UNIVERSIDADE DE SÃO PAULO
Instituto de Ciências Matemáticas e de Computação

Avaliação de Desempenho do Blockchain Hyperledger Fabric com Dados Médicos Heterogêneos

Ana Caroline Fernandes Spengler

Dissertação de Mestrado do Programa de Pós-Graduação em Ciências de Computação e Matemática Computacional (PPG-CCMC)

ICMC USP
SÃO CARLOS

Ana Caroline Fernandes Spengler

Avaliação de Desempenho do Blockchain Hyperledger Fabric com Dados Médicos Heterogêneos

Dissertação apresentada ao Instituto de Ciências Matemáticas e de Computação – ICMC-USP, como parte dos requisitos para obtenção do título de Mestre em Ciências – Ciências de Computação e Matemática Computacional. *VERSÃO REVISADA*

Área de Concentração: Ciências de Computação e Matemática Computacional

Orientador:: Prof. Dr. Paulo Sérgio Lopes de Souza

USP – São Carlos
Agosto de 2022

Ficha catalográfica elaborada pela Biblioteca Prof. Achille Bassi
e Seção Técnica de Informática, ICMC/USP,
com os dados inseridos pelo(a) autor(a)

S744a

Spengler, Ana Caroline Fernandes
Avaliação de Desempenho do Blockchain Hyperledger Fabric com Dados Médicos Heterogêneos / Ana Caroline Fernandes Spengler; orientador Paulo Sérgio Lopes de Souza. -- São Carlos, 2022.
223 p.

Dissertação (Mestrado - Programa de Pós-Graduação em Ciências de Computação e Matemática Computacional) -- Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, 2022.

1. Blockchain. 2. Avaliação desempenho. 3. Hyperledger Fabric. 4. Hyperledger Caliper. 5. Dados Médicos. I. Lopes de Souza, Paulo Sérgio , orient. II. Título.

Bibliotecários responsáveis pela estrutura de catalogação da publicação de acordo com a AACR2:
Gláucia Maria Saia Cristianini - CRB - 8/4938
Juliana de Souza Moraes - CRB - 8/6176

Ana Caroline Fernandes Spengler

Performance Evaluation of Hyperledger Fabric blockchain with Heterogeneous Medical Data

Dissertation submitted to the Instituto de Ciências Matemáticas e de Computação – ICMC-USP – in accordance with the requirements of the Computer and Mathematical Sciences Graduate Program, for the degree of Master in Science. *FINAL VERSION*

Concentration Area: Computer Science and Computational Mathematics

Advisor: Prof. Dr. Paulo Sérgio Lopes de Souza

USP – São Carlos
August 2022

AGRADECIMENTOS

Quero começar esses agradecimentos destacando que as palavras aqui escritas definem apenas uma parcela da minha gratidão. Espero ter a oportunidade de agradecer mais vezes.

A primeira pessoa que eu preciso agradecer é a minha mãe, Sandra, um exemplo de perseverança, foco e dedicação. Sempre me incentivou a tentar o meu máximo e o meu melhor e nunca me permitiu desistir dos meus sonhos, concordando com eles ou não. Meu pai, Paulo, vem em seguida e a ele eu agradeço por sempre me incentivar a sonhar grande e nunca duvidar da minha capacidade de alcançar meus objetivos. Não posso deixar de agradecer ao Rubens, por me mostrar como a vida é cheia de escolhas e que cada um deve trilhar o seu caminho. Acredito que minha curiosidade de entender a capa "Tanenbaum" me fez escolher a computação como profissão.

Deixo um agradecimento especial para os meus avós. Apesar de não estarem comigo em vida, meu vô Donato e minha vó Dalva, acredito que estão me observando e iluminando minhas escolhas. A falta que vocês fazem é enorme, mas as marcas que vocês deixaram em mim são eternas. Agradeço também a vó Olidia, que sempre me incentivou a estudar e sempre defendeu que estudar também é trabalho. Você me ensina por meio de exemplos.

Desde que eu cheguei em São Carlos tive a sorte de encontrar e seguir encontrando pessoas incríveis. Morar sozinha e tão longe da família é um desafio, mas foi facilitado pelo apoio e carinho dessas pessoas maravilhosas. Para citar alguns, Alex Barbosa, Paulo Bardes, Anayã, Gil Barbosa, Laís Chiachio, Luna Gallo, Lucas Chápeu, Matheus Tornelli, Lucas Ribeiro, Leo Sampaio, Guilherme Perego, Helena Aravechia, André Missaglia. Vocês acolheram essa sul-mato grossense perdida em terras paulistas e diariamente fizeram minha vida melhor.

Participar do LaSDPC foi um desafio que compensou com muito crescimento profissional e pessoal. Agradeço aos meus colegas de laboratório pelas dicas e sugestões durante o desenvolvimento do projeto. Em especial, agradeço o Guilherme Martins e Davi Conte pelos momentos de risadas e descontração do team TBL. Aos professores, agradeço pelos ensinamentos e suporte técnico. Também quero agradecer amizades que a pós-graduação me trouxe. Leo Natan e Lucas Lagoa, as atualizações de vida no caminho do bandejão fizeram a pandemia mais tolerável.

Por último agradeço ao meu orientador, Paulo Sérgio de Souza. Além de ser o professor que eu fiz mais matérias durante a graduação, também aceitou me orientar desde o projeto de TCC até esse momento. Agradeço pela oportunidade do mestrado e pelos direcionamentos dados durante a realização desse projeto, tanto de caráter profissional quanto pessoal.

“Understand well as I may,
my comprehension can only be
an infinitesimal fraction of all
I want to understand.”
(Ada Lovelace)

RESUMO

SPENGLER, A. C. F. **Avaliação de Desempenho do Blockchain Hyperledger Fabric com Dados Médicos Heterogêneos**. 2022. 223 p. Dissertação (Mestrado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2022.

Blockchain nasceu com o surgimento do *Bitcoin*, apresentando-se como uma infraestrutura para o armazenamento distribuído de transações financeiras com a garantia de imutabilidade. Pelas suas características, a *blockchain* passou a ser utilizado em diversos contextos de aplicações que podem se beneficiar dessa infraestrutura. Alguns exemplos dessas aplicações são: distribuição de mercadorias, educação e saúde. Essas aplicações apresentam demandas distintas daquelas focadas em transações financeiras, como manipulação de diferentes tipos de dados. Diante disso, é importante investigar e demonstrar o desempenho da *blockchain* frente aos seus diferentes componentes e a dados oriundos de outros domínios de aplicação, como a área da saúde. O objetivo deste trabalho é verificar o desempenho da *blockchain* sobre diferentes perspectivas, considerando aspectos distintos da rede quando utilizadas aplicações que demandam dados heterogêneos. A metodologia utilizada na avaliação considera o uso do *Hyperledger Fabric* para estabelecer uma rede de compartilhamento de informações médicas obtidas de uma base de dados reais. O desempenho dessa rede foi mensurado com estudos experimentais, por meio do *benchmark Hyperledger Caliper*, considerando métricas de vazão e latência para diferentes fatores contemplando o armazenamento, fluxo de dados, processo de criação de blocos, consenso e rede. Dentre os principais resultados dos estudos experimentais destacam-se: (i) associar a *blockchain* a um banco de dados aumenta a latência e diminui a vazão do sistema; (ii) o processo de criação de blocos é um fator determinístico no desempenho da aplicação *blockchain* quando são usados dados heterogêneos. Os resultados obtidos buscam auxiliar desenvolvedores de aplicações *blockchain* ao apontar aspectos decisivos para o desempenho de tais aplicações.

Palavras-chave: Blockchain, Avaliação desempenho, Hyperledger Fabric, Hyperledger Caliper, Dados Médicos.

ABSTRACT

SPENGLER, A. C. F. **Performance Evaluation of Hyperledger Fabric blockchain with Heterogeneous Medical Data**. 2022. 223 p. Dissertação (Mestrado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2022.

Blockchain was born with the emergence of Bitcoin, presenting itself as an infrastructure for the distributed storage of financial transactions with the guarantee of immutability. Due to its characteristics, the blockchain started to be used in several contexts of applications that can benefit from this infrastructure. Some examples of these applications are distribution of goods, education, and healthcare. These applications present demands different from those focused on financial transactions, such as handling different types of data. Therefore, it is important to investigate and demonstrate the performance of the blockchain against its different components and data from other application domains, such as healthcare. The objective of this work is to verify the performance of blockchain over different perspectives, considering different aspects of the network when used applications that demand heterogeneous data. The methodology used in the evaluation considers the use of Hyperledger Fabric to establish a network for sharing medical information obtained from a real database. The performance of this network was measured through experimental studies, by means of the benchmark Hyperledger Caliper, considering throughput and latency metrics for different factors contemplating storage, data flow, block creation process, consensus and network. Among the main results of the experimental studies we highlight: (i) associating blockchain with a database increases latency and decreases system throughput; (ii) the block creation process is a deterministic factor in the performance of the blockchain application when heterogeneous data is used. The obtained results seek to assist developers of blockchain applications by pointing out aspects decisive for the performance of such applications.

Keywords: Blockchain, Performance Evaluation, Hyperledger Fabric, Hyperledger Caliper, Medical Data.

LISTA DE ILUSTRAÇÕES

LISTA DE QUADROS

LISTA DE TABELAS

SUMÁRIO

CAPÍTULO

1

INTRODUÇÃO

O *Bitcoin* foi lançado em 2009 com o objetivo de eliminar a centralidade que instituições bancárias e governamentais representam sobre as transações financeiras. Isso se deve, principalmente, aos altos custos que bancos e outras instituições financeiras impõem sobre cada transação realizada dentro do seu domínio (SWAN, 2015). A grande inovação do *Bitcoin* é oferecer uma solução que dispensa esse terceiro elemento centralizador, propondo uma nova maneira distribuída de se operar tais transações.

Nesse cenário, os participantes estão espalhados em uma rede *Peer-to-Peer* em que a validade e a segurança das transações são garantidas através de um processo de consenso. As transações realizadas são consideradas imutáveis e estão visíveis a todos os integrantes da rede. A estrutura na qual as transações são armazenadas é chamada de *blockchain* (NAKAMOTO, 2009). É através desta estrutura que há a garantia de imutabilidade dos registros. Essa maneira de armazenar os dados é um dos principais subprodutos do lançamento do *Bitcoin*.

O uso da *blockchain* já está consolidado para aplicações financeiras, com o *Bitcoin* e outras criptomoedas ganhando cada dia mais destaque, se tornando relevantes para a economia de modo geral (EXAME, 2022). A *blockchain* foi desassociado do *Bitcoin* e passou a ser analisado em outros contextos além do financeiro. Propostas de uso na distribuição e controle de mercadorias, na transação de bens materiais e imateriais, na educação e na saúde estão sendo colocadas em práticas em todo o globo. Transações financeiras entre países podem se beneficiar com o uso da *blockchain*, pois com o seu uso é possível o acesso aos dados das transações através de uma única interface. Além disso, é viável acompanhar o *status* das transações a qualquer momento (GUPTA, 2017a).

Alguns exemplos de uso da *blockchain* são litados a seguir. Como a iniciativa da *IBM* anunciada em 2016 com o objetivo de facilitar transações entre bancos internacionais (D'ALTO, 2017). Essa iniciativa deriva da dificuldade de troca de valores entre países pela diferença da cotação das moedas, assim sendo, as transações demoram dias e precisam de vários intermediários

para serem concluídas. Com o uso da *blockchain* é possível que todos os participantes tenham acesso aos registros de forma global e de forma mais eficiente, eliminando a necessidade de intermediários nessas transações.

Uma iniciativa relacionada a redes de suplemento, também da *IBM* porém em parceria com *Mizuho Bank* [1]. Com essa iniciativa, as organizações pretendem melhorar o serviço de entrega de mercadorias, pois atualmente existe uma grande diferença entre o tempo de pagamento e recebimento de um produto. Além disso, tanto a entrega quanto o pagamento de um produto são realizados por meio de vários intermediários. Para melhorar a comunicação entre as partes do processo, usa-se a *blockchain* visando deixar os registros visíveis a todos os participantes no intuito de deixar o processo mais seguro e mais econômico. Um dos principais benefícios do uso da *blockchain* nesse contexto é o aumento da confiança, pois nenhuma das partes tem o controle total da aplicação e das informações. Também há melhora na eficiência pela facilidade de verificação do *status* do produto a ser compartilhado (GUPTA, 2017a).

Outro contexto em que a *blockchain* está sendo utilizado é em aplicações na educação. Uma iniciativa conhecida é a do governo do Quênia em fazer uso da *blockchain* para o registro dos dados das escolas do país (BORE *et al.*, 2017). O projeto determina que os dados de todos os alunos sejam mantidos de forma a possibilitar o acompanhamento do histórico escolar, faltas, e cursos realizado por esses alunos. Com essa iniciativa, permite-se um melhor controle da distribuição de recursos entre as escolas, avaliar a evasão escolar e verificar onde os recursos disponíveis estão sendo aplicados.

A *blockchain* tem um grande potencial para ser aplicado também na área médica, em que podem ser citadas ferramentas que utilizam dessa tecnologia para atender as demandas na interoperabilidade dos dados (Esposito *et al.*, 2018). O uso da *blockchain* é extremamente importante nessa área em virtude da necessidade de compartilhamento seguro de informações de pacientes, fato que contribui para melhores diagnósticos, e economia em exames e procedimentos. Trabalhar com dados médicos na *blockchain* é um desafio, pois, além da necessidade de compartilhar os dados heterogêneos, há uma série de garantias que necessitam ser atendidas por serem dados sensíveis. Com a *blockchain* é possível criar aplicações que garantam o compartilhamento de dados entre as diferentes instituições médicas e também certifiquem privacidade e segurança dos dados, essenciais para essa finalidade. Além disso, essas aplicações permitem que as instituições e pacientes dividam a responsabilidade sobre os dados, de forma que não seja necessário um terceiro órgão para controlar o acesso e distribuição desses dados.

De fato, as aplicações *blockchain* estão sendo usadas na área da saúde tanto para registro de dados médicos dos pacientes como para acompanhamento de consultas e procedimentos pelo hospital e planos de saúde (HöLBL *et al.*, 2018). No primeiro caso, o uso da *blockchain* permite o compartilhamento de dados dos pacientes entre médicos, consultórios e hospitais de maneira

[1] https://newsroom.ibm.com/2017-04-26-Mizuho-Financial-Group-Mizuho-Bank-and-IBM-Japan-Building-Trade-Financial-Platform-Using-Blockchain

segura e persistente, além de possibilitar que o paciente controle o acesso aos seus dados por outras instituições (GUPTA, 2017a). Em relação ao segundo caso, os planos de saúde têm maior controle sobre os procedimentos realizados pelos usuários e podem monitorar quais serviços são mais utilizados e oferecer planos mais específicos (GUPTA, 2017a).

Azaria *et al.* (2016) propôs um sistema descentralizado para o gerenciamento de dados médicos com a finalidade de fornecer aos pacientes um acesso fácil e amplo aos seus registros médicos. Nesse sistema, a *blockchain* é utilizado por permitir que os dados sejam armazenados de forma imutável ao mesmo tempo em que possibilita acesso paralelo aos registros, pela rede distribuída *Peer-to-Peer*. A plataforma *MedRec* que oferece um acesso distribuído aos dados médicos, em um sistema transparente voltado ao paciente. A proposta também pretende atender a demanda de pesquisadores por dados médicos. Com a *blockchain* é possível distribuir os dados para institutos de pesquisa garantindo o anonimato dos pacientes.

Outro exemplo do uso da *blockchain* na área médica é a aplicação concebida em (YUE *et al.*, 2016), o *Healthcare Data Gateway (HGD)*. Este sistema surgiu com a intenção de alavancar melhorias nos atendimentos de serviços médicos, em relação à qualidade do atendimento, de forma a facilitar o acesso aos dados do paciente e encontrar tendências relativas ao tipo de serviços prestados. A aplicação foi desenvolvida com a finalidade de permitir armazenamento e a distribuição de dados médicos sem o risco de violações a privacidade dos pacientes. Nessa proposta, o controle de compartilhamento de dados é responsabilidade do paciente o qual tem poder de definir quem tem acesso aos seus dados e por quanto tempo.

Com a análise dessas propostas de uso da *blockchain* verifica-se a ausência de avaliações em relação à aplicabilidade das mesmas. Desta forma, uma busca foi realizada, a fim de averiguar as propostas com relação ao desempenho de aplicações *blockchain*. Por meio dessa investigação foi possível notar as tendências em relação a quais aspectos da *blockchain* estão sendo analisados e quais são os objetivos dessas avaliações.

Outro aspecto que notado na busca realizada foram os tipos de aplicações que estão sendo avaliadas nesses estudos. Existe uma lacuna relacionada ao uso de dados heterogêneos, principalmente dado a pouca variedade de tipos de dados e tamanho dos mesmos, o que possibilita averiguação do uso da *blockchain* nesse contexto.

O objetivo geral deste trabalho é avaliar o desempenho da *blockchain* em aplicações para a distribuição de dados heterogêneos, considerando aplicações distintas do mercado financeiro (*Bitcoin*). Os objetivos específicos inclui investigar os diversos componentes da estrutura da *blockchain* para o armazenamento de dados heterogêneos, com a finalidade de também verificar os gargalos da rede *blockchain* através da análise do desempenho. Além disso, investigar o uso da *blockchain* no armazenamento de dados heterogêneos por um aspecto funcional, com foco no desempenho.

Este trabalho considera o uso da plataforma *Hyperledger Fabric* para criar uma rede

blockchain para o compartilhamento de dados heterogêneos encontrados em uma base de dados médicos reais. Esse cenário de aplicação foi escolhido por apresentarem essas características que os diferenciam dos tipos de dados cujo uso da *blockchain* já está consolidado, como em transações financeiras.

A rede *blockchain* foi avaliada por meio de estudos experimentais, com a coleta da vazão e da latência através do *benchmark Hyperledger Caliper*. Diferentes aspectos da estrutura da *blockchain* foram avaliados,sendo eles o armazenamento de dados, criação de blocos, consenso da rede, tamanho da rede e distribuição dos nós. Esse estudo também analisou o fluxo de dados para a investigação de um aspecto funcional da *blockchain*.

Os principais resultados obtidos nos experimentos realizados são relacionados ao armazenamento de dados e ao processo de criação dos blocos. O experimento que investiga o uso da *blockchain* com o banco de dados mostra o reflexo dessa associação no desempenho da rede, com o aumento da latência e diminuição da vazão para as operações de inserção de dados. Já a análise da criação de blocos mostra como o tamanho do bloco e seu intervalo de criação são determinísticos no desempenho de uma aplicação *blockchain* quando são usados dados heterogêneos.

A contribuição deste projeto de mestrado é quantificar o desempenho da *blockchain* com o uso de dados heterogêneos do contexto da saúde. Através dos experimentos conduzidos, identificou-se quais componentes da *blockchain* que têm maior influência no desempenho da rede considerando os cenários analisados. Para tanto, foram estabelecidos parâmetros representativos para os experimentos, tendo como base o contexto do *Bitcoin* e de um hospital do porte do Hospital das Clínicas (HC) da Faculdade de Medicina (FM) da USP, segundo dados de 2014 (IMPRENSA, 2014; CROMAN *et al.*, 2016).

Os resultados obtidos buscam auxiliar desenvolvedores de aplicações *blockchain* ao apontar aspectos decisivos para o desempenho de tais aplicações.

Esta dissertação está estruturada em nove capítulos, o primeiro é a introdução, no qual o contexto, motivação e objetivo foram apresentados. No segundo capítulo a *blockchain* é introduzida, abordando como os dados são inseridos na cadeia e a sua distribuição da cadeia na rede *Peer-to-Peer*. Por último, o capítulo apresenta plataformas que dão suporte ao desenvolvimento de aplicações *blockchain*, com destaque ao *Hyperledger Fabric* e *Hyperledger Caliper*.

O terceiro capítulo relata o levantamento bibliográfico que investiga como o teste e verificação da infraestrutura da *blockchain* são executados. São apresentadas as etapas realizadas no levantamento, em seguida a descrição e análise dos trabalhos encontrados e finaliza com o cenário atual desses trabalhos em conjunto com as lacunas encontradas.

Já o quarto capítulo enumera os experimentos desenvolvidos durante a realização deste trabalho e ressalta aspectos comuns entre eles, tais como a metodologia e métricas de avaliação.

Os capítulos cinco, seis, sete e oito são dedicados ao detalhamento dos experimentos

realizados. As etapas de execução de cada experimento é detalhada e os resultados obtidos são exibidos de duas formas, com tabelas e gráficos.

O nono capítulo conclui o trabalho, destacando os principais resultados encontrados de cada experimento. Por fim, os artigos publicados, as limitações do trabalho realizado e possíveis trabalhos futuros são apresentados.

CAPÍTULO

2

BLOCKCHAIN

2.1 Considerações Iniciais

Este capítulo tem como objetivo apresentar o referencial teórico no qual este trabalho se baseia. Para isso, será apresentado o cenário que levou ao lançamento do *Bitcoin* e a sua evolução como ferramenta, não só para o setor financeiro, mas também para o compartilhamento de registros de modo geral.

Além disso, descreve o funcionamento da infraestrutura necessária para garantir a manutenção, segurança e a credibilidade dos dados armazenados e aponta as qualidades e limitações desta infraestrutura. Para concluir, algumas plataformas de desenvolvimento de aplicações *blockchain* são retratadas.

2.2 Histórico

A presente Seção tem a finalidade de apresentar o contexto no qual o *Bitcoin* e a *blockchain* foram lançados. Com isso, introduzem-se os fatores que motivaram o lançamento dessa criptomoeda e qual o impacto dessa tecnologia nas formas de pagamento e transações de modo geral. No final, é relatada a evolução do *blockchain* fora do contexto do *Bitcoin*.

O primeiro estudo que menciona o termo *Bitcoin* é de autoria de Nakamoto (2009). Em seu artigo intitulado *"Bitcoin: A Peer-to-Peer Electronic Cash System"*, publicado em 2009, o autor introduziu ao mundo o *Bitcoin*. A promessa do autor foi a realização de transações monetárias sem a necessidade de envolver um órgão regulador, pois o protocolo desenvolvido teria a capacidade de regular e validar as transações realizadas.

