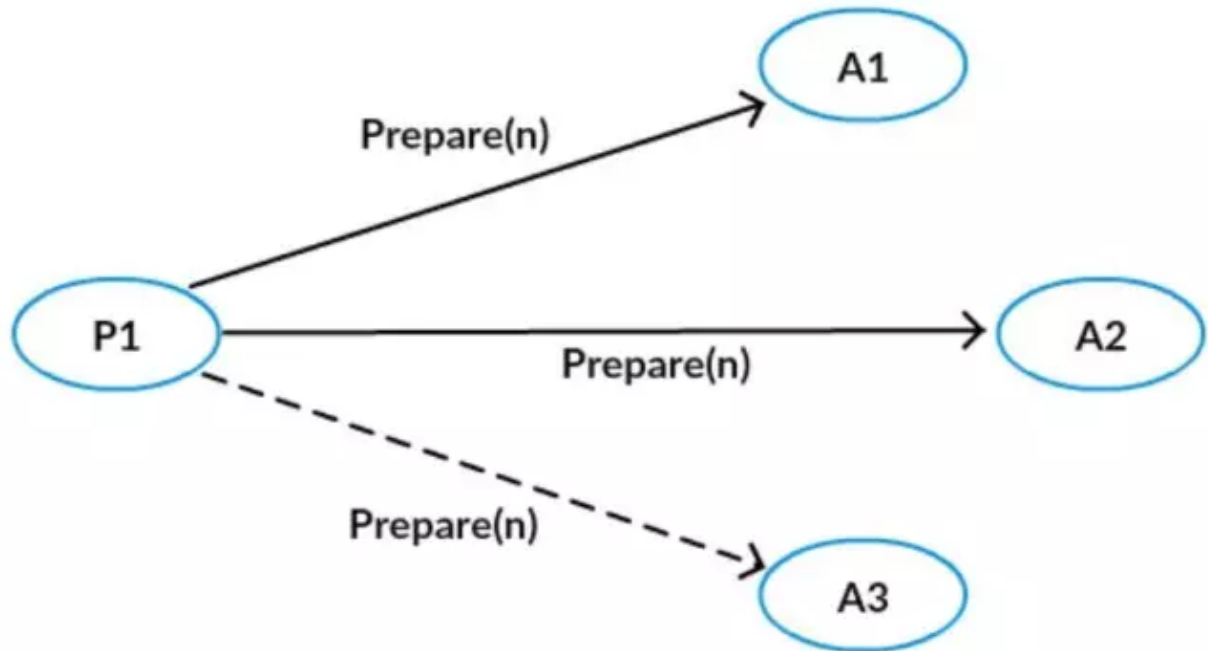
注意，提议跟提议的值是有区别的，后面会具体说明。协议中Proposer有两个行为，一个是向Acceptor发Prepare请求，另一个是向Acceptor发Accept请求；Acceptor则根据协议规则，对Proposer的请求作出应答；最后Learner可以根据Acceptor的状态，学习最终被确定的值。

方便讨论，在本文中，记 $\{n, v\}$ 为提议编号为 n ，提议的值为 v 的提议，记 $(m, \{n, v\})$ 为承诺了Prepare (m) 请求，并接受了提议 $\{n, v\}$ 。

