O ALGORITMO PAXOS

Giovanni Moreira Guimarães[1], Vitor Oliveira Ropke[2]

Resumo: O algoritmo Paxos, criado por Leslie Lamport, é uma solução para resolver o problema do consenso em sistemas distribuídos. Seu objetivo é garantir que, mesmo com falhas em alguns processos, os processos restantes possam chegar a uma decisão conjunta de maneira eficiente e confiável. O algoritmo funciona em três fases: proposição, aceitação e aprendizado. Um líder é eleito para gerenciar o processo de proposição e aceitação de propostas, garantindo o progresso. Os proponentes propõem valores, que precisam ser acordados pela maioria dos processos. O algoritmo Paxos é usado em uma variedade de aplicações, como o gerenciador de lock do Google Chubby, o serviço de gerenciamento de cluster da Microsoft Autopilot usado no motor de busca do Bing e o Amazon Web Services. Ele permitiu o desenvolvimento de sistemas distribuídos confiáveis e eficientes.
Palavras-chave: Paxos; sistemas distribuídos; processos; algoritmo; consenso.

1. INTRODUÇÃO

O algoritmo Paxos, criado por Leslie Lamport e dado ao mundo num artigo intitulado The Part-Time Parliament[1], na forma de uma pesquisa arqueológica fictícia de uma ilha grega chamada Paxos (que realmente existe, mas não a civilização como relatada no artigo) que tinha um sistema particular no qual os políticos trabalhavam meio período. Ninguém entendeu nada sobre aquele artigo, tanto que um tempo depois Lamport publicou Paxos Made Simple, em 2001[2].

Os principais conceitos do algoritmo Paxos incluem:

- **Líder**: o algoritmo requer a eleição de um líder para gerenciar o processo de proposição e aceitação de propostas. Se houver vários proponentes, suas propostas podem entrar em conflito indefinidamente. Para garantir o progresso, apenas um líder distinto propõe os valores. Eleger um líder é uma otimização.
- **Proponente**: propõe solicitações de clientes a um quórum de aceitantes.
- **Proposta**: um processo inicia uma proposta de um valor que deseja que seja acordado.
- **Acordo**: para que uma proposta seja aceita, ela precisa ser acordada pela maioria dos processos. Isso é conhecido como “condição de quórum”.
- **Promessa**: quando um processo recebe uma proposta, ele faz uma promessa de não aceitar nenhuma outra proposta até que a proposta atual seja aceita ou rejeitada.
- **Aceitador**: aceita propostas de proponentes e pode recusar se houver conflitos. Os proponentes enviam uma proposição a um quórum de aceitantes e só podem prosseguir se a proposição for aceita por um quórum.
- **Aceitação**: se uma proposta é acordada, os processos precisam aceitá-la. Significa que eles concordam em manter o valor proposto e não aceitar outra até que a decisão seja tomada.
- **Aprendiz**: aprende os valores propostos. É passivo e é a parte de replicação do protocolo.
- **Aprendizado**: depois que uma proposta for aceita, todos os processos aprendem o valor proposto.
- **Fases**: o algoritmo Paxos funciona em três fases: proposição, aceitação e aprendizado.

Esses conceitos são cruciais para entender como o algoritmo Paxos funciona, mas uma vez que são compreendidos, é possível aplicá-lo com sucesso em muitos sistemas distribuídos.

2. DESENVOLVIMENTO

Parte principal do artigo, deve conter a exposição ordenada e pormenorizada do assunto tratado. Deve conter a Fundamentação teórica, o método os resultados e as discussões. Divide-se em seções e subseções, conforme a NBR 6024, que variam em função da abordagem do tema e do método.

2.1. Forma de Funcionamento

Como já dito, o algoritmo Paxos funciona em três fases: proposição, aceitação e aprendizado. Na fase de proposição, um processo propõe um valor para ser decidido. Na fase de aceitação, os outros processos precisam aceitar a proposta. E, finalmente, na fase de aprendizado, o valor proposto é aprendido por todos os processos.

