Uma Análise Comparativa da Arquitetura e Desempenho de Plataformas de Corrente de Blocos Permissionadas para Contratos Inteligentes

**Guilherme A. Thomaz, Gustavo F. Camilo,
Lucas Airam C. de Souza, e Otto Carlos M. B. Duarte**

Grupo de Teleinformática e Automação (GTA)
Universidade Federal do Rio de Janeiro (UFRJ)

Resumo. *A corrente de blocos e os contratos inteligentes garantem segurança e automatização em cenários sem confiança, gerando soluções inovadoras em diversos setores produtivos. O projeto de código aberto Hyperledger impulsiona o emprego dessas tecnologias no meio corporativo provendo diferentes plataformas para o desenvolvimento de aplicações distribuídas. Este artigo analisa e compara duas plataformas amplamente utilizadas para o desenvolvimento de aplicações baseadas em correntes de blocos permissionadas: o Hyperledger Sawtooth e o Hyperledger Fabric. Dois protótipos desenvolvidos implementam contratos inteligentes para uma mesma aplicação de modo a avaliar o desempenho de cada ferramenta. Os resultados obtidos revelam que: i) o processamento paralelo de transações do Sawtooth apresenta um desempenho até 30% superior apenas se o número de transações conflitantes permanecer baixo; ii) o desempenho do modelo XO do Fabric é 4 vezes maior que o OX do Sawtooth, mas apresenta uma piora considerável com transações conflitantes; iii) o consenso do Sawtooth apresenta maior segurança e menor desempenho que o do Fabric; iv) as aplicações no Sawtooth consomem menos armazenamento em disco.*

1. Introdução

A corrente de blocos (*Blockchain*) é uma tecnologia disruptiva que garante auditabilidade, autenticação, pseudoanonimato e não repúdio na transferência de ativos digitais com confiança distribuída [Rebello et al. 2019b, Nakamoto 2008]. Os participantes da rede armazenam cópias completas de um registro imutável de transações assinadas pelos seus emissores e um protocolo de consenso garante que as mesmas transações sejam aplicadas, ou rejeitadas, em todos os pontos da rede. A corrente de blocos implementa lógicas arbitrárias a partir das transações com o uso de contratos inteligentes, possibilitando o desenvolvimento de aplicações distribuídas que vão além das criptomoedas [Wood et al. 2014].

O projeto Hyperledger da Linux Foundation desenvolve plataformas de código livre com a finalidade de fomentar tecnologias de correntes de blocos permissionadas e

Este trabalho foi realizado com recursos do CNPq, CAPES, FAPERJ e FAPESP (18/23292-0, 2015/24485-9 e 2014/50937-1).

Uma versão em inglês baseada neste artigo intitulada ”Architecture and Performance Comparison of Permissioned Blockchains Platforms for Smart Contracts”deve ser submetida a congressos ou revistas internacionais.

contratos inteligentes em casos de uso corporativos. As correntes de blocos permissionadas requerem a identificação e uma permissão para o usuário fazer parte da rede, criando um cenário com participantes que possuem um nível alto de confiança. Entre as seis plataformas mantidas pelo projeto até o momento, duas se destacam por serem as pioneiras e as mais utilizadas: Hyperledger Sawtooth, liderada pela Intel, e Hyperledger Fabric, liderada pela IBM [Rauchs et al. 2019]. Analisar as características das correntes de blocos permissionadas auxilia na difusão da tecnologia. Ademais, a limitação no desempenho ainda é um dos principais obstáculos para a adoção das correntes de blocos em cenários corporativos que exigem altas vazões transacionais e baixas latências. Dessa forma, é necessário avaliar experimentalmente os fatores que afetam o desempenho de aplicações de interesse do meio corporativo, como gerenciamento de dados, comercialização de ativos e automação de processos [Hamida et al. 2017].

Este artigo analisa e compara a arquitetura e o desempenho de duas plataformas Hyperledger de correntes de blocos permissionadas para o desenvolvimento de contratos inteligentes: Sawtooth e Fabric. Um protótipo com contratos inteligentes para comercialização automática de dados entre organizações (*marketplace*) é desenvolvido nas duas plataformas com a finalidade de avaliar os seus desempenhos [Camilo et al. 2020b]. Os resultados obtidos revelam que o uso do processamento paralelo de transações no Sawtooth aumenta a vazão transacional se o número de transações conflitantes permanecer baixo. Também foi observado um aumento na vazão transacional com o uso do protocolo de consenso Raft no Fabric e uma degradação do desempenho do Fabric em aplicações práticas que geram um grande volume de transações conflitantes, como em transferências que alteram o balanço financeiro de uma mesma organização. A comparação entre as plataformas mostrou que o Fabric apresenta um protocolo de consenso mais rápido e portanto, a taxa de crescimento da corrente no Fabric é maior.

O restante do artigo está organizado da seguinte forma. A Seção 2 discute os trabalhos relacionados. A Seção 3 detalha as arquiteturas das plataformas Hyperledger Sawtooth e Fabric. A Seção 4 apresenta os resultados da avaliação de desempenho do protótipo desenvolvido nas duas plataformas. Por fim, a Seção 5 conclui o artigo, sintetizando as diferenças mais importantes de cada plataforma e fornecendo as direções para trabalhos futuros.

