SEMICT IFMS 2019
Seminário de Iniciação Científica e Tecnológica
25 e 26 de julho de 2019

IMPLEMENTAÇÃO PARALELA DA PROVA DE TRABALHO DO BLOCKCHAIN EM PYTHON

Thiago Martins Silva[1], Rafael Freitas Schmid[1]

[1]Instituto de Educação, Ciência e Tecnologia de Mato do Sul – Aquidauana IF-MS

thiagoobleyk@gmail.com, rafael.schmid@ifms.edu.br

Resumo

Os mineradores são de suma importância na Blockchain. Através da prova de trabalho, eles garantem o funcionamento da rede de forma segura. Entretanto, realizar essa prova não é uma tarefa fácil, pois o número de verificações cresce de forma exponencial à dificuldade definida pela rede. Neste trabalho, comparamos os tempos de execução de uma implementação sequencial com uma paralela da prova de trabalho. Os resultados mostraram que a versão paralela, em um processador com 4 núcleos, pode ser até 4 vezes mais rápida que a versão sequencial.

Palavras-chave: Blockchain, Mineradores, Prova de Trabalho.

Introdução

O Bitcoin surgiu em meados de 2008 no artigo *Bitcoin: A Peer-to-Peer Electronic Cash System*, de Satoshi Nakamoto (NAKAMOTO, 2008), essa moeda digital originou a Blockchain. Podemos resumir a Blockchain como uma "estrutura de dados que armazena transações de forma ordenada e ligada ao bloco anterior, servindo como um sistema de registros distribuído" (CHICARINO, 2017).

Os responsáveis por atualizarem essa estrutura são os mineradores, por meio de uma prova, conhecida como prova de trabalho que, dentre outras palavras, "é um desafio criptográfico utilizado para garantir que um nó realizou uma certa quantidade de trabalho" (PIRES, 2017, p.30). Já a mineração é o processo pelo qual os mineradores incluem as transações em um bloco e geram um cabeçalho (que é obtido através da prova de trabalho) válido para ele (CHICARINO, 2017).

Qualquer pessoa pode se tornar um minerador de Bitcoins, o que acaba sendo um negócio muito competitivo (KARASINSKI, 2014). Sendo assim, existe o interesse de otimizar a prova de trabalho, seja através de hardwares especializados, ou softwares. Neste trabalho, efetuamos a implementação da prova de trabalho de forma paralela e comparamos com os tempos da versão sequencial.

Metodologia e desenvolvimento

A prova de trabalho consiste em encontrar um nonce que, concatenado ao bloco, gera um hash que inicie com o número de zeros da dificuldade estabelecida pela rede. Para essa tarefa foram implementadas duas versões do algoritmo: uma sequência e outra paralela. A versão sequencial incrementa o nonce sequencialmente a cada iteração, verificando se ele satisfaz os requisitos da prova de trabalho. Enquanto isso, a versão paralela divide essa tarefa em 8 processos independentes, que fazem essa verificação em paralelo.

Na Figura 1 é apresentado o algoritmo da prova de trabalho sequencial. O nonce é inicializado em 0 e incrementado em um a cada iteração do laço. A condição VERIFICA_HASH faz a verificação se o hash encontrado satisfaz o número de zeros definido pela dificuldade.

Figura 1. Algoritmo Sequencial da Prova de Trabalho.

```
1: procedure SEQUENCIAL(bloco, dificuldade)
2:     nonce ← 0
3:     while true do
4:         hash ← HASH256(bloco + nonce)
5:         if VERIFICA_HASH(hash, dificuldade) then
6:             Break
7:         nonce ← nonce + 1
```

A Figura 2 apresenta a versão paralela do algoritmo. Nesta versão, são chamados 8 processos independentes (de 0 a 7) que farão as verificações em *nonces* diferentes e de forma independente. Cada um dos processos verificam 200 nonces sequenciais e depois pulam 1600 posições para fazer as próximas 200 verificações. Por exemplo, o processo 0 verifica de 0 a 199 e o processo 1 examina de 200 a 399, depois disso ambos pulam para as posições 1600 e 1800 respectivamente, e assim todos os demais processos.

Figura 2. Algoritmo Paralelo da Prova de Trabalho.

```
1: procedure PARALELA(bloco, dificuldade)
2:     nonce ← 0
3:     for processo = 0, 7 do
4:         PROVA_DE_TRABALHO(bloco, dificuldade, nonce)
5:         nonce ← nonce + 200
6: procedure PROVA_DE_TRABALHO(bloco, dificuldade, nonce)
7:     while true do
8:         novo_nonce ← nonce + 200
9:         while nonce < novo_nonce do
10:            hash ← hash256(bloco + nonce)
11:            if VERIFICA_HASH(hash, dificuldade) then
12:                Break
13:            nonce ← nonce + 1
14:        nonce ← nonce + 1400
```

Resultados Experimentais

Os testes foram executados em uma máquina com processador Intel(R) Xeon(R), CPU E5-1620 v3 @ 3.50GHz, 32 GB de memória RAM e usando a versão 3.5 do Python. Tais testes foram realizados em 3 blocos

distintos iniciando da dificuldade 4, já que uma dificuldade menor pode ser realizado muito rápido pelo processador. Nosso algoritmo considera os números como hexadecimais, o que implica que a dificuldade 1 exige um hash que inicie com 4 bits em zero, a dificuldade 4 seriam necessários 16 bits em 0 (4 bits por número hexadecimal x 4 da dificuldade), e assim por diante.