No referido artigo, o termo *Bitcoin* descreve o protocolo *Bitcoin*, cuja finalidade é possibilitar a transferência da criptomoeda *bitcoin* (*BTC*). Além disso, definiu-se o termo *blockchain*, o nome dado para a cadeia de dados distribuído, base do funcionamento do protocolo (SWAN,

2015). A infraestrutura da *blockchain* será detalhada na próxima seção.

Neste trabalho, o *Bitcoin* será bastante utilizado para exemplificar como a *blockchain* garante segurança e imutabilidade dos dados. Quando o termo *Bitcoin* inicia-se com *B* maiúsculo refere-se ao protocolo e, quando *bitcoin* inicia-se com *b* minúsculo refere-se à moeda.

Transações financeiras *online* têm a necessidade de um terceiro elemento/agente, tais como bancos, controladoras de cartão de crédito, para atuar como intermediador. Esse elemento intermediador tem a função de dar credibilidade às operações feitas, garantindo, inclusive que as transações sejam registradas apenas uma vez. A motivação por trás da criação do *Bitcoin*, conforme Nakamoto (2009), é intenção de eliminar a necessidade dessa terceira parte.

Apesar de sistemas centralizados serem uma forma simples de controlar e validar as transações, também é uma solução custosa. Isso se deve à série de garantias que esse órgão regulador oferece aos participantes da transação (SWAN, 2015). A inevitabilidade de intermediar disputas e o risco de fraude de estorno de pagamentos são fatores que impactam diretamente no custo das operações. O elevado custo acarreta na limitação do tamanho e da quantidade de transações a serem realizadas (NAKAMOTO, 2009).

Além disso, como as transações se apoiam em um sistema central, há uma maior fragilidade contra fraudes e *cyberattacks*, que expõe todos os participantes no caso desse sistema ser comprometido (GUPTA, 2017a). Com isso, não só as transações correm o risco de serem expostas, como também as informações contratuais de clientes e comerciantes que realizaram essas transações.

Nesse cenário, o *Bitcoin* propõe uma alternativa para a redução desses custos operacionais. Os pagamentos realizados nesse sistema são registrados em uma cadeia de dados pública distribuída em uma rede *Peer-to-Peer* na qual a maioria dos computadores membros da rede e, permanecem continuamente visíveis (SWAN, 2015). A segurança é garantida com o armazenamento cronológico das transações e, com a validação de toda a rede sobre a legitimidade da transação (NAKAMOTO, 2009).

Os *bitcoins* são criados como recompensa pelo trabalho de processamento computacional, conhecido como mineração, em que os participantes oferecem poder computacional, resolvendo um *puzzle* conhecido como *Proof-of-Work*, para verificar e registrar pagamentos na cadeia de dados pública (NAKAMOTO, 2009). Os participantes que interagem nesse processo são conhecidos como mineradores e recebem em troca pelo serviço prestado taxas de transação (*Transactions Fees*) e os *bitcoins* recém criados (SWAN, 2015). Com a intenção de prevenir possíveis ataques, a rede do *Bitcoin* se apoia nesses incentivos para manter os participantes honestos. Devido ao alto custo do processo de validação das transações a desonestidade se torna desvantajosa, uma vez que o beneficio recebido na realização desse processo, as taxas de transação coletados e as moedas criadas são perdidos, porém o seu alto custo é mantido, já que os cálculos já foram realizados, acarretando em prejuízos para nós fraudulentos (NAKAMOTO,

2009).

O *Bitcoin* não foi a primeira inciativa que almejava realizar pagamentos sem a necessidade de um órgão regulador em ambiente distribuído, porém foi a primeira que conseguiu resolver dois problemas até então insolúveis. O primeiro problema se refere à falta de mecanismos que impedissem o gasto de uma quantia já não pertencente ao remetente de uma transação. Essa questão é conhecida como *double-spending*. O segundo, é o Problema dos Generais Bizantinos (*Byzantine Generals Problems*), que diz respeito à dificuldade de múltiplas partes que não confiam uma nas outras em manter um mecanismo de comunicação coordenada (NAKAMOTO, 2009).

O impacto do artigo Nakamoto (2009) não se limita às transações financeiras com o *bitcoin*, mas se estende ao protocolo por trás dessa inovação. Esse protocolo é uma combinação de fatores tais quais as redes *Peer-to-Peer*, que permitem a descentralização do sistema; uma cadeia pública que armazena os dados chamada de *blockchain*; o conjunto de regras de consenso para a validação das transações; e o mecanismo de autenticação global descentralizado para validar a cadeia de registros (o algoritmo de *Proof-of-Work*) (ANTONOPOULOS, 2017). Esses elementos são explicados na próxima seção.

O lançamento do *Bitcoin*, é conhecido como *blockchain 1.0* sendo caracterizado como a primeira inovação da *blockchain*. A evolução da *blockchain* é chamada de *blockchain 2.0* e é marcada pelo distanciamento do *Bitcoin* a fim de aproveitar a funcionalidade da *blockchain* como dinheiro programável (GUPTA, 2017b).

A *blockchain 3.0*, é marcada pela criação do *smart contract*, apoiado pela plataforma *Ethereum* (GUPTA, 2017b). O *Ethereum* é uma plataforma de desenvolvimento de aplicações *blockchain*. Nessa plataforma, é possível criar aplicações usando uma infraestrutura genérica, definindo as regras através de uma especificação denominada de *smart contract*. Essa estrutura funciona como um contrato assinado em cartório, onde as regras são definidas e as partes integrantes em uma transação as seguem. Essa infraestrutura facilita o desprendimento da *blockchain* do contexto de criptomoedas (NARAYANAN *et al.*, 2016).

A *blockchain 4.0*, traz a concepção do *Proof-of-Stake* para substituir o *Proof-of-Work* (GUPTA, 2017b). O *Proof-of-Work* é utilizado no processo de criação de um bloco, a fim de validar os dados a serem armazenados na cadeia global. Apesar da eficácia desse algoritmo, há um alto custo associado a sua execução, o que dificulta a popularização da infraestrutura. O *Proof-of-Stake* apresenta o mesmo objetivo do seu antecessor, garantir segurança e consenso na rede, porém visa à economia de energia, tornando o processo mais economicamente viável (LISK, 2019).

Uma estimativa de avanço da tecnologia é encontrar uma alternativa eficaz de validação das transações. Atualmente, esse processo é realizado por todos os participantes da rede, que sequencialmente verificam todas as transações transmitidas. Esse processo sequencial tem um

custo e se caracteriza como um gargalo no desempenho das aplicações. Dessa forma, a ideia de distribuir esse processo, com o objetivo de eliminar a necessidade da aprovação de todos os nós se apresenta como uma necessidade. Essa nova proposta é chamada de *blockchain scaling* (GUPTA, 2017b).

2.3 Infraestrutura Básica Utilizada pela *Blockchain*

Nesta Seção é apresentada a infraestrutura por trás do *Bitcoin*, a *blockchain* em conjunto com a rede na qual a *blockchain* se alicerça. Em seguida, é abordada como as transações são validadas e distribuídas nessa rede de maneira global.

A *blockchain* recebe esse nome pelo modo como os registros são mantidos: uma sequência de blocos ligados em forma de cadeia. O bloco consiste da estrutura básica do *blockchain*, onde as transações são armazenadas. A *blockchain* mantém os registros de todas as transações desde o primeiro bloco criado, o *genesis block*, até o mais recente adicionado na cadeia (SWAN, 2015).

Essa sequência de blocos está espalhada em uma rede distribuída *Peer-to-Peer* (*P2P*), de forma que a maioria, potencialmente todos, dos participantes têm uma cópia local da cadeia (SWAN, 2015). A escolha de uma rede *P2P* é determinante para o funcionamento distribuído dessa infraestrutura, pois nesse tipo de arquitetura os pares se comunicam de forma direta entre si, sem a necessidade de um servidor central (KUROSE; ROSS, 2007).

Os computadores que contribuem nessa rede são chamados de nós. Em uma rede pública, como a do *Bitcoin*, não há restrições em relação à capacidade e identidades dos nós participantes. Qualquer computador, que tem acesso acesso aos protocolos, pode ser um nó da rede (ANTONOPOULOS, 2017). Existem também protocolos que utilizam de uma rede privada, em que há uma limitação na quantidade e na identidade de nós. Apesar dessa restrição, ao se conhecer os participantes da rede é possível aumentar a gama de dados que podem ser compartilhados (BUTERIN, 2015). A diferença entre elas será detalhada posteriormente na presente seção.

O processo de criação de um bloco é denominado mineração, sendo o mecanismo pelo qual os registros são validados. Esse nome é dado pelo processo se assemelhar ao modo como outros produtos de valor, como ouro, são extraídos, devido ao esforço dedicado nesse trabalho. No caso do *Bitcoin*, esse processo também é responsável pela criação de novas moedas (NARAYANAN *et al.*, 2016).

Os nós que se dedicam a minerar blocos são chamados de nós mineradores. A ideia desse processo é que cada minerador gaste um certo poder computacional e recursos energéticos para resolver um problema matemático. Por esse trabalho realizado, há uma recompensa associada. No *Bitcoin*, o nó recebe uma certa quantia de *bitcoins*, referente à quantidade de moeda criada nesse processo somado à taxa de transferência, denominado *Transactions Fees* (NAKAMOTO,

2009).

O problema computacional popularmente utilizado, e lançado em conjunto com o *Bitcoin*, é o *Proof-of-Work* (*PoW*). Esse algoritmo consiste em um processo de tentativa e erro, por força bruta, a fim de encontrar um resultado pré-determinado. Esse problema tem como propriedade gastar um alto poder computacional para ser executado, porém tem um baixo custo de verificação, o tornando ideal nesse cenário (ANTONOPOULOS, 2017).

Os nós mineradores estão constantemente competindo pela criação de um novo bloco. Quando um novo bloco é criado, ele é transmitido na rede para os outros nós. Quando é determinado que o bloco criado é válido, inicia-se a competição pelo próximo bloco. Ao ser propagado pela rede, o novo bloco é inserido na cópia local da cadeia de todos os nós participantes da rede (NAKAMOTO, 2009).

Antes de inserir o bloco na cópia local, o nó realiza a verificação de cada transação armazenada pelo bloco. Após a execução dessa etapa, se todas as transações foram consideradas válidas, o nó adiciona o bloco na sua cópia local da *blockchain*. A validade de uma transação é determinada pela aplicação.

Dessa maneira é o que o consenso é garantido na rede, pois todos os nós participam da validação das transações. No caso dos nós não concordarem na validade das transações, o bloco não é inserido na cadeia e a recompensa por minerar o bloco é perdida. Com isso, há um incentivo para os nós mineradores só adicionem transações válidas na rede (ANTONOPOULOS, 2017).

Nas próximas subseções, são descritos com mais detalhes cada um desses tópicos.

2.3.1 Blockchain

O bloco é a unidade básica da *blockchain* e cada implementação da *blockchain* tem uma estrutura definida para o bloco. No *Bitcoin* o bloco é formado pelos seguintes componentes:

- **Tamanho do bloco:** tamanho do bloco a ser criado em *bytes*;
- ***Header*:** identificador único de cada bloco na cadeia;
- **Contador de transações:** quantidade de transações armazenadas no bloco;
- **Transações:** as transações a serem registradas no bloco criado.

O *header* de um bloco, além de garantir a unicidade de cada bloco criado em uma rede *blockchain* também tem a função de conectar os blocos em formato de cadeia. Dado essa função, o *header* é composto por:

- **Versão:** número de versão para controle de *software* e protocolos da rede;

- ***Hash* do bloco anterior:** referencia a *hash* do bloco anterior da cadeia. É dessa forma que os blocos são ligados em formato de cadeia e a imutabilidade dos dados é garantida. A forma como a cadeia é criada é descrita nesta Seção.
- **Raiz da *Merkle tree*:** uma estrutura utilizada pelo *Bitcoin* que contém um resumo de todas as transações armazenadas pelo bloco. Como as transações são armazenadas na *Merkle tree* será detalhado nesta Seção.
- ***Timestamp*:** o instante da criação do bloco. Armazenar essa informação prova que as informações armazenadas no bloco existiram no momento da criação do mesmo. Além disso, também define a ordem que os blocos serão adicionados na cadeia.
- ***Difficult Target*:** a dificuldade de mineração usada pelo algoritmo do *Proof-of-Work* na criação do bloco. O *Proof-of-Work* envolve a busca por um valor que quando inserido na *hash* faz com que o seu resultado se aproxime de uma determinada resposta. A resposta esperada é classificada como o *difficult target* e influencia diretamente no ritmo de criação dos blocos.
- ***Nonce*:** contador usado pelo *Proof-of-Work* para atingir um determinado *difficult target*. O *nonce* é incrementado a cada erro na tentativa de acertar o *difficult target*.

A Figura 1 exemplifica os componentes que formam o bloco e seu *header*. Os dois últimos componentes do *header*, a dificuldade de mineração e o *nonce*, são referentes ao processo de mineração de novos blocos através do *Proof-of-Work* no *Bitcoin* e serão explicados no decorrer deste Capítulo.

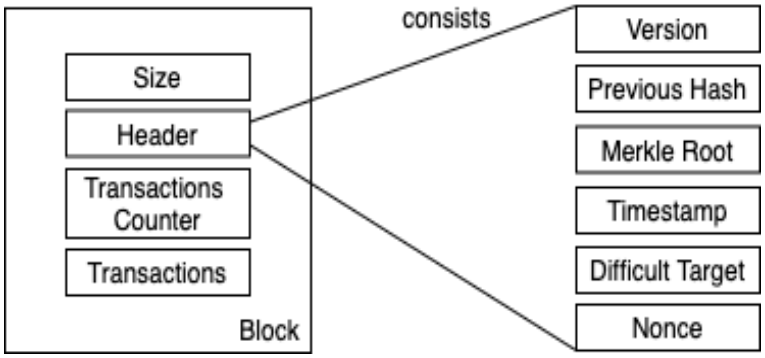


Figura 1 – Composição do bloco

O *header* de um bloco é armazenado em *hash*. Uma *hash* é uma função matemática na qual a entrada pode ter qualquer tamanho, porém a saída apresenta um tamanho fixo. A vantagem desse cálculo é a sua baixa complexidade, o que permite que a saída seja gerada em um tempo computacionalmente razoável (NARAYANAN *et al.*, 2016). No *Bitcoin*, o assume-se como padrão o tamanho *256-bits* para a saída.

A cadeia é mantida intacta pelo fato do identificador único de cada bloco ser o resultado da combinação entre o identificador do bloco anterior, e outros elementos do bloco, calculados por uma função *hash* (NAKAMOTO, 2009). A Figura 2 ilustra a ligação existente entre os blocos através da dependência da *hash* do bloco anterior.

Essa dependência impede que um novo bloco seja adicionado entre dois blocos já existentes, pois para conseguir alterar um bloco é necessário calcular o novo *hash* do bloco a ser alterado, e o *hash* do filho deste bloco, e assim por diante, tornando essa alteração computacionalmente inviável, uma vez que novos blocos estão sendo adicionados constantemente (NAKAMOTO, 2009).

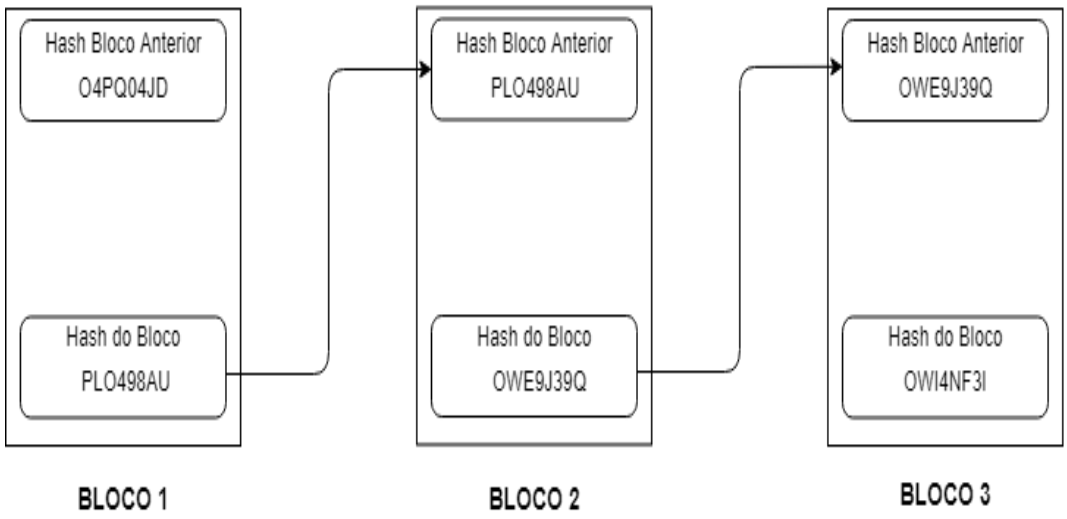


Figura 2 – Ligação entre os blocos na cadeia

A Tabela 1 mostra o resultado gerado após a entrada ser calculada por um função *hash SHA-256*, utilizada pelo *Bitcoin*. Essa Tabela tem como objetivo ilustrar como pequenas alterações na entrada de uma função *hash* alteram completamente a saída.

Entrada	***Hash***
Teste	89F308210C7C7820BAD0974F31E751BFA433D2066A93E808947C3188DEDBA6E3
teste	46070D4BF934FB0D4B06D9E2C46E346944E322444900A435D7D9A95E6D7435F5
teste1	15BF532D22345576B4A51B96DA4754C039EF3458494066D76828E893D69EBD1E

Tabela 1 – Exemplo de resultado da função *hash* para três entradas distintas

Existem duas formas de armazenar os registros na *blockchain*: *On-Chain* e *Off-chain*. Na primeira os registros são salvos inteiramente no bloco. Já na segunda, o bloco contém apenas a referência para as transações, sendo elas armazenadas em um banco de dados externo. O Quadro 1 apresenta os tipos de dados e as vantagens e desvantagens de ambas as formas de armazenamento.

Quadro 1 – Comparação entre as formas de armazenamento

	On-Chain	***Off-Chain***
Tipo de dados	Dados padronizados Tamanho fixo	Tipo de dado flexível Tamanho variável
Prós	Os dados visíveis e reconhecidos	Múltiplos tipos de dados Comunicação com o banco de dados existente
Contras	Limitado pelo tipo e volume dos dados Incapacidade de integração com o sistema vigente	Reconhecimento dos dados depende do conhecimento da aplicação Requer camadas de integração Maior possibilidade de perda de dados

As Figuras 3 e 4 têm como objetivo ilustrar as diferenças entre o armazenamento *On-chain* e *Off-chain* na construção do *hash* armazenado no bloco. Os dados relativos ao bloco, como tamanho do bloco e da cadeia, o *timestamp* e o *hash* do bloco anterior, permanecem compondo o *hash* do bloco, independente da forma de armazenamento escolhida. A diferença entre as formas de armazenamento é relativa aos dados. No caso do *On-chain* os dados formam uma parte desse *hash*, já no *Off-chain* apenas uma referência aos dados compõe o *hash* formado.

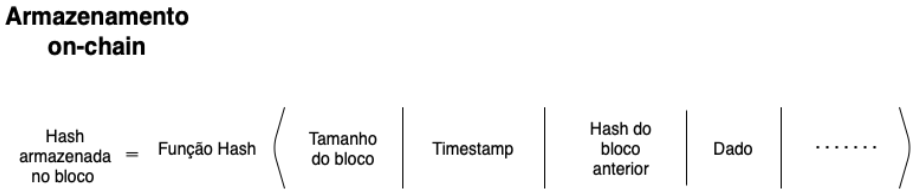


Figura 3 – *Hash* do armazenamento *On-Chain*

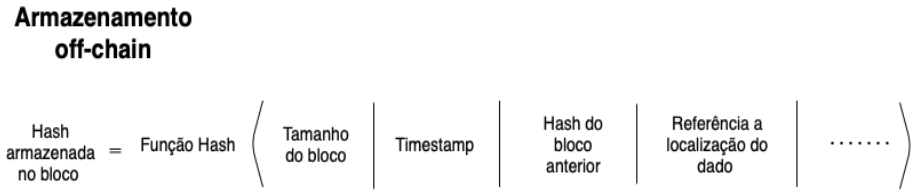


Figura 4 – *Hash* do armazenamento *Off-Chain*

No *Bitcoin* as transações são armazenadas dentro da cadeia em uma estrutura de árvore chamada de *Merkle Tree*. As transações são armazenadas em formato de *hash*. Primeiro, cada transação tem o seu formato modificado para *hash*, depois, elas são combinadas duas a duas e têm o seu formato modificado para *hash* mais uma vez. Esse processo ocorre sucessivamente até que reste apenas uma única *hash*, que será salva no bloco. Essa *hash* final é chamada de raiz da *Merkle Tree*. Quando um número ímpar de transações é armazenado, a transação mais à esquerda

na árvore é duplicada, não modificando o número de combinações necessárias para montar a raiz (NAKAMOTO, 2009).

A Figura 5 ilustra como a raiz da *Merkle Tree* é montada. Nos nós folhas, as transações são alteradas para o formato *hash*. A combinação dessas transações é modificada mais uma vez e no último nível formam a raiz, que será armazenada no bloco.