2. Trabalhos relacionados

O emprego de correntes de blocos permissionadas e contratos inteligentes no meio corporativo em cenários de Internet das Coisas (*Internet of Things* - IoT), 5G e automação [de Souza et al. 2020] tem sido investigado com sucesso. Camilo *et al.* propõem um sistema que garante a comercialização segura e o controle de acesso automático de dados IoT seguindo um modelo de reputação e confiança [Camilo et al. 2020a]. Os autores utilizam a corrente de blocos para mitigar as dispendiosas taxas de serviço, pontos únicos de falha e comprometimento da privacidade, produzidos pela centralização do controle de acesso aos dados. O protótipo implementado no Fabric alcança uma vazão transacional de até 70 transações por segundo, valor superior ao requisitado pelos sistemas de comércio eletrônico nacionais. Rebello *et al.* desenvolvem uma arquitetura baseada em corrente de blocos para garantir isolamento entre fatias de rede e implementam um protótipo no Fabric [Rebello et al. 2019a]. A pro-

posta se destaca por utilizar os canais isolados da plataforma para garantir privacidade e por utilizar contratos inteligentes para automatizar o gerenciamento e a configuração de funções virtuais de rede, essenciais para o 5G. Palma *et al.* apresentam a corrente de blocos como alternativa para o armazenamento dos certificados de graduação e históricos acadêmicos de estudantes do ensino superior brasileiro e implementam o protótipo no Ethereum [Palma et al. 2019]. Os autores utilizam contratos inteligentes para minimizar a interferência humana na emissão de certificados e se servem das propriedades de autenticação e da garantia de integridade da corrente de blocos para mitigar fraudes.

Existem muitas arquiteturas, ferramentas e configurações para correntes de blocos e contratos inteligentes com diferentes características e a escolha da plataforma apropriada para cada caso de uso é uma tarefa complexa. Trabalhos que analisam e comparam a arquitetura e o desempenho de correntes de blocos permissionadas são essenciais para a adoção dessa tecnologia nas organizações. Benahmed *et. al.* avaliam os fatores que mais contribuem para o desempenho das aplicações em correntes de blocos [Benahmed et al. 2019]. O trabalho compara diferentes contratos inteligentes do Fabric, mas não considera diferentes plataformas de correntes de blocos.

Polge *et. al.* comparam cinco plataformas de correntes de blocos industriais quanto ao suporte da comunidade, desempenho, escalabilidade, privacidade e critério de adoção, incluindo o Hyperledger Fabric [Polge et al. 2021]. Os autores realizam uma revisão bibliográfica e não realizam implementações de cenários experimentais para comparar as métricas.

Caro *et al.* apresentam um caso de uso da corrente de blocos para rastreamento da cadeia de suprimentos de produtos agrícolas e alimentos e comparam o desempenho das implementações no Sawtooth e no Ethereum [Caro et al. 2018]. Os resultados apresentam uma latência mais de setecentas vezes menor e um consumo de CPU mais de seis vezes menor no Sawtooth, demonstrando o potencial de desempenho da plataforma da corrente de blocos permissionada em cenários práticos. O artigo foca na implementação prática do caso de uso para apresentar resultados preliminares e não explora a arquitetura das plataformas e os fatores que mais afetam o desempenho da aplicação.

Dinh *et al.* desenvolvem o *BLOCKBENCH* para comparar correntes de blocos privadas utilizando diversas métricas e identificam os fatores limitantes do desempenho [Dinh et al. 2017]. O trabalho não se concentra nas diferenças entre as arquiteturas das plataformas Hyperledger.

Muller *et. al.* comparam as plataformas Sawtooth e Fabric e considera um caso de uso de cadeia de suprimentos de pescados [Muller 2019]. Apesar de a tese apresentar aspectos gerais das plataformas, o foco é na comparação do fluxo de processamento das transações. O autor não implementa os casos de uso e não realiza uma comparação do desempenho das plataformas. Rasolroveicy *et. al.* comparam as características, o uso de recursos e a latência de quatro plataformas de correntes de blocos para registro de dados IoT, incluindo o Fabric e o Sawtooth [Rasolroveicy and Fokaefs 2020]. O artigo não avalia a influência do protocolo de consenso no desempenho e não apresenta medidas de vazão transacional e uso do armazenamento em disco, que são essenciais para avaliar a escalabilidade das ferramentas em um cenário IoT, com milhares de sensores registrando dados Wang *et al.* analisa o desempenho de quatro plataformas de correntes de blocos,

incluindo o Fabric e o Sawtooth [Wang et al. 2020]. Os autores resumem características das plataformas e avaliam a influência da taxa de envio das transações na vazão transacional e na latência com diferentes transações. O artigo não realiza um estudo aprofundado das plataformas Hyperledger e não avalia de forma independente a influência do modelo de execução de transações, dos protocolos de consenso e da concorrência de transações no desempenho.

Ao contrário dos artigos citados anteriormente, este artigo realiza uma análise comparativa entre as arquiteturas das duas plataformas pioneiras e amplamente utilizadas mantidas pelo projeto Hyperledger. Para cada plataforma, o artigo analisa: arquitetura da rede, implementação dos contratos inteligentes, armazenamento do estado da corrente de blocos, modelo de processamento das transações, protocolos de consenso e controle de permissões. O artigo também implementa contratos inteligentes baseados em um caso de uso de comercialização entre organizações. Os resultados demonstram como o desempenho é afetado pelas diferenças no processamento de transações, no protocolo de consenso e na forma com o contrato inteligente lê e escreve na corrente de blocos.

3. As Plataformas Hyperledger Sawtooth e Hyperledger Fabric

O Hyperledger é um projeto da Linux Foundation que visa desenvolver plataformas, arcabouços, ferramentas e bibliotecas de código aberto para a difusão de implementações da tecnologia de corrente de blocos para uso empresarial. As plataformas Hyperledger são modulares e garantem flexibilidade na configuração das opções de consenso, contratos inteligentes e políticas de permissão de acesso, atendendo assim um grande número de aplicações. As plataformas de corrente de blocos permissionadas Hyperledger Fabric[2] e Hyperleger Sawtooth[3] são as duas principais plataformas do projeto Hyperledger.

As correntes de blocos permissionadas visam aplicações empresariais em que os nós se identificam para participar da rede. Essas aplicações usualmente empregam protocolos de consenso baseados em quorum que apresentam uma maior vazão em transações por segundo quando comparados aos protocolos baseados em prova das redes não permissionadas como Bitcoin e Ethereum [Rebello et al. 2020].

