Submission date: 02/07/2020
Resubmission date: 28/08/2020
Camera ready submission date: 11/09/2020

1st round notification: 18/08/2020
2nd round notification: 29/08/2020
Available online: 09/10/2020
Publication date: 14/05/2021

Section: regular article

Uma Avaliação de Desempenho de Soluções *Off-chain* baseadas em Sistemas de Armazenamento Distribuído

A Performance Evaluation of Off-Chain Solutions Based on Distributed Storage Systems

Filipe Herculano Rocha[1], **Allysson Allex Araújo**[2], **Pamella Soares**[1], **Raphael Lima Saraiva** [1] **e Jerffeson Teixeira de Souza**[1]

[1]Programa de Pós-graduação em Ciência da Computação – Universidade Estadual do Ceará (UECE)
Fortaleza, Ceará – Brasil

[2]Grupo de Estudos em Sistemas de Informação e Inovação Digital (GESID) – Universidade Federal do Ceará (UFC)
Cratéus, Ceará – Brasil

filipeherculanorocha@gmail.com, allysson.araujo@crateus.ufc.br, {pamella.soares, raphael.saraiva}@aluno.uece.br, jeff@larces.uece.br

Abstract. *Off-chain strategies based on distributed storage systems have been approached towards overcoming blockchain's scalability issues. Despite both academic and market relevance, we identified a research gap regarding the performance comparison between the well-known IPFS, Sia, and Swarm systems. Given this opportunity, we carried out a pioneer experimental evaluation aiming to contribute to the aggregation of information that could support the decision of adopting off-chain solutions. We concluded that each system stood out in different ways, depending on the project's particularities and the evaluated metrics.*

Keywords. *Blockchain; Off-chain Strategies; Distributed Storage System; Smart Contracts; dApps; Experimental Evaluation*

Resumo. *Estratégias off-chain baseadas em sistemas de armazenamento distribuídos têm sido adotadas para contornar o desafio da escalabilidade de armazenamento em blockchain. Apesar da relevância acadêmica e mercadológica, identifica-se uma lacuna quanto a realização de análises empíricas comparando o desempenho dos sistemas com mais visibilidade nesse contexto, isto é, o IPFS, Sia e Swarm. Diante dessa oportunidade, este trabalho realizou, de forma pioneira, uma avaliação experimental visando contribuir para a agregação de informações que apoiem a tomada de decisão sobre tais soluções off-chain. Concluiu-se que cada sistema avaliado destacou-se de diferentes formas dependendo das particularidades do projeto e das métricas avaliadas.*

Palavras-Chave. *Blockchain; Estratégias Off-chain; Sistema de Armazenamento Distribuído; Contratos Inteligentes; dApps; Avaliação Experimental*

Cite as: Rocha, F. H., Araújo, A. A., Soares, P., Saraiva, R. & Souza, J. (2021). A Performance Evaluation of Off-Chain Solutions Based on Distributed Storage Systems (Uma Avaliação de Desempenho de Soluções *Off-chain* baseadas em Sistemas de Armazenamento Distribuído). iSys: Revista Brasileira de Sistemas de Informação (Brazilian Journal of Information Systems), 14(1), 4-23. DOI: 10.5753/isys.2021.808

1. Introdução

O *blockchain*, que teve seu desenvolvimento difundido após a criação da moeda digital Bitcoin [Nakamoto 2009], constitui uma das novas infraestruturas digitais baseadas na internet com amplo potencial disruptivo [Rodrigues et al. 2018]. Em termos genéricos, o *blockchain* pode ser entendido como um banco de dados distribuído, protegido por criptografia e governado por um mecanismo de consenso [Laurence 2019], ou seja, o *blockchain* é essencialmente um registro de eventos digitais. O *blockchain* está entre as tecnologias mais promissoras da atualidade e com grandes perspectivas de inovação em diversos segmentos da sociedade [Tapscott and Tapscott 2016].

De acordo com Iansiti e Lakhani (2017), o uso de *blockchain* pode reduzir drasticamente o custo de transações e, se amplamente adotado, pode remodelar a economia. Tal perspectiva se deve à ampla extensão de finalidades que o *blockchain* pode impactar [Holgate et al. 2017]. Vislumbrando tamanho potencial, Beck *et al.* (2017) destacam atualmente como um ótimo momento para o desenvolvimento de pesquisas na área de Sistemas de Informação sobre as implicações e possibilidades desta tecnologia inovadora.

Nesse sentido, após a popularização do *Bitcoin* [da Silva Rodrigues 2017], diversas outras soluções descentralizadas se disseminaram, agora adotando redes de *blockchain* genéricas. A de maior visibilidade até este momento é a rede *Ethereum* [Chinchilla 2014], lançada em 2015, com cerca de 94 milhões de usuários atualmente [Etherscan 2020]. *Blockchains* de caráter genérico não somente se restringem ao funcionamento de criptomoedas, elas podem conter outras informações em suas transações. No caso, a *Ethereum* possui a capacidade de construir contratos inteligentes (do inglês, *smart contracts*) utilizando a linguagem *Solidity* ou *Vyper*. Contratos inteligentes produzem programas que mudam o estado de um computador global (referenciado como *Ethereum Virtual Machine* ou EVM), onde qualquer usuário pode criá-los ou executá-los.

Ademais, aumentar a disponibilidade e adotar infraestruturas escaláveis tornaram-se elementos fundamentais no desenvolvimento e manutenção de sistemas de informação modernos [Xie et al. 2019, Jiménez-Peris et al. 2002]. Nesse contexto, um desafio crítico atualmente na adoção de *blockchains* é o armazenamento necessário para se manter um *full node*. Um *full node* é uma entidade na rede distribuída cujo objetivo é, entre outros, manter dados de cada uma das mudanças de estado da EVM, ou seja, manter todas informações de transações em cada um dos blocos em sua mais pura forma, sem otimizações usando *Merkle Trees* [Project 2020].

Em redes mais simples, a única informação de usuários que poderia ser armazenada nos blocos eram as transações monetárias que, em comparação, consomem menos recursos. Com a introdução de contratos inteligentes, outros tipos de dados como arquivos de mídia podem ser armazenados, visto que a linguagem *Solidity* permite o uso de um tipo de dado primitivo em sua forma mais básica, o 31 *bytes* [Ethereum 2020]. Isso possibilita que diferentes casos de uso possam ser implementados como, por exemplo, quando uma entidade governamental deseja armazenar seus documentos legais de gastos públicos de forma acessível e infraudável. Tal desafio reflete-se sempre que um novo *full node* entra pela primeira vez em uma rede, demandando, assim, a sincronização com seu estado atual. O novo membro requisita de um outro *full node*, este já ativo, os dados necessários para

que possa se tornar um *full node*. Isso pode levar dias ou até semanas [Szilágyi 2019], dependendo da capacidade da máquina [Project 2020].

Portanto, constata-se que o processo de desenvolvimento de projetos baseado em *blockchain* utilizando contratos inteligentes pode demandar estratégias que lidem com o aumento na escalabilidade de armazenamento. Para essa demanda, hoje existem algumas alternativas que promovem o armazenamento de dados *off-chain* a partir de sistemas de armazenamento distribuído, isto é, ao invés de armazenar os dados na própria *blockchain*, os dados são alocados em serviços descentralizados de armazenamento, onde somente suas referências ficarão nos blocos. Atualmente existem várias soluções com tal propósito, dentre as quais pode-se destacar com grande visibilidade o IPFS, Sia e Swarm [Benisi et al. 2020]. No entanto, identifica-se na literatura uma lacuna concernente a análise de desempenho entre esses sistemas, sendo encontradas apenas análises de cunho bibliográfico.

Diante da oportunidade contextualizada previamente, e considerando a relevância acadêmica e profissional do tema em questão, este trabalho tem como objetivo principal realizar um experimento computacional com o propósito de comparar e analisar quantitativamente o desempenho das soluções IPFS, Sia e Swarm. Através do presente estudo empírico, contribui-se de forma pioneira para a consolidação de informações que subsidiem a tomada de decisão sobre a adoção de soluções *off-chain*.

2. Fundamentação Teórica

Nesta seção, serão apresentados conceitos fundamentais sobre *Blockchain*, Contratos Inteligentes, Aplicações Descentralizadas e Sistemas de Armazenamento Distribuído. Tais conceitos serão importantes para contextualização prévia do estudo empírico.

2.1. Blockchain

Uma *blockchain* é uma estrutura de dados distribuída *append-only*, o que a faz ser uma lista crescente de transações na qual é possível apenas realizar a inserção de registros. Esses registros são replicados e mantidos entre os membros de uma rede. Semelhante a um livro contábil distribuído, a *blockchain* dispõe de alguns recursos importantes que a faz uma tecnologia tão proeminente nos dias atuais, como a imutabilidade e não-repudiabilidade [Jiang et al. 2018]. Em outras palavras, os registros digitais gravados em uma *blockchain* são resistentes à violações e não possuem uma autoridade central, sendo eles mantidos de maneira distribuída em cada nó participante da rede [Yaga et al. 2019].