协议过程

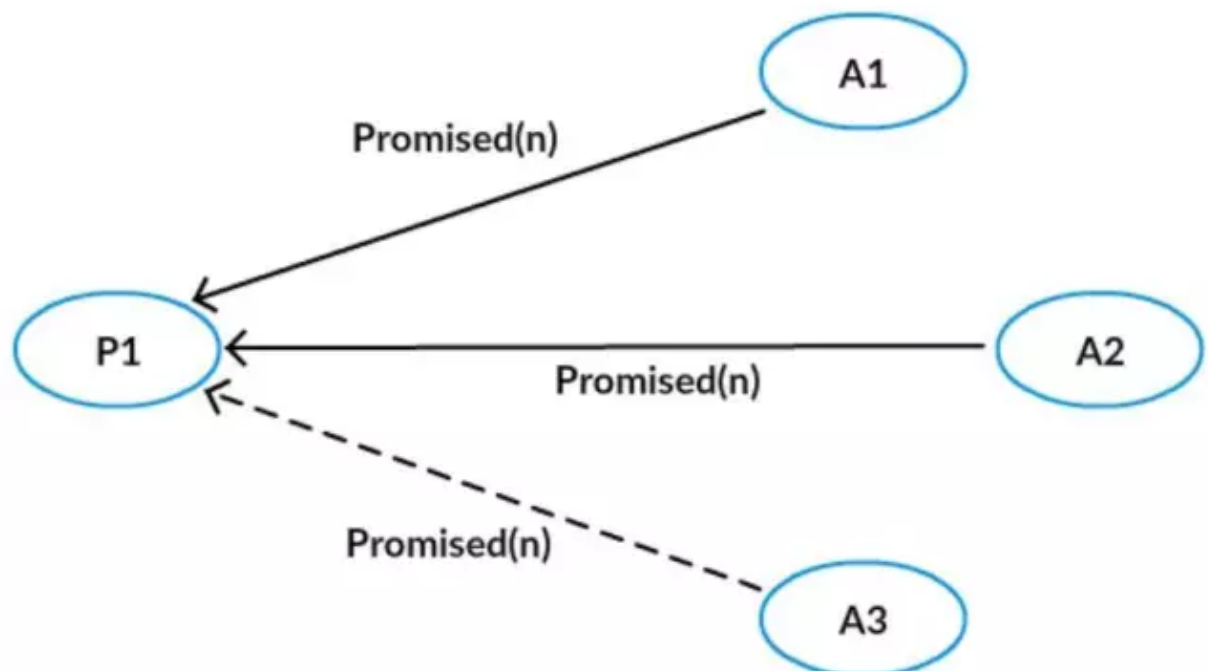
第一阶段A

Proposer选择一个提议编号 n ，向所有的Acceptor广播Prepare (n) 请求。



第一阶段B

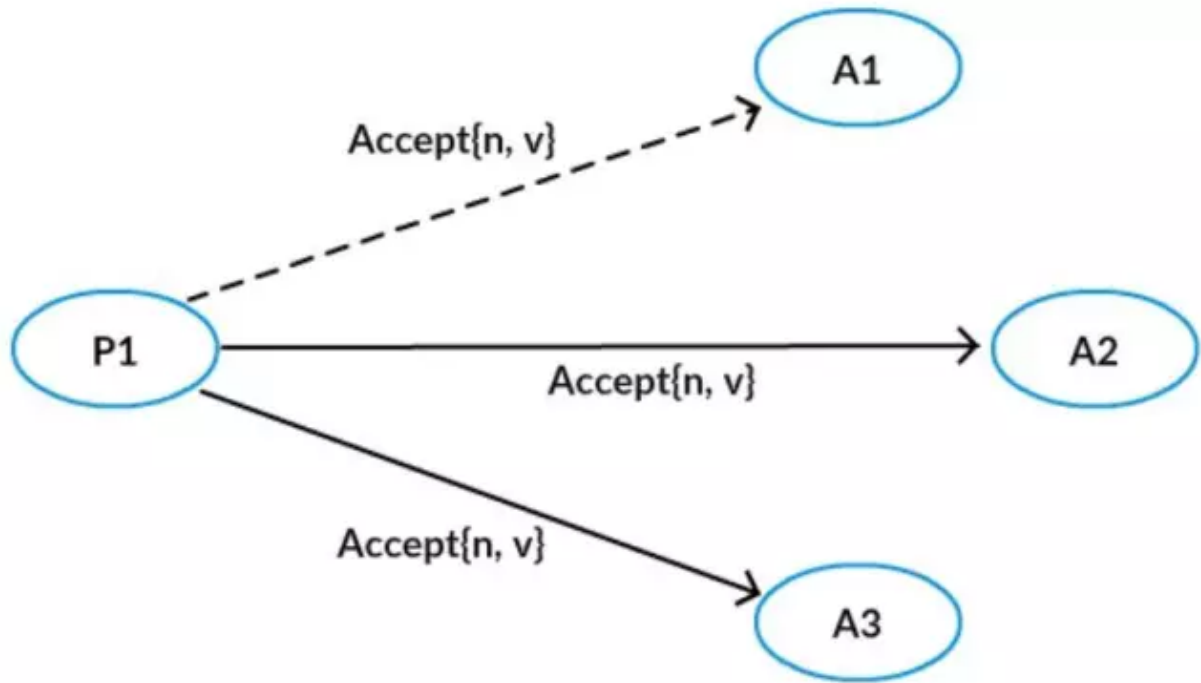
Acceptor接收到Prepare (n) 请求，若提议编号n比之前接收的Prepare请求都要大，则承诺将不会接收提议编号比n小的提议，并且带上之前Accept的提议中编号小于n的最大的提议，否则不予理会。