No início, um proponente envia uma mensagem de preparação com um número de proposta N e o valor de uma solicitação do cliente para um quórum de aceitadores. Os aceitadores aceitam a proposta e respondem com uma mensagem de preparação de resposta, se o número da proposta N for maior do que qualquer número de proposta visto anteriormente. Caso contrário, os aceitadores não respondem ou enviam uma confirmação negativa. Se vários proponentes propõem valores diferentes, as mensagens de preparação permitem que os aceitadores detectem conflitos.
Em uma segunda fase, o proponente envia uma mensagem de aceitação para um quórum de aceitadores. A mensagem de aceitação contém os mesmos valores de uma mensagem de preparação. Se o número de proposta N for grande o suficiente, o acordo é bem-sucedido e o aceitante transmite uma mensagem de aprendizagem para informar os aprendizes sobre o valor escolhido. escolhido. Se um aprendiz receber mensagens de aprendizado de um quórum de aceitadores, ele confirmará o valor escolhido. Veja a figura 1. [3]

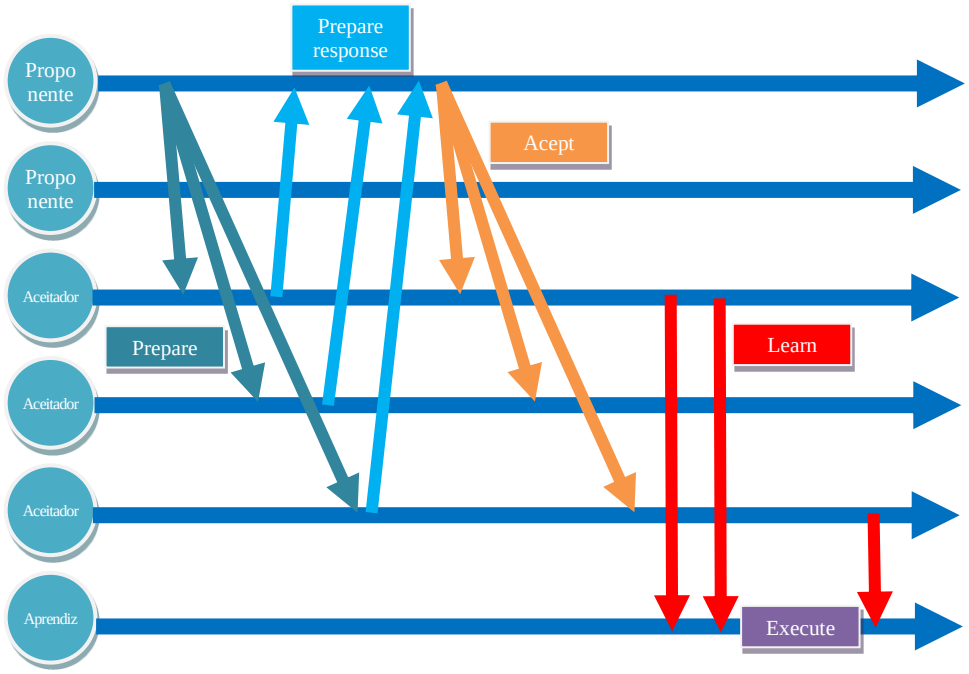

Figura 1. Paxos: caso base sem falhas

O Paxos considera apenas um único contrato, mas a maioria dos sistemas exige um fluxo de contratos. Para acordos múltiplos, o Multi-Paxos é usado. Multi-Paxos assume que apenas um único proponente é escolhido como líder e que o líder é bastante estável.

Quando o líder é escolhido, a primeira fase com mensagens de preparação não é mais necessária e o número de atrasos para chegar a um acordo é reduzido. Um caso mais comum onde os papéis são combinados (réplica é aceitador e aprendiz) é mostrado na figura 2. Através da formação de quóruns de aceitadores, Paxos assim como Multi-Paxos podem tolerar falhas não bizantinas. Contanto que um quórum de aceitadores possa ser formado, Paxos e Multi-Paxos não param.