A estrutura da *blockchain* é formada por uma cadeia de blocos cujo encadeamento dos blocos é feito quando o *hash* do bloco anterior é adicionado ao conteúdo do bloco atual, como mostrado na Figura 1. Este *hash* é uma impressão digital desses dados e bloqueia os blocos em ordem cronológica [Laurence 2019]. Tal característica faz com que não haja a possibilidade de que uma transação seja alterada com antecedência sem alterar seu bloco e todos os blocos posteriores [Aitzhan and Svetinovic 2018]. O bloco é formado por um conjunto de transações organizadas em uma estrutura de dados definida como *Merkle Tree*. Nessa estrutura, as transações são agrupadas em pares e o *hash* de cada das uma transações são armazenadas em um nó pai e, assim, sucessivamente até o

nó raiz da *Merkle Tree* [Narayanan et al. 2016]. Por sua vez, o apontador *hash* da *Merkle Tree*, composta pelas transações, é armazenado no cabeçalho do bloco.

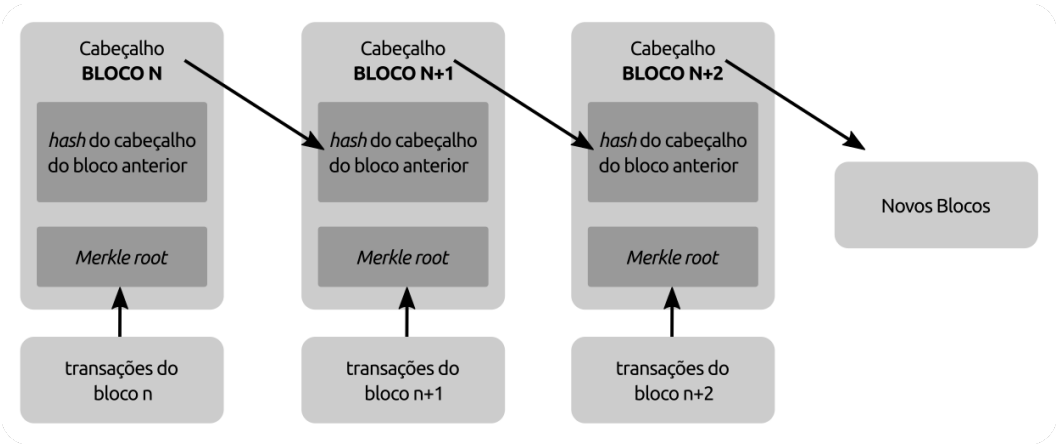


Figura 1. Representação simplificada do encadeamento dos blocos.

Por ser uma rede distribuída, o comportamento da *blockchain* é controlado através do voto coletivo de nós anônimos [Aitzhan and Svetinovic 2018]. Contudo, os nós individuais podem travar ou comportar-se maliciosamente, por exemplo. Por esse motivo, a execução de um protocolo de consenso tolerante a falhas como parte do funcionamento das *blockchains* pode garantir que todos os nós concordem com a ordem em que as entradas são anexadas à blockchain [Cachin and Vukolić 2017]. O conceito de "consenso" é relacionado ao problema de sincronização de estados em sistemas distribuídos, com o intuito de que diferentes participantes da rede podem concordar com um em mesmo estado. A falta de uma entidade central para fazer estas decisões é uma das qualidades de redes *blockchain*, pois isso favorece a não censura e a independência de autoridades que definem as permissões de acesso a dados. De forma resumida, a Figura 2 mostra as etapas gerais do funcionamento de uma *blockchain*.

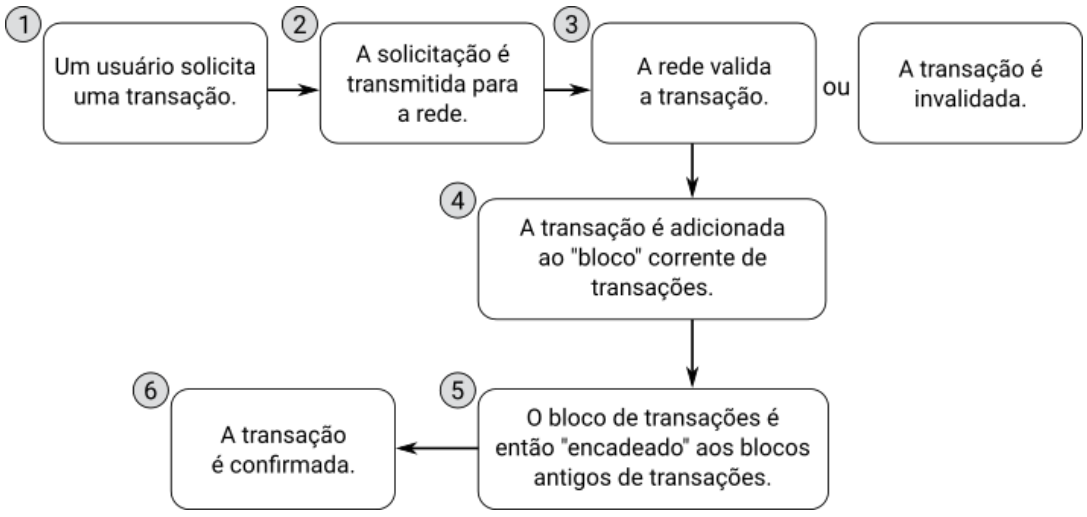


Figura 2. Funcionamento de uma blockchain. Adaptado de [Laurence 2019].

Cada *blockchain* pode operar com um ou mais tipos de algoritmos de consenso, os quais podem ser determinados pela ameaça esperada e o grau de confiança que a rede possui nos nós que participam da *blockchain*. No caso das redes Bitcoin e Ethereum, utiliza-se atualmente o algoritmo de consenso denominado *Proof-of-Work* (PoW) [Laurence 2019]. Em suma, tal procolo de consenso funciona da seguinte maneira: como cada bloco tem um de seus campos presentes denominado de `nonce`, o nó minerador, através de um algoritmo de força bruta, testa diversos números sucessivamente até que o bloco seja considerado correto. Para que um bloco seja aceito, o nó minerador deve computar o valor da *hash* do bloco inteiro, onde o valor resultante deve ser menor que a dificuldade imposta pela rede naquele período. O principal ganho da PoW é que não existe uma estratégia mais eficiente de achar um valor para o `nonce` que não seja através de enumerar todas as possibilidades e que sua checagem é trivial e barata, ou seja, torna-se difícil forjar blocos e menos complexo checar a corretude deles.

Vale destacar que a Ethereum possui uma prova de conceito mais recente que utiliza *Proof-of-Stake* (PoS), implementado no novo protocolo chamado Casper que será disponibilizado na versão *Ethereum* 2.0 [Ethereum 2020], porém como ainda se encontra em estado de desenvolvimento não será abordado neste trabalho.

2.1.1. Contratos Inteligentes e Aplicações Descentralizadas

Como já mencionado, a tecnologia *blockchain* tem passado por diferentes estágios desde sua concepção. A rede Ethereum é uma segunda evolução do conceito de *blockchain*. Esta tecnologia introduziu uma linguagem de programação e a construiu dentro da estrutura tradicional de *blockchain* [Laurence 2019]. Por meio de *scripts* inseridos na *blockchain*, foi possível contemplar a transferência de muitos outros tipos de ativos, e outros mercados além de criptomoedas. Tais *scripts* são denominados contratos inteligentes (do inglês, *smart contracts*). Em outras palavras, contratos inteligentes são programas que executam códigos armazenados na própria *blockchain*, são imutáveis e determinísticos. Esse termo já tinha sido usado anteriormente na década de 90 quando o estudioso Nick Szabo cunhou o termo [Szabo 1996] e definiu-o como um conjunto de promessas especificadas em forma digital, incluindo protocolos nos quais podem-se interagir com outras promessas [Chinchilla 2014].

Para que contratos possam ser usados, eles primeiro precisam ser compilados em uma linguagem de baixo nível que irá executar na *Ethereum Virtual Machine* (EVM). A EVM é definida como um computador global, pois cada nó da rede executa as suas mudanças de estado acarretadas pelo último bloco minerado, produzindo transações monetárias assim como executando métodos definidos em contratos inteligentes. Essa máquina virtual se torna global, visto que o estado do mesmo permanece igual em todos os nós da rede, não tendo assim uma máquina centralizada, mas sim distribuída. Sempre que um novo bloco é minerado, todos os nós da rede computam as mudanças de estados presentes nele e assim acabam convergindo novamente para o mesmo estado. Uma vez compilado, o código é disponibilizado para o *Ethereum* através de uma transação especial de construção de contratos. Cada contrato é definido por um endereço, eles não possuem dono e são executados autonomamente. O endereço do contrato pode ser utilizado como

recipiente de transações, podendo ser enviado Ether ao contrato ou chamando uma de suas funções já definidas e compiladas. A passagem de parâmetros para métodos de contratos é feita através da passagem destes dados em um campo especial da transação chamado `data`, de acordo com [Antonopoulos and Wood 2018]. Dessa maneira, usuários ou contratos podem executar métodos de outros contratos colocando o endereço e `data` apropriados em uma transação.