As Tabelas 1 e 2 apresentam os tempos de execução da prova de trabalho de cada um dos 3 blocos das versões sequencial e paralela, respectivamente. Para cada uma das dificuldades foram realizadas 10 execuções e a média foi considerada. Essas tabelas nos mostram que o tempo de um bloco para outro pode variar bastante, mesmo que consideremos a mesma dificuldade. Por exemplo, na tabela 1 o tempo de minerar o bloco 2 com dificuldade 5 é muito mais rápido que o bloco 1, porém quando verificamos o tempo para minerar os mesmos blocos com dificuldade 6, o tempo do primeiro bloco é muito melhor.

Tabela 1. Tempos da Execução Sequencial.

	DIFICULDADES				
BLOCOS	**4**	**5**	**6**	**7**	**8**
1	0,01	2,52	2,51	167,77	274,99
2	0,06	0,05	39,42	111,89	258,45
3	0,13	1,78	13,67	625,04	3.583,19

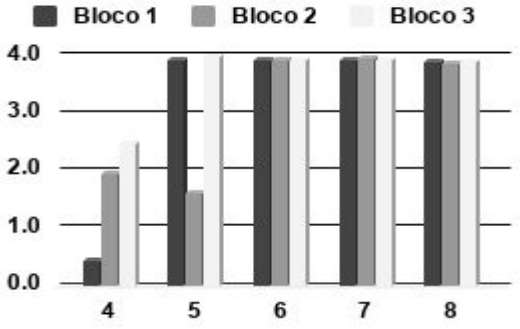
Tabela 2. Tempos da Execução Paralela.

	DIFICULDADES				
BLOCOS	**4**	**5**	**6**	**7**	**8**
1	0,02	0,64	0,64	42,64	70,77
2	0,03	0,03	10,04	28,33	66,98
3	0,05	0,45	3,48	159,07	918,87

Para entender melhor as diferenças de desempenho entres os códigos foi criada a Figura 3. Nela é apresentado o gráfico do speedup da versão paralela em relação à versão sequencial. Speedup é uma medida de desempenho onde se divide o tempo de execução antes da otimização (sequencial) pelo tempo de execução após a otimização (paralelo), caso o número seja maior que 1, a versão otimizada obteve ganhos em relação à outra versão. Nesta figura é possível ver que em apenas 1 caso, com dificuldade 4, a versão sequencial foi melhor que a paralela, isso compreende que para dificuldades menores, o sequencial tende a ser melhor que o paralelo.

A versão paralela nas dificuldades 5, 6, 7 e 8 foi quase 4 vezes melhor que o sequencial, com a única exceção do bloco 2 com dificuldade 5, que foi apenas 1,6 vezes melhor. Ao analisar esse caso, identificamos que o bloco 2 consegue realizar a prova de trabalho para as dificuldades 4 e 5 que coincidentemente é o mesmo nonce. Como trata-se de um valor pequeno de nonce, apenas 34724, a versão sequencial acaba tendo uma vantagem, apesar de ainda se sair pior nesse caso.

Figura 3. *Speedup* da execução paralela em relação à execução sequencial.



Conclusão

Nossos resultados mostraram que a paralelização da prova de trabalho obteve ganho de 4 vezes em relação ao sequencial, número proporcional aos 4 núcleos de processamento da máquina. Considerando que GPUs possuem centenas de núcleos de processamento, como trabalho futuro pretendemos implementar uma versão paralela em placa de vídeo desse problema.

Agradecimentos

Primeiro lugar ao meu bom DEUS onipresente, onisciente e onipotente; a toda a minha família; a toda a minha equipe especialmente meus orientadores e ao IFMS pelo o apoio às pesquisas.

Referências

NAKAMOTO, Satoshi et al. Bitcoin: A peer-to-peer electronic cash system. 2008.

CHICARINO, V. R. et al. Uso de blockchain para privacidade e segurança em internet das coisas. **Livro de Minicursos do VII Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais. Brasília: SBC**, 2017.

PIRES, T. P. *Tecnologia Blockchain e suas aplicações para provimento de transparência em transações eletrônicas*. 2016. 57f. Bacharelado em Engenharia de Redes de Comunicação - Universidade de Brasília, Brasília 2017.

KARASINSKI**,** Vinicius. **Tecmundo Explica: como funcionam as Bitcoins?**. Tecmundo, 18 mar 2014. Disponível em:<https://www.tecmundo.com.br/bitcoin/52445-tecmundo-explica-como-funcionam-as-bitcoins-video-.htm> Acesso em: 14 jun. 2019 .