3.1. A plataforma Hyperledger Sawtooth

A arquitetura da plataforma Hyperledger Sawtooth é formada por nós validadores e clientes, como mostra a Figura 1. Os nós validadores armazenam uma cópia da corrente de blocos, validam as transações e blocos e se comunicam com os outros nós validadores em uma rede par-a-par através de um protocolo de fofocas (*gossip*). Por sua vez, os clientes se comunicam com os validadores por requisições através de uma interface de programa de aplicação em transferência representacional de estado (*Application Program Interface - Representational State Transfer* - API-REST).

Os contratos inteligentes do Sawtooth são denominados famílias de transações que são formadas por dois componentes: um processador de transações e um aplicativo cliente. O processador de transações recebe as transações dos clientes e aplica as alterações na corrente de blocos, seguindo as regras definidas pelo contrato inteligente. Todos os nós

[2]https://www.hyperledger.org/use/fabric. Acesso em: 30/07/2021

[3]https://www.hyperledger.org/use/sawtooth. Acesso em: 30/07/2021

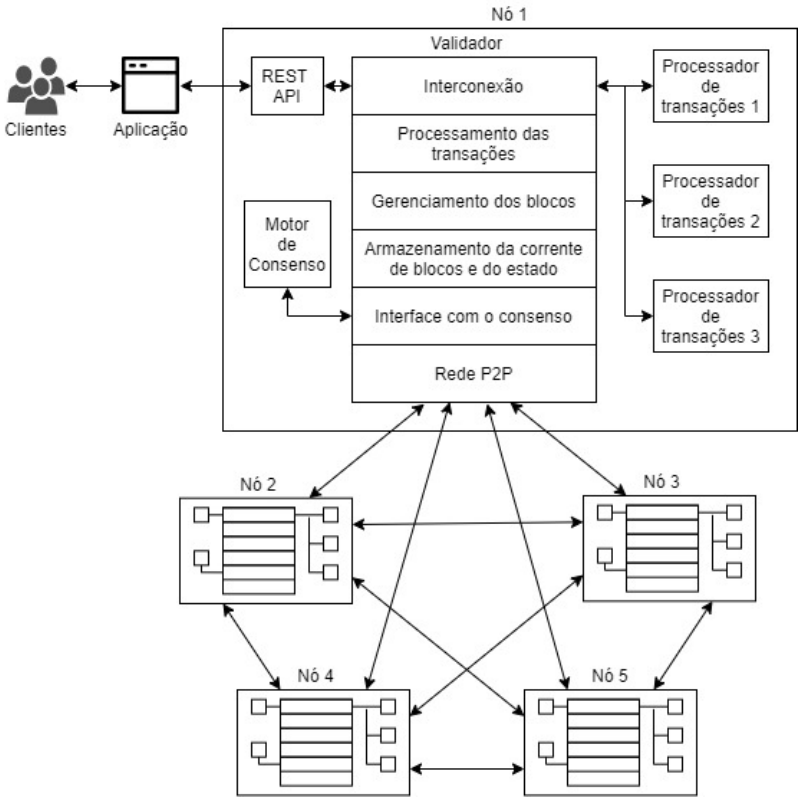


Figura 1. Arquitetura da plataforma Hyperledger Sawtooth. Os nós validadores processam as transações dos clientes, armazenam a corrente de blocos e se comunicam para garantir o consenso em uma rede par-a-par.

validadores devem ter o mesmo conjunto de processadores de transações instalados. O aplicativo cliente envia as transações para um validador, no formato definido pelo contrato inteligente. A plataforma disponibiliza algumas famílias de transações para configuração e testes, além de um kit de desenvolvimento de software (*Software Development Kit* - SDK) nas linguagens Go, JavaScript, Python, Rust, C++, Java e Swift[4] para desenvolvimento de novas famílias de transações.

Dados do estado global da corrente de blocos, que são frequentemente consultados, são armazenados em uma estrutura de árvore de Merkle. Essa técnica evita buscas extensas na corrente e fornece uma forma ágil de verificar a integridade do estado global. Uma transação pode depender das alterações aplicadas no estado global por outra transação, chamada transação predecessora. Essas transações são chamadas de conflitantes, pois devem ser aplicadas em ordem.

O processamento das transações segue o modelo tradicional, chamado ordenar-executar (*order-execute* - OX) [Muller 2019] e as etapas são apresentadas abaixo:

1. **Envio**: os clientes propõem uma ou mais transações e as enviam para seus nós validadores em uma estrutura ordenada e assinada chamada lote. O lote inteiro é descartado caso uma transação do lote seja inválida. Para que todos os nós

[4]A SDK nas linguagens C++, Java e Swift ainda encontra-se em fase experimental. Disponível em: https://sawtooth.hyperledger.org/docs/core/releases/latest/app_developers_guide/sdk_table.html. Acesso em: 30/03/2021.

validadores processem todos os lotes de transações, um nó validador recebe os lotes dos outros;

2. **Ordenação**: os validadores garantem que as transações conflitantes sejam aplicadas na ordem adequada. As transações enviadas pelos clientes contêm os endereços do estado a serem lidos e escritos e os validadores utilizam essa informação para determinar as transações predecessoras;
3. **Execução**: os processadores de transações computam as alterações do estado global. A execução das transações pode seguir um modelo serial, em que todas as transações são executadas sequencialmente na ordem definida, ou paralelo, em que transações não conflitantes são executadas concorrentemente. Isso evita que transações sejam descartadas por falta de informações sobre as alterações aplicadas pelas predecessoras. Entretanto, no caso de um número grande de transações conflitantes em um mesmo bloco, a vazão transacional é consideravelmente menor devido à execução sequencial;
4. **Confirmação**: um nó validador propõe e difunde um bloco de acordo com o protocolo de consenso. O bloco difundido contém os identificadores (ID) dos lotes de transações e apenas as transações válidas são aplicadas na corrente de blocos. Os nós adicionam o novo bloco quando todos os lotes de transações forem executados.