Os contratos inteligentes possibilitaram a descentralização do controle lógico e das funções de pagamentos das aplicações [Antonopoulos and Wood 2018] visto que a maioria dos sistemas anteriores à introdução desse conceito, eram centralizados, ou seja, controlados por uma única entidade. Assim, por meio dessa disruptiva tecnologia, se fortaleceu o desenvolvimento de Aplicações Descentralizadas (do inglês, *Decentralized Apps* ou dApps), aplicações cuja relação cliente-servidor segue um modelo totalmente descentralizado. Antonopoulos e Wood (2018) definem um dApp como sendo "um aplicativo que é principalmente e totalmente descentralizado" o qual tem essas características nos mais diversos aspectos e partes da aplicação. Raval (2016) caracteriza os dApps com as seguintes propriedades:

- Código aberto: devido à natureza confiável da blockchain, os dApps precisam tornar seus códigos de código aberto, para que possíveis auditorias de terceiros se tornem possíveis;
- Suporte interno à criptomoeda: a moeda interna é o veículo que executa o ecossistema para um aplicativo específico. Com os *tokens*, é possível para um dApp quantificar todos os créditos e transações entre os participantes do sistema, incluindo provedores de conteúdo e consumidores;
- Consenso descentralizado: o consenso entre os nós descentralizados é a base da transparência;
- Nenhum ponto central de falha: um sistema totalmente descentralizado não deve ter um ponto central de falha, pois todos os componentes dos aplicativos serão hospedados e executados no *blockchain*.

2.2. Sistemas de Armazenamento Distribuído

Para mitigar o desafio do armazenamento tem sido proposta a combinação da *blockchain* com sistemas de armazenamento existentes, podendo ser eles bancos de dados centralizados, bancos de dados NoSQL ou bancos de dados distribuídos [Pop et al. 2019]. Tal estratégia, denominada *off-chain*, possibilita a terceirização do armazenamento em banco de dados fora da *blockchain* sem sacrificar características fundamentais, como a imutabilidade e disponibilidade de dados [Shukla and Samet 2020]. Dentre as estratégias *off-chain* pode-se destacar o uso de Sistemas de Arquivos Distribuídos [Benet 2014], como *Inter Planetary File System* (IPFS), Sia e Swarm os quais serão introduzidos a seguir.

O **IPFS** é um sistema distribuído para acesso e armazenamento de arquivos, sites, aplicações e dados em geral potencializado por sua comunidade [Benet 2014]. Na prática, o IPFS solicita a diversos computadores da rede compartilharem os dados de uma determinada página, tornando qualquer usuário um provedor de dados. Cada dado que usa o protocolo IPFS possui um *Content Identifier* (CID), que é o resultado da *hash* do dado armazenado. Este valor é específico para cada dado, mesmo que seja bem menor

que o próprio conteúdo. O IPFS faz uso da estrutura de dados *Directed Acyclic Graphs* (DAG). Um DAG é um tipo de grafo onde não é permitido ciclos e as arestas possuem direcionamento. Mais especificamente IPFS usa Merkle-DAG, que são DAGs onde cada nó possui um CID. Como cada nó no DAG possui a informação que indica quem são os seus descendentes, o valor do CID é diretamente relacionado à topologia do grafo. Sabendo disso, dois nós que possuem o mesmo CID constroem o mesmo grafo. Isto se torna uma peça chave no funcionamento do IPFS, pois com isso é possível sincronizar de forma eficiente diferentes grafos que representam a estrutura dos arquivos no protocolo. Ao criar uma representação de um arquivo o IPFS primeiro o divide em diversos blocos. Com isso, diferentes partes do arquivo podem vir de diversas fontes e também podem ser autenticados mais rapidamente. Outra funcionalidade importante é que se existem dois arquivos similares armazenados no protocolo, eles podem compartilhar partes da Merkle-DAG. Por exemplo, se um projeto de software for atualizado, somente os arquivos que tiveram mudanças terão seus endereços alterados. Isso proporciona uma maior eficiência ao transferir versões diferentes de um grande projeto.

Por sua vez, o **Sia** é um sistema de armazenamento em nuvem descentralizado assegurado por sua *blockchain* criada pela empresa Nebulous em 2014 [Vorick and Champine 2014]. O Sia faz uso de discos rígidos não utilizados, distribuídos pelo mundo, criando um mercado mais confiável e de menor custo que soluções tradicionais. A *blockchain* do Sia permite que esse mercado de hardware funcione sem um intermediário, assegurando transações de armazenamento com contratos inteligentes por meio do uso da criptomoeda Siacoin. Nenhuma pessoa ou organização pode censurar ou impedir o acesso aos dados. Os arquivos armazenados são privados e são armazenados por toda a rede, o que elimina a chance de um ponto único de falha e permite uma alta disponibilidade.

Finalmente, a **Swarm** é um serviço de armazenamento *peer-to-peer* que é resistente a *Distributed Denial of Service*, tolerante a falhas e à prova de censura, sendo posteriormente integrado a Ethereum [Hartman et al. 1999].

O Swarm está profundamente integrado com multiprotocolo da camada de rede do *Ethereum*, assim como sua *blockchain* para resolução de nomes de domínios (em outras palavras, a mesma funcionalidade do DNS) e garantia de disponibilidade de conteúdo. Seu objetivo primário é promover a descentralização e redundância do armazenamento da *blockchain* do *Ethereum*, em particular armazenando e distribuindo código e dados de apps. O Swarm tem duas principais funcionalidades que o diferenciam de outras soluções. Enquanto outros serviços permitem que se registre e compartilhe conteúdo hospedado por um servidor na rede, ele promove a hospedagem como um serviço de armazenamento em *cloud*. No Swarm, o conteúdo pode ser armazenado e posteriormente buscado, sem a necessidade de se manter conectado alimentando a rede. A segunda funcionalidade é o sistema de incentivo. Como esse sistema faz parte da *stack* do *Ethereum*, este já faz uso da própria criptomoeda para garantir a disponibilidade dos dados na rede. Até a data deste trabalho, o sistema de incentivo se encontra em desenvolvimento e ainda não está disponível para produção, isso significa que a persistência dos dados ainda não é garantida.

3. Trabalhos Relacionados

A partir da busca bibliográfica para o presente trabalho foi possível identificar uma lacuna a respeito de análises empíricas comparando diferentes sistemas de armazenamento distribuído atrelado à *blockchain*. Não foram identificados trabalhos empíricos sobre a comparação de desempenho entre esses sistemas, sendo encontrados apenas revisões de literatura abordando o tema em questão.

Casino *et al*. (2020) realizam uma análise de *blockchain* e de sistema de Armazenamento de Arquivo Descentralizado (DFS, do inglês, *Decentralised File Storage*) mais utilizados. Os autores discutiram sobre desafios e oportunidades com foco na imutabilidade e na influência sobre requisitos da *General Data Protection Regulation* (GDPR) utilizando ambas as tecnologias. Os autores apresentam um estudo sobre problemas referentes a ataques maliciosos em relação aos sistemas de *blockchain* e DFS, e enfatizam que, apesar do uso de tais tecnologias serem relevantes, faz-se necessário buscar soluções contra os possíveis usos maliciosos. O trabalho realiza uma breve apresentação de alguns DFSs existentes, como IPFS, Swarm, Storj, Maidsafe, Sia e IndImm em relação à imutabilidade. Além disso, os autores identificam quatro emergentes ameaças de vulnerabilidade que estão relacionada à dados pessoais, conteúdo jurídico, malware e atualização/descontinuação dos sistemas, e levantam estratégias advindas dos próprios sistemas que podem ser adotadas para mitigar tais problemáticas.

Por sua vez, Benisi *et al*. (2020) realizam um levantamento bibliográfico sobre sistemas de armazenamento baseados em *blockchain*. Os autores relatam os problemas de sistemas de armazenamento atuais, como o armazenamento em nuvem, por exemplo. O trabalho sugere que os sistemas de armazenamento baseados em *blockchain* podem "superar muitos problemas e deficiências dos sistemas de armazenamento tradicionais". Ademais, apresentam uma visão geral do funcionamento das tecnologias e, posteriormente, realizam comparações entre sistemas baseados em *blockchain* e de sistemas tradicionais em termos de privacidade, segurança, banda larga e custo, dentre outros critérios. Além disso, os autores denotam em quais contextos científicos esses sistemas estão sendo utilizados, assim como discutem diferentes protocolos de consenso nessas aplicações. Por fim, são apresentados os desafios considerando ainda questões de segurança, problemas na falta de tomada de decisão com base em dados, restrições legais, questões de escalabilidade, controle de acesso, dentre outros.

Parte dos trabalhos encontrados na literatura tratam especificamente dos sistemas de armazenamento como solução para problemas de escalabilidade da rede *blockchain*. Primeiramente, Eberhardt e Tai (2017) apresentam lições aprendidas sobre diferentes padrões de técnicas *off-chain* introduzidos em um conjunto de projetos experimentais de *blockchain*. Os autores denominam de *Content-Addressable Storage Pattern* o padrão que utiliza sistemas de armazenamento distribuído, e o apresenta em dado contexto. Os autores realizam uma breve discussão enfatizando que o custo de armazenamento em uma aplicação pode ser reduzido com o uso dessas técnicas, mas que cuidados em relação à exposição dos dados devem ser levados em consideração. Por fim, mencionam IPFS e Swarm como as tecnologias que podem ser utilizadas nesse tipo de padrão.