第二阶段A

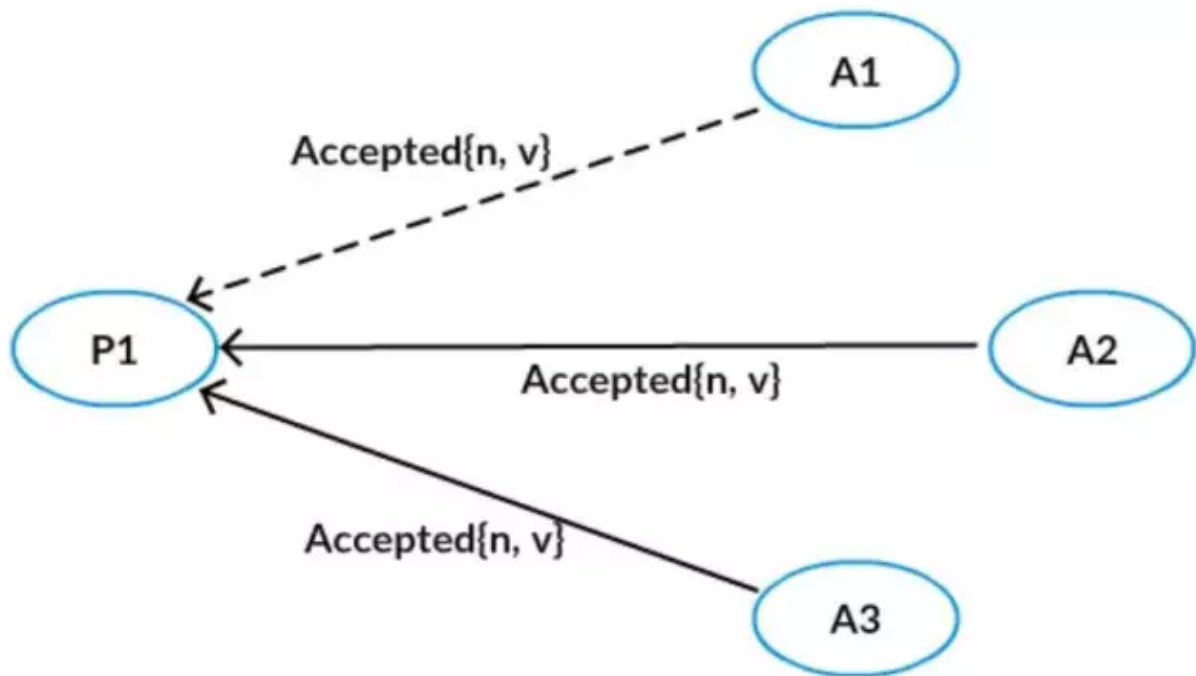
Proposer得到了多数Acceptor的承诺后，如果没有发现有一个Acceptor接受过一个值，那么向所有的Acceptor发起自己的值和提议编号n，否则，从所有接受过的值中选择对应