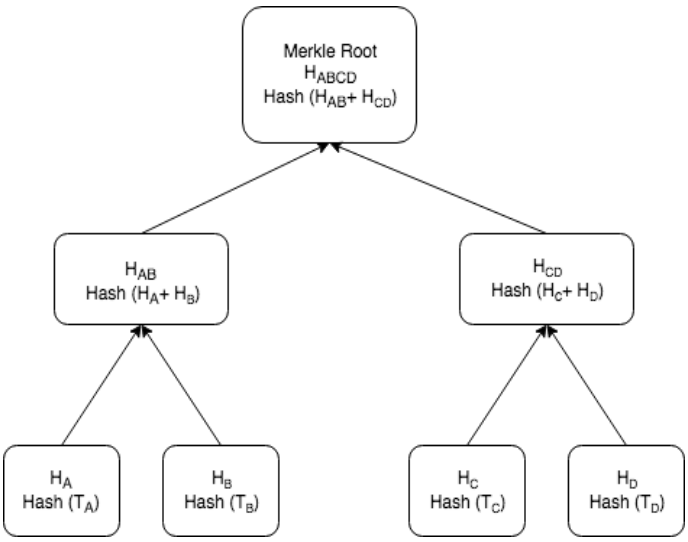


Figura 5 – *Merkle Tree* Adaptado de (ANTONOPOULOS, 2017)

Ao criar um bloco, o nó minerador tem a função de propagar esse bloco pela rede. Devido à demora de propagação, dois blocos podem ser criados em um período de tempo próximo, fazendo que, por algum tempo, existam duas cópias diferentes da *blockchain* espalhados pela rede. Os blocos são informados desse *fork* da cadeia e mantêm a cópia dos blocos criados por ambos os nós, pois ainda não é possível determinar qual bloco será escolhido para compor a cadeia principal. Dessa forma, os nós da rede armazenam duas cadeias, a primária e a secundária. Porém cada nó tem um registro diferente de qual cadeia é qual, e isso se deve pela divergência em relação aos tempos de chegada dos blocos nos nós. O bloco que foi notificado primeiro para o nó será considerado parte da sua cadeia principal, e o segundo é considerado parte da cadeia secundária. Quando um novo bloco é criado a cadeia a ser escolhida, é aquela em que o nó minerador do bloco atual considera principal (ANTONOPOULOS, 2017).

Os *forks* da cadeia impactam diretamente na permanência de uma transação na *blockchain*. No cenário de um *fork*, a cadeia secundária é descartada e os registros armazenados nela voltam para a lista de transações não inseridas. A permanência na cadeia depende da criação de novos blocos posteriores. Por padrão, no *Bitcoin*, as transações levam seis rodadas para serem validadas e garantidas que são parte da cadeia principal (NARAYANAN *et al.*, 2016).

A necessidade de criar mais blocos para confirmar a permanência de uma transação na cadeia impacta no desempenho do sistema, pois as transações não estão disponíveis assim que realizadas. Essa característica inviabiliza aplicações de tempo real usando a infraestrutura da

blockchain (GUPTA, 2017a).

2.3.2 Rede

A *blockchain* está distribuída por uma rede de nós organizados em uma arquitetura *Peer-to-Peer*. Nessa arquitetura os nós são vistos como iguais, não existindo um nó principal que atua de forma centralizada. Desse modo, se faz necessário que os nós estejam ativos, ou seja, que recebam e transmitam informação a fim de manter a rede ativa. Um dos desafios de uma arquitetura *P2P* é sustentar o interesse dos nós em permanecerem ativos, o que pode ser resolvido por meio de incentivos. Outra adversidade dessa arquitetura é a segurança, devido à falta de um nó central que controle os demais (KUROSE; ROSS, 2007).

A escolha por uma arquitetura *P2P* foi uma necessidade, considerando a proposta de descentralização de controle para o acesso a informações contidas nos nós. O *Bitcoin* contornou os problemas de incentivo com o pagamento em *bitcoins* pelos serviço de mineração de blocos. Os desafios de segurança foram tratados através dos protocolos de consenso na rede (NAKAMOTO, 2009). Outras aplicações que utilizam da *blockchain* se baseiam nos mesmos princípios do *Bitcoin* para lidar com esses obstáculos.

A rede em que está inserida a *blockchain* pode ser de dois tipos, pública ou privada. Em uma rede pública não existe um controle central sobre os integrantes da rede. Dessa forma, é necessário haver regras definidas para a participação e um controle rígido sobre a inserção dos dados. Nesse tipo de rede, os usuários não são conhecidos, e necessitam de incentivos para permanecerem honestos, sendo esse o caso do *Bitcoin* (BUTERIN, 2015).

Na rede privada, a participação e acesso aos dados são restritos e controlados por uma entidade, que pode ser uma empresa, um conjunto de empresas ou o governo, por exemplo. Essa entidade atua como órgão que gerencia a aplicação, que continua descentralizada permitindo o acesso e inserção de registros de forma paralela. Em uma rede privada, os nós são conhecidos e não precisam de incentivos para serem mantidos, uma vez que é do interesse da iniciativa privada manter a aplicação (BUTERIN, 2015).

Dentro de uma rede privada, as regras de consenso podem ser facilmente alteradas, e as transações revertidas. Também apresentam a vantagem de serem mais baratas que a rede pública, umas vez que um número menor de nós precisam validar uma transação, gerando menos gasto computacional do processo de mineração. Como a rede é conhecida, outra vantagem é que falhas podem ser manualmente corrigidas (BUTERIN, 2015). O quadro 2 apresenta as principais características que diferenciam as redes públicas das redes privadas.

Outro tipo de rede que pode ser encontrada é a *consortium*, um híbrido entre os dois tipos apresentados anteriormente. Nessa categoria, o processo de consenso é controlado por um grupo de nós pré-estabelecidos, e o direito de escrita pode ser restrito ou não aos participantes da rede. Não há um consenso sobre quais características diferenciam uma rede privada da *consortium*,

porém é estabelecido que ela agrega os atributos da não confiança em uma rede pública com um ponto central de controle da rede privada (BUTERIN, 2015).

Quadro 2 – Características das redes Públicas e Privadas

	Pública	***Privada***
Acesso	Qualquer entidade	Organizacional
Autoridade	Descentralizada	Parcialmente descentralizada
Velocidade das transações	Lento	Rápido
Custo das transações	Alto	Baixo
Manipulação dos dados	Leitura e escrita por qualquer entidade	Leitura e escrita limitada a organização
Imutabilidade	Total	Parcial
Eficiência	Baixa	Alta

Há outra classificação das redes da *blockchain*: *permissioned* e *permissionless*. A principal diferença é quem pode ou não ter acesso aos registros compartilhados na rede. Nas redes *permissioned* é necessário uma permissão do administrador ou administradores para a admissão à rede. Ao contrário, nas redes *permissionless* não é preciso autorização para associação e interação na rede (BLOCKCHAINS, 2020).

Alguns autores classificam as rede *permissioned* como a rede privada e as *permissionless* como a pública (BLOCKCHAINS, 2020). Isso se deve a ao acesso aos dados ser a principal característica que justifica essa diferenciação em ambos os casos. Porém, outros autores definem como outro ponto de caracterização de uma rede *blockchain*. Como em Wüst e Gervais (2018) que define a caracterização em função das permissões de leitura e escrita dos participantes. A rede pública *permissionless* é um em cenários em que os participantes da rede com possibilidade de escrita não são fixos nem conhecidos, como no *Bitcoin*. Caso haja restrição na escrita contudo a leitura é permitida a qualquer entidade então a rede é classificada como pública *permissioned*. No cenário de restrição tanto de escrita quanto de leitura ela é classificada como rede privada *permissioned*.

Os nós dessa rede podem ter diversas funções e, de modo geral, todos os integrantes atuam como distribuidores das transações e dos blocos. Porém, também podem apresentar outras funções, como por exemplo nem todo nó precisa manter uma cópia completa da cadeia, podendo haver nós específicos para essa atribuição. Os nós também podem se especializar em serem nós mineradores de blocos, podendo até apresentar um *hardware* próprio para tal. Os nós também podem realizar ambas as funções (ANTONOPOULOS, 2017).

A Figura 6 ilustra como os nós e suas funções podem ser distribuídos em uma rede. O objetivo desse exemplo é mostrar as possíveis funções executadas pelos nós. Os participantes podem ser mineradores e/ou armazenar a cadeia, porém todos propagam as transações na rede e armazenam aquelas que ainda não foram inseridas na cadeia.

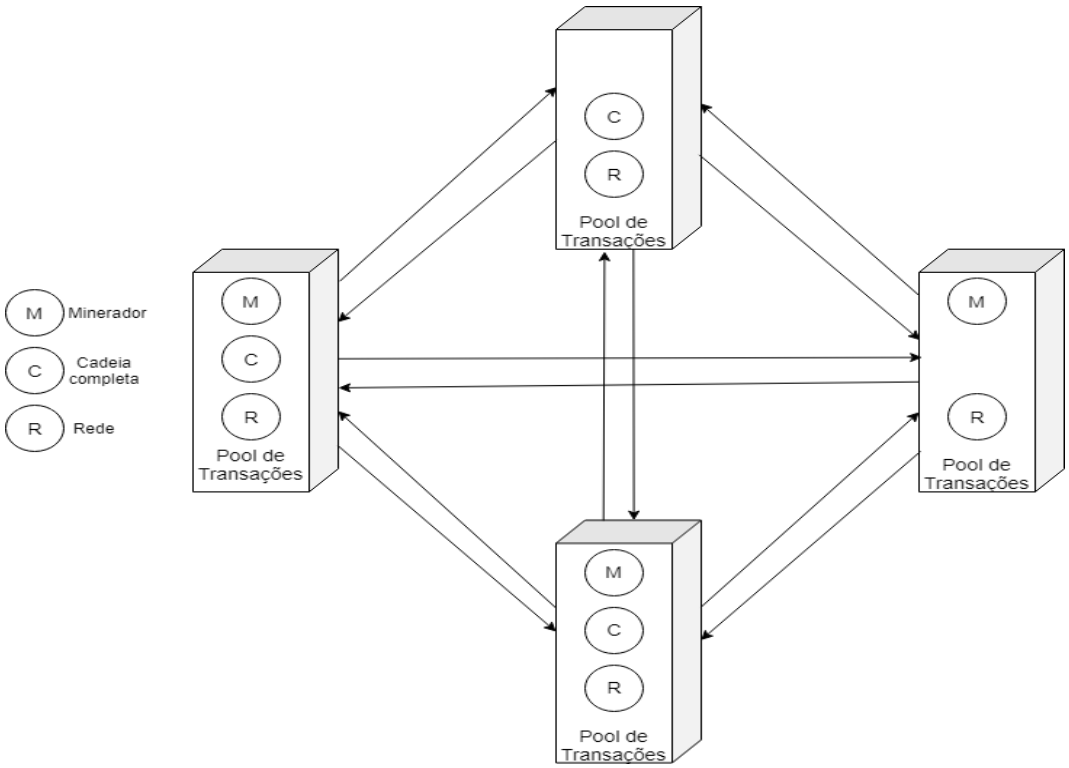


Figura 6 – Exemplo de rede com as funções de cada nó Adaptado de (ANTONOPOULOS, 2017)

Os nós também apresentam uma lista temporária de transações que ainda não foram adicionadas na cadeia. Essas transações ficam armazenadas em uma estrutura chamada *Memory Pool* ou *Transaction Pool*. Nessa estrutura, os registros não têm uma hierarquia definida. Quando os transações chegam, elas são verificadas e se forem validadas são armazenadas na *Memory Pool* e propagadas na rede pelo nó (ANTONOPOULOS, 2017).

Em um dado momento, cada nó tem uma lista diferente de transações na *Memory Pool*, essa diferença se dá pela velocidade de propagação em uma rede *P2P*. São essas transações/registros armazenadas nos nós que são utilizados para montar o bloco no processo de mineração (NARAYANAN *et al.*, 2016). No caso do *Bitcoin*, os nós mineradores dão preferência para as transações que têm o maior incentivo financeiro (NAKAMOTO, 2009). Além das diferenças entre as transações recebidas e validadas por cada nó, a latência da rede também influencia no desempenho do sistema. Há uma incerteza em relação ao tempo que uma transação será incluída na cadeia devido à demora na propagação da mesma pela rede (ANTONOPOULOS, 2017).

2.3.3 Mineração

A mineração é o processo em que novos blocos e novas moedas são criados no *Bitcoin*. Os nós mineradores espalhados nessa rede *P2P* estão competindo a todo momento pela criação de um bloco. Para isso, a primeira ação do nó é montar o bloco que se deseja criar. É necessário construir o *header* do bloco, constituído pela versão do bloco que se deseja criar, o *hash* do bloco anterior, os dados que serão armazenados no formato da *Merkle Tree*, o *timestamp* de criação do

bloco, o *difficult target* e o *nonce* utilizados no algoritmo do *Proof-of-Work* (ANTONOPOULOS, 2017).

O algoritmo do *PoW* consiste em exaustivamente comparar a *hash* produzida pela combinação entre os elementos do cabeçalho do bloco ao *difficult target*. O valor do *difficult target* pode ser caracterizado como a dificuldade atual de se minerar um bloco, ele é um parâmetro definido dinamicamente na rede do *Bitcoin* e tem como objetivo manter a taxa de criação de um novo bloco a cada dez minutos. Caso os blocos criados forem mais rápido que esse tempo estipulado, a dificuldade aumenta. No cenário oposto, em que a criação esteja demorando mais então a dificuldade diminui. Esse valor é estabelecido dentro da rede e independe do número de transações ou dos valores da mesma. O *nonce* é o contador modificado para tentar atingir o *difficult target*. Esse valor é inicializado com zero e é incrementado após cada erro em acertar o valor esperado (NARAYANAN *et al.*, 2016).

Em outras palavras, o processo de mineração com o *Proof-of-Work* consiste em resolver um *hash* onde o *difficult target* é a saída que se deseja se alcançar e o *nonce* é o valor variável, a fim de atingir esse *target*. Esse processo é realizado por meio de tentativa e erro, em que o *nonce* é incrementado a cada novo esforço de se atingir esse objetivo (NAKAMOTO, 2009). A Figura 7 ilustra como o é realizado o calculo do *hash puzzle*, em que as informações são concatenadas em uma função *hash* a fim de aproximar do valor determinado pelo *difficult target*.

Figura 7 – *Hash puzzle* do *Proof-of-Work*

Como visto na seção 2.3.1, funções *hash* são amplamente usadas em aplicação que usam a *blockchain*. No *PoW*, um tipo diferente de função *hash* é utilizado. As funções *hash* criptográficas apresentam as mesmas características da primeira, contudo, são adicionadas outras três particularidades: *Collision Resistance*, *Hiding* e *Puzzle Friendliness* (NARAYANAN *et al.*, 2016).

A primeira, *Collision Resistance* determina que é improvável que duas entradas distintas resultem em uma mesma saída. Essa natureza não garante que é impossível a existência de colisões, contudo o número de possibilidades é grande ao ponto de ser computacionalmente inviável achar uma colisão. No *Bitcoin*, o número de possibilidades é a raiz quadrada de 2^{256}, sendo *256-bits* o tamanho da *hash*, ou seja, há 2^{128} possibilidades a serem investigadas a fim de encontrar uma colisão (NARAYANAN *et al.*, 2016).

Outra propriedade, conhecida como *Hiding* define que dada uma saída é improvável determinar qual foi a entrada que a originou. Essa propriedade é limitada, pois se o número de possibilidades de entrada for conhecido, é fácil calcular cada uma e descobrir qual derivou a

saída. Entretanto, no caso de *hash* de *256-bits* (como no *Bitcoin*), o número de possibilidades a serem testadas é de 2^{256}, tornando-se computacionalmente inviável o cálculo de todas as possibilidades (NARAYANAN *et al.*, 2016).

A terceira propriedade é conhecida como *Puzzle Friendliness* que estabelece que mesmo com parte da entrada e uma saída conhecida, ainda assim é difícil encontrar um valor que direcione a entrada para o valor de saída desejado. Seu teorema afirma que para uma saída de *n-bits* é improvável encontrar a parte restante da entrada em um tempo menor que 2^n (NARAYANAN *et al.*, 2016).

As propriedades citadas acima têm como objetivo ilustrar a dificuldade desse processo. Só há um meio de obter a resposta desejada, através de um processo exaustivo de tentativa e erro. A ideia por trás do algoritmo de *PoW* é tornar a criação de blocos o mais próximo do aleatório possível, ou seja, todos os nós têm a mesma probabilidade de minerar um bloco. Dessa forma, há a garantia da total descentralização do processo de mineração, pois não há interferência central em relação a qual nó cria um novo bloco (NARAYANAN *et al.*, 2016).

O aumento da relevância do *Bitcoin* no mercado financeiro tornou consideravelmente rentável o processo de mineração de blocos. Os nós são numerosos e estão executando cálculos a todo momento para somente um deles efetivamente construir o bloco. Nisso, há um significativo gasto de energia elétrica associado a essa infraestrutura, o que eleva o custo desse processo (LISK, 2019).

Em outros contextos de aplicações *blockchain*, o custo do algoritmo e tempo de execução do *Proof-of-Work* são inviáveis. Nesse sentido, o *Proof-of-Stake* (*PoS*) é apresentado como alternativa acessível. No *PoS*, o criador do próximo bloco é determinado de maneira aleatória, que é, em parte, baseada pela quantidade de moedas que um usuário detém e, em alguns casos, por quanto tempo ele está mantendo essa moeda (NARAYANAN *et al.*, 2016).

Para participar do processo de criação de um bloco, os nós têm que possuir uma parcela das moedas do sistema. Ao contrário do *Proof-of-Work*, no algoritmo do *Proof-of-Stake* não há criação de moedas, desta forma o nó recolhe os *transactions fee* das transações que serão inseridas na cadeias (LISK, 2019).

A probabilidade de criar um bloco e receber a recompensa associada é proporcional à quantidade de moedas de um nó, ou seja, quanto maior a parcela de moedas que um nó possui, maior a probabilidade dele ser escolhido para criar o bloco. Esse mecanismo de probabilidade visa à descentralização do processo de criação de blocos, caso contrário, o nó com o maior número de moedas do sistema sempre criaria o próximo bloco e aumentaria a sua riqueza e, como resultado, seu controle do sistema (LISK, 2019).

Para competir e ter a possibilidade de criar o bloco, os nós devem, além de possuir uma quantidade de moedas, também bloqueá-las. Essas medidas garantem a segurança da rede. A primeira medida implica que para um usuário, ou grupo de usuários, ter controle sobre a

rede, é necessário deter 51% das moedas da aplicação. Esse cenário é impraticável, uma vez que seria extremamente caro adquirir essa parcela das moedas. Além disso, os outros usuários notariam a tentativa de controle da rede e poderiam reagir a isso, tirando suas moedas da rede ou aumentando o seu valor para assim, evitar a compra (LISK, 2019).

Já a medida de bloquear as moedas torna completamente prejudicial um usuário atacar a rede, pois diminuiria o valor de suas próprias moedas. Na prática, os usuários com a maior participação têm mais interesse em manter e proteger a rede, porque qualquer ataque diminuiria a reputação e o preço da moeda que eles detêm (LISK, 2019).

2.3.4 Consenso

Uma das adversidades enfrentadas por quem utiliza os serviços presentes em redes distribuídas é a latência que pode ocorrer na comunicação entre os nós. Isso pode dificultar o consenso entre os nós e acarretar no problema conhecido como Problema dos Generais Bizantinos (sigla *PGB* e em inglês, *Byzantine Generals Problems*). Esse é um problema clássico da computação distribuída dada a dificuldade de se chegar a um consenso em uma rede onde não se conhecem todos os nós participantes. Nos estudos relacionados ao *PGB* provou-se que é necessário assegurar que menos de um terço dos nós sejam maliciosos, pois caso o contrário, é impossível certificar a segurança da rede (NARAYANAN *et al.*, 2016).

O *PGB* foi considerado insolúvel, porém, o *Bitcoin* e demais aplicações baseadas nele, superam esse desafio com o uso de incentivos. Os nós mineradores só ganham a bonificação quando além do bloco, todas as transações que o constituem forem validadas juntamente com ele. Dessa forma, aceitar transições inválidas se torna desvantajoso, já que um alto poder computacional foi gasto para minerar o bloco, para ele ser invalidado pela rede. Os nós que certificarem muitos nós invalidados, são considerados não-honestos e podem ser removidos da rede (NARAYANAN *et al.*, 2016).

O processo de consenso é constante: os nós garantem a validade dos registros e blocos ininterruptamente. Assim, gradualmente, a probabilidade de que os dados sejam inválidos é reduzida consideravelmente, visto que um conjunto numeroso de nós já realizou esta validação. Além de validar, os nós também têm que concordar com a ordem na qual as transações ocorreram (NARAYANAN *et al.*, 2016).

O consenso pode ser definido como o processo de verificar se um registro é válido seguindo uma lista de pré-requisitos. A segurança da rede é garantida através do consenso entre os nós que tanto os registros são válidos e seguros quanto os blocos. Para isso os nós precisam de forma independente verificar se cada registro segue uma lista de critérios, definida por cada aplicação (ANTONOPOULOS, 2017).

Após o bloco ser criado, ele também é validado pelos nós da rede, de forma independente. Ao receber um bloco, o nó valida o bloco antes de distribuí-lo pela rede. Essa ratificação é

baseada em uma lista de critérios que o bloco deve seguir para os nós o validar e transmiti-lo pela rede. A última etapa do processo de consenso da rede é inserir o bloco dentro da cadeia (ANTONOPOULOS, 2017).

O processo de dupla verificação, com as transações sendo validadas individualmente e os blocos passando pelo mesmo processo, é custoso para o desempenho das aplicações. Tanto as transações quanto os blocos são validados através de um processo sequencial, que avalia cada um deles de acordo com a lista de critérios pré-estabelecida. Somado a isso, a verificação é realizada duas vezes, pois as transações já validadas são inseridas no bloco que também será avaliado (ANTONOPOULOS, 2017).

A latência da rede também pode causar o problema de *double spending* o qual é caracterizado pela tentativa de usar uma mesma quantia de moedas mais de uma vez em transações diferentes, tentando contar com a latência da rede para criar uma nova transação com as mesmas moedas e, dessa forma, gerar um *fork* na cadeia principal. Entretanto, com o protocolo de consenso, as transações demoram para serem validadas de maneira que as duas transações ficam evidentes para todos os nós da rede e é descoberta a tentativa de fraude. Uma transação só é considerada inserida na cadeia, quando um número considerável de blocos foram criados depois dela ser incorporada na *blockchain*. No caso do *Bitcoin*, é necessário que seis blocos sejam criados para garantidamente uma transação não ser removida por conta de *forks* (NARAYANAN *et al.*, 2016).