O principal protocolo de consenso da plataforma Sawtooth é a Prova de Tempo Decorrido (*Proof of Elapsed Time* - PoET). O participante deve fornecer uma prova de que esperou um tempo aleatório para propor o bloco. Este tipo de consenso exige o uso de *hardware* específico com suporte à tecnologia Intel SGX (*Software Guard Extensions* - SGX) para que participantes maliciosos não manipulem a geração de blocos [van Schaik et al. 2020]. Uma alternativa ao *hardware* específico em um cenário de desenvolvimento é o uso do simulador de PoET cuja implementação é insegura em ambientes com nós maliciosos, pois garante apenas tolerância a falhas de parada (*Crash Fault Tolerance* - CFT). Em redes com poucos nós com certo nível de confiança, o Sawtooth suporta o protocolo prático tolerante a falhas bizantinas (*Pratical Byzantine Fault Tolerance* - PBFT)[5], um protocolo baseado em prova no qual os participantes precisam se identificar e autenticar para que haja troca de mensagens.

Nós validadores e clientes se identificam por suas chaves públicas que são usadas para definir quais clientes podem submeter transações e quais nós validadores podem se conectar à rede P2P. Desta forma, o Sawtooth oferece suporte a redes públicas em que todos os clientes podem submeter transações e qualquer validador pode participar do consenso.

3.2. A plataforma Hyperledger Fabric

A arquitetura do Fabric é formada por clientes, nós pares e nós ordenadores como mostra a Figura 2. Todos os participantes são identificados como membros de uma organização. Os clientes interagem com a corrente de blocos através de transações enviadas para os pares. Os nós pares são responsáveis por armazenar uma cópia da corrente

[5]O protocolo de consenso Sawtooth Raft ainda se encontra em fase de desenvolvimento de acordo com o repositório em linha (*online*) disponível em: https://github.com/hyperledger/sawtooth-raft. Acesso em: 31/03/2021

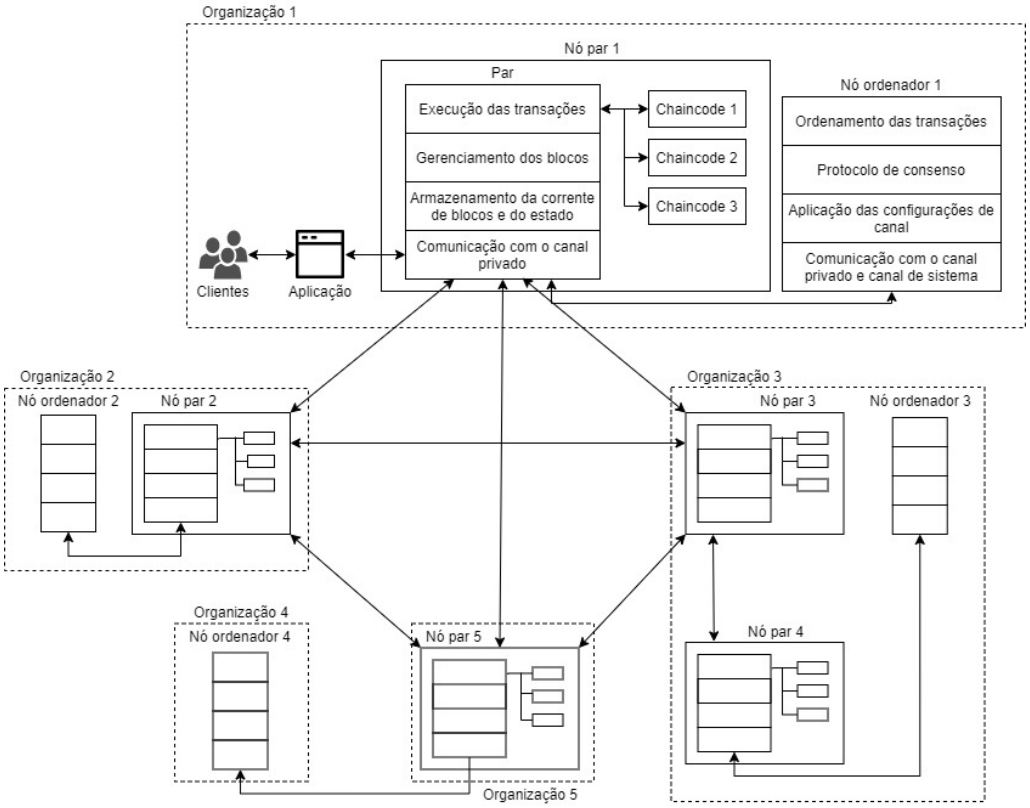


Figura 2. Arquitetura da plataforma Hyperledger Fabric. Os nós pares executam as transações dos clientes e podem se comunicar com os pares de outra organização ou com o serviço de ordem. Os nós ordenadores ordenam as transações executadas em blocos e se comunicam por um canal de sistema, omitido na figura.

de blocos e executar as transações dos clientes. A função dos nós ordenadores é garantir consenso na ordem das transações executadas pelos pares e difundir os blocos de transações para a rede. Diferente do Sawtooth, no qual os validadores concentram todo o processamento das transações, o Fabric separa os papéis entre pares e ordenadores e alcança maior paralelismo. Pares âncoras são responsáveis por se comunicarem com pares de outras organizações através de um protocolo de fofocas (*gossip*) e pares que se comunicam com ordenadores são denominados líderes. Na Figura 2, os nós 1, 2, 3 e 5 são pares âncoras e os nós 1, 2, 4 e 5 são pares líderes. [Rebello et al. 2019b].

Os *chaincodes* no Fabric são equivalentes aos processadores de transação no Sawtooth. Os pares com um chaincode instalado recebem propostas de transações dos clientes e retornam para o cliente uma resposta assinada, denominada transação endossada. Uma vantagem do Fabric em relação ao Sawtooth está no fato de que nem todos os pares precisam instalar os mesmos chaincodes e executar as mesmas transações, diminuindo o número de transações que cada par executa. Uma política de endosso define o número mínimo de pares que precisam executar uma proposta de transação para que ela seja adicionada à corrente de blocos.