的提议编号最大的，作为提议的值，提议编号仍然为n。



第二阶段B

Acceptor接收到提议后，如果该提议编号不违反自己做过的承诺，则接受该提议。

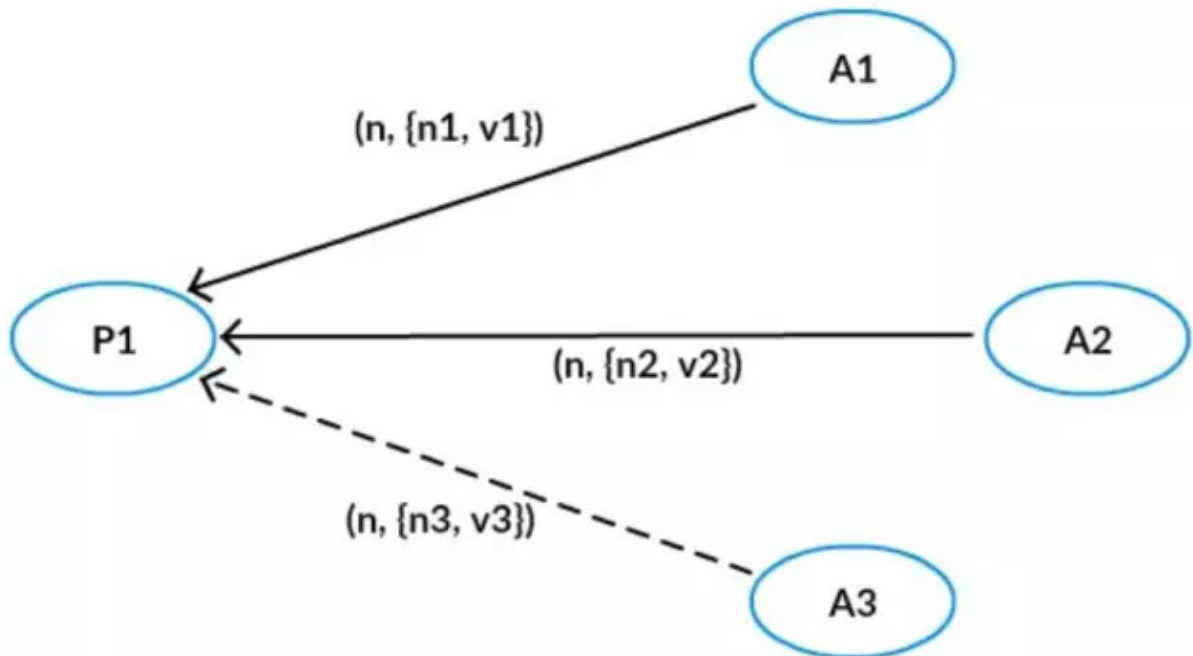


需要注意的是，Proposer发出Prepare (n) 请求后，得到多数派的应答，然后可以随便再选择一个多数派广播Accept请求，而不一定要将Accept请求发给有应答的Acceptor，这是常见的Paxos理解误区。

小结

上面的图例中，P1广播了Prepare请求，但是给A3的丢失，不过A1、A2成功返回了，即该Prepare请求得到多数派的应答，然后它可以广播Accept请求，但是给A1的丢了，不过A2，A3成功接受了这个提议。因为这个提议被多数派（A2，A3形成多数派）接受，我们称被多数派接受的提议对应的值被Chosen。

三个Acceptor之前都没有接受过Accept请求，所以不用返回接受过的提议，但是如果接受过提议，则根据第一阶段B，要带上之前Accept的提议中编号小于n的最大的提议。



Proposer广播Prepare请求之后，收到了A1和A2的应答，应答中携带了它们之前接受过的{n1, v1}和{n2, v2}，Proposer则根据n1，n2的大小关系，选择较大的那个提议对应的值，比如 $n1 > n2$ ，那么就选择v1作为提议的值，最后它向Acceptor广播提议{n, v1}。

Paxos协议最终解决什么问题？

当一个提议被多数派接受后，这个提议对应的值被Chosen（选定），一旦有一个值被Chosen，那么只要按照协议的规则继续交互，后续被Chosen的值都是同一个值，也就是这个Chosen值的一致性问题。

协议证明

上文就是基本Paxos协议的全部内容，其实是一个非常确定的数学问题。下面用数学语言表达，进而用严谨的数学语言加以证明。

Paxos原命题

如果一个提议 $\{n_0, v_0\}$ 被大多数Acceptor接受, 那么不存在提议 $\{n_1, v_1\}$ 被大多数Acceptor接受, 其中 $n_0 < n_1$, $v_0 \neq v_1$ 。

Paxos原命题加强

如果一个提议 $\{n_0, v_0\}$ 被大多数Acceptor接受, 那么不存在Acceptor接受提议 $\{n_1, v_1\}$, 其中 $n_0 < n_1$, $v_0 \neq v_1$ 。

Paxos原命题进一步加强

如果一个提议 $\{n_0, v_0\}$ 被大多数Acceptor接受, 那么不存在Proposer发出提议 $\{n_1, v_1\}$, 其中 $n_0 < n_1$, $v_0 \neq v_1$ 。

如果“Paxos原命题进一步加强”成立, 那么“Paxos原命题”显然成立。下面我们通过证明“Paxos原命题进一步加强”, 从而证明“Paxos原命题”。论文中是使用数学归纳法进行证明的, 这里用比较紧凑的语言重新表述证明过程。