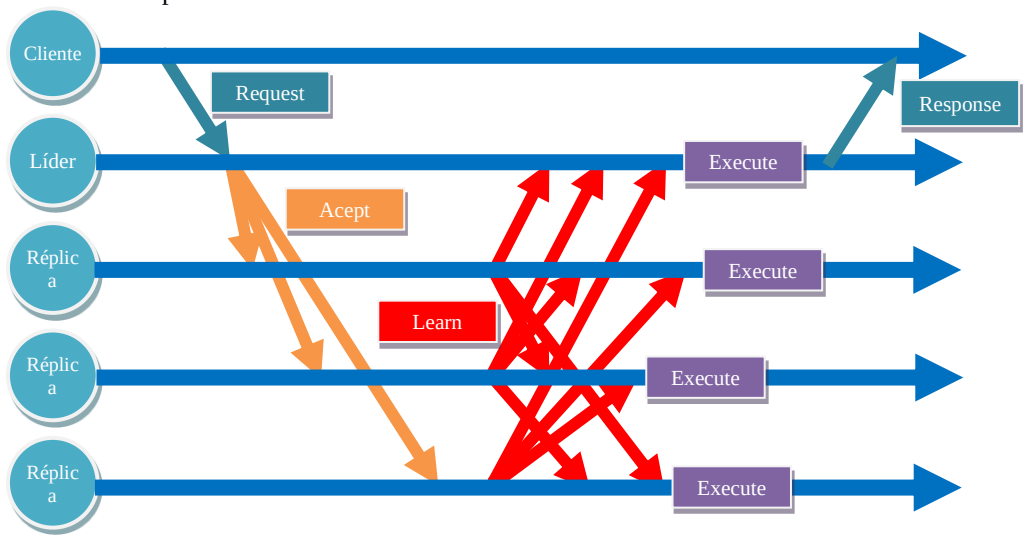

Figura 2. Multi-paxos: caso base sem falhas

2.2. Variante 1Paxos

O 1Paxos, uma variante Multi-Paxos adaptada para uma máquina multicore, foi proposta por Rachid Guerraoui[4]. O objetivo do 1Paxos é reduzir ao mínimo o número de mensagens por convênio. O conceito-chave do 1Paxos é o número reduzido de aceitadores. Se houver apenas um único aceitador, o número de mensagens do proponente para os aceitadores é reduzido. Além disso, como apenas um aceitador recebe uma aceitação, apenas uma única transmissão para os alunos é realizada. No 1Paxos os tipos de mensagens têm o mesmo significado que no Paxos.

2.3. Caso Livre de Falhas

1Paxos escolhe o líder, semelhante ao Multi-Paxos, por meio de preparar mensagens de resposta. Um proponente que deseja se tornar líder envia uma mensagem de preparação para o aceitador ativo. Se o número da proposta da mensagem de preparação for maior do que todos os números de propostas anteriores, o aceitante responde com uma mensagem de resposta de preparação. Supondo que o líder e o aceitador sejam eleitos, o líder pode enviar uma mensagem de aceitação ao aceitador. O aceitador decide se aceita a proposta e transmite o valor aceito para os alunos, dependendo do número da proposta. O caso livre de falhas com um líder já escolhido é mostrado na figura 3.