2.3.5 Qualidades & Limitações

Comparado com bases de dados tradicionais, a *blockchain* apresenta diversas vantagens, uma delas é a imutabilidade, ou seja, uma vez inseridos na cadeia os dados não podem ser alterados, somente novas transações modificam informações já armazenadas. Essa característica é permitida pelo algoritmo de *Proof-of-Work* que agrega um alto custo computacional ao modo como as transações são armazenadas, tornando, dessa maneira, praticamente impossível modificar informações já armazenadas na *blockchain* (NAKAMOTO, 2009).

Outra vantagem é a segurança, tanto pela descentralização da rede quanto pelo consenso. O primeiro torna a rede resistente a possíveis ataques por não conter apenas um ponto de falha. E a segunda, permite que nós desonestos sejam rapidamente detectados e removidos. O consenso da rede acaba por eliminar um problema de segurança natural de redes *Peer-to-Peer*, no qual os participantes não são conhecidos (GUPTA, 2017a).

A redundância dos dados também é uma propriedade das aplicações *blockchain* que o torna vantajoso sobre base de dados tradicionais. Considerando que os dados estão espalhados nos diversos nós da rede, há uma dificuldade na perda de dados em caso de falha em um nó, ou até mesmo de vários, se comparado à base de dados centralizadas em um único nó (GUPTA, 2017a).

Em soluções que apresentam múltiplas partes as quais precisam compartilhar, ler e escrever informações de forma que todos têm acesso aos dados paralelamente a *blockchain* se destaca. Para aplicações em que há a necessidade que as transações e registros sejam conhecidas e validadas por todos os integrantes, pois os intermediários são ineficientes, ele também se sobressai (GUPTA, 2017a).

Apesar das vantagens apresentadas pelo uso da *blockchain* nas aplicações, ele é uma solução computacionalmente cara por demandar uma adaptação do sistema já utilizado. Porém, em aplicações onde os dados são armazenados fora da cadeia (armazenamento *Off-chain*) esse custo pode ser reduzido, pela maior possibilidade de integração entre os sistemas. Ainda que represente uma solução viável, a recuperação dos registros em um banco de dados externo à cadeia influencia de forma negativa no desempenho da aplicação (GUPTA, 2017a).

Além disso, o custo do processo de inserção dos dados, a mineração, também é alto, mostrado na seção 2.3.3. Em redes privadas, esse custo pode ser diminuído, afrouxando as regras para criar um bloco, com a adoção de um algoritmo de mineração menos custoso, como o *Proof-of-Stake* (GUPTA, 2017a).

Outra dificuldade é a necessidade de manter os nós participantes e ativos na rede. Essa questão é natural de redes *P2P*, pela dependência da rede na quantidade e qualidade da atuação desses nós, visto que não há uma unidade controladora que os mantém. Em aplicações de *blockchain* privadas, a organização ou organizações que controlam a aplicação podem manter os nós ativos sem a necessidade de incentivos, entretanto, em aplicações públicas, há a necessidade de criar um estímulo para manter a rede funcionando (GUPTA, 2017a).

A infraestrutura da *blockchain* não é de forma nativa preparada para lidar com dados que precisem de garantias com relação a privacidade. Dentro de uma rede, os dados das transações são abertos para todos os participantes da rede, sendo assim, para os registros serem mantidos sigilosos, é necessário remodelar as transações utilizando de funções de criptografia (GUPTA, 2017a).

2.4 Virtualização e Banco de Dados

Esta seção contém uma breve descrição de dois itens principais que compõem a base para o uso da plataforma de desenvolvimento *blockchain Hyperledger Fabric* (que será explicado na Seção 2.5.2): *Docker* e *CouchDB*.

2.4.1 Docker

A execução de uma rede *blockchain* com o *Hyperledger Fabric* (descrito a seguir na Seção 2.5.2) envolve o uso de imagens *Docker* as quais contêm as instruções para a criação de *containers* que atuam nas funções da rede descritas na Seção (HYPERLEDGER, 2019). *Docker* é uma

plataforma *open-source* com a finalidade de facilitar o desenvolvimento de sistemas de software. O objetivo do *Docker* é separar a aplicação da infraestrutura a ser utilizada, possibilitando, assim, o desenvolvimento mais rápido e padronizado de software usando virtualização em nível de sistema operacional. O isolamento proposto pelo *Docker* é alcançado através da conteinerização, em estruturas chamadas de *container* (INC, 2021).

O *container* empacota o código e suas dependências (*frameworks*, bibliotecas, etc.) para a implantação de uma aplicação de maneira rápida e confiável. Diferentemente das máquinas virtuais (VMs), os containers compartilham o *kernel* do sistema com as demais aplicações, sendo executados no sistema operacional já existente. As VMs executam o seu próprio sistema operacional, aumentando o seu tamanho, sua complexidade e seu uso de recursos computacionais (INC, 2021).

Além de compartilhar recursos, há outras vantagens do uso de *containers*, tais como a portabilidade, ou seja, é possível construir *containers* localmente e usá-los em qualquer ambiente *docker*. A modularidade também é outro benefício, permitindo que atualizações sejam feitas sem a necessidade de interromper a aplicação. Além disso, o uso de *containers* permite a reversão, isto é, voltar ao estado anterior da aplicação (INC, 2021).

O que permite essas vantagens descritas com o uso do *Docker* são as chamadas imagens *docker*, que podem ser definidas como arquivos com instruções para a criação de *containers*. Uma imagem *docker* contém o código da aplicação, bibliotecas, ferramentas, dependências e outros arquivos necessários para fazer uma aplicação funcionar. Executar uma imagem permite instanciar um ou mais *containers* que executam determinada aplicação (INC, 2021).

O conjunto de imagens *docker* disponibilizadas pelo *Hyperledger Fabric* permite a criação de *containers* que atuam como os nós integrantes da rede, o banco de dados *CouchDB*, as *Certificates Authority* (CAs) e os nós do tipo *Orderer*. O uso de imagens permite a modularização do *Fabric* em aplicações isoladas, o que facilita a manutenção do sistema e provê maior estabilidade à infraestrutura *blockchain* (HYPERLEDGER, 2019). Além disso, o uso de *containers docker* permite que essas funcionalidades sejam sempre executadas da mesma maneira, independentemente do sistema operacional das máquinas utilizadas.

2.4.2 CouchDB

O *CouchDB* é um banco de dados não-relacional orientado a documentos, recomendado quando o valor do par chave-valor é composto por múltiplos valores.

Cada base de dados no *CouchDB* é formada por um conjunto de documentos e unicamente nomeados independentes entre si onde os dados são armazenados em formato não estruturado. O *CouchDB* fornece *APIs RESTful* para leitura e atualização (operações de adicionar, editar e remover) dos documentos da base de dados. Os documentos são registrados em formato *JSON*, o que viabiliza flexibilidade no registro de informações com a possibilidade de armazenamento de

diversos tipos de dados como *strings*, números, datas e *arrays* (FOUNDATION, 2020).

Os documentos identificados de maneira única, sendo indexados utilizando uma árvore B (*B-Tree*) de acordo com o nome (*DocID*) e uma identificação sequencial (*SequenceID*), valor usado para monitorar as atualizações em um determinado documento. Armazenar os documentos indexados em árvore B permite buscas, inserções e remoções em tempo logarítmico e os índices da árvore são atualizados a cada adição ou remoção de documentos (FOUNDATION, 2020).

O acesso ao documento não é bloqueado quando um cliente o acessa, dessa forma o *CouchDB* permite o acesso simultâneo de vários clientes ao mesmo tempo. As operações de leitura no *CouchDB* usam um modelo de *Multi-Version Concurrency Control* (MVCC) onde cada cliente vê o documento como ele está naquele instante, até o final da operação de leitura. Assim, o *CouchDB* garante a disponibilidade das informações porém com a limitação da consistência eventual (FOUNDATION, 2020).

Atualizações de documentos, ou seja, adição, edição e remoção são operações atômicas na estrutura do *CouchDB*: ou são totalmente bem sucedidas ou falham completamente. O banco de dados não armazena documentos parcialmente modificados (FOUNDATION, 2020).

Ao contrário de bancos de dados relacionais, o *CouchDB* não apresenta esquemas de tabelas em que as informações estão rigidamente estruturadas o que permite a realização de diversos tipos de busca. Desta forma, as buscas por documentos são essencialmente baseadas no identificador único de cada documento. No intuito de expandir a capacidade de filtrar e agrupar os documentos, o *CouchDB* oferece um modelo de visualização (*views*) gerado através de funções de *MapReduce* para agregar e gerar relatório dos documentos armazenados na base de dados. As visualizações são construídas sob demanda e não afetam os dados, sendo definidas em documentos separados e unicamente nomeadas (FOUNDATION, 2020).

Views são uma representação dinâmica da base de dados. Gerar essas visualizações consome recursos computacionais e tempo, uma vez que uma varredura da base de dados é necessária, tornando desvantajoso criar novamente a visualização toda vez em que ela é consultada. Para manter as buscas de *views* eficientes, o *CouchDB* as mantêm indexadas e atualizadas continuamente para refletir as alterações na base de dados quando documentos são adicionados, atualizados ou removidos (FOUNDATION, 2020).

A busca de informações em documentos no *CouchDB* é facilitada pelo uso de *search indexes*. Eles permitem consultas na base de dados sem a necessidade de examinar cada linha do documento, o que torna a busca mais rápida e eficiente. Os *search indexes* podem ser criados com um ou mais campos de um documento (FOUNDATION, 2020).

2.5 Plataformas de Desenvolvimento

Nas seções anteriores desse capítulo foram descritas quais características que envolvem uma aplicação *blockchain*. De forma geral, essa infraestrutura é composta por blocos, onde as informações são armazenadas, ligados em formato de cadeia o que garante a imutabilidade dos registros armazenados. O processo de criação desses blocos é chamado de mineração e tem como objetivo garantir a validade das informações armazenadas. A cadeia formada é distribuída em uma rede *Peer-to-Peer*, no qual um processo de consenso assegura que todos os participantes concordam com as informações armazenadas na *blockchain*.

A descrição dessas características mostra a complexidade de desenvolver aplicações *blockchain*. Cada implementação precisa inicializar uma rede *blockchain* independente, bem como construir e testar todos os códigos de rede e transição de estado necessários. Além disso, há os desafios de garantir o consenso de aplicações descentralizadas sem incentivos para aplicações de pequeno porte. O *Bitcoin*, por exemplo, soluciona essa questão com o incentivo da cripto-moeda, porém aplicações menos consolidadas não necessariamente têm essa possibilidade (BUTERIN, 2021).

Um exemplo de implementação independente do protocolo *Bitcoin* é o *Namecoin* cujo objetivo é criar uma solução segura e descentralizada para o registro de *Domain Name System* (DNS). Foi desenvolvida através de um *fork* no código-fonte do *Bitcoin*. A finalidade de um DNS descentralizado é possibilitar qualquer pessoa publique informações na *Internet* sem ser suprimida ou censurada. A ideia é que *URLs* permanentemente registrados na *blockchain* seriam resistentes à apreensão de domínios pelo governo, como na China onde o governo tem controle sobre o domínio *.cn* (NARAYANAN *et al.*, 2016).

O *Bitcoin* não foi construído com componentes como contas, usuários, extrato e pagamentos, no lugar há uma linguagem de *script* transacional. Esses conceitos citados são derivados das primitivas dessa linguagem a qual permite um conceito simples de *smart-contract* (que será definido nas próximas subseções) (BUTERIN, 2021). Dessa maneira, a *blockchain Bitcoin* pode ser usado para registro de informações além do *bitcoin* de forma limitada. As restrições de uso se deve a falta de *Turing-completeness* o que significa que os *scripts* desenvolvidos tem limitação na complexidade e tempos de execução previsíveis por não existir *loops* ou capacidades complexas de controle de fluxo e controle de fluxo condicional. Outra característica que limita a linguagem é a falta de armazenamento de estados ou seja, não há nenhum estado anterior à execução do *script*, ou estado salvo após a sua execução (ANTONOPOULOS, 2017).

Um exemplo de aplicações que usam essa flexibilidade oferecida pelo *Bitcoin* são as *Colored Coins*. O termo se refere a um conjunto de tecnologias que usa as transações do *Bitcoin* para o registro da criação, propriedade e transferência de ativos fora do contexto da cripto-moeda. As *Colored coins* são usadas para rastrear tanto ativos digitais quantos físicos. O termo vem da ideia de "colorir"ou marcar um conjunto de *bitcoins* para representar algum coisa além do

valor do *bitcoin*. Elas são criadas, transferidas e vistas em carteiras especiais que interpretam o protocolo *Colored Coins* associado as transações *bitcoin*. As implementações mais proeminentes são *Open Assets* e *Colored Coins by Colu*. Esses dois sistemas tem implementações diferentes e não são compatíveis, as *colored coins* criadas em um sistema não podem ser vistas ou usadas em outro sistema (ANTONOPOULOS, 2017).

Esses exemplos de implementações *blockchain* tem como objetivo ilustrar formas como essas aplicações foram desenvolvidas a partir da consolidação da *blockchain* para a distribuição e armazenamento de informações. Como destacado, essas soluções apresentam limitações, seja pelo custo de desenvolvimento e falta de segurança, cenário do *Namecoin*, ou pela limitação de suas possibilidades, exemplo das *Colored Coins*. Nesse contexto, o *Ethereum* (BUTERIN, 2014) foi a primeiro *framework* com o objetivo de facilitar o desenvolvimento de aplicações *blockchain* de propósito geral com a garantia de segurança. A iniciativa *Hyperledger* (DHILLON; METCALF; HOOPER, 2017) propôs um conjunto de ferramentas para o desenvolvimento de aplicações *blockchain* com o objetivo de promover a *blockchain* para as empresas e tornar seu uso mais popular. Isso se deve a necessidades específicas que aplicações com essa inciativa demandam, como a necessidade de identificar participantes e cadeia distribuída em redes privadas, por exemplo.

Nas próximas Subseções são descritas essas principais iniciativas com o foco no *Hyperledger Fabric* e *Hyperledger Caliper* usados para a realização deste trabalho.

2.5.1 Ethereum

O *Ethereum* é uma plataforma *open-source* para aplicações distribuídas em um ambiente seguro e descentralizado. Seu objetivo é abstrair a complexidade da infraestrutura da *blockchain* de forma a tornar mais simples o desenvolvimento de aplicações *blockchain*. O *Ethereum* provê um ambiente de programação robusto e completo que permite o desenvolvimento de aplicações fundamentadas no conceito de contrato, que é uma estrutura utilizada para representar regras para transferência de posses, transações e possíveis estados de um objeto (BUTERIN, 2014).

A plataforma fornece mecanismos para simplificação do desenvolvimento de soluções baseadas na troca de mensagens, que podem envolver a transferência de valores monetários ou dados. Essas aplicações são essencialmente descentralizadas, sem um elemento intermediário e garantem alta disponibilidade. Desta forma, em poucas linhas de código, pode-se criar uma aplicação para gerenciar transações ou distribuir dados entre usuários geograficamente distantes (BUTERIN, 2014).

O funcionamento do *Ethereum* é baseado na infraestrutura do *Bitcoin*, porém com a proposta de dinamizar as possíveis aplicações que podem ser executadas na plataforma. Com isso, não se restringe a aplicações financeiras ou trocas de moedas virtuais, caso do *Bitcoin*. Para isso, a plataforma utiliza o conceito de estados, que representam a transferência atômica de

dados sobre a infraestrutura da *blockchain* (BUTERIN, 2014).

Para esse fim, há os *smart-contracts* que são a estrutura básica do *Ethereum*. Eles representam uma entidade do mundo real, descrevendo suas funcionalidades e os possíveis estados que o objeto pode assumir. Os *smart-contracts* fundamentam o desenvolvimento de qualquer aplicação na plataforma (BUTERIN, 2014).

As entidades no *Ethereum* estão associadas às contas, que podem ser de dois tipos: conta de contrato (*contract account*) e conta externa (*external account*). A primeira, identifica os *smarts-contracts*. Este tipo de conta pode trocar mensagens com outras contas do mesmo tipo ou contas externas, porém não podem iniciar uma transação por conta própria (necessitam da solicitação de uma conta externa) (BUTERIN, 2014).

Já as *external accounts* identificam entidades externas que possuem chaves de acesso privadas e podem trocar mensagens entre si ou com contas de contrato. As contas externas podem acessar funcionalidades de um *smart contract* ou iniciar uma transferência de dados. Qualquer evento no *Ethereum* é desencadeado por uma requisição de uma conta externa. Todas as contas possuem um estado associado a elas e um endereço de identificação, para a troca de mensagens (BUTERIN, 2014).

Para o desenvolvimento de aplicações no *Ethereum*, há várias linguagens de programação que originam do *C++*, *Java* e *Python*. Dentre elas, destacam-se, o *Vyper* (baseada no *Python*), *Bamboo* (que permite representar um *smart-contract* como uma máquina de estados finita) e o *Solidity*, que é a linguagem mais consolidada para o desenvolvimento de aplicações na plataforma *Ethereum* (CONDON, 2017).

Todas os programas escritos em qualquer uma dessas linguagens são interpretadas e transformados em *bytecode*, e são executados pela *Ethereum Virtual Machine* (*EVM*). Com a *EVM* é possível replicar o mesmo código em um ambiente composto de máquinas distintas e, desta forma, garantir a disponibilidade de uma aplicação na rede (CONDON, 2017).

Dentro da plataforma do *Ethereum* existem alguns tipos de *DAOs* (*Decentralized Autonomous Organization*), que são organizações nas quais os membros participantes dividem investimentos e rendimentos. O direito de participação na organização está diretamente relacionado à quantidade de votos que esse membro tem, que por sua vez é referente à quantia investida por ele. O mais famoso deles é o *TheDAO*, que englobava 10% das moedas do *Ethereum* (GRAHAM, 2016).

A importância dessas organizações motivou o *DAO exploit*, uma série de ataques que ocorrem contra o *The DAO*. Nesses ataques, por volta de 50 milhões de dólares foram perdidos nas transações da plataforma por conta de inconsistências encontradas na execução dos *smart-contracts* dessas organizações (JENTZSCH, 2016).

O *DAO exploit* ocorreu devido a um mecanismo de execução que permitia que os membros que desejassem sair da organização pudessem recolher seus investimentos. Essa

possibilidade permitia que ela se chamasse recursivamente, de modo que os valores fossem retirados indefinidamente até a conclusão das chamadas, o que era determinado pelo próprio invasor. Essa brecha se deve ao não determinismo da ordem de execução dos comandos dos *smart-contracts* e do fato de que cada comando representa uma transação. Sendo assim, podem ocorrer erros na ordem de execução por não haver garantias que essas transações sejam reconhecidas na ordem correta (GRAHAM, 2016).

2.5.2 Hyperledger Fabric

O *Hyperledger* pode ser definido como um esforço colaborativo *Open-Source* para promover a *blockchain* em vários setores da tecnologia. Esse estimulo é uma colaboração global, hospedada pela *The Linux Foundation*, que inclui líderes em finanças, serviços bancários, *internet of things*, cadeias de suprimento, manufatura e tecnologia (DHILLON; METCALF; HOOPER, 2017).

A partir desta definição, a proposta é que o *Hyperledger*, em vez de ser uma única plataforma, como o *Ethereum*, tenha vários projetos sobre sua tutela. A chamada *Umbrella Strategy* faz com que o *Hyperledger* seja uma encubadora e promova um conjunto de inciativas relacionadas à *blockchain*, visando fornecer padrões robustos e eficientes de desenvolvimento de aplicações com o objetivo de facilitar a adoção comercial da infraestrutura (DHILLON; METCALF; HOOPER, 2017). A Figura 8 apresenta os projetos *Hyperledger*.

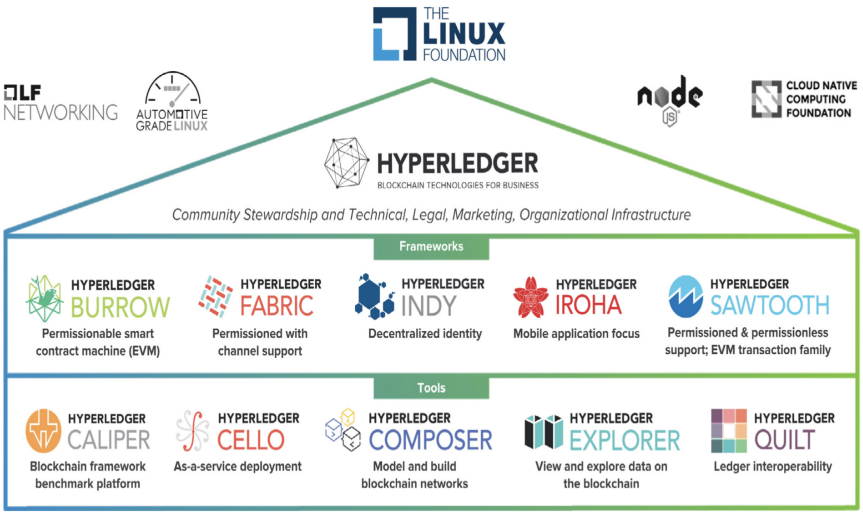


Figura 8 – Projetos sobre a tutela *Hyperledger* Retirada de (LINUXFOUNDATION, 2019)

Nestes termos, o *Hyperledger* visa prover uma interface de programação (*API*) para aplicações *blockchain* que permita fácil desenvolvimento de aplicações que atendam requisitos de interoperabilidade, escalabilidade e disponibilidade (DHILLON; METCALF; HOOPER, 2017).

As principais iniciativas são: o *Hyperledger Sawtooth* (INTEL, 2017), em parceria com a *Intel*, o *Hyperledger Iroha* (SORAMITSU, 2017), uma parceria com empresas japonesas e o *Hyperledger Fabric* (HYPERLEDGER, 2019), desenvolvido em parceria com a *IBM*, que é a implementação mais consolidada atualmente. Além desses, também vale destacar o *Hyperledger Cello* (ATTRIBUTION, 2019), uma implementação do modelo de *blockchain-as-a-service* e o *Hyperldger Calliper* (PROJECT, 2020), uma série de *benchmarks* com a finalidade de facilitar a validação de aplicações *blockchain* (cujo os objetivos e arquitetura são detalhados na próxima Seção).