归纳法证明

假设, 提议 $\{m, v\}$ (简称提议 m) 被多数派接受, 那么提议 m 到 n (如果存在) 对应的值都为 v , 其中 n 不小于 m 。

这里对 n 进行归纳假设, 当 $n = m$ 时, 结论显然成立。

设 $n = k$ 时结论成立, 即如果提议 $\{m, v\}$ 被多数派接受,

那么提议 m 到 k 对应的值都为 v , 其中 k 不小于 m 。

当 $n = k+1$ 时, 若提议 $k+1$ 不存在, 那么结论成立。

若提议 $k+1$ 存在, 对应的值为 v_1 ,

因为提议 m 已经被多数派接受, 又 $k+1$ 的Prepare被多数派承诺并返回结果。

基于两个多数派必有交集, 易知提议 $k+1$ 的第一阶段B有带提议回来。

那么 v_1 是从返回的提议中选出来的, 不妨设这个值是选自提议 $\{t, v_1\}$ 。

根据第二阶段B, 因为 t 是返回的提议中编号最大, 所以 $t \geq m$ 。

又由第一阶段A, 知道 $t < n$ 。所以根据假设 t 对应的值为 v 。

即有 $v_1 = v$ 。所以由 $n = k$ 结论成立, 可以推出 $n = k+1$ 成立。

于是对于任意的提议编号不小于 m 的提议 n , 对应的值都为 v 。

所以命题成立。

反证法证明

假设存在，不妨设 n_1 是满足条件的最小提议编号。

即存在提议 $\{n_1, v_1\}$ ，其中 $n_0 < n_1$ ， $v_0 \neq v_1$ 。----- (A)

那么提议 $n_0, n_0+1, n_0+2, \dots, n_1-1$ 对应的值为 v_0 。----- (B)

由于存在提议 $\{n_1, v_1\}$ ，则说明大多数Acceptor已经接收 n_1 的Prepare，并承诺将不会接受提议编号比 n_1 小的提议。

又因为 $\{n_0, v_0\}$ 被大多数Acceptor接受，

所以存在一个Acceptor既对 n_1 的Prepare进行了承诺，又接受了提议 n_0 。

由协议的第二阶段B知，这个Acceptor先接受了 $\{n_0, v_0\}$ 。

所以发出 $\{n_1, v_1\}$ 提议的Proposer会从大多数的Acceptor返回中得知，

至少某个编号不小于 n_0 而且值为 v_0 的提议已经被接受。----- (C)

由协议的第二阶段A知，

该Proposer会从已经被接受的值中选择一个提议编号最大的，作为提议的值。

由 (C) 知该提议编号不小于 n_0 ，由协议第二阶段B知，该提议编号小于 n_1 ，

于是由 (B) 知 $v_1 = v_0$ ，与 (A) 矛盾。

所以命题成立。

通过上面的证明过程，我们反过来回味一下协议中的细节。

- 为什么要被多数派接受？

因为两个多数派之间必有交集，所以Paxos协议一般是 $2F+1$ 个Acceptor，然后允许最多 F 个Acceptor停机，而保证协议依然能够正常进行，最终得到一个确定的值。

- 为什么需要做一个承诺？

可以保证第二阶段A中Proposer的选择不会受到未来变化的干扰。另外，对于一个Acceptor而言，这个承诺决定了它回应提议编号较大的Prepare请求，和接受提议编号较小的Accept请求的先后顺序。

- 为什么第二阶段A要从返回的协议中选择一个编号最大的？

这样选出来的提议编号一定不小于已经被多数派接受的提议编号，进而可以根据假设得到该提议编号对应的值是Chosen的那个值。

原文的第一阶段B

Acceptor接收到Prepare (n) 请求，若提议编号n比之前接收的Prepare请求都要大，则承诺将不会接收提议编号比n小的提议，并且带上之前Accept的提议中编号最大的提议，否则不予理会。

相对上面的表达少了“比n小的”，通过邮件向Leslie Lamport请教了这个问题，他表示接受一个提议，包含回应了一个Prepare请求。这个有点隐晦，但也完全合理，有了这个条件，上面的证明也就通顺了。就是说Acceptor接受过的提议的编号总是不大于承诺过的提议编号，于是可以将这个“比n小的”去掉，在实际工程实践中我们往往只保存接受过的提议中编号最大的，以及承诺过的Prepare请求编号最大的。