O Fabric propõe um modelo novo de processamento de transações chamado executar-ordenar (*Execute-Order* - XO) em que os pares executam todas as transações

em paralelo, com o objetivo de aumentar a vazão transacional [Gorenflo et al. 2020]. As etapas executadas pelo Fabric listadas abaixo são:

1. **Envio**: os clientes enviam propostas de transação para os nós pares. Pares que recebem propostas de transação devem possuir um chaincode instalado e são chamados de pares de endosso;
2. **Execução**: os pares de endosso recebem as propostas de transação e calculam as alterações a serem aplicadas na corrente de blocos, de acordo com a lógica do contrato inteligente. As transações são executadas em paralelo, na ordem em que chegam aos pares. Em seguida, os nós de endosso retornam para os clientes as transações endossadas contendo os resultados das execuções;
3. **Ordenação**: os clientes enviam as transações endossadas para os nós ordenadores que ordenam as transações em um candidato a bloco. Os nós ordenadores acordam um bloco executando o protocolo de consenso;
4. **Confirmação**: o bloco é enviado para os pares líderes da cada organização que difundem para os outros pares. Os pares validam as transações e aplicam as alterações nas suas cópias da corrente de blocos. Diferente do Sawtooth, as transações inválidas são registradas na corrente de blocos para garantir auditabilidade.

O estado global armazena informações da corrente de blocos em um banco de dados no formato de pares chave-valor. Cada transação especifica um conjunto de leitura e um conjunto de escrita com informações a serem lidas e escritas no estado global. Diferente do Sawtooth que evita conflitos executando transações previamente ordenadas, o Fabric executa todas as transações concorrentemente e pode gerar conflitos, como ilustrado na Figura 3. Se uma transação for executada sem que suas predecessoras tenham sido aplicadas, os pares lerão informações de uma versão desatualizada do estado global. Os pares realizam uma validação de controle de concorrência multiversão (*Multi-Version Concurrency Control* - MVCC) na fase de confirmação para impedir inconsistências. Transações que leem a versão atual do estado global são validadas e as transações dependentes no mesmo bloco são invalidadas [Chacko et al. 2021]. Quando a transação é invalidada, o

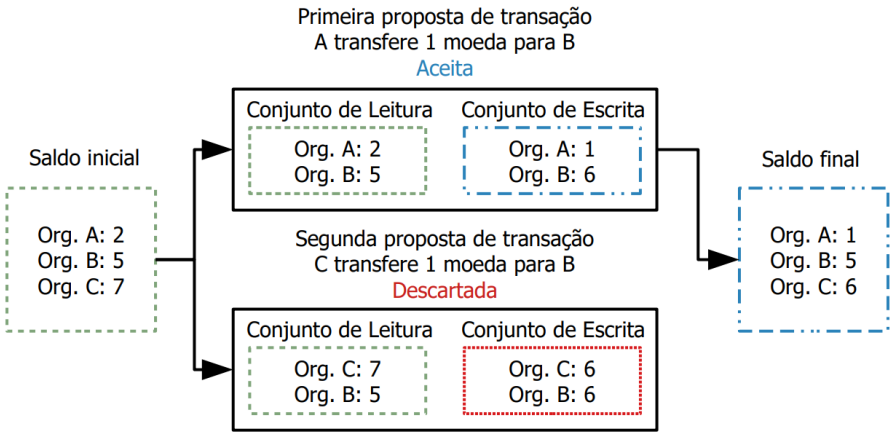


Figura 3. Transações conflitantes no Fabric. A versão do estado é indicada pelo estilo de linha do retângulo. A segunda transação é executada pelo par sem que as alterações da primeira tenham sido aplicadas e será descartada.

cliente reenvia a proposta de transação para ser processada e ordenada novamente, reduzindo o desempenho da rede, pois os participantes desperdiçam recursos computacionais processando transações inválidas.

O protocolo de consenso disponível no Fabric é o Raft[6]. O protocolo do Fabric é baseado em quorum, de modo que os participantes trocam mensagens para votar no próximo bloco. Essa categoria de protocolos é determinística e não permite bifurcações, que ocorrem nos protocolos baseados em prova quando dois participantes propõem blocos ao mesmo tempo sem se conhecerem. Entretanto, diferente do PBFT, o Raft tolera apenas falhas de parada e é vulnerável a agentes maliciosos (bizantinos). Esses protocolos são bem mais simples que protocolos tolerantes a falhas Bizantinas e, então, alcançam vazões transacionais mais altas em redes permissionadas.

As organizações utilizam autoridades de certificação (*Certification Authorities - CA*) para identificar os participantes e um provedor de serviço de associação (*Membership Service Provider - MSP*) é definido na configuração da rede para determinar as permissões de cada identidade. Um diferencial do Fabric em relação ao Sawtooth e às correntes de bloco públicas é que a comunicação entre pares, clientes e ordenadores, indicada na Figura 2 pelas setas, ocorre em um canal privado. Diferentes canais logicamente isolados podem ser implementados em uma mesma rede no Fabric, possibilitando comunicações privadas entre organizações que decidem formar um consórcio.

4. Experimentos e Resultados

O objetivo da análise de desempenho é comparar a vazão transacional, em transações válidas por segundo, das plataformas Hyperledger Sawtooth e Hyperledger Fabric em uma aplicação de contrato inteligente para comercialização de dados. A Figura 4 mostra os pseudo-códigos parciais do contrato inteligente da aplicação implementado nas duas plataformas. Quando um cliente envia uma transação de compra, o contrato inteligente lê os saldos dos participantes da negociação e a informação do anúncio previamente armazenado e, em seguida, escreve os novos saldos e a informação referente a compra. A aplicação manipula balanços bancários, registra as negociações na corrente e automatiza o gerenciamento dos dados. Três das quatro aplicações citadas por Hamida *et. al.* como mais representativas de interesse do meio corporativo foram prototipadas em experimentos para representar os cenários mais realistas possíveis [Hamida et al. 2017]. Os protótipos foram implementados em um computador com Intel i9-10900 CPU 2.80 GHz com 32 GB RAM e 20 *threads* de processamento utilizando contêineres Docker. Os resultados apresentam o valor médio com um intervalo de confiança de 95%.