O texto a seguir descreve alguns aspectos do *Hyperledger Fabric*, que é considerado a implementação *hyperledger* mais consolidada e utilizada atualmente (ANDROULAKI *et al.*, 2018). Além disso, também é a plataforma escolhida para desenvolvimento desse projeto.

O *Hyperledger Fabric* fornece uma estrutura para o desenvolvimento de soluções *blockchain* com uma arquitetura modular. Assim, diferentes componentes da *blockchain* podem ser modificados sem alterar a funcionalidade de outras partes da infraestrutura. Diferentemente de outras plataformas, oferece uma solução para redes privadas, os membros são conhecidos por meio de *Membership Service Provider* (*MSP*), uma lista que atribui as funções dos participantes da rede (CACHIN, 2016).

Cada componente da rede tem encapsulada uma identidade digital, através da qual é possível determinar quais as permissões no uso de recursos e acesso às informações que cada componente da rede tem. A identidade dos participantes é verificada pelo *MSP*, que mapeia as atribuições/papéis de cada membro da rede em função de sua identidade. Mapear identidade em papéis é essencial para o funcionamento da rede do *Fabric* pois permite que os componentes determinem as permissões de cada integrante da rede (HYPERLEDGER, 2019).

Os integrantes da rede também têm associado um *Certificate Authority (CA)*, que é usado para emitir certificados para administradores e nós da rede. Os certificados emitidos pelo *CAs* podem ser usados para assinar uma transação, indicando que determinada organização confirma o resultado de uma transação: uma condição para a transação ser incluída na cadeia. Os *CAs* também são usados para apontar a qual organização determinado participante é pertencente. Organizações distintas utilizam diferentes *CAs*, dessa forma, há várias *CAs* em uma única rede (HYPERLEDGER, 2019).

Um diferencial do *Hyperledger Fabric* é a possibilidade de criar *channels* (estrutura lógica composta por um conjuntos de nós) permitindo que um grupo de participantes da rede crie uma cadeia separada. Ou seja, caso dois ou mais participantes formem um *channel*, somente estes mesmos participantes terão acesso às transações realizadas de forma isolada dos outros integrantes da rede. Essa é uma opção para redes em que alguns participantes podem ser concorrentes e, por isso, não desejam que todas as transações realizadas sejam conhecidas por todos (ANDROULAKI *et al.*, 2018).

O *Hyperledger Fabric* tem duas entidades que lidam com a lógica de processamento das transações: o *smart-contract* e o *chaincode*. Esses termos podem ser usado de forma intercambiável, pois juntos controlam a organização das transações. No contexto do *Fabric*, os *smart-contracts* definem a lógica de execução que determina como deve ser feita a inserção de dados na cadeia. O *chaincode* é usado por administradores da rede para agrupar *smart-contracts* com mesmo propósito e implementar a lógica de execução definida por eles (HYPERLEDGER, 2019).

A implantação de um *chaincode* na rede disponibiliza todos os seus *smart-contracts* para os participantes. Dessa forma, apenas os administradores da rede precisam definir o *chaincode*, e os demais integrantes lidam somente com o *smart-contract* (HYPERLEDGER, 2019). Na infraestrutura do *Hyperledger Fabric*, os *smart-contracts* podem ser desenvolvidos em *JavaScript*, *GO* ou *Java*.

A cadeia distribuída no *Hyperledger Fabric* é composta por dois componentes de armazenamento: o *world state database* e a *blockchain*. O primeiro pode ser caracterizado como o estado da cadeia em um devido momento. Os dados são armazenados no formato chave-valor e podem estar nos bancos de dados *LevelDB* ou *CouchDB*. A *blockchain*, por sua vez, armazena o registro de cada transação que modificou o valor do *world state*, como um registro global. As transações são coletadas e arquivadas na cadeia de forma ordenada, garantindo assim possibilidade de recuperar o valor do *world state* no caso da inserção de um novo nó na rede ou em caso de falha de um nó (HYPERLEDGER, 2019). Como as informações são armazenadas no formato de cadeia, os dados são imutáveis, como explicado na Seção 2.3.4.

A Figura 9 ilustra como a cadeia é formada no *Hyperledger Fabric*.

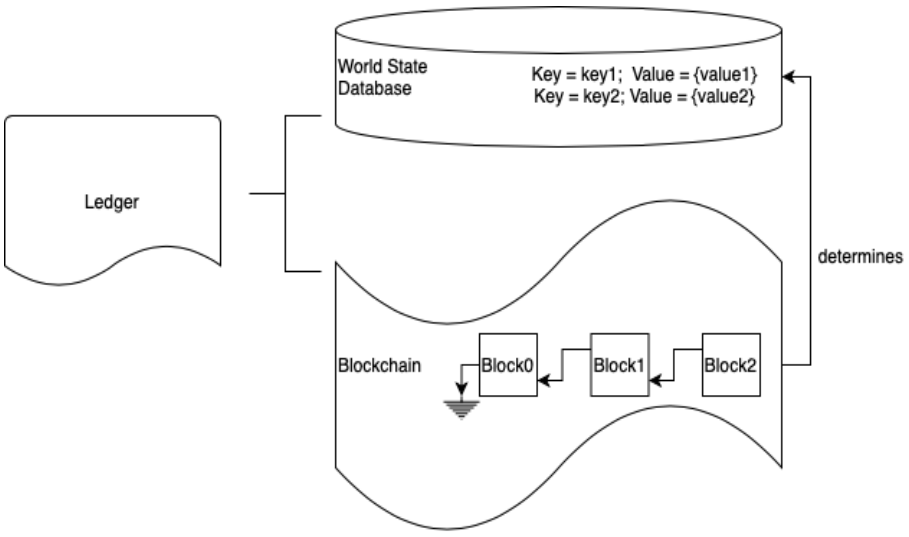


Figura 9 – Composição da cadeia no *Hyperledger Fabric* Adaptado de (HYPERLEDGER, 2019)

Armazenar os valores das chaves no *world state* é uma estratégia para facilitar a busca. Dessa forma, elimina a necessidade de uma busca em toda a cadeia para recuperar o valor de

uma chave em pesquisas específicas (HYPERLEDGER, 2019). O *LevelDB* é um banco de dados não relacional e padrão do *Fabric*, sendo o ideal quando há somente um valor armazenado (DEAN; GHEMAWAT, 2020). Já o *CouchDB*, que é um banco de dados não relacional orientado a documentos, é usado quando o valor do par chave-valor é composto por múltiplos valores. Essa escolha se deve a estrutura do *CouchDB*, em que os dados são armazenados em documentos *JSON*, dessa forma é possível utilizar diversos tipos de dados (FOUNDATION, 2020).

Dentro do *Hyperledger Fabric*, cada *smart-contract* está associado com uma política de validação de transações (ANDROULAKI *et al.*, 2018). Essa política determina quantos e quais nós têm que aprovar uma transação para esta ser considerada válida. Todas as transações são armazenadas na cadeia, sendo elas válidas ou não, porém somente as transações válidas alteram o *state world*. Depois de distribuídas pela rede, as transações são validadas em duas etapas: a primeira consiste em verificar se a transação seguiu a política de validação correspondente, ou seja, se foram validadas pelas organizações necessárias; a segunda é verificar se o valor no *state world* condiz com a transação que está sendo avaliada (HYPERLEDGER, 2019).

Assim como nas demais implementações da *blockchain*, o elemento básico da rede do *Fabric* é o nó, onde a cadeia e os *smart-contracts* estão armazenados. Dentro da rede do *Fabric* os nós pertencem a organizações que representam entidades do mundo real, tais como empresas indivíduos ou instituições (HYPERLEDGER, 2019). A rede é administrada por um conjunto de organizações e os nós têm função centralizadora, atuando como pontos de ligação entre as organizações e a rede. Um nó, e consequentemente uma organização, pode participar de mais de *channel*, armazenando a cadeia e os *smart-contracts* associados a cada *channel* do qual o nó é integrante (ANDROULAKI *et al.*, 2018).

Cada rede criada dentro do *Hyperledger Fabric* inclui um *Ordering Service*, uma abstração que tem a função de estabelecer qual é a ordem das transações. O nó ou nós *Orderder*, que implementam o *Ordering Service*, são essenciais para manter a cadeia de cada nó da rede consistente e permite que os nós da rede foquem em validar e armazenar as transações na cadeia, sem necessidade de lidar com a ordenação das transações (HYPERLEDGER, 2019). Uma vez que os nós podem estar distantes um dos outros e não tem uma visão comum com relação às transações, acordar sobre a ordem seria custoso e causaria uma sobrecarga na comunicação. A criação de uma rede é caracterizada pela inicialização de um nó do tipo *Ordering Service*, que é visto como o ponto de administração inicial de uma rede (HYPERLEDGER, 2019).

O nó ou nós do tipo *Orderer* tem como função o processo de coletar as transações propostas, ordená-las e agrupar nos blocos que serão distribuídos pela rede. No *Hyperledger Fabric*, os blocos criados pelo *Ordering Service* são irrefutáveis, ou seja, uma vez que a transação é escrita em um bloco, a sua posição na cadeia é imutável (ANDROULAKI *et al.*, 2018). Essa característica diferencia o *Fabric* de outras implementações *blockchain*, pois as transações são adicionadas ao bloco de maneira determinística, assim, a cadeia não apresenta *forks* (descrito na Seção 2.3.1) como no *Bitcoin* (HYPERLEDGER, 2019).

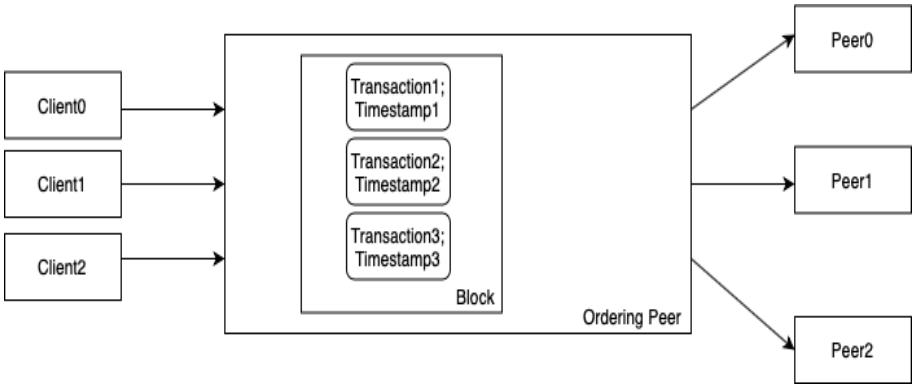


Figura 10 – Ordering service no *Hyperledger Fabric* Adaptado de (HYPERLEDGER, 2019)

A Figura 10 ilustra o processo do nó *Orderer* ao receber as transações dos clientes, inclui-las em um bloco e distribuir entre os demais nós da rede. Para cada transação é registrado um *timestamp* que não é associado ao seu momento de criação mas ao momento de classificação pelo *Orderer* para ser inserida no bloco.

Há três diferentes implementações do *Ordering Service* as quais o *Fabric* oferece suporte, com todas com a mesma finalidade de criação do bloco e distribuição desse bloco na rede.

- ***Solo:*** Nessa implementação há apenas um um único nó *Orderer*, por isso ela não é considerada *crash fault tolerante*, ou seja, caso esse nó apresente alguma falha o funcionamento de toda rede é comprometido. Por ser uma implementação mais simples não apresenta custos administrativos de manutenção de múltiplos nós. Porém, na perspectiva dos outros componentes da rede, o processamento das transações em blocos é o mesmo.

- ***Raft:*** É baseada no algoritmo de consenso *Raft*, um algoritmo para o gerenciamento da replicação de *logs* em um sistema distribuído. Foi desenvolvido com o objetivo principal de ser abrangente, em comparação com algoritmos semelhantes sobretudo o *Paxos*, para facilitar o ensino de estratégias de consenso para o registro de *logs* sem perder a precisão (ONGARO; OUSTERHOUT, 2014).

  O *Raft* segue um modelo de líder e seguidor, em que o líder é eleito dentro de cada *channel* e sua decisão é reproduzida pelos demais nós *Orderer*. Cada *channel* apresenta uma instância do *Raft*, o que permite a eleição um líder diferente para cada *channel*. Essa diferença de hierarquia entre os nós do *Ordering Service* é abstraída para o restante da rede.

  Os nós *Orderer* dessa implementação podem estar em três estágios: seguidor, candidato e líder. Todos os nós são inicializados como seguidores, nesse estágio eles podem receber entradas de um líder, caso tenha, ou votar para um líder. Se nenhuma entrada é recebida por um período de tempo, o nó é promovido para o estágio de candidato. Nesse estágio, o nó requisita votos de outros nós e caso receba a maioria dos votos é promovido para

líder. O líder tem a função de receber as transações, montar o bloco e replicar entre os nós *Orderer*.

Ao contrário da implementação *Solo*, o modelo *Raft* é considerado *crash fault tolerante*, ou seja, caso ocorra a falha de um nó *Orderer*, mesmo que esse seja um líder, não há perdas na rede. A única condição necessária é que a maioria dos nós *Orderer* permaneçam sem falhas assim pode haver a eleição um novo líder.

Comparado com a implementação *Kafka* é considerada uma maneira mais simples tanto para configuração quanto para administração do *Ordering Service* na rede do *Fabric*.

- ***Kafka:*** O *Apache Kafka* é um sistema de mensagem do tipo *publish-subscribe*, ou seja, há uma aplicação que publica e uma que se inscreve para receber as mensagens executado como um *cluster* de um ou mais servidores. O *cluster Kafka* atua como intermediário que coleta os dados de uma fonte e posteriormente entrega para uma aplicação que consumirá esses dados (FOUNDATION, 2017).

  O *cluster Kafka* funciona como uma instância única do serviço de mensagem servindo como um de *log* de confirmação externo de um sistema distribuído. Esse *log* ajuda a replicar dados entre os nós e atua como um mecanismo de ressincronização para que os nós com falha restaurem seus dados.

  Sua arquitetura é composta por *producers*, *consumers* e o próprio *cluster* (FOUNDATION, 2017). O *producer* é qualquer aplicação que publica mensagens no *cluster*, nesse cenário, os clientes da rede. Já o *consumer* é qualquer aplicação que recebe as mensagens do *Kafka*, os nós integrantes da rede excluindo os nós *Orderer*. Por último, *cluster Kafka* é a implementação do *Ordering Service* na coleta e distribuição das transações ordenadas nos blocos.

  Semelhante a implementação do *Raft*, o *Apache Kafka* também implementa o modelo líder e seguidor. As configurações do *Kafka* são gerenciadas pelo *Apache Zookeeper* mantendo a sincronização entre os elementos do *cluster Kafka*. Em caso de falha de um nó, o *Zookeeper* elege um substituto e recupera o estado anterior das informações. Dessa forma, também é considerada *crash fault tolerante*.

  Devido a sua arquitetura há uma sobrecarga na administração da rede ao incluir um *cluster Kafka* para o gerenciamento do *Ordering Service* na arquitetura do *Fabric*.

Além de criar os blocos no *Fabric*, o nó ou nós *Orderer* também mantêm a lista das organizações que podem criar *channels*. O *Ordering Service* aplica o controle de acesso aos *channels*, restringindo quem pode ler, escrever e configurar (HYPERLEDGER, 2019).

A Figura 11 exemplifica uma rede criada com o *Hyperledger Fabric* mostrando a interação entre os elementos da rede descritos acima. Nesse exemplo possível observar cinco nós participantes da rede, um deles está atuando como nó *Orderer* e os demais como nós pertencentes

a organizações. Os nós foram nomeados semanticamente, identificando a qual organização determinado nó pertence. Dentro de uma mesma organização os nós não apresentam hierarquia, porém podem ser separados em nós validadores ou não. Os nós identificados como validadores atuam na primeira fase do processo de validação das transações (que será detalhada nessa Seção). Uma possível relação que os clientes podem ter com a rede também foi incluída na Figura. No *Fabric*, os clientes interagem com as organizações não necessariamente importando qual nó pertencente a organização vai atender a requisição.

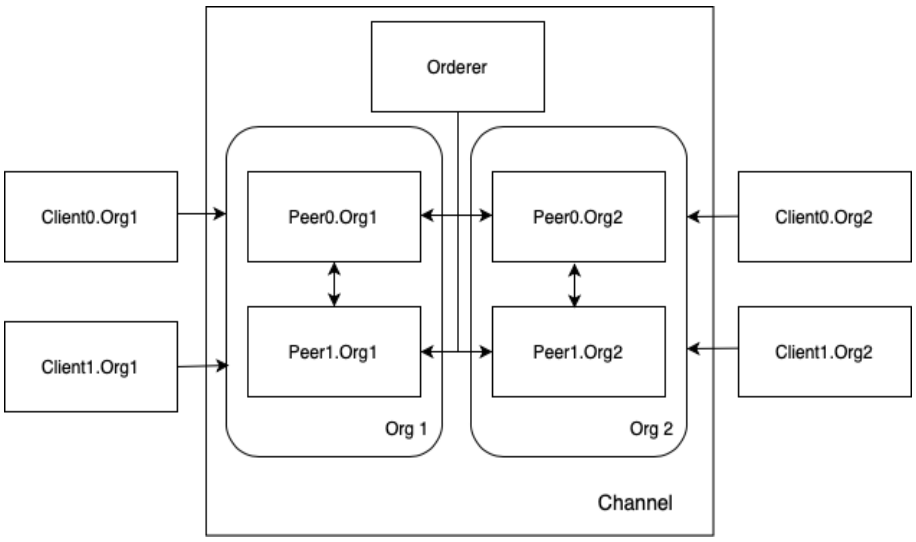


Figura 11 – Exemplo de rede desenvolvida com o *Hyperledger Fabric* Adaptado de (HYPERLEDGER, 2019)

No *Hyperledger Fabric* implementa-se rede de comunicação baseada na arquitetura P2P com adição de algumas particularidades. O protocolo de comunicação utilizado é próprio e chamado de *Gossip data dissemination protocol*. Este protocolo desempenha três funções principais na rede do *Fabric*, primeiro mantém a rede descobrindo novos nós e a adesão desses ao *channel*, identificando continuamente os nós disponíveis, e eventualmente detectando nós que foram desconectados. Segundo, dissemina os dados da cadeia entre todos os nós de um *channel*. Qualquer nó esteja fora de sincronia com o resto do *channel* que esta inserido, o protocolo identifica os blocos em falta e sincroniza, copiando os dados corretos. E, por último, atualiza os nós recém-conectados, permitindo a transferência de estado dados da cadeia entre os nós (HYPERLEDGER, 2019).

A comunicação entre os nós no *Fabric* é feita baseada na troca de mensagens entre nós por meio de RPC (utilizando a implementação gRPC – *Google Remote Procedure Call*) ou por comunicação RESTful por meio de servidores HTTP hospedados nos nós (HYPERLEDGER, 2019).

O bloco criado no pelo *ordering service* é composto de três seções: o *header* (cabeçalho), as transações armazenadas e o metadado do bloco. A Figura 12 demonstra quais são os

componentes do bloco e como as transações são armazenadas na arquitetura do *Hyperledger Fabric*.

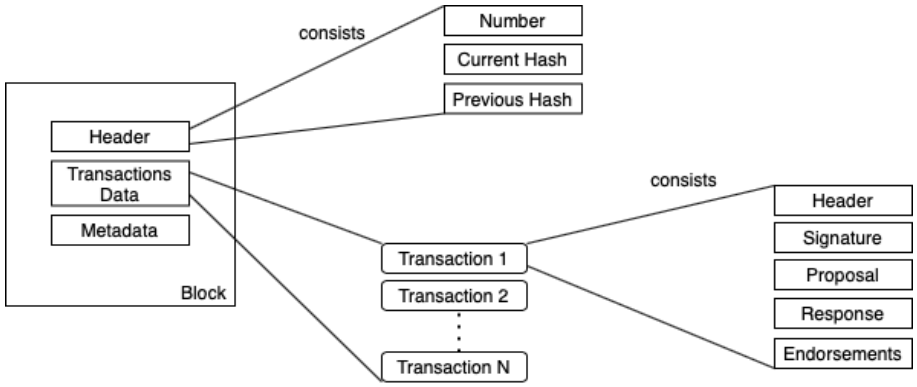


Figura 12 – Composição do bloco na arquitetura do *Fabric* Adaptado de (HYPERLEDGER, 2019)

O *header* do bloco é composto por três partes:

- **Número:** inicia com zero para o bloco *genesis* (o primeiro bloco criado o qual inicia a cadeia) e é adicionado um a cada bloco criado
- ***Hash* do bloco atual:** formada pela concatenação da *hash* de todas as transações contidas no bloco que está sendo criado
- ***Hash* do bloco anterior:** garante que o bloco está ligado a seu antecessor e, por consequência, a imutabilidade da cadeia

As transações são armazenadas no bloco na ordem determinada pelo *ordering service*. As transações são compostas pelos seguintes elementos:

- ***Header:*** contém informações relevantes sobre o metadado da transação.
- **Assinatura:** é uma assinatura criptográfica criada pela aplicação cliente e utilizada para verificar se os detalhes da transação não foram manipulados.
- **Proposta:** é a codificação dos valores de entrada de uma proposta de atualização da *blockchain*. Esta proposta, em conjunto com o valor atual do *world state* determina o seu novo valor.
- **Resposta:** capta os valores de antes e depois do *world state* no caso da transação seja validada. No cenário positivo, esse valor será aplicado à *blockchain* para atualizar o *world state*.
- **Validadores:** é a lista das respostas das organizações necessárias para garantir que a transação seja válida (a quantidade de respostas é determinada pela política de validação definida no *smart-contract*)

A última parte é o metadado do bloco que registra o *timestamp* da criação do bloco, certificado, chave pública e a assinatura do bloco. Além disso, para cada transação armazenada no bloco é registrado um indicador que determina se a transação é válida ou inválida.