Leslie Lamport也表示在去掉“比n小的”的情况下，就算接受一个提议不包含回应一个Prepare请求，最终结论也是对的，因为前者明显可以推导出后者，去掉反而把条件加强了。

假如返回的提议中有编号大于n的，比如{m, v}，那么肯定存在多数派承诺拒绝小于m的Accept请求，所以提议{n, v}不可能被多数派接受。

学习过程

如果一个提议被多数Acceptor接受，则这个提议对应的值被选定。

一个简单直接的学习方法就是，获取所有Acceptor接受过的提议，然后看哪个提议被多数的Acceptor接受，那么该提议对应的值就是被选定的。

另外，也可以把Learner看作一个Proposer，根据协议流程，发起一个正常的提议，然后看这个提议是否被多数Acceptor接受。

这里强调“一个提议被多数Acceptor接受”，而不是“一个值被多数Acceptor”接受，如果是后者会有什么问题？

提议{3, v3}, {5, v3}分别被B、C接受，即有v3被多数派接受，但不能说明v3被选定（Chosen），只有提议{7, v1}被多数派（A和C组成）接受，我们才能说v1被选定，而这个选定的值随着协议继续进行不会改变。

总结

“与其预测未来，不如限制未来”，这应该是Paxos协议的核心思想。如果你在阅读Paxos的这篇论文时感到困惑，不妨找到“限制”的段落回味一番。Paxos协议本身是比较简单的，如何将Paxos协议工程化，才是真正的难题。

目前在微信核心存储PaxosStore中，每分钟调用Paxos协议过程数十亿次量级，而《微信PaxosStore内存云揭秘：十亿Paxos/分钟的挑战》一文，则对内存云子系统做了展开。

后续我们将发表更多的实践方案，包括万亿级别的海量闪存存储，支持单表亿行的NewSQL解决方案，以及有别于业界的开源实现，PaxosStore架构方案基于非租约的Paxos实现等内容。