O primeiro experimento avalia a influência da execução serial e paralela de transações no desempenho do Sawtooth. Um cliente envia 800 transações de compra para um nó validador que executa o consenso *dev-mode* dedicado a cenários de desenvolvimento. Esse algoritmo é leve e rápido devido a sua simplicidade e não deve ser usado em ambientes de produção por não oferecer nenhuma segurança. O número máximo de transações por bloco é 1000, como usado em outros trabalhos [Corso 2019]. Primeiro, o cliente envia um grupo de 800 transações, seguido de 2 grupos de 400 transações, e assim por diante. Transações em um mesmo grupo escrevem no saldo de uma mesma

[6]Os protocolos Solo e Kafka do Fabric foram descontinuados de acordo com o repositório em linha (*online*) disponível em https://github.com/hyperledger/fabric/releases. Acesso em: 18/04/2021

```
struct AdvertTransaction {
    // Informacoes do emissor
    OrganizationID    string
    Signature         string

    // Informacoes do anuncio
    TransactionID     string
    Title             string
    Description       string
    Price             string
    DataType          string
    IPAdress          string
}
function ApplyAdvert (AdvertTransaction advert) {
    // Endereco no estado do anuncio
    advertState = getState (TransactionID)
    if advertState alredy exists:
        return error

    // Escreve as informacoes no estado
    putState (TransactionID,advert)
    return success
}
```

(a) Anúncio.

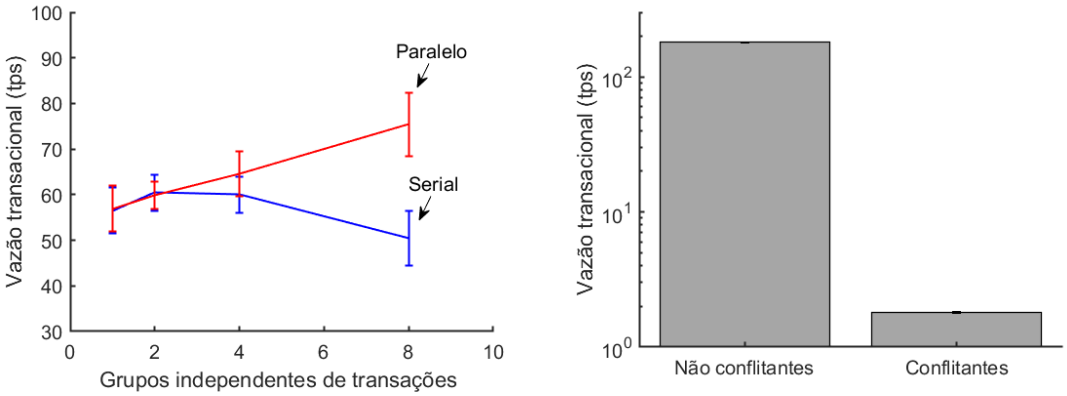
```
struct BuyTransaction {
    // Informacoes do emissor
    OrganizationID    string
    Signature         string

    // Informacoes da compra
    TransactionID     string
    AdvertID          string
    IPAddress         string
}
function ApplyBuy (BuyTransaction buy) {
    // Endereco no estado da compra
    buyState = getState(TransactionID)
    if buyState alredy exists:
        return error

    // Endereco no estado do anuncio
    advertState = getState(AdvertID)
    if advertState does not exist:
        return error
    price = advertState.Price
    AdvertOrgID = advertState.OrganizationID

    // Endereco no estado dos saldos
    buyOrgState = getState (OrganizationID)
    advertOrgState = getState(advertOrgID)
    if buyOrgState < price:
        return error
    else:
        buyOrgBalance = buyOrgState - price
        advertOrgBalance = advertOrgState + price

    // Escreve info. da compra e os saldos
    putState(TransactionID,buy)
    putState(OrganizationID,buyOrgBalance)
    putState(AdvertOrgID,advertOrgBalance)
    return succeess
}
```

(b) Compra.

Figura 4. Pseudo-códigos parciais da transação de anúncio (a) e de transação de compra (b). As transações são estruturas que contêm os campos com as informações para o anúncio ou compra, além da identificação do emissor e sua organização. As funções *getState* e *putState*, fornecidas pela SDK, leem e escrevem dados na corrente de blocos, respectivamente.

organização e, portanto, são conflitantes, enquanto transações de grupos diferentes manipulam saldos de organizações diferentes e não são conflitantes. A Figura 5(a) revela que o processamento paralelo alcança uma vazão até 30% maior que a execução serial. No esquema serial, todas as transações são processadas sequencialmente, formando uma única fila de execução, enquanto que no esquema paralelo, os grupos de transações independentes formam filas que são executadas em paralelo. Portanto, as transações de grupos diferentes são executadas paralelamente, aumentando o desempenho.

O segundo experimento avalia o desempenho do modelo XO do Fabric com transações conflitantes e não conflitantes. A rede contém dois pares e cinco ordenadores com consenso Raft. O número de transações por bloco é 100, como em trabalhos anteriores [Thakkar et al. 2018]. Oito clientes enviam um total de 800 transações concorrentemente e medem o tempo de processamento. A Figura 5(b) mostra que o Fabric processa uma transação concorrente em cada um dos dez blocos e as outras 99 são invalidadas pelo controle de concorrência multiversão, levando a uma vazão 100 vezes menor que no caso de transações não conflitantes. Uma aplicação financeira corporativa com muitos clientes manipulando o saldo de uma mesma organização gera muitas transações conflitantes. Nesse cenário, os pares desperdiçam muitos recursos computacionais com transações inválidas e com as transações reenviadas pelos clientes. Isso demonstra que a