Dentro do *Hyperledger Fabric*, há dois tipos de transação: *query* e *update*. A primeira é a mais simples, consistindo apenas da leitura de um valor armazenado na blockchain. Essa transação não precisa do processo de validação e consenso, visto na Seção 2.3.4. Já a transação de *update* altera a cadeia, sendo necessária a validação dos demais participantes da rede. O processo de validação de uma transação de *update* dentro da rede do *Fabric* ocorre em três etapas: proposta, ordenação e criação dos blocos e, por último, validação e confirmação (HYPERLEDGER, 2019).

Na primeira fase, os clientes notificam os nós validadores (determinados no *smart-contract*) sobre a proposta de transação que desejam realizar. Os nós validadores fornecem um parecer sobre a modificação proposta porém não aplicam a atualização à sua cópia da cadeia. Esses nós validam as transações de modo individual e retornam a resposta (se a transação é válida ou não). Quando há um número suficiente de respostas (valor também configurado no *smart-contract*) essa fase termina e a transação pode ser ou descartada ou seguir para a próxima etapa (HYPERLEDGER, 2019).

Na segunda fase, as transações validadas são ordenadas e posteriormente, agrupadas e inseridas em um bloco. As transações aprovadas na primeira fase são enviadas para o nó *Orderer*, que tem a função de ordenar as transações e criar o bloco que será distribuído na rede. O estabelecimento da ordem das transações é necessário pois é com base nela que os nós validam as transações e o bloco na etapa seguinte (HYPERLEDGER, 2019).

Na terceira fase, os blocos citados anteriormente são distribuídos para os nós da rede, onde cada transação é validada por cada nó (incluindo os nós não caracterizados como validadores) antes de ser aplicada à sua cópia local da cadeia. Cada nó valida as transações seguindo a ordem determinada na fase anterior de maneira independente. A ordenação das transações garante que os nós concordem ou não sobre as transações na mesma sequência, de forma que a cadeia se mantém a mesma nos diferentes nós da rede (HYPERLEDGER, 2019).

A Figura 13 apresenta o fluxo de uma transação de *update*, ou seja, que propõe uma mudança na cadeia. Os valores 1 e 2 na Figura mostram a primeira fase do processo de validação, o 3 ilustra a segunda etapa e o 4 a terceira. Em 1, o cliente notifica ao nó validador a transação e em 2 o nó retorna se a transação corresponde aos critérios necessários para ser validada. Se a transação for validada por esse nó, ela é enviada ao nó *Orderer*, em 3, onde as transações serão ordenadas e agrupadas em blocos. Em 4 essas transações são distribuídas na rede, onde serão validadas de forma individual por todos os nós da rede, incluindo os nós validadores. Somente nessa etapa que a cadeia é modificada em cada nó da rede.

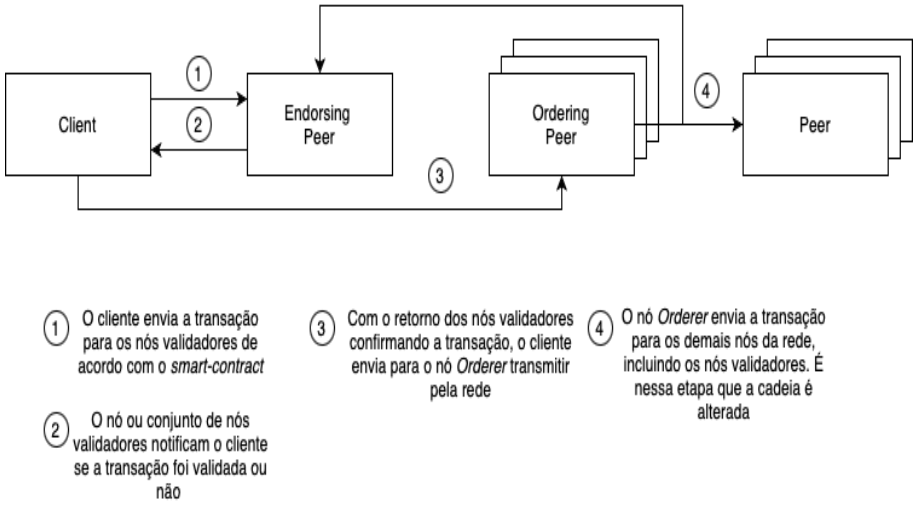


Figura 13 – Processo de validação de uma transação na rede do *Hyperledger Fabric* Adaptado de (HYPERLEDGER, 2019)

2.5.3 Hyperledger Caliper

Como já citado anteriormente, o *Hyperledger Caliper* é um conjunto de *benchmarks* que têm como objetivo definir um padrão para avaliar o desempenho de soluções *blockchain*. Os resultados encontrados com o uso do *benchmark* podem ser aplicados em outros projetos *Hyperledger*, assim como o *Hyperlegder Fabric* e as demais plataformas *blockchain* sob a tutela do *Hyperledger*, como ponto de referência no desenvolvimento baseado nessas plataformas *blockchain* (DHILLON; METCALF; HOOPER, 2017).

Atualmente, o *Hyperledger Caliper* permite obter resultados sobre a quantidade de transações aceitas na cadeia (*success rate*), vazão e latência. Além disso, é possível obter resultados do consumo de recursos computacionais, os quais incluem uso de *CPU*, memória e rede. Esses resultados são apresentados em forma de relatório, gerados na saída padrão (*stdout*) e também em arquivo *HTML* (DHILLON; METCALF; HOOPER, 2017).

Atualmente, o *Caliper* tem adaptadores para seguintes projetos *Hyperledger*: *Bezu*, *Burrow*, *Fabric*, *Iroha* e *Sawtooth*; e também ao *Ethereum* e *FISCO BCOS* (plataformas de desenvolvimento que não estão sob a tutela do *Hyperledger*) (DHILLON; METCALF; HOOPER, 2017).

De modo geral, o *Caliper* gera um *workload* para o sistema que se deseja avaliar, *system under test* (*SUT*), e continuamente monitora sua resposta. A partir desse resultado, o *Caliper* gera um relatório. O *framework* utiliza arquivos de configuração, onde são definidas a caracterização do *benchmark*, a configuração da rede e a carga de trabalho utilizada (PROJECT, 2020).

O arquivo *benchmark file configuration* descreve a execução do *benchmark*. Neste arquivo é definida a quantidade de repetições para cada *workload*. Além disso, ele coordenada a invocação dos *workloads* avaliados em cada execução. Também é nesse arquivo onde é determinada a taxa

de transações que devem ser submetidas para cada operação. A configuração desse arquivo é independente da plataforma a ser testada e, dessa forma, é possível utilizar o mesmo arquivo de *benchmark* para testar múltiplas plataformas blockchain(PROJECT, 2020).

Outro arquivo de configuração é o *network file configuration*, que representa a topologia da rede da plataforma a ser testada. Este arquivo contém informações relacionadas aos endereços dos nós integrantes da rede, quais são os clientes presentes na rede e quais são os *smart-contracts utilizados*. Essa configuração é totalmente dependente da plataforma testada e apresenta um formato específico de organização para cada sistema a ser testado (*SUT*) (PROJECT, 2020).

Há também os arquivos de *workload* onde são definidos os parâmetros para geração da carga de trabalho utilizada pelo *benchmark*. Esses arquivos são essenciais, pois é neles onde se encontra a lógica de construção e envio de transações que implementam o *benchmark*. Os *workloads* desenvolvidos determinam que tipo de transação será realizada, como leitura ou *update* (no caso do *Fabric*). Cada execução pode ter associado um ou mais módulos de *workload*, sendo necessário separar os arquivos de acordo com a semântica desejada (PROJECT, 2020).

Além dos arquivos de entrada mencionados, pode haver artefatos adicionais necessários para executar um *benchmark*. A necessidade desses elementos adicionais variam para diferentes *benchmarks* e execuções. Entre os artefatos pode incluir, materiais criptográficos necessários para interagir com o *SUT*, *smart-contract* a ser implementado pelo *Caliper*, arquivos de configuração de tempo real e pacotes pré-instalados para os módulos de *workload*. Para a execução deste trabalho, o artefato extra utilizado é o código fonte do *smart-contract* implementado (PROJECT, 2020).

A Figura 14 ilustra a arquitetura do *Caliper*. O módulo de *workload*, configuração do *benchmark* e da rede são os arquivos de entrada da execução do *Caliper* no qual as configurações são descritas. Outro parametro de entrada são os artefatos do *benchmark*, que depende do *SUT* utilizado. Já o *SUT* representa o sistema em teste, ou seja as plataformas *blockchain* com os quais o *Caliper* tem adaptadores; para este trabalho é usado o *Hyperlegder Fabric*. Através da configuração estabelecida nos arquivos de *workload*, o *Caliper* gera a carga de trabalho e transmite para o *SUT*. Através da carga gerada, o *Caliper* faz o monitoramento da quantidade de transações aceitas, vazão, latência e uso de recursos computacionais. Os dados coletados são agrupados e um relatório é gerado.

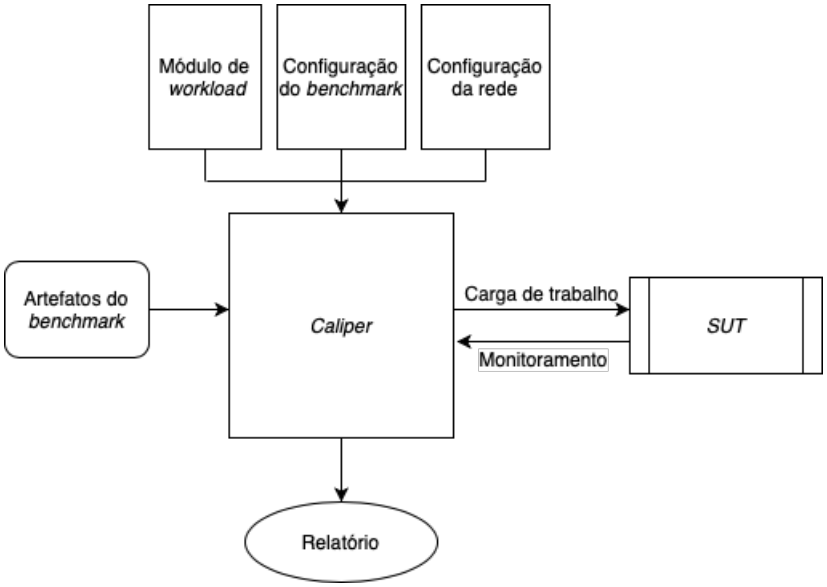


Figura 14 – Arquitetura do *Hyperledger Caliper*. Adaptado de (PROJECT, 2020)

O *Caliper* divide a execução do *benchmark* entre dois tipos de processo: o *master* e os *workers*. O processo *master* é responsável por toda lógica de execução do *benchmark*, com exceção do processamento da carga de trabalho, a qual é atribuído aos *workers*. A execução do *benchmark* é realizada nos seguintes passos (PROJECT, 2020):

1. O processo *master* inicializa o *Caliper* e a plataforma a ser testada, instala os *smart-contracts* associados à execução na plataforma.

2. Finalizado a etapa anterior, o *master* faz o escalonamento das rodadas de execução, dividindo a carga de trabalho entre os *workers*.

3. Quando os processos do tipo *worker* terminam a execução, o *master* é responsável por coletar os resultados e gerar o relatório de execução.

4. Por último, o *master* finaliza a execução do *benchmark*.

Já a execução da carga de trabalho é processada nos *workers*. A divisão da carga de trabalho é balanceada entre o número de *workers* presentes, de forma determinada na configuração do *benchmark*. Os *workers* alternam entre duas tarefas: execução da carga de trabalho e esperar serem liberados para executar a próxima rodada de testes (PROJECT, 2020).

2.6 Considerações Finais

Este Capítulo iniciou com uma introdução do contexto do lançamento do *Bitcoin* e a infraestrutura que fundamenta a plataforma, a *blockchain*. Além disso, foi apresentada uma breve

evolução da *blockchain*, que surgiu como infraestrutura de suporte de uma única criptomoeda e se tornou uma ferramenta de apoio ao armazenamento de registros em diversos cenários.

Em um segundo momento, houve a descrição do funcionamento da *blockchain*. Essa infraestrutura compreende a inserção de registros na cadeia, o processo de criação dos blocos que formam essa cadeia, características da rede na qual a cadeia está distribuída, validação desses dados antes de serem armazenados e o processo de consenso da rede. Em conjunto, foram levantados os principais pontos nos quais a tecnologia se destaca e em quais ela é limitada.

Por último, foram apresentadas as duas principais plataformas que dão suporte ao desenvolvimento de aplicações *blockchain*: *Ethereum* e *Hyperledger Fabric*. Em conjunto com essas plataformas, o *Hyperledger Caliper* um *benchmark* para a avaliação de plataformas *blockchain* também foi introduzido.

O próximo Capítulo contém a descrição dos trabalhos relacionados a esse projeto de mestrado. Em um primeiro momento é descrito quais as etapas que foram realizadas no levantamento bibliográfico, em sequência é apresentado em detalhes os trabalhos encontrados na busca realizada. Ao final do Capítulo é evidenciado as principais lacunas da área através da análise da literatura obtida.

CAPÍTULO

3

TRABALHOS RELACIONADOS

3.1 Considerações Iniciais

Este Capítulo apresenta o levantamento bibliográfico de trabalhos relacionados aos testes e verificações de plataformas e/ou componentes *blockchain*. Em um primeiro momento, serão descritas as etapas realizadas durante a execução do levantamento bibliográfico. Em seguida, os trabalhos encontrados serão detalhados de forma a evidenciar a metodologia de teste utilizada e os resultados encontrados. Além disso, também serão apontadas quais são as contribuições e lacunas desses projetos.

Os resultados encontrados nesta etapa do trabalho são essenciais para caracterizar o estado da arte das pesquisas realizadas no contexto de avaliação da *blockchain*. Através da análise desse cenário, é possível levantar quais são as lacunas ainda não atendidas pelos estudos, para assim determinar o direcionamento dos trabalhos que podem ser realizados a fim de complementar a literatura.

3.2 Levantamento Bibliográfico

O Capítulo 1 apresenta exemplos de aplicações que utilizam a *blockchain* em áreas distintas à financeira, como na educação e na saúde. A avaliação dessas aplicações permitiu constatar que há poucos estudos que analisam a aplicabilidade e a viabilidade desses *frameworks*. Nesse cenário, o objetivo da busca descrita nesta Seção é investigar como os componentes e plataformas *blockchain* são testadas e avaliadas.

O processo de levantamento de trabalhos realizado neste trabalho não seguiu todas as etapas de uma revisão sistemática, tal como proposto por (KITCHENHAM, 2004), porém foi inspirada nas etapas que caracterizam esse processo. A metodologia empregada nessa investigação são descritas nesta Seção.

A primeira etapa elaborada é referente à definição do objetivo da revisão da literatura. Dessa forma, o objetivo desse levantamento é identificar o estado da arte de pesquisas relacionadas e os testes de componentes e plataformas *blockchain*. Além disso, visa averiguar quais as tendências de pesquisas da área juntamente com a análise de possíveis brechas não englobadas pelos trabalhos da área.

A busca realizada não limitou ao domínio da aplicação ou plataforma de desenvolvimento no qual a *blockchain* é avaliado. Isso se deve à intenção de apurar como são realizados os testes em todos os possíveis contextos de uso da *blockchain*. Dessa forma tentou-se identificar quais são os meios para a verificação de aplicações, componentes e plataformas *blockchain* de modo a mapear em quais áreas esses trabalhos estão concentrados.

Com o objetivo definido foi possível determinar quais as questões de pesquisa que o levantamento bibliográfico pretende satisfazer. São elas:

QP1: Quais são os elementos da infraestrutura *blockchain* que são avaliados pelos estudos?

QP2: Quais são as plataformas *blockchain* utilizadas e/ou avaliadas no estudos?

QP3: Qual a metodologia utilizada nesses estudos, em relação à forma de avaliação, tipo de aplicação e ambiente de execução?

Outra etapa que qualifica um mapeamento sistemático é a escolha de uma base de dados relevante. Neste trabalho, optou-se pela ferramenta de busca *Scopus*.

A chave de busca foi construída com base na ferramenta de busca selecionada. A Tabela 2 mostra a chave de busca utilizada.

(“test” OR “validation” OR “verification”) AND (“blockchain”)

Tabela 2 – Chave de busca

Seguindo as etapas do mapeamento sistemático descritos em (KITCHENHAM, 2004), é necessário definir critérios para a seleção dos estudos de modo a direcionar a investigação a responder as questões de pesquisa. O Quadro 3 apresenta a relação dos critérios empregados nessa etapa.

Quadro 3 – Relação dos critérios de inclusão e exclusão

Critérios de Inclusão	Critérios de Exclusão
Estudos que apresentem teste, validação ou verificação de componentes da *blockchain*.	Estudos de componentes da *blockchain* que não descrevam no mínimo um dos itens: teste, validação e verificação.
Estudos que apresentem teste, validação ou verificação de plataformas *blockchain*.	Estudos de plataformas *blockchain* que não apresentem no mínimo um dos itens: teste, validação e verificação.
-	Estudos que apresentem uma proposta de aplicação, sem um teste, validação ou verificação das funcionalidades da mesma.
-	Estudos que apresentem perspectivas de estudos, sem uma análise ou avaliação de componentes ou plataformas.
-	Chamadas de eventos.
-	Estudos em idioma diferente do inglês.

A definição dos critérios de inclusão e exclusão permitiu a análise dos trabalhos encontrados, de forma a selecionar ou eliminar artigos de acordo com os parâmetros estabelecidos. A avaliação ocorreu em três fases, primeiramente, os trabalhos retornados na busca foram analisados de acordo com título. Em um segundo momento, ocorreu a avaliação dos resumos, e para a terceira fase os conteúdos dos artigos foram lidos para que pudéssemos coletar detalhes dos mesmos.

A Tabela 3 mostra a quantidade de estudos selecionados em cada etapa de avaliação. Através da Tabela 3, é possível notar a quantidade de trabalhos que foram eliminados pela leitura do título, isso se deve ao alto interesse em estudos relacionados à *blockchain*. Esse cenário gera uma grande quantidade de trabalhos retornados que são relacionados à infraestrutura, porém não apresentam referências a teste, verificação e validação desses componentes.

Fase	Quantidade de Estudos Selecionados
I	364
II	46
III	24

Tabela 3 – Estudos selecionados após a aplicação dos critérios de inclusão e exclusão.

O levantamento bibliográfico elaborado resultou na avaliação de 18 (dezoito) trabalhos, os quais são descritos e analisados nas próximas seções. A análise desses trabalhos permitiu apurar quais são as propostas apresentadas no contexto da *blockchain* e como as mesmas são validadas, principalmente em relação à avaliação do desempenho. Além disso, foi possível identificar quais as tendências de avaliações e qual a metodologia utilizada. Mapear esses resultados permitiu reconhecer as lacunas e perspectivas de trabalhos futuros.

Apesar do resultado satisfatório na realização das etapas descritas, o levantamento bibliográfico realizado não seguiu o rigor científico de um mapeamento sistemático da literatura. Isso se deve à limitação do número de participantes na realização desse processo, pois a falta de uma avaliação de pares é uma restrição da busca realizada. Outro fator é o uso de apenas uma única base de dados. Por último, não houve uma avaliação de qualidade dos trabalhos encontrados, sendo esta a etapa final descrita em (KITCHENHAM, 2004).

3.3 Descrição dos Trabalhos

Os trabalhos desta Seção foram agrupados de acordo com o tipo de estudo realizado sobre as funcionalidades e elementos da *blockchain*. Os sete primeiros trabalhos descritos contêm uma proposta com relação à análise da ordem de execução dos *smart contracts* da plataforma *Ethereum*. Esses estudos foram baseados no *DAO exploit*, descrito na Seção 2.5.1. Do total de trabalhos retratados, três deles apresentam questões diversas sobre a *blockchain*. O primeiro deles contém uma análise sobre as transações do *Bitcoin*. O segundo discorre a respeito de uma avaliação realizada a um algoritmo de consenso. O último mostra uma solução na qual a *blockchain* é apresentado como serviço.

Os catorze trabalhos restantes foram agrupados por incluírem em seus estudos a avaliação de desempenho de plataformas *blockchain*. Esses trabalhos podem ser divididos entre aqueles que apresentam uma comparação de performance entre duas ou mais plataformas ou, os que pretendem avaliar apenas uma. Apesar dessa diferença, mostram o mesmo tipo de avaliação de desempenho e questões de pesquisa semelhantes. Dois dos trabalhos descritos introduzem uma perspectiva diferente de avaliação, trazendo uma avaliação de requisitos.

O *DAO exploit* ocorreu devido à ordem de execução dos comandos dos *smart contracts*, pois a ordem de execução não era conhecida. Considerando essa brecha na segurança, os autores do trabalho (BAI *et al.*, 2018) propõem métodos formais para garantir a correta ordem de execução dos *smart contracts* para assim assegurar a segurança das aplicações. Além disso, os autores apresentam uma descrição geral sobre os *smart contracts* e como eles podem ser representados, tanto como tuplas quanto máquinas de estado finito. Outra contribuição, é o que os autores chamaram de *PROMELA*, um modelo oferecido para verificar as propriedades dos *smart contracts*.

Seguindo o raciocínio de solução para o problema da incerteza na ordem de execução dos *smart contracts* em (ABDELLATIF; BROUSMICHE, 2018), aponta-se que soluções que usam a documentação de vulnerabilidades e verificação formal não capturam o comportamento da *blockchain* e dos usuários. Desta forma, são ineficientes para solucionar a questão da ordem de execução. Dado esse cenário, os autores apresentam um modelo formal de verificação em ambiente de execução, para que dessa forma as vulnerabilidades dos *smart contracts* possam ser analisadas usando um modelo estatístico. Como forma de validação da proposta, são apresen-

tados os resultados práticos, através da análise de *smart contracts* para o registro de nomes na plataforma *Ethereum*.