作者简介

郑建军，微信工程师，目前负责微信基础存储服务，致力于强一致、高可用的大规模分布式存储系统的设计与研发。

今日荐文

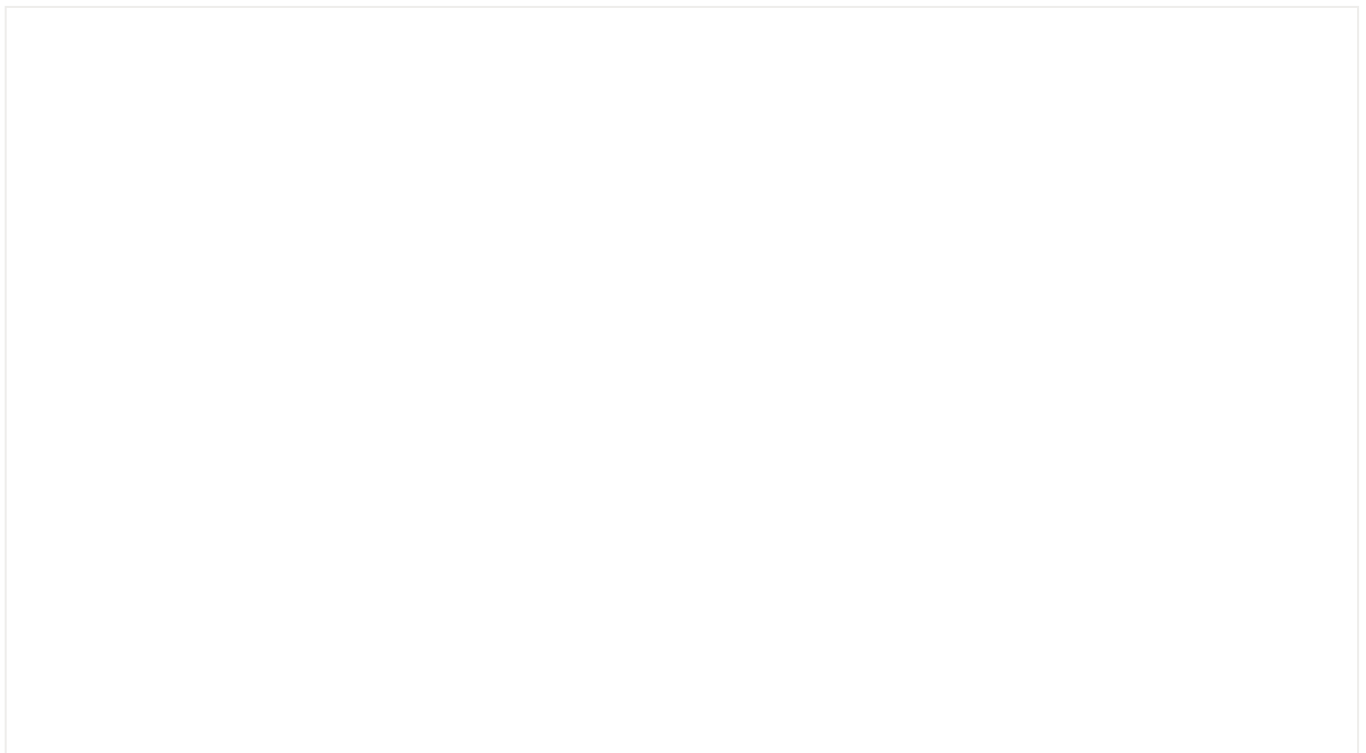
点击下方图片即可阅读

微信PaxosStore内存云揭秘：十亿Paxos/分钟的挑战

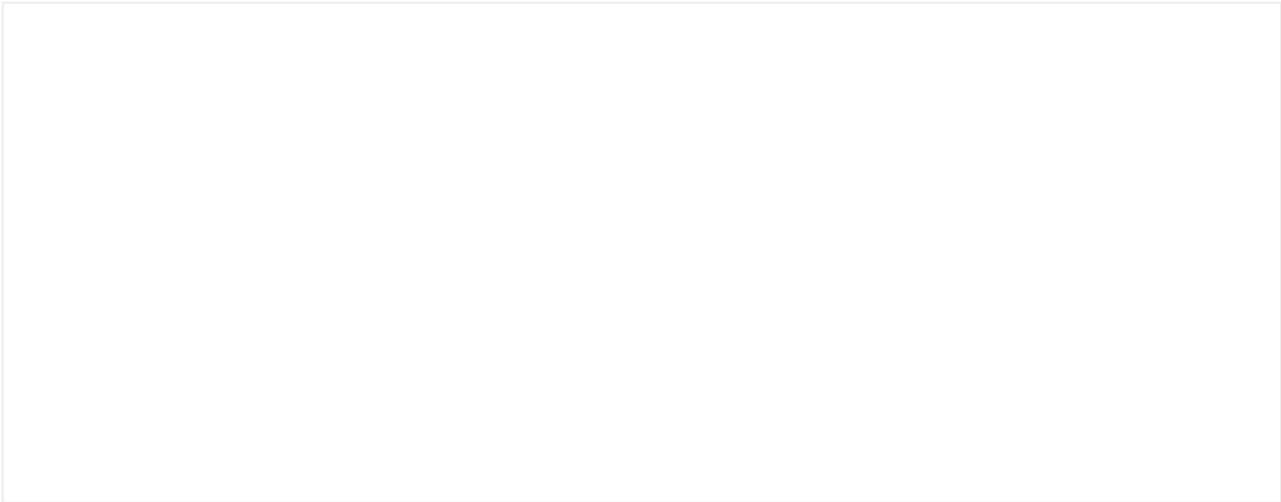


对于大多数人来说，使用微服务架构来操纵和管理大规模应用是一项比较困难的工作，而且运行复杂的容器管理架构也并不轻松。正因为如此，Amazon EC2 Container Service (ECS) 作为一项高度可扩展的高性能容器管理服务，得到了很多人的青睐。

本期在线课堂特别邀请AWS首席云计算技术顾问结合案例为大家详细讲解如何更好地设置、管理和扩展AmazonECS，点击[阅读原文](#)免费报名！



喜欢我们的会点赞，爱我们的会分享！



阅读原文

喜欢此内容的人还喜欢

放弃大厂高薪的程序员，涌进体制内

InfoQ

一年只卖三次地，加速淘汰小房企

涛哥杂谈

市场监管总局定点扶贫（对口支援）工作领导小组办公室被党中央、国务院授予“全国脱贫攻坚先进集体”荣誉称号

市说新语