Finalmente, Xie *et al*. (2019) realizam um levantamento bibliográfico com foco

em escalabilidade nos sistemas *blockchain*. Dentre as diversas categorias levantadas, os autores apresentam soluções que envolvem o armazenamento em *blockchain*, juntamente com sistemas de armazenamento distribuído, para solucionar problemas de atraso no tempo de intervalo do bloco. Nesta categoria, o IPFS, *Distributed Hash Table* (DHT) e Bigchain são descritas e consideradas como tecnologias para solução desse problema.

4. Estudo Empírico

O estudo empírico conduzido nesta pesquisa considerou as três principais soluções de armazenamento *off-chain* disponíveis no mercado: IPFS, Sia e Swarm [Benisi et al. 2020]. Para tal investigação, realizou-se um experimento computacional baseado em um conjunto de testes de desempenho a partir de múltiplas métricas, cujas configurações e processo de coleta são descritos a seguir.

4.1. Configurações do Experimento

Para concretização dos testes de desempenho fez-se necessário i) implementar um protótipo de dApp através de contrato inteligente implantado na *blockchain* da *Ethereum* ii) e configurar um ambiente de simulação em *Nodejs* o qual coordena os nós na rede.

Em relação ao contrato inteligente, foram testados o comportamento de diversas soluções em um mesmo ambiente. O objetivo do contrato é, a princípio, ter duas operações que um usuário poderá realizar, são elas: 1) armazenar um conjunto de vetores de *bytes* e 2) retornar o último vetor de *bytes* armazenado por um usuário (um pdf com gastos semestrais, por exemplo).

```
pragma solidity 0.5.11;
contract SlowStorage {
    mapping (address => bytes[]) dataBase;
    function storeData(bytes memory rawData) public {
        dataBase[msg.sender].push(rawData);
    }
    function retrieveDataArray() public view returns (bytes memory)
        bytes memory ret = "\x00";
        if(dataBase[msg.sender].length > 0) {
            uint256 len = dataBase[msg.sender].length;
            ret = dataBase[msg.sender][len-1];
        }
        return ret;
    }
}
```

(a) Caso Base.

```
pragma solidity 0.5.11;
contract FastStorage {
    mapping (address => string[]) dataBase;
    function storeData(string memory ipfsHash) public {
        dataBase[msg.sender].push(ipfsHash);
    }
    function retrieveData() public view returns (string memory){
        string memory ret = "";
        if(dataBase[msg.sender].length > 0) {
            uint256 len = dataBase[msg.sender].length;
            ret = dataBase[msg.sender][len-1];
        }
        return ret;
    }
}
```

(b) IPFS.

Figura 3. Contratos Inteligentes para o Caso Base e IPFS.

A Figura 3a apresenta o código do **Caso Base** implementado em *Solidity*. Conforme pode-se observar, tem-se os dois métodos que serão utilizados na simulação, chamados `storeData` e `retrieveDataArray`. Sempre que um usuário tenta armazenar um vetor de *bytes*, seu endereço é buscado no `dataBase` e, logo depois, é armazenado o dado. Da mesma forma, se o usuário tenta visualizar o dado armazenado, seu endereço é usado como índice da procura. A estrutura de dados `dataBase` é usada como meio de armazenamento simples de dados, onde os valores são associados a uma chave. Essa chave, no caso, é o endereço do usuário que executou o método de armazenamento ou busca. Essa estrutura é o único dado do usuário que será armazenado no banco de dados da *blockchain* do *Ethereum*, e este tem um potencial de crescimento diretamente relacionado à quantidade de chamadas de armazenamento de dados realizadas durante a simulação.

Os códigos referentes aos **casos de teste** estudados usarão outro valor no lugar de um conjunto de *bytes* para representar o dado armazenado, enquanto o resto do contrato será idêntico ao Caso Base. O código em *Solidity* do contrato que será usado para testar a solução em IPFS está demonstrado na Figura 3b.

Quanto ao ambiente de simulação, utilizou-se a *Ganache* [Group 2019], uma *Blockchain* privada a qual possibilita a implantação de contratos inteligentes na *Ethereum* localmente. Seu principal objetivo é disponibilizar um ambiente seguro e isolado de testes, no qual é possível determinar parâmetros que ditam o funcionamento da rede. A Tabela 1 descreve os parâmetros escolhidos, incluindo a respectiva justificativa.

Parâmetro	Valores e Motivação
Tempo médio de mineração	12 segundos por bloco, baseados nas informações de mineração atuais em redes Ethereum, segundo [BitInforCharts 2019]
Limite de gás do bloco	8.000.000 unidades de gás, ou seja, a quantidade máxima de gás que pode ser gasta nas transações do bloco. Média baseada em anos anteriores [Etherscan 2019].
Quantidade de usuários	1.000 usuários, de acordo com o site [Etherscan 2020] existem cerca de 102 milhões de endereços únicos na rede Ethereum. A fim de que simulação não tenha problema de desempenho na questão de gerência desses usuários, a quantidade foi reduzida. Para que não haja rapidamente uma escassez de recursos em produzir transações, cada usuário tem um balanço com cerca de 10^{15} Ether.
Valor da unidade do gás	22 Gwei, valor médio baseado em dados de [BitInforCharts 2019].
Transações por segundo	8 transações por segundo, de acordo com o site [Etherscan 2020] até a data deste trabalho baseando-se na quantidade de transações por hora na Ethereum.
Taxa média de gás	0,0013 Ether, baseando-se de acordo com o site [Etherscan 2020], quantidade suficiente para produzir a operação requisitada pelo usuário.

Tabela 1. Parâmetros da Rede *Blockchain* na *Ganache*.

A Figura 4 sintetiza o procedimento adotado na simulação. Existem três entidades, são elas: a *blockchain* em *Ganache*, a simulação em *Nodejs* e um *script* em *Python* responsável pela geração de gráficos e dados. O estudo empírico ocorreu em outubro de 2019 e todos os códigos-fontes referentes ao contrato inteligente, a aplicação em *Nodejs* e os *scripts* de teste em *Python* encontram-se disponíveis no repositório do *Github*[1].

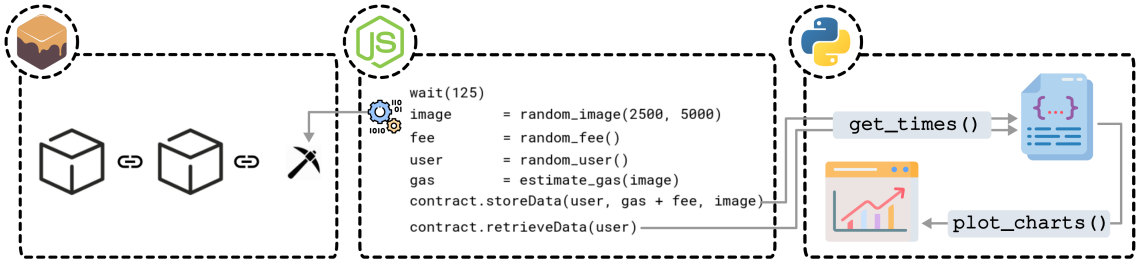


Figura 4. Visão Geral da Simulação.

Os dados utilizados na simulação poderiam ter sido buscados de um conjunto finito de imagens e estas transformadas em um vetor de *bytes* para que fossem armazenadas no contrato. No entanto, isso introduz o problema de que elas poderiam ser selecionadas mais de uma vez, favorecendo a implementação de alguns dos objetos de estudo deste trabalho. Visto que o produto final das imagens seria um vetor de *bytes*, optou-se, ao invés de converter uma imagem real, produzir um vetor de tamanho e conteúdo aleatório.

[1]https://github.com/filipeherculano/dapp

A probabilidade de colisão entre duas imagens de tamanho acima de 2,5KB é menor que 1 em 256^{2500}, produzindo assim um cenário mais verídico onde usuários armazenariam imagens distintas a todo momento. Portanto, conforme a Figura 4, pode-se observar que a simulação espera 125 milissegundos e, logo após isso, produz uma imagem aleatoriamente em um vetor de *bytes* de 2,5KB a 5KB de tamanho. Outros dois valores aleatórios são o usuário que será usado como ator nas transações e a taxa a ser aplicada em cada uma, que ficaram em média em 0,0013 Ether. Antes de realizar a operação de armazenamento, a simulação estima a quantidade exata de gás que será necessário na operação para que a taxa seja exatamente paga somente ao minerador. Durante esse processo, tem-se um *script* em *Python* que usa os dados extraídos das operações para criar os gráficos. Os testes realizados foram executados em uma máquina de 16GB de memória RAM com um processador *Intel Core* i7 da oitava geração com uma conexão de internet banda larga de 20 Mbps. Na Tabela 2 são descritos cada um dos procedimentos realizados para a coleta das métricas analisadas.