No artigo (ELLUL; PACE, 2018) é descrita uma outra abordagem usando verificação em tempo de execução como forma de examinar as dependências e veracidade dos *smart contracts*. Segundo os autores, a verificação estática em tempo de compilação apresenta limitações que sugerem que a verificação em tempo de execução seja a ideal no contexto dos *smart contracts*. Como ferramenta de validação dessa declaração, os autores apresentam uma ferramenta parcialmente implementada de *proof-of-concept* chamada *ContractLarva*. Seu diferencial, segundo os autores, é apresentar mecanismos de reação a violações ocorridas aos *smart contracts*.

Também usando verificação em tempo de execução, no artigo (BHARGAVAN *et al.*, 2016), os autores oferecem um *framework* para análise e verificação de segurança e funcionalidade em tempo de execução dos *smart contracts*. Os autores consideram o compilador do *Solidity* (linguagem de *script* do *Ethereum*) não confiável e por isso desenvolveram uma outra forma de verificação. Esse *framework* foi desenvolvido em *F**, uma linguagem funcional destinada à verificação de programas escritos em *Solidity*. Com esse *framework* é possível comparar os resultados da execução com os resultados gerados pela *EVM* (*Ethereum Virtual Machine*) e verificar falhas.

Diferente dos trabalhos até agora apresentados, no trabalho (GRISHCHENKO; MAFFEI; SCHNEIDEWIND, 2018a) é apresentada uma solução para a análise estática da *Ethereum Virtual Machine (EVM)*. O *framework EtherTrust* apresentado tem como objetivo permitir a investigação do *bytecode* da *EVM*. Complementar a esse *framework*, há um propósito em apresentar o estado da arte no teste, na validação e na verificação dos *smart contracts* do *Ethereum*, com a intenção de mostrar os estudos já propostos e quais as possibilidades de trabalhos futuros.

O artigo (GRISHCHENKO; MAFFEI; SCHNEIDEWIND, 2018b) também traz uma solução que permite a investigação do *bytecode* da *EVM*, apresentando uma demonstração sobre a análise semântica do *bytecode* da *Ethereum Virtual Machine*. Segundo os autores, há uma série de erros e imprecisões nas ferramentas atuais de análise semântica e verificação dos *smart contracts* da plataforma *Ethereum*. Os autores apresentam uma demonstração, com o uso de exemplos, que as técnicas atuais de análise são imprecisas e necessitam de modificações. Essa falha valida a busca por soluções que permitam um processo de exploração mais rigoroso. O artigo também apresenta a definição formal de propriedades de segurança dos *smart contracts*, como integridade, atomicidade e independência dos parâmetros.

Por último, em (SERGEY; HOBOR, 2017) é apresentada uma visão sobre os problemas de computação de alto desempenho que podem ocorrer na execução de *smart contracts*. Para os autores a execução de *smart contracts* segue a mesma lógica aplicada na execução de *threads* em ambiente de memória compartilhada, sendo assim, apresenta os mesmos problemas e as mesmas soluções. Nesse trabalho é destacada a falta de controle sobre a ordem de execução dos *smart contracts* por seus comandos estarem em transações diferentes. Dependendo da aplicação, pode-

se gerar erros de atualização de variáveis, assim como acontece em um ambiente de memória compartilhada. Nesse sentido, os autores propõem soluções que remetem àquelas já consolidadas nesses ambientes, como, por exemplo, o uso de *locks* e *mutex* de forma que seja nativa dos *smart contracts*, sem precisar de uma terceira entidade para garantir a ordem de execução.

Os próximos três trabalhos encontrados apontam estudos diversos sobre a avaliação e verificação do comportamento da *blockchain* porém, não foi possível agrupá-los usando um fator em comum.

Em um contexto de avaliação e verificação de transações, no artigo (MERCANTI; BISTARELLI; SANTINI, 2018) os autores coletaram as transações do *Bitcoin* com o intuito de investigar se o comportamento padrão é atingido. De acordo com os autores, transações padrão são aquelas que apresentam o comportamento esperado pelo sistema, as quais não apresentam sinais de tentativa de provocar erros ou falhas. Para os autores, esse estudo é importante para entender como o *Bitcoin* tem sido explorado. Através da análise pode-se concluir que o padrão de comportamento é respeitado com uma pequena porcentagem de 0,03% das transações constando como não-padrão. Outro resultado apresentado foi a classificação de transações em sete tipos para os padrões e, em nove para as não-padrão.

No cenário de avaliação de algoritmos de consenso, no trabalho (THIN *et al.*, 2018) os autores destacam a falta de estudos relacionados à avaliação dos algoritmos de consenso a fim de garantir segurança e precisão do sistema. Com o objetivo de atender essa demanda, os autores realizaram uma análise do protocolo de consenso *Tendermint*, que é uma variação do algoritmo de consenso *Proof-of-Stake*. Os resultados obtidos mostram que 67% da rede precisa estar em conformidade para haver o consenso. Mesmo com esses resultados, os autores apontam que a minoria dos nós ainda podem influenciar o consenso da rede.

Baseados no lançamento da *Blockchain-as-a-Service* nos *data centers* da *Microsoft*, os autores do trabalho (MELO *et al.*, 2018) propuseram uma forma de avaliação com a finalidade de averiguar a viabilidade do projeto. O conceito *Blockchain-as-a-Service* é caracterizado pela possibilidade de criar e gerenciar uma rede *blockchain* sobre ambientes de computação em nuvem. A metodologia apresentada tem como objetivo a verificação dos requisitos de confiabilidade e disponibilidade. Os testes foram realizados em duas máquinas, em que uma atua como mestre e a outra como escrava, configuradas com o *Hyperledger Cello*. Segundo os autores, os testes evidenciam que o sistema não suporta mais de 31 dias sem falhas.

O trabalho de (WEBER *et al.*, 2017) objetiva comparar o desempenho de duas plataformas. Os autores apresentam uma análise sobre as limitações em relação à disponibilidade de dados em comparação a duas plataformas, o *Bitcoin* e o *Ethereum*. Esse estudo objetiva identificar questões de disponibilidade de dados dessas plataformas.

Com o resultado alcançado, foi possível pontuar quantas transações são validadas assim que recebidas na rede e quantas têm que esperar certas condições serem atendidas para ocorrer

a inserção. Outra conclusão mostrada foi o impacto do *Transaction Fee* no tempo de inserção de uma transação. Também foi exposta a relação de transações que não são inseridas devidos a *forks* na cadeia e qual o percentual delas em relação ao total de transações realizadas. Além disso foi mostrado o impacto da rede no tempo de inserção das transações.

Para cada plataforma foram inseridos nós na rede com a finalidade de coletar o tempo em que as transações são inseridas na rede de forma permanente. O tempo total de inserção calculado baseou-se no tempo de inserção da transação na cadeia somado ao tempo de criação de novos blocos que a tornem imutável, mesmo com a presença de *forks* na cadeia. No caso do *Bitcoin* são seis blocos e para o *Ethereum* são doze.

Para os testes do *Bitcoin* utilizou-se uma implementação modificada do *BitcoinDark* para coletar as transações e determinar o tempo para serem aceitas pela rede permanentemente. Os dados foram coletados em duas ocasiões distintas, em Novembro de 2016 e em Abril de 2017, em um período de vinte e quatro horas em cada coleta. Em cada experimento coletou-se cerca de trezentas mil transações.

Um nó de observação foi inserido na rede para o *Ethereum*, baseado na implementação do cliente *Geth*. Para os experimentos, definiu-se o *Transaction Fee* como zero, ou seja, as transações não apresentam bonificações, de forma que não são privilegiadas na *Transaction Pool*. As demais configurações foram ajustadas como padrão. Esse nó foi conectado com outros quinhentos nós da rede do *Ethereum*, permitindo que mais de seis milhões de transações fossem coletadas. Os dados foram coletados por em torno de noventa e sete dias.

O trabalho de (WEBER *et al.*, 2017) se mostra relevante, no contexto deste projeto de mestrado, por apresentar o detalhamento da avaliação de desempenho baseado em plataformas com o uso de redes públicas. Além disso, a verificação sobre a porcentagem de transações que são removidas da cadeia por conta de *forks* é essencial para validar os estudos sobre a aplicabilidade da *blockchain* em áreas não correlatas à financeira. Outras áreas que podem se beneficiar do uso da *blockchain* precisam ter assegurado o acesso aos dados, mostrando a importância desse tipo de análise.

Mesmo com um número significativo de resultados, o artigo é limitado por não apontar o impacto de seus resultados para possíveis aplicações do uso da *blockchain*. Isso gera uma lacuna relativa ao potencial do uso da plataforma em aplicações diferentes das financeiras, possibilitando explorar outros tipos de dados e configurações.

Outro trabalho que visa comparar duas plataformas é o (PONGNUMKUL; SIRIPANPORNCHANA; THAJCHAYAPONG, 2017), no qual o *Hyperledger* e o *Ethereum* são avaliados e comparados em um contexto em que a rede é privada. Como diferencial dos outros trabalhos, além da análise de vazão e latência, também apresenta a performance em relação ao tempo de execução.

Os resultados apresentados pelo artigo focam na comparação entre as plataformas. O

método utilizado pelos autores indica que o *Hyperledger* atinge maior vazão e menor latência quando a carga de trabalho atinge 10 mil transações, se comparado ao *Ethereum*, com o mesmo número de transações. O estudo também mostrou que, com os mesmos recursos computacionais, o *Ethereum* é capaz de suportar mais operações paralelamente do que a plataforma do *Hyperledger*. O artigo também traz como contribuição a divulgação de uma metodologia de teste que permite a reprodutibilidade.

Para alcançar esses resultados, foi desenvolvida uma aplicação sintética para transferência monetária com as funções: criar conta, depositar e transferência. Essas funções foram utilizadas para realizar as transações, escolhidas de modo arbitrário e com valor aleatório para cada operação. Foram lançadas em tempo não sincronizado, variando a frequência das chegadas. A frequência de chegada foi de 1 transação até 10 mil requisições por teste, sendo realizadas dez baterias de teste com o intuito de calcular a média e apontar os resultados.

Dentro do contexto deste trabalho, o estudo mostra a importância da caracterização das plataformas para diferentes tipos de contexto que possam ser utilizadas. Dessa forma, mostra que uma plataforma não se sobrepõe a outra, mas dependendo da necessidade, deve-se avaliar qual melhor se adapta ao contexto. A metodologia utilizada é outra contribuição do artigo, uma vez que apresenta três tipos de transações, demonstrando mais variedades de transações que os outros trabalhos encontrados e por consequência, apontando para a necessidade de mais estudos com essas características.

Se por um lado o trabalho apresenta uma pequena variedade de transações, por outro, falha em destacar os resultados isolando cada uma dessas funções e não explorando o impacto de diferentes transações em um mesmo contexto. A intenção dos autores, em comparar as duas plataformas foi concluída, porém era esperado uma análise mais aprofundada sobre quais as peculiaridades de cada uma que levaram a tais resultados. Somados a isso, a discussão sobre o impacto dos resultados para futuras aplicações que desejem utilizar a *blockchain* foi limitada. Dessa forma, o trabalho evidencia aspectos que precisam ser explorados.

Com o objetivo de comparar plataformas, o trabalho (DINH *et al.*, 2017) propõe um *framework* chamado *BlockBench* que possibilita comparar a performance de plataformas distintas. Além disso, o *framework* também busca a identificação de gargalos para contribuir com possíveis melhorias no desempenho das plataformas avaliadas. Esse artigo considera como desempenho a vazão, latência e acrescenta tolerância a falhas.

Nesse artigo as plataformas *Hyperledger*, *Ethereum* e *Parity* foram avaliadas e comparadas. Os resultados mostraram que em termos de vazão o *Hyperledger* supera os outros dois sistemas, porém o *Parity* mostra a menor latência das três plataformas. No cenário de falhas de nós, o funcionamento tanto do *Ethereum* quanto do *Parity* não é afetado, evidenciando a tolerância a falhas dessas plataformas. Quando comparado à eficiência das três plataformas, o *Hyperledger* se destaca, seguido pelo *Parity* e o *Ethereum*.

Para comparar as plataformas foi criado o *BlockBench*, um conjunto de *benchmarks* desenvolvidos com a finalidade de avaliá-las e compará-las. Para esse trabalho, os autores produziram seis *benchmarks*, porém apontam como sugestão para trabalhos futuros a criação de outros para, assim, generalizar os resultados.

Os *benchmarks* foram lançados com a intenção de avaliar determinada camada da plataforma. Nesse trabalho, os autores se basearam na divisão da plataforma em quatro camadas. A camada mais baixa é a de protocolo que contém o protocolo de consenso da plataforma a ser avaliada. A segunda é a camada de dados onde está a estrutura, o conteúdo e as operações com relação aos dados. A terceira é a camada de execução que inclui os detalhes do ambiente de execução. A camada mais externa é a de aplicação que apresenta as classes de aplicações. Com essa divisão foram criados dois macro-*benchmarks* para avaliar a camada de aplicação e quatro micro-*benchmarks* com o objetivo de examinar as demais camadas.

Com os resultados da avaliação de cada camada, os autores concluíram que as aplicações mais estáveis se concentram nas *criptomoedas*. Outra conclusão dos autores é que as plataformas poderiam explorar os *multi-cores* disponíveis nas máquinas a fim de otimizar as operações realizadas. Por fim, eles apontaram o *Ethereum* como a plataforma mais madura em termos de base de código, usuários e comunidade de desenvolvimento.

A contribuição desse trabalho e seu diferencial em relação aos demais encontrados é a descrição da *blockchain* com o uso de camadas e como elas poderiam ser avaliadas de forma independente. Também é mostrada a falta de dados disponíveis que garantidamente testem o funcionamento das plataformas e a necessidade de produzi-los. Dos trabalhos analisados, foi o único que trouxe uma justificativa para o uso de poucos nós nos testes realizados no trabalho.

Ao propor a comparação o desempenho das plataformas, o trabalho não mencionou que a plataforma *Parity* é um cliente do *Ethereum* e que, por isso, ambas apresentam características em comum por se basearem na mesma implementação. Essa falta de caracterização, tanto em aprofundar o funcionamento do *Parity* quanto em mostrar os pontos que elas têm diferentes, deixou a comparação sem justificativa. Como outros trabalhos apresentados, não há uma preocupação em apresentar um impacto dos resultados para futuros desenvolvedores de aplicações *blockchain*, não dando a finalidade esperada ao estudo.

Com outra perspectiva de comparação, o trabalho (ROUHANI; DETERS, 2017) propõe estabelecer o efeito de diferentes clientes na performance de aplicações que usam a *blockchain Ethereum*. Para isso, comparou dois clientes do *Ethereum* o *Geth* e o *Parity*.

O principal resultado dessas avaliações é a conclusão que 89,8% das transações são mais rápidas no cliente *Parity* em comparação com o cliente *Geth*. Outro fator observado nos testes realizados é que o aumento da capacidade de armazenamento da *RAM* impacta positivamente no desempenho de ambos os sistemas.

Os resultados apresentados pelos autores foram baseados em testes realizados que

lançaram um total de duas mil transações para cada cliente e calcularam o tempo no qual elas foram inseridas na cadeia. Uma variação desse teste foi a alteração da capacidade de *RAM* disponível nos computadores utilizados para os testes a fim de avaliar o impacto dessa mudança no tempo de inserção de uma transação.

A descrição da metodologia utilizada não especifica as condições nas quais esses testes foram realizados. A falta de detalhamento em relação à rede utilizada, à quantidade de nós e como eles estavam distribuídos impossibilita a reprodutibilidade do teste e prejudica a credibilidade dos resultados apresentados. Outro fator que impacta no resultado final da avaliação de desempenho é o tipo da transação realizada, porém o estudo apresentado não detalhou as informações relativas às transações, interferindo na validade do resultado apresentado.

Além dos problemas relativos à falta de reprodutibilidade dos testes realizados, não há uma discussão sobre os resultados alcançados. Os resultados teriam que ser analisados de acordo com a proposta de cada cliente, considerando decisões de projeto e particularidades que impactam no desempenho. Da forma apresentada, desconsiderando esses fatores, perde a validade da comparação.

Outros estudos optaram por apresentar a avaliação de apenas uma plataforma, como em (BALIGA *et al.*, 2018). Neste artigo os autores pretendem caracterizar a performance do *Hyperledger Fabric* baseado na vazão e latência do sistema. Além disso, os autores pretendem avaliar quais fatores da plataforma que modificados apresentam maior impacto no desempenho do sistema.

Através dos testes realizados, os autores perceberam que um dos gargalos apresentado pela plataforma é a validação sequencial das transações realizadas por cada nó. Também foi destacada a interferência da política de validação dessas transações sobre o desempenho do sistema. A política de validação se refere a quais características uma transação deve apresentar para ser considerada válida e segura. Em conjunto com esse resultado, também foi avaliada a influência do tamanho do bloco na vazão do sistema. Segundo os autores, diminuir o tamanho do bloco reduz a vazão pois acaba necessitando que um número maior de blocos seja criado e com isso, mais eventos ocorrem, acarretando em queda do desempenho de forma geral.

Os testes realizados neste trabalho, (BALIGA *et al.*, 2018), podem ser divididos em três categorias. A primeira categoria pretende avaliar a latência e vazão, com essa finalidade foi utilizado um *benchmark* chamado *Caliper* para gerar o *workload*. Nessa categoria de testes o *Caliper* também foi utilizado para coletar o *timestamp* das transações, de modo a permitir o cálculo das métricas. A segunda categoria de testes constitui em avaliação da escalabilidade. Nesse teste o número de nós foi variado com o objetivo de calcular o impacto dessa mudança no desempenho da plataforma, considerando as métricas de vazão e latência.

Na terceira categoria de testes foram criados cinco *benchmarks* com o objetivo de classificar o impacto da mudança de alguns parâmetros de configuração no desempenho da

plataforma. Os parâmetros que sofreram alterações foram: o tamanho do conjunto de requisições, o tamanho das transações, os *chaincodes* e o valor do *Transaction Fee*. As modificações ocorreram de forma isolada, permitindo a avaliação individual de cada propriedade.

Como contribuição para futuros desenvolvedores de aplicações com a *blockchain*, os autores destacam a modularização como uma futura solução para mapear novas funções. Também destacam que essa solução deve ser acompanhada por uma avaliação sobre o impacto no desempenho da aplicação. Outra conclusão que os autores identificaram foi a limitação do número de nós que uma rede *consortium*, definida na Seção 2.3.2, consegue garantir a performance desejável.

As conclusões evidenciadas neste trabalho trazem uma discussão relevante sobre as possibilidades de desenvolvimento de aplicações com o *Hyperledger Fabric*, porém não cobrem aspectos no que diz respeito ao tipo e o tamanho dos dados. Os próprios autores destacaram essa limitação.

O estudo (THAKKAR; NATHAN; VISHWANATHAN, 2018) também contém uma análise detalhada do *Hyperledger Fabric*. Os objetivos se assemelham ao artigo (BALIGA *et al.*, 2018) em que os autores caracterizam a performance da *blockchain* mediante a análise da vazão e latência do sistema. Juntamente com esse resultado, também foi avaliado o impacto no desempenho de mudanças nas configurações da plataforma.

Assim como o trabalho de (BALIGA *et al.*, 2018), os autores concluíram que a validação serial das transações é um gargalo apresentado pela plataforma, porém extrapolam esse resultado e incluem outros dois fatores que impactam significativamente o desempenho. As operações de criptografia e as chamadas para o banco de dados foram listadas como outros gargalos da plataforma. A primeira se refere ao cálculo da *hash* apresentado na Seção 2.3.1. A segunda a forma de armazenamento *Off-Chain* descrito na mesma seção.

Os autores também avaliaram como a taxa de chegada das transações influenciam na latência e na vazão da plataforma. Em conjunto com essa análise, também foi investigado o peso do tamanho do bloco no desempenho do *Hyperledger Fabric*. Os resultados observados mostram a relação entre tamanho do bloco, taxa de chegada das transações e a latência, quando o tamanho do bloco é inferior à taxa de chegada das transações a latência aumenta. A latência diminui conforme a taxa de chegada ultrapassa o tamanho do bloco.

Nesse artigo também foi avaliado o uso de *CPU* para verificar a taxa de utilização desse recurso em relação ao crescimento da taxa de chegada de transações. Outra contribuição exibida no artigo é a comparação da performance de dois bancos de dados, o *GoLevelDB* e o *CouchDB*, na plataforma, uma vez que o *Hyperledger Fabric* oferece suporte a ambos.

Os resultados foram obtidos através de um *benchmark*, criado pelos autores, em uma configuração de rede com oito nós disponíveis, divididos em quatro organizações distintas, com dois nós em cada. Os autores justificaram o desenvolvimento de um *benchmark* próprio pela falta

de opções que garantidamente testem o desempenho da *blockchain*. A partir dessa configuração, os experimentos foram realizados em um intervalo de mil segundos.

A principal contribuição desse trabalho foi mostrar quais os principais fatores que influenciam no desempenho da plataforma. Além dessa identificação, para cada fator apresentado, os autores propuseram melhorias que poderiam ser aplicadas a fim de diminuir o impacto desses gargalos. Os autores apontam como futuros trabalhos a avaliação da plataforma com diferentes configurações, considerando nós geograficamente distantes.

Apesar da descrição de cada resultado apresentado ser completa, assim como outros trabalhos, há uma limitação na discussão de como essas conclusões podem ajudar no desenvolvimento de novas aplicações usando a *blockchain*. Em relação à metodologia apresentada, não houve a especificação do tipo de dado utilizado. Essa lacuna compromete a validação dos resultados apresentados, ao mesmo tempo que deixa em aberto a falta de estudos que variem o tipo e o tamanho de dados para avaliar o desempenho da *blockchain*.

O artigo (CHEPURNOY; RATHEE, 2018) exibe uma perspectiva distinta de avaliação da *blockchain*. Os trabalhos anteriores propuseram a avaliação do desempenho das plataformas *blockchain* por meio da análise dos tempos de latência e da vazão do sistema. Esse difere ao apresentar a investigação da plataforma através de uma verificação de requisitos.