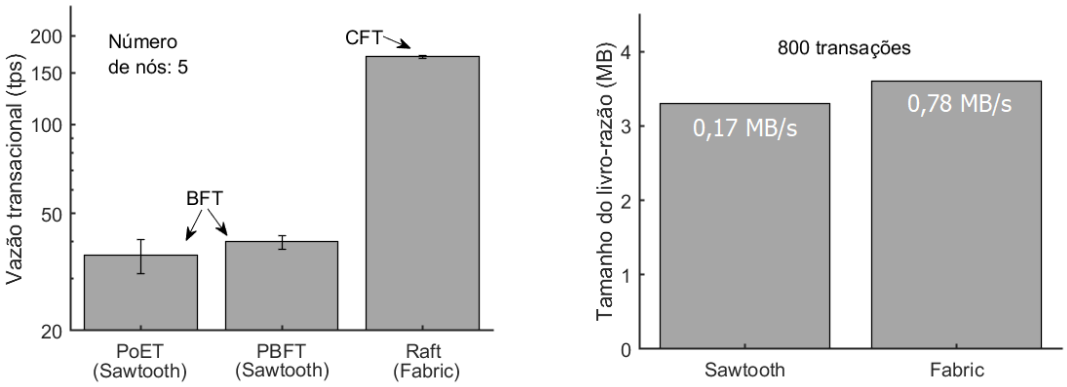
(a) O paralelismo do Sawtooth apresenta um ganho de desempenho em relação ao modelo serial com o aumento no número de transações não conflitantes.

(b) O modelo XO do Fabric apresenta uma vazão maior quando não ocorrem transações conflitantes.

Figura 5. Impacto dos modelos OX do Sawtooth e XO do Fabric no desempenho.

aplicação real apresenta desempenho muito menor que a capacidade de processamento da rede, devido à limitação do modelo XO do Fabric.

O terceiro experimento compara os protocolos de consenso das plataformas em um cenário permissionado. Cinco nós participam no consenso e o Sawtooth executa as transações não conflitantes em paralelo de maneira similar ao Fabric para garantir cenários análogos em ambas as plataformas. A Figura 6(a) mostra que o protocolo de consenso Raft do Fabric alcança uma vazão transacional muito maior por ser computacionalmente mais simples que protocolos tolerantes a falhas bizantinas como o PBFT e o PoET do Sawtooth. O simulador do PoET executa os mesmos procedimentos que sua versão tolerante a falhas bizantinas, apesar de não utilizar ambiente de execução confiável do Intel SGX. O Sawtooth PBFT apresenta uma vazão maior que o Sawtooth PoET em pequenas redes permissionadas. Os resultados do algoritmo PoET apresentam uma barra de erro maior, devido ao tempo intrinsecamente aleatório de cada rodada experimental. Conclui-se que os protocolos PBFT e PoET do Sawtooth, por serem tolerantes a falhas bizanti-



(a) Impacto do protocolo de consenso no desempenho do Fabric e do Sawtooth com cinco participantes.

(b) Os tamanhos das correntes de blocos são similares e a taxa de crescimento no Fabric é maior.

Figura 6. Impacto do protocolo de consenso e comparação do tamanho da corrente de blocos.

nas, são recomendados em cenários que exigem um nível maior de confiança, como no caso de organizações competidoras no mercado interessadas em obter vantagens financeiras [Rebello et al. 2019]. Além disso, o Raft do Fabric apresenta uma vazão 4,2 vezes maior que o PBFT do Sawtooth e 4,7 vezes maior que o PoET do Sawtooth, mas oferece menor garantia de segurança por tolerar apenas falhas de parada.

A Figura 6(b) compara o tamanho das correntes de blocos do Sawtooth e do Fabric em MB após processar as mesmas 800 transações válidas. Apesar dos estados globais serem diferentes, esses resultados revelam que o armazenamento utilizado pelo livro-razão do Fabric é similar ao do Sawtooth, quando as plataformas registram um mesmo número de transações. A A Figura 6(b) mostra que a taxa de crescimento da corrente de blocos do Fabric, em MB por segundo, é maior do que a do Sawtooth. Isso ocorre porque a vazão transacional do Fabric é maior. É importante ressaltar que o Fabric armazena também as transações inválidas, diferentemente do Sawtooth. Portanto, os nós do Sawtooth consomem menos armazenamento em disco que os nós do Fabric para o registro do livro-razão.

5. Conclusões

O artigo apresentou as principais características das plataformas Hyperledger Sawtooth e Fabric para o desenvolvimento de contratos inteligentes. A Tabela 1 resume diferenças importantes entre as ferramentas que devem ser levadas em consideração no processo de escolha da corrente de blocos permissionada a ser adotada em cada cenário. A avaliação de desempenho das plataformas se baseou em um caso de uso de comercialização de dados em redes permissionadas, que reflete bem as necessidades práticas do meio corporativo, e um contrato inteligente foi implementado para esta fina-

Tabela 1. Tabela comparativa entre plataformas.

Critério de comparação	Hyperledger Sawtooth	Hyperledger Fabric
Infraestrutura e rede	Validadores, rede par-a-par e cliente	Organizações, nós ordenadores, nós pares, autoridades certificadoras e canais
Estado global	Árvore de Merkle	Estrutura chave-valor
Processamento das transações	Clientes enviam transações e todos os validadores as ordenam, executam e aplicam em blocos, nessa ordem	Clientes enviam transações, pares de endosso executam as transações, nós ordenadores ordenam as transações em blocos e os pares aplicam o bloco de transações, nessa ordem
Contrato Inteligente	Implementação na forma de família de transações. Todos os validadores devem possuir o mesmo conjunto de famílias de transações	Implementação na forma de chaincodes. Pares podem instalar chaincodes diferentes
Consenso	PoET e PBFT	Raft
Privacidade	Participantes identificados com suas chaves públicas. Não há o conceito de canal	Participantes identificados pelas autoridades certificadoras. Canais com correntes de blocos diferentes.
Permissionamento	Configurações locais e globais dos validadores permitem implementar redes públicas	Provedor de serviço de associação define as permissões de cada identidade. Não há a possibilidade de implementar redes públicas, pois todos os participantes precisam se identificar como membros de uma organização.