Métrica	Processo de Coleta
Facilidade de implementação	Considerou-se a quantidade de alterações realizadas no código do Caso Base em relação ao código do caso teste. Foi utilizada a ferramenta Diff[2] do Linux para medir as diferenças entre os códigos, tais medidas foram o número de inserções e de deleções entre o código base e alguma das implementações. Além disso, o número de dependências que precisaram ser instaladas no sistema foram contabilizadas.
Estresse	Coletado a partir de um script em Python que executa 1.000 vezes o mesmo procedimento. Esse procedimento cria um conjunto de bytes de forma randômica e de tamanho fixo armazenado na ferramenta em teste. Os primeiros 1000 conjuntos foram de 1 byte, totalizando 1KB de armazenamento. Depois, mais 1000 conjuntos de 10 bytes, e assim sucessivamente até chegar a 1000 conjuntos de 1.000.000.000 bytes, totalizando um armazenamento de 1TB. Em cada teste, foi medido o tempo que levou para um dado ser armazenado a fim de determinar se a rede teria algum tipo de dificuldade em armazenar os dados de diferentes tamanhos.
Custo de execução	O custo de execução foi calculado para o Caso Base e caso testes, e convertido para a moeda em Reais. O custo necessário para rodar cada uma das soluções em Ether foi somado e convertido para Reais, assim como o custo adicional que cada um dos softwares pode ou não adicionar. O teste no \DH app foi utilizado para a extração e cálculo desses dados.
Consumo de memória por bloco	Por conta que a memória da blockchain aumenta à medida que novos blocos são minerados, foi calculado o consumo total de memória ao fim do teste usando o \DH app assim como a quantidade de blocos minerados nesse período, ao final, obteve-se a média de gasto de memória por bloco.
Número de transações por bloco	Foi contabilizado no teste do \DH app o número de transações contidas em cada bloco minerado, ao fim produzindo uma média.
Tempo de publicação	O tempo de publicação foi mensurado pela diferença no tempo entre a requisição de armazenamento e quando a transação é finalmente minerada em um bloco, sabendo que entre esses tempos existe o período de espera previsto pelo uso de alguma das soluções. Foi feito então uma média em cima de todos os valores das transações.
Tempo de busca	O tempo que levou para se fazer uma busca do dado na blockchain foi calculado, assim como a busca em um dos softwares.

Tabela 2. Procedimentos de Coleta das Métricas.

5. Resultados e Análises

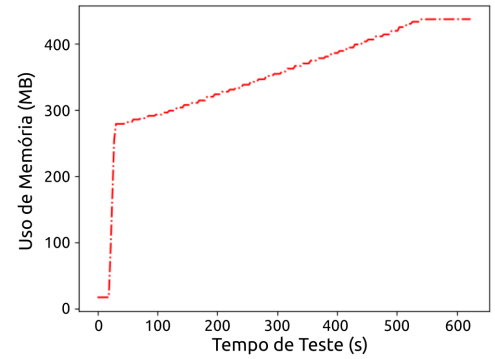
Nesta seção serão apresentados os resultados dos testes de desempenho a partir das múltiplas métricas coletadas, e suas respectivas análises em relação ao Caso Base, IPFS, Sia e Swarm.

5.1. Caso Base

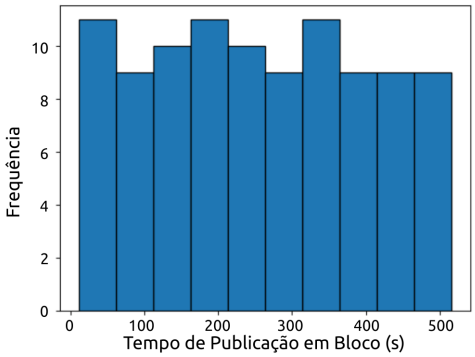
Objetivando criar um contraponto ao não uso dos sistemas abordados neste trabalho, adaptou-se o dApp para que execute testes armazenando os dados diretamente na *blockchain*. Devido à rapidez em que a *blockchain* cresce e o tempo necessário ao executar cada

[2]https://man7.org/linux/man-pages/man1/diff.1.html

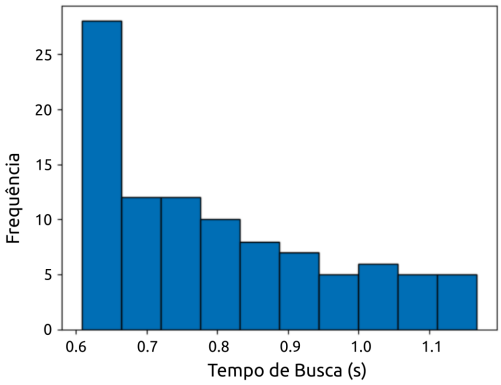
experimento, optou-se por executar somente 25% da carga utilizada nos demais serviços. Essa decisão não afetará os tempos médios comparados ao fim da análise.



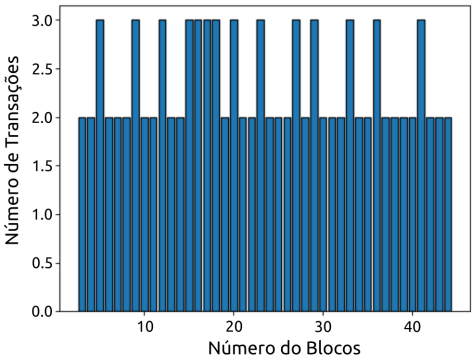
(a) Consumo de memória da *blockchain*.



(b) Tempo de mineração de transações.



(c) Tempo de busca de dados.



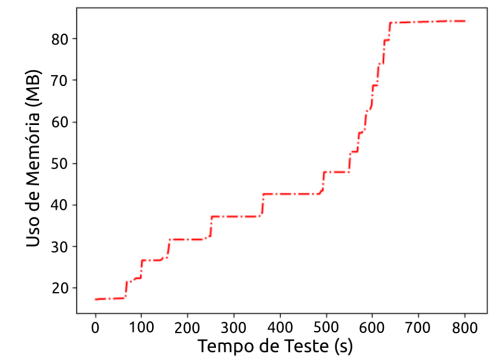
(d) Número de transações por bloco.

Figura 5. Resultados dos Testes de Desempenho no Caso Base.

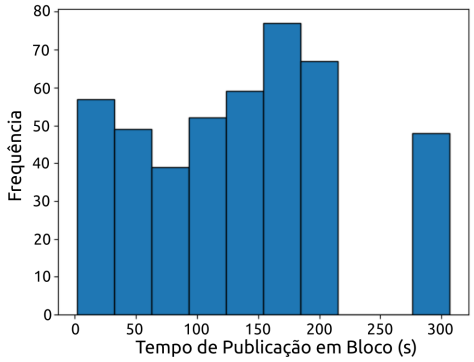
Inicialmente, ao verificar a Figura 5a ,pode-se perceber que o consumo de memória da *blockchain* cresce rapidamente, chegando ao máximo de 474MB executando somente 100 transações com uma média de consumo de memória de 11,28MB por bloco. Por sua vez, ao analisar a Figura 5b, constata-se um intervalo de tempo decorrido entre a requisição da operação de armazenamento e a publicação desta transação em um bloco. Os tempos vão até cerca de 550 segundos, tendo em média 257,92 segundos para a publicação de cada transação. Através da Figura 5c pode-se perceber que o tempo de busca é bem baixo na maioria dos testes, tendo um máximo de 1,4 segundos e uma média de 0,8 segundos. Por último, a Figura 5d representa a quantidade de transações encontradas em cada bloco. Pode-se verificar que a quantidade de transações é, em média, de 2,33 por bloco. Foi registrado que cada transação gasta em média 2.694.948 unidades de gás, acrescidos de uma taxa média de 0,0013 ETH. Dessa forma, com o valor da unidade de gás em 22 Gwei (que é o equivalente a 0,000000022 Ether), tem-se um gasto médio por cada transação de 0,059288856 Ether acrescidos de 0,0013 Ether de taxa, totalizando um gasto de 0,060588856 Ether. Utilizou-se em tal teste uma distribuição uniforme que randomiza o tamanho do dado armazenado entre 2,5MB e 5MB, tendo então uma média de 3,75MB. Visto que, o gasto por transação foi de R$ 43,92, tem-se um custo por MB de R$ 11,71.

5.1.1. IPFS

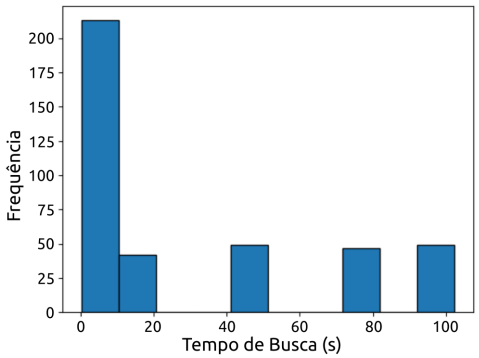
A Figura 6a representa o consumo de memória da *blockchain* observado durante a execução do dApp. O uso máximo de memória foi por volta de 85 MB, crescendo com uma média 7,7 MB de consumo de memória por bloco. Na Figura 6b, constata-se um intervalo de frequência que representa o tempo decorrido entre um *broadcast* de uma transação provinda de um usuário até sua publicação em um bloco minerado. Conclui-se que os tempos variam de 0 até 350 segundos, produzindo uma média de tempo de publicação de 135,03 segundos. Observa-se também que o tempo de busca de um dado ficou em até 120 segundos, como demonstrado na Figura 6c, e uma média de 29,67 segundos.