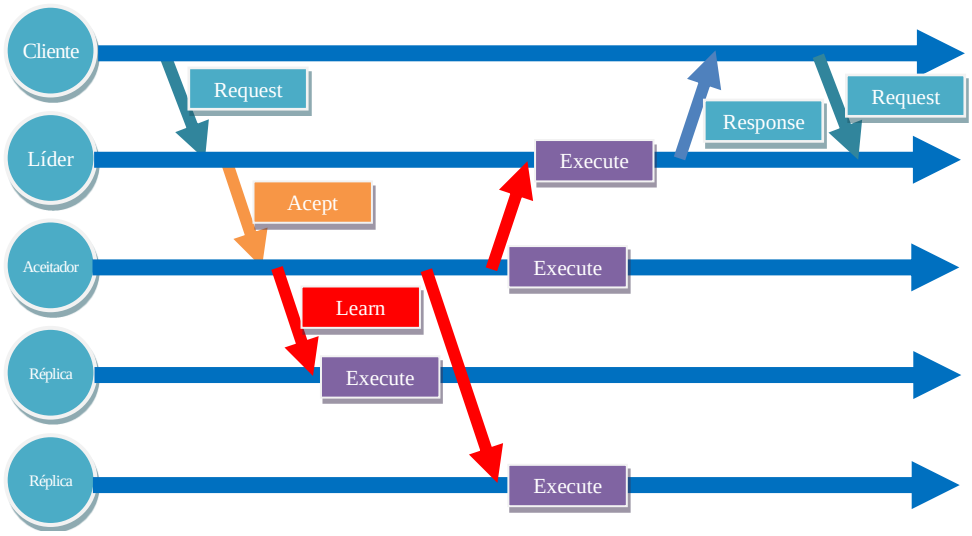

Figura 3. 1Paxos: caso base sem falhas

2.4. Falha do Aceitador

O líder é a única réplica autorizada a alterar o aceitador ativo. A mudança de um aceitador para outro deve ser confirmada pela maioria das réplicas para evitar mais de um par aceitador/líder no sistema. O acordo sobre um novo aceitante é alcançado por um protocolo de consenso de utilidade. O líder primeiro garante que ainda é o líder solicitando a todas as réplicas que enviem a ele o id do líder. Se o líder tiver recebido a maioria das respostas com seu id, ele anuncia a mudança de aceitador para as outras réplicas. Em seguida, o líder envia uma mensagem de preparação para o novo aceitante. A mensagem de preparação contém valores propostos não confirmados para cobrir o caso se o aceitador recebeu uma mensagem de aceitação, mas não transmitiu a mensagem de aprendizado antes de falhar.

2.5. Falha do Líder

Trocar o líder é semelhante a uma falha do aceitador. Uma réplica pode tentar se tornar líder se suspeitar que o líder não responde mais. Uma réplica primeiro adquire o id do aceitador solicitando o id de todas as réplicas. Se a réplica receber a maioria das respostas com o mesmo id, ela pode prosseguir e anunciar a mudança de liderança. O novo líder requer uma maioria de réplicas para evitar uma mudança de aceitador durante a eleição do líder. Após a eleição, o novo líder anuncia sua liderança a todas as réplicas. Se um líder antigo recebeu uma mensagem de mudança de liderança, ele assume que foi substituído como líder.

2.6. Falha Tanto do Líder Quanto do Aceitante

Se o líder e o aceitador falharem, o protocolo para até que o líder ou o aceitador responda novamente. A decisão de lidar com esse caso esperando é baseada na probabilidade. Assumindo que a probabilidade de duas réplicas falharem ao

mesmo tempo já é pequena, a probabilidade de exatamente o líder e o aceitador falharem é ainda menor.

2.7. Aplicações

O algoritmo Paxos é usado em:

- Gerenciador de lock/Servidor de nomes do Google Chubby[5]
- Apache Zookeeper[6]
- Transações em Cassandra
- Google Spanner, Megastore
- Serviço de gerenciamento de cluster da Microsoft Autopilot usado no motor de busca do Bing
- Controlador VMware NSX
- Amazon Web Services, DynamoDB

3. CONCLUSÕES

O Paxos é um algoritmo projetado para solucionar o problema do consenso em um sistema distribuído, um desafio importante em ciência da computação. O desafio é garantir que, mesmo que ocorram falhas em alguns dos processos, os processos restantes possam chegar a uma decisão em conjunto de uma maneira eficiente e confiável. Antes dele, haviam outras soluções propostas para o problema, mas nenhuma era tão eficiente e elegante quanto.

Ele foi originalmente desenvolvido para ser usado em sistemas distribuídos de alta disponibilidade, como sistemas de telecomunicações e sistemas bancários. No entanto, se mostrou útil em muitos outros contextos, como sistemas de gerenciamento de arquivos distribuídos e sistemas de processamento distribuído de dados.

É graças ao Paxos que podemos ter sistemas distribuídos que funcionam corretamente, evitando inconsistências entre as réplicas, mesmo que ocorram falhas em alguns dos processos.

4. REFERÊNCIAS BIBLIOGRÁFICAS

[1] LAMPORT, LESLIE. "The Part-Time Parliament." ACM Transactions on Computer Systems 16.2 (1998): 133-169.
[2] LAMPORT, LESLIE. "Paxos Made Simple." ACM SIGACT News 32.4 (2001): 18-25.
[3] LAMPORT, LESLIE. "Paxos Made Practical." ACM SIGACT News 35.4 (2004): 59-60.
[4] Guerraoui, Rachid; David, Tudor Alexandru; Yabandeh, Maysam, Consensus Inside. 2014. Disponível em <https://infoscience.epfl.ch/record/201600> Acesso em: 17 maio 2023
[5] COULOURIS, GEORGE; et al., Sistemas Distribuídos – Conceitos e Projeto. 5.ed. Porto Alegre: Bookman, 2013. 1048p.
[6] ONGARO, DIEGO, and JOHN OUSTERHOUT. "In search of an understandable consensus algorithm (extended version)." USENIX Annual Technical Conference (2014).