lidade. Os resultados dos experimentos demonstraram que o Sawtooth apresentou uma vazão 30% maior com o processamento paralelo, quando o número de conflitos permanece baixo. Ademais, o consenso Raft do Fabric, apesar de vulnerável a agentes maliciosos, apresentou um desempenho muito maior que os protocolos de consenso do Sawtooth em cenários equivalentes. Entretanto, o modelo XO pode levar a muitas transações inválidas e, consequentemente, a um desempenho muito baixo. Além disso, os nós do Sawtooth vão consumir menos espaço em disco para armazenar as transações, pois não vão registrar transações inválidas. Em ambas as plataformas, o desenvolvedor determina como as informações são lidas ou escritas na corrente de blocos e seleciona o protocolo de consenso adequado para cada cenário dependendo do nível de desempenho e segurança esperados. Os desenvolvedores de contratos inteligentes devem reduzir a frequência com que um mesmo endereço da corrente de blocos é alterado para otimizar o desempenho das aplicações.

Como trabalhos futuros é prevista a extensão da análise comparativa para outras plataformas de corrente de blocos.

Referências

Benahmed, S., Pidikseev, I., Hussain, R., Lee, J., Kazmi, S. A., Oracevic, A., and Hussain, F. (2019). A comparative analysis of distributed ledger technologies for smart contract development. In *2019 IEEE 30th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, pages 1–6. IEEE.

Camilo, G. F. et al. (2020a). A Secure Personal-Data Trading System Based on Blockchain, Trust, and Reputation. In *IEEE Blockchain*.

Camilo, G. F. et al. (2020b). AutAvailChain: Automatic and Secure Data Availability through Blockchain. In *GLOBECOM 2020 - 2020 IEEE Global Communications Conference*, pages 1–6.

Caro, M. P., Ali, M. S., Vecchio, M., and Giaffreda, R. (2018). Blockchain-based traceability in agri-food supply chain management: A practical implementation. In *2018 IoT Vertical and Topical Summit on Agriculture-Tuscany (IOT Tuscany)*, pages 1–4. IEEE.

Chacko, J. A., Mayer, R., and Jacobsen, H.-A. (2021). Why Do My Blockchain Transactions Fail? A Study of Hyperledger Fabric. *arXiv preprint arXiv:2103.04681*.

Corso, A. (2019). Performance Analysis of Proof-of-Elapsed-Time (PoET) Consensus in the Sawtooth Blockchain Framework.

de Souza, L. A. C. et al. (2020). DFedForest: Decentralized Federated Forest. In *2020 IEEE Blockchain*, pages 90–97.

Dinh, T. T. A., Wang, J., Chen, G., Liu, R., Ooi, B. C., and Tan, K.-L. (2017). Blockbench: A framework for analyzing private blockchains. In *Proceedings of the 2017 ACM International Conference on Management of Data*, pages 1085–1100.

Gorenflo, C., Golab, L., and Keshav, S. (2020). XOX Fabric: A hybrid approach to blockchain transaction execution. In *2020 IEEE ICBC*, pages 1–9. IEEE.

Hamida, E. B., Brousmiche, K. L., Levard, H., and Thea, E. (2017). Blockchain for enterprise: overview, opportunities and challenges. In *The Thirteenth International Conference on Wireless and Mobile Communications (ICWMC 2017)*.

Muller, R. R. H. (2019). Demystification of the blockchain phenomenon: A matter of state management. Master's thesis, Radboud University.

Nakamoto, S. (2008). Bitcoin: A peer-to-peer electronic cash system. Disponível em https://bitcoin.org/bitcoin.pdf. Acessado em 18 de abril de 2021.

Palma, L. M. et al. (2019). Blockchain and smart contracts for higher education registry in Brazil. *IJNM*, 29(3):e2061.

Polge, J., Robert, J., and Le Traon, Y. (2021). Permissioned blockchain frameworks in the industry: A comparison. *ICT Express*, 7(2):229–233.

Rasolroveicy, M. and Fokaefs, M. (2020). Performance evaluation of distributed ledger technologies for iot data registry: A comparative study. In *2020 Fourth World Conference on Smart Trends in Systems, Security and Sustainability (WorldS4)*, pages 137–144. IEEE.

Rauchs, M., Blandin, A., Bear, K., and McKeon, S. B. (2019). 2nd global enterprise blockchain benchmarking study. *Available at SSRN 3461765*.

Rebello, G. A. F., Alvarenga, I. D., Sanz, I. J., and Duarte, O. C. M. B. (2019). BSec-NFVO: A Blockchain-Based Security for Network Function Virtualization Orchestration. In *2019 IEEE International Conference on Communications (ICC)*, pages 1–6.

Rebello, G. A. F., Camilo, G. F., Guimaraes, L., de Souza, L. A. C., and Duarte, O. (2020). Security and performance analysis of quorum-based blockchain consensus protocols. Technical report, Electrical Engineering Program, COPPE/UFRJ.

Rebello, G. A. F. et al. (2019a). Providing a sliced, secure, and isolated software infrastructure of virtual functions through blockchain technology. In *IEEE HPSR*, pages 1–6.

Rebello, G. A. F. et al. (2019b). Segurança na Internet do Futuro: Provendo Confiança Distribuída através de Correntes de Blocos na Virtualização de Funções de Rede. *Minicursos do SBRC*, 2019.

Thakkar, P., Nathan, S., and Viswanathan, B. (2018). Performance Benchmarking and Optimizing Hyperledger Fabric Blockchain Platform. In *IEEE MASCOTS*, pages 264–276. IEEE.

van Schaik, S., Kwong, A., Genkin, D., and Yarom, Y. (2020). SGAxe: How SGX fails in practice.

Wang, R., Ye, K., Meng, T., and Xu, C.-Z. (2020). Performance evaluation on blockchain systems: A case study on ethereum, fabric, sawtooth and fisco-bcos. In *International Conference on Services Computing*, pages 120–134. Springer.

Wood, G. et al. (2014). Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper*, 151(2014):1–32.