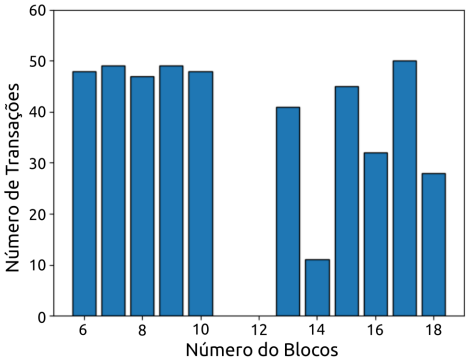
(a) Consumo de memória da *blockchain*.



(b) Tempo de mineração de transações.



(c) Tempo de busca de dados.



(d) Número de transações por bloco.

Figura 6. Resultados dos Testes de Desempenho no IPFS.

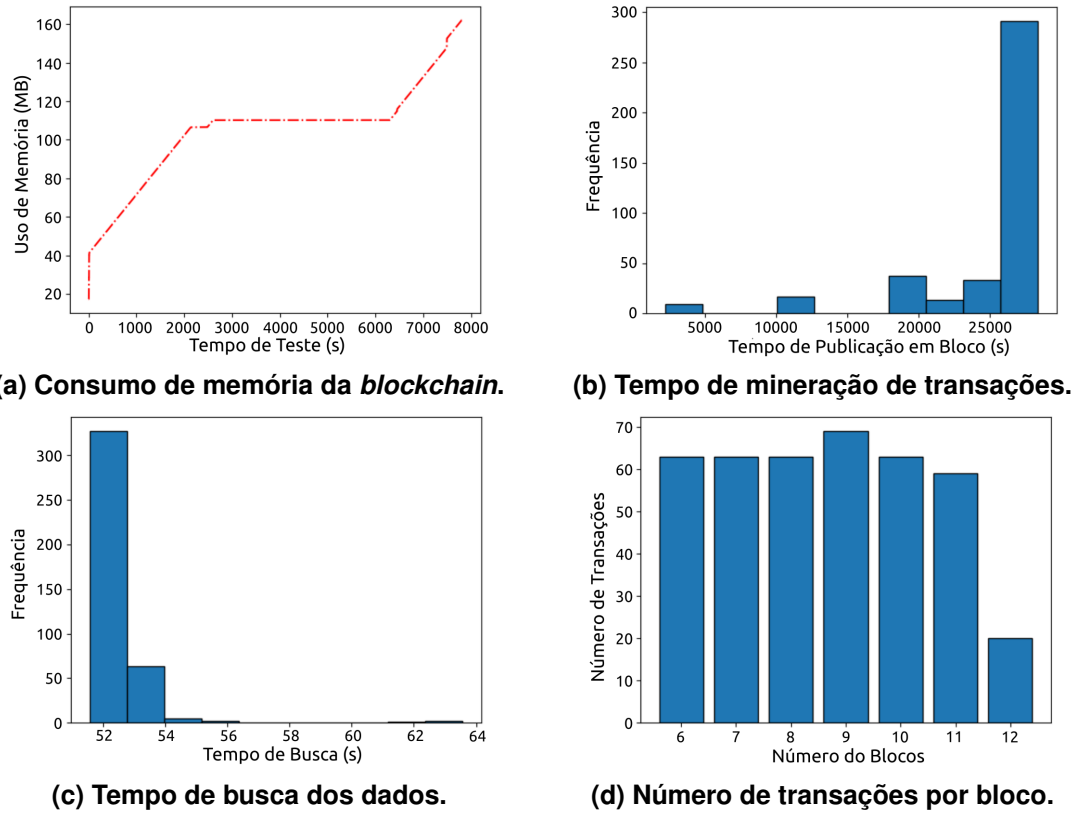
Na Figura 6c também pode-se ver que a maioria dos tempos de busca foram aproximadamente de 10 segundos. Por último, observa-se que o número de transações presentes em cada bloco é em média de 40,72, conforme Figura 6d. Essa quantidade tem pouca variação porque o dApp sempre gasta em cada transação cerca de 106.555 unidades de gás acrescidos de uma taxa média de 0.0013 Ether. Tem-se que o valor da unidade de gás ficou registrado como 22 Gwei, que equivale a 0.000000022 Ether. Considerando que o gasto de cada transação foi de 106.555 unidades, verifica-se um gasto de 0,00234421 Ether acrescidos de 0,0013 de taxa para o minerador, totalizando 0,00364421 Ether.

Cada execução de teste randomiza o tamanho do dado que será armazenado entre

2,5MB e 5MB. Como adotou-se uma distribuição uniforme de valores, tem-se uma média de 3,75MB. Dado que cada transação custa R$ 2,73, verifica-se um custo por MB de aproximadamente R$ 0,73. Os serviços disponibilizados pelo IPFS não têm custo extra, por padrão, visto que é sustentado pela própria comunidade, por isso não dispõe-se um gasto somado a esse valor final. Para adaptar o dApp do Caso Base para a solução aplicada do IPFS foi preciso a instalação do IPFS *daemon*, assim como, um pacote de *JavaScript* chamado "ipfs". A quantidade de alterações no código foram de 33 adições e 28 deleções.

5.1.2. Sia

Na Figura 7a, pode-se analisar o crescimento da *blockchain* observado durante a execução do teste no dApp. O máximo de memória utilizada ao longo do teste foi por volta de 161 MB, crescendo em uma média de 23 MB por bloco.



(a) Consumo de memória da *blockchain*. **(b) Tempo de mineração de transações.**

(c) Tempo de busca dos dados. **(d) Número de transações por bloco.**

Figura 7. Resultados dos Testes de Desempenho no Sia.

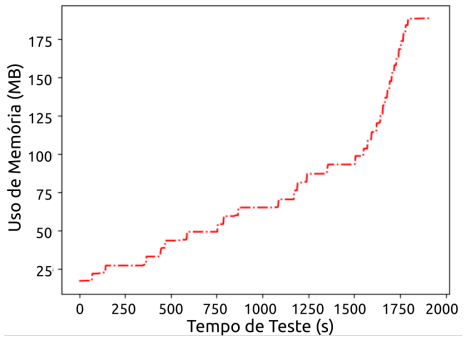
Na Figura 7b pode-se avaliar a frequência de tempos de armazenamento. Estes oscilam até próximos de 28.500 segundos, um valor bastante alto. A maioria dos tempos de armazenamento ficaram acima de 25.000 segundos, criando uma média de tempo de armazenamento de 25.464,38 segundos. Esse tempo é definido pelo período entre a transação de armazenamento ter sido divulgada na rede e o tempo em que a transação apareceu em um bloco minerado. Através da Figura 7c verifica-se que os tempos de busca ficaram em média de 52,50 segundos. Já na Figura 7d denota-se que o número

de transações por bloco ficou bastante uniforme com uma média de 57,14 transações por bloco. Tem-se também uma média de gasto de unidades de gás sendo de 6.382 acrescidos de uma taxa média de 0,0013 Ether. Cada unidade de gás custa 0,000000022 Ether, então registra-se um gasto de 0,000140404 Ether acrescidos de 0,0013 da taxa para o minerador, resultando em 0,001440404 Ether.

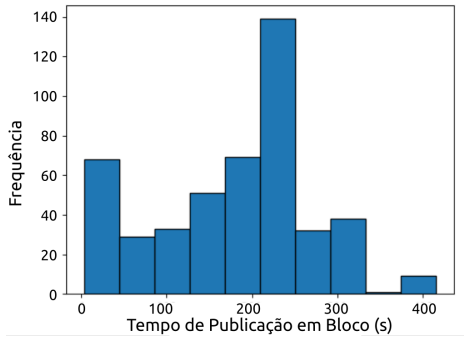
Nos referidos testes também foram usados valores randomizados uniformemente para representar o tamanho de um arquivo. Esses valores oscilam entre 2,5MB e 5MB, tendo em média 3,75MB por arquivo. Como cada transação custa R$ 1,08, tem-se um gasto por MB de R$ 0,28. Os gastos em Sia são contratados de tal forma que podem chegar a valores tão baixos quanto 2 dólares por TB armazenado. Como os cálculos estão sendo feitos em MB, esse valor é muito baixo para interferir no resultado final.

5.1.3. Swarm

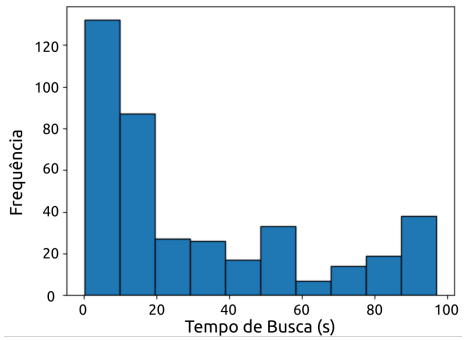
Ao analisar a Figura 8a, verifica-se o crescimento do consumo de memória ao longo do teste. Foram executadas 400 operações de armazenamento, assim como 400 operações de busca. Ao fim, obteve-se um uso máximo de memória de 188 MB. Ao averiguar todos os blocos que possuem transações, percebe-se uma média de consumo de memória de 8,95 MB por bloco.



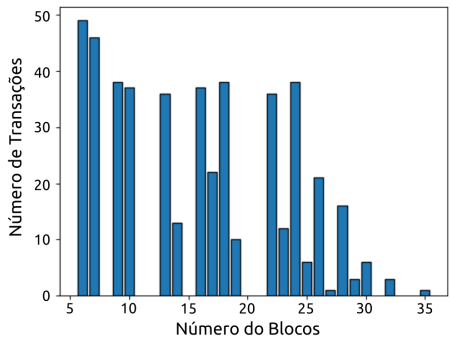
(a) Consumo de memória da *blockchain*.



(b) Tempo de mineração de transações.



(c) Tempo de busca de dados.



(d) Número de transações por bloco.

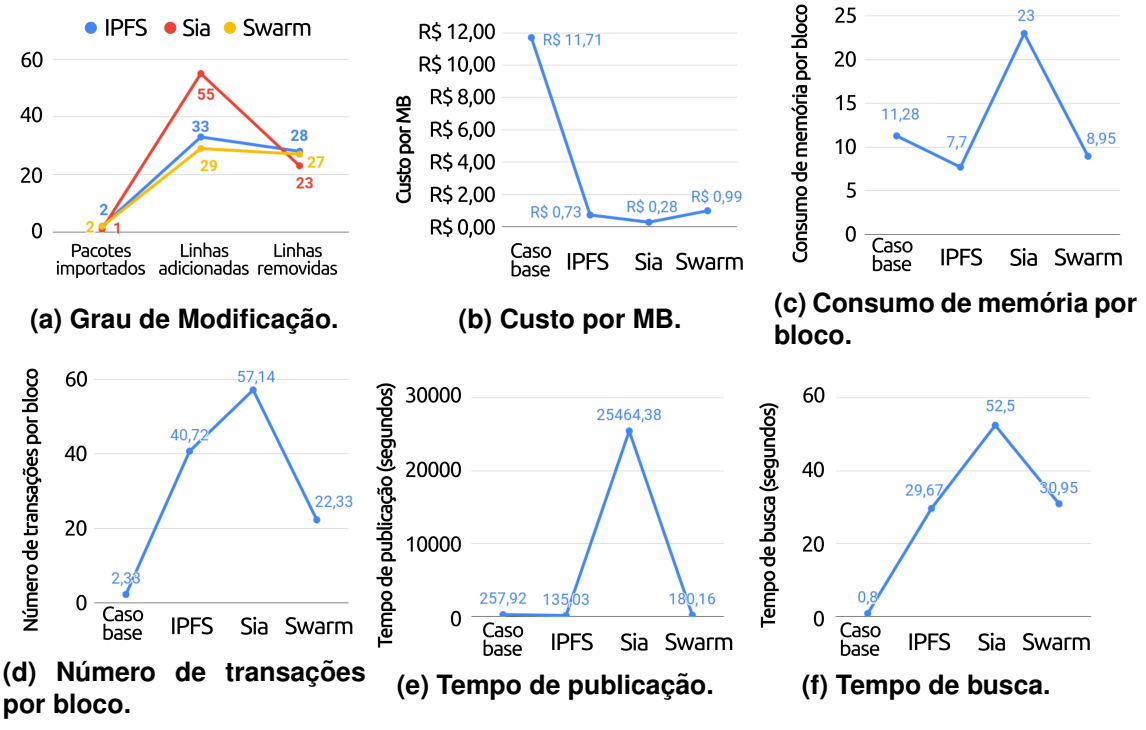
Figura 8. Resultados dos Testes de Desempenho no Swarm.

Na Figura 8b observa-se um intervalo de frequência do tempo de publicação de uma transação em um bloco. Os valores variam e alcançam até 440 segundos com uma

média de 180,16 segundos. Na Figura 8c reflete-se o tempo de busca de dados durante a execução do teste no dApp, o qual chega até 96 segundos com uma média de 30,95 segundos. Por fim, tem-se na Figura 8d a quantidade de transações por bloco o qual possui uma média de 22,33 transações. Conclui-se também que a média de unidades de gás gastas por transações foi de 167.694. Como custo da unidade de gás foi de 22 Gwei, ou 0,000000022 Ether, registra-se um gasto de 0,00368927878 Ether por transação acrescidos de 0,0013 Ether de taxa para o minerador. Visto que adotou-se um teste randomizado uniformemente, tem-se um tamanho médio dos valores armazenados de 3,75MB, resultando em um custo por MB de R$ 0,99. O Swarm não possui um gasto fixo por padrão, então não será adicionado nenhum custo a esse valor por MB.

5.2. Síntese Comparativa

Na Figura 9 são apresentados os gráficos com resultados dos testes de estresse realizados no dApp. Na métrica de grau de modificação (Figura 9a) tem-se que o sistema de mais fácil integração foi o Swarm, observa-se pouca diferença em relação ao Caso Base. O código foi alterado somente com 29 adições e 27 deleções. Contudo, o número de pacotes instalados foi levemente maior que no Sia. Por outro lado, o Sia foi o sistema que mais teve adições de linhas no código.



(a) Grau de Modificação. **(b) Custo por MB.** **(c) Consumo de memória por bloco.**

(d) Número de transações por bloco. **(e) Tempo de publicação.** **(f) Tempo de busca.**

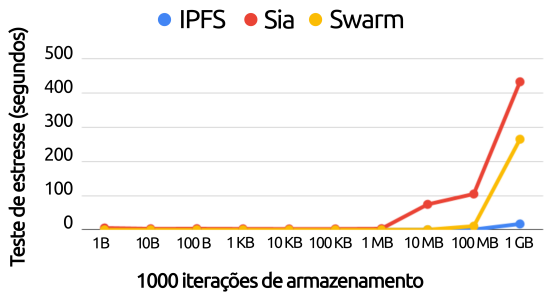
Figura 9. Comparação entre Caso Base, IPFS, Sia e Swarm em relação às métricas avaliadas.

O custo de cada solução por MB teve menor resultado no Sia (Figura 9b). Isso é refletido diretamente por causa dos valores armazenados na *blockchain*. No Caso Base, o próprio dado foi armazenado, o que ocasionou um custo excessivo. Já nas soluções

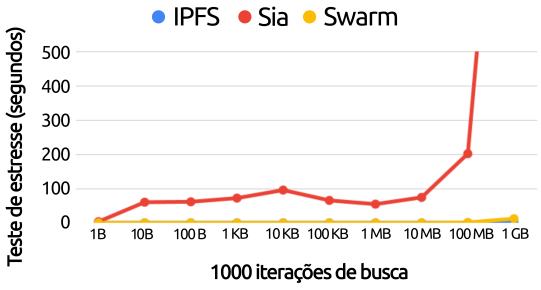
alternativas, o único dado que precisava ser armazenado na *blockchain* era a referência para localizá-lo em cada um dos sistemas. No caso do IPFS e do Swarm, seria o resultado da *hash* do próprio dado, e no Sia o caminho do arquivo armazenado no sistema que nos testes foi, por exemplo, "img0.bin".

O consumo de memória por bloco foi melhor no IPFS (Figura 9c). Quanto ao número de transações por bloco, Sia obteve resultados consideráveis (Figura 9d). O tempo de publicação também foi melhor quando o dado foi armazenamento no IPFS (Figura 9e). Por fim, a Figura 9f mostra que o tempo de busca no Caso Base foi o que se apresentou mais rápido, pois pelo fato de não haver necessidade de requisitar um dado através da rede, mas de forma local, diretamente na *blockchain*, é de se esperar que o tempo no Caso Base obtivesse melhor resultado em relação aos outros. Porém, devido ao gasto de memória excessivo é viável permitir um *trade-off* entre memória e velocidade.

Em relação aos resultados dos testes de estresse foi observado que o Sia demonstrou pior desempenho, tanto no armazenamento quanto na busca. Pode-se verificar nas Figuras 10a e 10b que os resultados foram sempre superiores às outras duas soluções em relação ao tempo gasto, sobre o que se pode concluir que IPFS apresenta-se melhor nos testes de estresse. Observa-se na Figura 10a que, a partir de valores de 1MB, o sistema IPFS tem resultados positivos em comparação aos tempos apresentados no Sia. Pode-se ressaltar, também, que a diferença fica cada vez maior à medida que o tamanho do dado armazenado aumenta, ficando até 15 vezes mais lento.



(a) Comparação dos tempos em teste de estresse de armazenamento.



(b) Comparação dos tempos em teste de estresse de busca.

Figura 10. Testes de Estresse.

Finalmente, a Figura 10b apresenta a superioridade do Swarm se estendendo até testes com arquivos de 100KB, mas, a partir disso os valores começam a degradar. Em geral, conclui-se que, de todos os sistemas (IPFS, Sia e Swarm), o IPFS é o que responde melhor ao crescimento do tamanho dos dados, tanto nos testes de armazenamento quanto nos de busca. A partir de um arquivo de 1MB de tamanho, o IPFS apresenta menores tempos de armazenamento (0,0969s) em comparação aos outros sistemas. Por sua vez, a partir de um arquivo de 100MB de tamanho, o IPFS apresenta menores tempos de busca (0,2388s) em comparação aos outros sistemas.

6. Considerações Finais

Visando contornar o desafio da escalabilidade de armazenamento em blockchain, estratégias *off-chain* têm sido adotadas, ou seja, ao invés de armazenar os dados na própria blockchain, eles serão alocados em serviços descentralizados de armazenamento, onde somente suas referências ficarão nos blocos. No entanto, atualmente existem várias soluções no mercado com diferentes características baseadas em Sistemas de Armazenamento Distribuído, dentre as quais pode-se destacar o IPFS, Sia e Swarm. Diante desse contexto, o presente trabalho contribuiu, de forma pioneira, para concretização de uma avaliação experimental e comparativa a fim de subsidiar uma análise mais profunda sobre a a adoção de estratégias *off-chain* e, consequentemente, aprimorar a tomada de decisão sobre qual opção de Sistema de Armazenamento Distribuído escolher.

Através de diferentes métricas e testes de desempenho, o experimento computacional conduzido neste trabalho possibilitou investigar, para cada sistema, critérios como consumo de memória, tempo de mineração de transações, tempo de busca de dados, transações por bloco, grau de modificação e custo por MB. Observou-se que cada escolha depende das particularidades do projeto. Os resultados mostraram que, quanto ao grau de modificação, o Swarm demonstrou ser o de mais fácil integração ao ser codificado. Por sua vez, o Sia apresentou o menor custo por MB, custando R$ 0,28/MB, enquanto posicionou-se como o sistema com maior quantidade de transações por bloco. Em relação ao uso de memória por bloco, IPFS demonstrou melhor desempenho. O Caso Base teve resultados positivos apenas na métrica de tempo de busca de dados. De forma geral, quanto aos testes de estresse de armazenamento e busca, IPFS e Swarm obtiveram melhores resultados, respectivamente. Concluiu-se que caso a procura seja por um sistema mais confiável, com soluções consolidadas e escalável, a Sia demonstra-se proeminente. Caso verifique-se o requisito de uma alta performance, o IPFS posiciona-se com mais potencial. No entanto, caso o projeto usufrua da *stack* do Ethereum, torna-se mais proeminente utilizar o Swarm enquanto sistema nativo, visto que estará integrado a solução final.

Em termos de trabalhos futuros, pretende-se ampliar o estudo comparativo incluindo mais métricas, cenários de avaliação e outras soluções *off-chain*, como o Storj (que ainda está em versão beta e o acesso indisponível), Koppercoin e Retricoin (os quais ainda encontram-se disponíveis apenas em artigos científicos). Almeja-se também realizar uma análise qualitativa, visando investigar questões referentes às comunidades, documentações, particularidades de implementação e sistemas de incentivo.

Referências

Aitzhan, N. Z. and Svetinovic, D. (2018). Security and privacy in decentralized energy trading through multi-signatures, blockchain and anonymous messaging streams. *IEEE Transactions on Dependable and Secure Computing*, 15(5):840–852.

Antonopoulos, A. M. and Wood, G. (2018). *Mastering ethereum: building smart contracts and dapps*. O'Reilly Media.

Beck, R., Avital, M., Rossi, M., and Thatcher, J. B. Blockchain technology in business and information systems research. *Bus Inf Syst Eng*.

Benet, J. (2014). Ipfs-content addressed, versioned, p2p file system. *arXiv preprint arXiv:1407.3561*.

Benisi, N. Z., Aminian, M., and Javadi, B. (2020). Blockchain-based decentralized storage networks: A survey. *Journal of Network and Computer Applications*, page 102656.

BitInforCharts (2019). Ethereum (eth) price stats and information. Disponível em: `https://bitinfocharts.com/ethereum/`. Acesso em: 27 set. 2019.

Cachin, C. and Vukolić, M. (2017). Blockchain consensus protocols in the wild. *arXiv preprint arXiv:1707.01873*.

Casino, F., Politou, E., Alepis, E., and Patsakis, C. (2019). Immutability and decentralized storage: An analysis of emerging threats. *IEEE Access*, 8:4737–4744.

Chinchilla, C. (2014). Ethereum: A next-generation smart contract and decentralized application platform. Disponível em: `https://github.com/ethereum/wiki/wiki/White-Paper`. Acesso em: 27 jun. 2020.

da Silva Rodrigues, C. K. (2017). Sistema bitcoin: uma análise da segurança das transações. *iSys-Brazilian Journal of Information Systems*, 10(3):5–23.

Eberhardt, J. and Tai, S. (2017). On or off the blockchain? insights on off-chaining computation and data. In *European Conference on Service-Oriented and Cloud Computing*, pages 3–15. Springer.

Ethereum (2020). Eth 2.0. Disponível em: `https://www.ethereum.org/learn/#eth-2-0`. Acesso em: 27 jun. 2020.

Etherscan (2019). Ethereum gas limit history. Disponível em: `https://etherscan.io/chart/gaslimit`. Acesso em: 27 set. 2019.

Etherscan (2020). Ethereum unique address growth rate. Disponível em: `https://etherscan.io/chart/address`. Acesso em: 27 jun. 2020.

Group, T. B. (2019). One click blockchain. Disponível em: `https://www.trufflesuite.com/ganache`. Acesso em: 27 jun. 2020.

Hartman, J. H., Murdock, I., and Spalink, T. (1999). The swarm scalable storage system. In *Proceedings. 19th IEEE International Conference on Distributed Computing Systems (Cat. No. 99CB37003)*, pages 74–81. IEEE.

Holgate, R., Furlonger, D., and Howard, R. (2017). Toolkit: Government use cases for blockchain. gartner. *International Journal of Community Currency Research*.

Jiang, S., Cao, J., Wu, H., Yang, Y., Ma, M., and He, J. (2018). Blochie: a blockchain-based platform for healthcare information exchange. In *2018 ieee international conference on smart computing (smartcomp)*, pages 49–56. IEEE.

Jiménez-Peris, R., Patiño-Martínez, M., Kemme, B., and Alonso, G. (2002). Improving the scalability of fault-tolerant database clusters. In *Proceedings 22nd International Conference on Distributed Computing Systems*, pages 477–484. IEEE.

Lakhani, K. and Iansiti, M. (2017). The truth about blockchain. *Harvard Business Review*, 95(1):119–127.

Laurence, T. (2019). *Blockchain for dummies*. John Wiley & Sons.

Nakamoto, S. (2009). Bitcoin: A peer-to-peer electronic cash system.

Narayanan, A., Bonneau, J., Felten, E., Miller, A., and Goldfeder, S. (2016). *Bitcoin and cryptocurrency technologies:* a comprehensive introduction. Princeton University Press, New Jersey.

Pop, C., Antal, M., Cioara, T., Anghel, I., Sera, D., Salomie, I., Raveduto, G., Ziu, D., Croce, V., and Bertoncini, M. (2019). Blockchain-based scalable and tamper-evident solution for registering energy data. *Sensors*, 19(14):3033.

Project, B. (2020). Running a full node. Disponível em: `https://bitcoin.org/en/full-node`. Acesso em: 27 jun. 2020.

Raval, S. (2016). *Decentralized applications: harnessing Bitcoin's blockchain technology*. O'Reilly Media, Inc.

Rodrigues, D., Cunha, A., Meirelles, F., and Diniz, E. (2018). Benefits of blockchain for digital social currency.

Shukla, P. A. and Samet, S. (2020). Systematization of knowledge on scalability aspect of blockchain systems. In *Future of Information and Communication Conference*, pages 130–138. Springer.

Szabo, N. (1996). Smart contracts: building blocks for digital markets. *EXTROPY: The Journal of Transhumanist Thought,(16)*, 18(2).

Szilágyi, P. (2019). Geth v1.9.0. Disponível em: `https://blog.ethereum.org/2019/07/10/geth-v1-9-0/`. Acesso em: 27 jun. 2020.

Tapscott, D. and Tapscott, A. (2016). The impact of the blockchain goes beyond financial services. *Harvard Business Review*, 10(7).

Vorick, D. and Champine, L. (2014). Sia: Simple decentralized storage. *Nebulous Inc*.

Xie, J., Yu, F. R., Huang, T., Xie, R., Liu, J., and Liu, Y. (2019). A survey on the scalability of blockchain systems. *IEEE Network*, 33(5):166–173.

Yaga, D., Mell, P., Roby, N., and Scarfone, K. (2019). Blockchain technology overview. *arXiv preprint arXiv:1906.11078*.