O objetivo dos autores era determinar quais testes devem ser aplicados a qualquer sistema *blockchain* com a finalidade de avaliar as propriedades do mesmo. Com o foco no desenvolvimento de plataformas *blockchain* e não em aplicações, os autores escolheram a *blockchain Scorex* para validar as propriedades escolhidas.

Dentre os vários elementos da *blockchain*, os autores escolheram quatro para apresentar uma lista de propriedades que devem atender. Os elementos escolhidos foram: a *Memory Pool*, descrita na seção 2.3.2; a cadeia de blocos, o estado mínimo, descrito como o estado da cadeia antes de *fork*, e os nós da rede. Apesar da escolha da plataforma, as propriedades listadas foram descritas de forma genérica, sem funcionalidades específicas da plataforma em questão.

A contribuição desse estudo para este trabalho de mestrado é limitada. As propriedades apresentadas se referem a teste de verificação que devem ocorrer no desenvolvimento de plataformas *blockchain* e não caracterizam questões de pesquisas relacionadas à avaliação de desempenho. A relevância desse artigo é a caracterização apresentada em relação às propriedades dos elementos, de forma a garantir o funcionamento adequado de futuras plataformas a serem desenvolvidas.

Com uma proposta que também não inclui avaliação de desempenho, o artigo (CROMAN *et al.*, 2016) apresenta uma investigação detalhada do funcionamento do *Bitcoin* com o objetivo de explorar questões de escalabilidade. Segundo os autores, o alvo dessa pesquisa é averiguar se há a possibilidade do *Bitcoin*, e de forma mais generalizada a *blockchain*, ser competitivo com as atuais formas de pagamento, como o cartão de crédito.

As conclusões apresentadas mostram que há uma necessidade de reestruturação do *Bitcoin* a fim de torná-lo competitivo com outras plataformas de pagamento. Segundo os autores a alterar o tamanho do bloco, apesar de se apresentar como uma solução, não produz uma solução satisfatória. Para os autores, apesar de contribuir com a performance, a melhora ainda fica aquém das soluções já existentes. Desse modo, os autores buscaram soluções que outras plataformas *blockchain* implementam a fim de apresentar resultados mais concretos.

Desse modo, os autores dividiram o *Bitcoin* em cinco camadas de forma a isolar suas funcionalidades e identificar quais eram os gargalos apresentados. A primeira camada é a de rede na qual foi identificado que o processo sequencial de validação das transações se mostra como gargalo de desempenho. Essa conclusão também foi apresentada pelos trabalhos (THAKKAR; NATHAN; VISHWANATHAN, 2018) e (BALIGA *et al.*, 2018), porém observando a plataforma *Hyperledger Fabric*. Nessa mesma camada, os autores também identificaram que a transmissão em duplicata de cada transação, a primeira vez quando a transação é lançada na rede e a segunda quando o bloco que contém a transação é transmitido na rede, também é um fator que impacta no desempenho.

A segunda camada apresentada é a de consenso. Os autores apontam o algoritmo escolhido de *Proof-of-Work* como gargalo do *Bitcoin*. Esse algoritmo é detalhado na Seção 2.3.3, onde é explicado que a dificuldade de mineração equilibra a concorrência porém gera uma queda no desempenho da plataforma. Nesse cenário, os autores sugerem a adoção do algoritmo de consenso *Proof-of-Stake* o qual propõe uma solução menos custosa, também já descrito na Seção 2.3.3.

A terceira camada apresentada é a de armazenamento na qual os autores evidenciam que a replicação excessiva de dados como um gargalo da performance. Isso se deve ao fato que no *Bitcoin* os dados são replicados por todos os nós da rede. Para essa camada, os autores sugerem uma abordagem que distribua o armazenamento dos dados nos nós.

Os autores apresentam a camada de visão, a qual apontam como gargalo a necessidade de obter todos os dados da cadeia para verificar o estado atual das transações. Neste caso, os autores sugerem a criação de visões que permitam a compreensão do estado atual das transações sem a obrigação de analisar todas as transações até então adicionadas na cadeia.

A camada *side* é a última a ser apresentada e se refere a operações realizadas com o banco de dados exterior a cadeia de blocos. Armazenar as referências de um dado na cadeia de blocos e utilizar um banco de dados externo é uma estratégia utilizada a fim de diminuir o tamanho da cadeia a ser armazenada. Porém, como apontam os autores, há um custo elevado nessas operações.

Apesar de não conter uma metodologia de avaliação de desempenho, o estudo apresenta um diagnóstico detalhado do *Bitcoin*, apontando para uma forma de compreender a plataforma com o uso de camadas definidas de acordo com a sua funcionalidade. A divisão em camadas

também é mostrado em (DINH *et al.*, 2017), com o objetivo de caracterizar elementos e facilitar a avaliação da plataforma. O trabalho também se mostra relevante pela análise de uma rede publica, que contem desafios mais abrangentes que uma rede privada.

Em (SHEN; GUO; YANG, 2019) os autores pretendem avaliar uma aplicação para o armazenamento de dados médicos de maneira distribuída. Para isso, usam a *blockchain* e outras ferramentas de armazenamento em nuvem em uma plataforma que chamam de *MedChain*.

Como resultado do trabalho, os autores destacam a criação do protótipo *MedChain* que inclui um módulo da *blockchain*, um módulo de diretório e um cliente teste. Com a avaliação do protótipo, eles concluíram que a aplicação apresenta maior eficiência do que as soluções anteriores encontradas por eles, considerando o *overhead* de comunicação e armazenamento. Além disso, apresenta maior flexibilidade por separar dados em mutáveis e imutáveis, armazenando os dados de maneira *off-chain*. Por último, concluíram que em questão de escalabilidade o protótipo não é pior que outras soluções.

A performance do *MedChain* foi avaliada com o uso de *benchmarks* que demonstram empiricamente o *overhead* em diferentes processos de compartilhamento de dados. Houve um foco na análise de segurança e na análise da eficiência. Os resultados compararam a performance da aplicação proposta neste trabalho com a de outros projetos do gênero.

A análise deste trabalho permite verificar que houve um foco no desenvolvimento da aplicação. Isso pode ser observado pelo detalhamento sobre o protótipo. A avaliação da aplicação é apresentada com menos detalhes, faltando, inclusive, a descrição dos dados que foram utilizados. Esses dados foram caracterizados apenas como dados médicos de maneira genérica. A estrutura do teste realizado também é limitada, com apenas um cliente ativo realizando requisições.

Outro trabalho no qual foca nos dados médicos é (ROEHRS *et al.*, 2019). Os autores apresentam a implementação e avaliação de um modelo de distribuição de dados médicos usando a *blockchain*. Através da análise dos resultados, eles concluíram que o uso da *blockchain* para a distribuição dos dados tem baixo tempo de resposta ao mesmo tempo que aumenta a disponibilidade dos dados.

Os autores optaram por implementar a própria *blockchain*, não se apoiando em uma plataforma de desenvolvimento de aplicações. Nessa implementação, foi avaliado o tempo de resposta, o espaço de memória ocupado e o uso de CPU. Para a isso foram utilizados 10 *superpeers* simulando o acesso simultâneo de 40000 clientes.

É interessante notar que os autores tentaram reproduzir o ambiente de uma aplicação real. Além de utilizar um número grande de clientes acessando os dados de forma simultânea, também usaram de dados de duas bases de dados hospitalares. Esse cenário de testes é o principal diferencial que esse trabalho apresenta de outros encontrados no levantamento bibliográfico da literatura.

Outro fator que vale destacar deste trabalho é a opção pela implementação própria de

uma *blockchain*. A explicação para essa escolha, porém, não é justificada, baseando somente em uma forma de diferenciar de outros trabalhos. Somado a isso, os autores não detalham essa nova implementação com relação aos componentes da *blockchain*.

Com a proposta de comparar duas plataformas de desenvolvimento foi encontrado o trabalho (OLIVEIRA *et al.*, 2019). Nele as plataformas *Parity* e *Multichain* foram avaliadas. A avaliação dos *frameworks* foi realizada através da inserção de transações geradas de acordo com a probabilidade atual de chegada de novas transações na rede do *Bitcoin*.

As métricas coletadas foram: tempo de validação das transações, tempo de mineração das transações, tempo de busca de uma transação e tempo de busca de um bloco. Os autores concluíram que o *Multichain* apresenta melhor performance para as quatro métricas avaliadas, isso se deve a simplicidade das operações a serem realizadas. O *Parity* apresenta os *smart contracts*, que aumenta o tempo para a realização das transações.

Esse trabalho contém como diferencial o uso de modelo de distribuição de probabilidade de cada métrica investigada. Além disso, usar a frequência de chegada de transações de uma aplicação real e consolidada como o *Bitcoin* aproxima os resultados encontrados ao cenário de uma aplicação real. Apesar de ser uma proposta com diferenças que destaca sua contribuição, a descrição dos testes não é detalhada. Não há informações com relação a quantidade de nós, número de clientes e versões das plataformas, por exemplo.

O trabalho (ISMAIL *et al.*, 2019). Neste artigo os autores apresentam uma taxonomia com diferentes plataformas de desenvolvimento e simuladores da literatura. Além disso, avaliam o *Bitcoin Simulator* em termos de bandwidth e total de execuções.

Com a avaliação, os autores afirmam que a performance da blockchain em termos de *bandwidth* e tempo de execução total cai com o aumento do número de validações. Além disso, concluíram que dependendo do tipo de validação, é necessário todos os nós em uma rede pública. Ademais, verificaram que aumentar o número de blocos não aumenta o *bandwidth* ao mesmo tempo que aumenta o tempo de resposta. Por fim, os autores completam que a performance pode ser melhorada com o aumento do tamanho do bloco enquanto o tempo de resposta permanece constante.

O ambiente de simulação foi projetado para testar o simulador com o uso de nós geograficamente separados. Os autores optaram por utilizar nós separados em prédios distintos dentro da universidade. Os resultados foram coletados em rodadas de 25 experimentos.

Apesar do experimento se limitar à distância de prédios dentro do mesmo complexo, a escolha de usar de nós geograficamente separados é interessante por apontar a necessidade de avaliar como nós em diferentes lugares interagem. Outro ponto positivo é o levantamento do que está sendo utilizado para desenvolver e simular aplicações *blockchain*. Um ponto negativo deste trabalho é a falha na definição das características da plataforma que está sendo analisada, inviabilizando a total compreensão dos resultados.

Em (ZOU; LV; ZHAO, 2021) os autores propõem uma implementação de uma solução *blockchain* chamada *SPChain* com foco na distribuição de histórico médico de pacientes. Para motivar as instituições médicas a participar do *SPChain* os autores construíram um sistema de reputação com o qual as instituições podem trocar informações para usar em pesquisas.

Nesse trabalho, os autores avaliaram seu sistema com base no custo de armazenamento, vazão e latência. A proposta foi comparada com o desempenho de outras já existentes e os autores consideram sua proposta competitiva, afirmando que o *SPChain* atinge um alto rendimento com mínima sobrecarga de armazenamento. Além disso, também realizaram testes de ataques a rede e consideram sua proposta tolerante aos cenários avaliados.

Esse estudo mostra como a *blockchain* pode ser implementada em um contexto específico, mostrando os desafios de adaptação da *blockchain* do setor financeiro para o médico. O uso de uma implementação própria, porém, impossibilita a generalização dos resultados. Dessa forma, a analise realizada não avança sinaficamente o estado da arte da avaliação de desempenho de redes *blockchain* em cenários diferentes do financeiro.

Os autores em (CHEN *et al.*, 2021) analisam a *blockchain* com foco na garantia de privacidade dos dados. O cenário de teste escolhido foi a área da saúde a plataforma *blockchain* utilizada foi o *Hyperledger Fabric*.

A avaliação realizada teve como objetivo verificar questões de segurança de informações em redes *blockchain* tais quais, preservação da privacidade, segurança das buscas, integridade e confiabilidade dos dados e se o sistema mantém a organização dos dados. A escalabilidade da aplicação também foi analisada com o aumento do número de integrantes da rede.

Através da analise da sobrecarga criada pelos processos de criptografia, desempenho dos *smart contracts* e escalabidade os autores concluem que é viável e incentivam o uso da *blockchain* como ferramenta para distribuição de dados sensíveis, como da área médica. O estudo não se trata de uma avaliação de desempenho com o uso de métricas de performance como latência e vazão. Além disso, não se aprofunda em aspectos fundamentais da infraestrutura da *blockchain*, como consenso e criação de blocos.

3.4 Análise dos Trabalhos

A leitura aprofundada desses trabalhos permitiu averiguar as tendências e as lacunas no contexto de testes, verificação ou validação de desempenho de aplicações *blockchain*. Na Seção 3.3, os trabalhos foram descritos e de forma pontual evidenciaram quais as contribuições e pontos fracos. Esta Seção pretende destacar os resultados encontrados, e também apontar quais as lacunas, desafios e perspectivas da área.

A motivação deste trabalho de mestrado se deve à falta de estudos que avaliem a viabilidade de aplicações *blockchain* não relacionadas à transferência monetária. A busca por

trabalhos relacionados tem como objetivo averiguar o que os pesquisadores da área consideravam importante avaliar na *blockchain* nos diversos contextos no qual ela pode ser aplicada, inclusive o financeiro. Esses resultados estão resumidos na Tabela 4.

Avaliação	**Quantidade**
Componente	8
Plataforma	13
Híbrido	3

Tabela 4 – O que é avaliado

Os resultados apontam para um empate entre a quantidade de trabalhos que avaliam componentes da *blockchain* isolados e os que avaliam a resposta do sistema como um todo. Os estudos que avaliam componentes estão concentrados na análise de comportamento dos *smart-contracts*. Já os artigos que trazem um parecer em relação à plataforma avaliam o desempenho em termos de vazão e latência. Foram classificados como *Híbrido* os artigos que apresentavam tanto a análise da plataforma quanto uma avaliação de componentes específicos.

O levantamento bibliográfico também tem como meta a identificação das plataformas utilizadas por esses trabalhos. A Tabela 5 condensa o que foi encontrado. Apesar dos trabalhos apresentarem clientes diferentes das plataformas *Hyperledger* e *Ethereum*, eles não foram destacados pois os clientes foram desenvolvidos com base na mesma implementação.

Plataforma	**Quantidade**
Ethereum	14
Hyperledger	6
Bitcoin	4
Scorex	1
Implementação Própria	3

Tabela 5 – Plataformas utilizadas

A tabela permite constatar que o *Ethereum* tem um maior número de trabalhos que o avaliam. Isso pode ser resultado de dois fatores, o primeiro é o *DAO exploit* o qual baseou uma parcela considerável dos dos trabalhos encontrados. Já o segundo fator é a abertura que a plataforma apresenta para o uso tanto para rede pública quanto privada, expandindo o tipo de avaliação que pode ser realizada.

O levantamento bibliográfico não limitou a busca em uma área específica, por isso era esperado um número maior de trabalhos que avaliassem o *Bitcoin*. Uma possível causa para esse resultado é a restrição do tipo de aplicação de plataforma, uma vez que o tipo de transação que é realizada pelo *Bitcoin* já é consolidada e seu resultado esperado.

Outro objetivo da análise dos trabalhos relacionados é explorar a metodologia utilizada. Os estudos que investigam as plataformas focam em avaliar o desempenho baseado na vazão e latência do sistema, para isso configuram um número limitados de nós e testam utilizando uma aplicação escolhida. Nesse cenário, os tempos de resposta são coletados para operações de escrita e leitura, variando a frequência com as quais as requisições chegam. Um dos elementos que os trabalhos se diferem é o ambiente de execução, o qual pode variar no uso de rede públicas ou privada. A Tabela 6 ilustra os resultados encontrados.

Ambiente de Execução	**Quantidade**
Rede Privada	10
Rede Pública	4

Tabela 6 – Ambiente de execução

Um dos fatores que os artigos se diferenciam em relação à metodologia é o tipo de aplicação utilizada. Apesar da generalidade no uso de aplicações financeiras, há uma variedade de origens dessas aplicações. A Tabela 7 sintetiza como os autores selecionaram as aplicações. *Sintética* se referencia as aplicações desenvolvidas pelos próprios autores, já as *Real* se relaciona a aplicações que estão disponíveis de maneira *Open-Source* e não foram idealizadas pelos autores do estudo.

Aplicação	**Quantidade**
Sintética	6
Real	3

Tabela 7 – Aplicação

Esses resultados evidenciam a falta de variedade na configuração dos cenários da validação que são realizados. O provável motivo para esse quadro é o recente interesse em averiguar questões de aplicabilidade da *blockchain* fora do contexto de *Bitcoin*.

A partir desses resultados é possível inferir quais são as possibilidades de contribuição no contexto de validação da *blockchain*. Um desses exemplos é a falta de estudos que desenvolvessem questões de segurança. Apesar dos vários mecanismos de proteção contra ataques, a *blockchain* ainda é vulnerável. Essa vulnerabilidade está associada a arquitetura de rede utilizada (*Peer-to-Peer*) e aos vários componentes presentes na infraestrutura da *blockchain*. Ainda que não fosse o foco do levantamento bibliográfico realizado, não houve limitações na busca que justifiquem a falta de artigos que avaliassem esses possíveis ataques.

Com a análise realizada também foi possível constatar que os testes são executados em um ambiente de execução limitado. Essa restrição se deve ao uso de poucos nós na rede e à localização geográfica dos mesmos. Com relação ao primeiro, o que pode ser constatado é que a

configuração dos experimentos se limitava ao uso de quatro nós, com poucas exceções utilizando oito nós. Somado a isso, os nós utilizados se encontravam na mesma rede, impossibilitando a verificação de condições diversas de rede.

O levantamento executado também evidencia que não há variação no que concerne ao tipo e tamanho de dado utilizado nas validações propostos nesse estudos. As avaliações apresentadas pelos projetos estão limitadas ao contexto financeiro, deixando em aberto as possibilidades de estudos que avaliem a *blockchain* em contextos diversos. Para isso é necessário variar os dados, tanto em relação ao tipo de dado, como documentos de texto ou imagens quanto no tamanho desses dados.

Outro aspecto que o levantamento bibliográfico não encontrou foi a análise em relação ao impacto do uso da criptografia como solução para garantir privacidade dos dados. Em transações monetárias que utilizam a *blockchain*, como no *Bitcoin*, as transações são realizadas de forma que tanto o remetente quanto o destinatário não sejam conhecidos de maneira pública. Já o valor a ser transmitido é conhecido por todos os participantes da rede. Quando a *blockchain* é aplicado em outras áreas, como a educação ou a saúde, há a necessidade de proteger o dado a ser transmitido com o objetivo de garantir a privacidade. Uma solução é o uso de criptografia para assegurar a privacidade. Adicionar essa camada de abstração tem um custo que não é avaliado pelos trabalhos encontrados.

A investigação bibliográfica permitiu levantar como os resultados dos cálculos de vazão e latência são apresentados pelos trabalhos. Durante a execução desse levantamento, não foram encontrados estudos que divulgassem os resultados utilizando análise estatística completa. Nesses trabalhos, os tempos de vazão e latência são calculados considerando a média dos valores encontrados. Com isso, há perda de informação derivada da generalização dos dados. Nesse sentido, o uso de, no mínimo, medidas como mediana, moda e desvio padrão são apropriadas para destacar o comportamento das execuções.

O Quadro 4 sintetiza o estado atual dos estudos relacionados a teste, verificação ou validação de aplicações e infraestrutura *blockchain*. Essa Tabela resume o que a literatura já considera padrão, como as métricas de avaliação e a maneira como os testes de tolerância a falhas são realizados. Além disso, apresenta quais as lacunas não abordadas pelos trabalhos acerca do uso de criptografia para garantir a privacidade dos dados e aplicações que possibilitem o uso de dados heterogêneos.

A revisão bibliográfica também realça quais as perspectivas e objetivos de futuros estudos sobre a *blockchain*. Apesar das inúmeras possibilidades de aplicação que a *blockchain* apresenta, a investigação conduzida evidenciou a necessidade de tornar essa infraestrutura mais flexível em relação às tecnologias de propósito similar.

Em aplicações financeiras, o uso da *blockchain* ainda não se compara a operações realizadas por operadoras de cartão de crédito, por exemplo. O próprio *Bitcoin* não garante

Quadro 4 – Aspectos analisados dos trabalhos

Aspectos Analisados	**Estado atual**
Principais métricas usadas	Vazão/Latência
Teste de tolerância a falhas	Poucos trabalhos mostram comportamento frente à perda de nós.
Teste de segurança	Poucos trabalhos simulam ataques.
Tipos de dados considerados	Dados de aplicações financeiras. Faltam estudos sobre dados heterogêneos.
Criptografia	Não foram encontrados neste levantamento trabalhos que usem criptografia e que precisem de privacidade aos dados.
Suporte da estatística	Estudos consideram apenas média simples. Não usam outras medidas que deem suporte às análises.
Nicho de aplicação	Restrito às aplicações financeiras.
Veículos de publicação	Maioria em congressos científicos.

transações em tempo real o que inviabiliza sua utilização em pagamentos rotineiros. Esta impraticabilidade de operação de tempo real se deve à necessidade de comunicação entre todos os nós, para que assim as transações sejam conhecidas pela rede. Outro fator que determina o tempo das operações é o cálculo realizado pelos nós mineradores devido ao algoritmo de consenso.

Outra lacuna observada nos trabalhos encontrados está em apresentar alternativas para os gargalos já encontrados pelos casos de testes analisados nessa revisão bibliográfica. Dentre eles estão o custo de execução e tempo de processamento do algoritmo de consenso; comunicação com o banco de dados em cadeias de armazenamento *Off-Chain*, devido ao número de operações e seus respectivos custos de acesso; e problemas de rede, como a demora de comunicação e duplicidade de transmissão das transações na rede.

Considerando a relevância dos tópicos apresentados, este trabalho tem como foco a análise da *blockchain* quando aplicado a dados heterogêneos. Essa escolha se deve pela relevância desse tipo de estudo para a aplicação da *blockchain* em outras áreas. A descrição do trabalho e da metodologia são discutidos de forma aprofundada no próximo capítulo, de definição do projeto.

Ao considerar este contexto específico, foram observados os desafios presentes na realização deste projeto. Como apontado pelo levantamento bibliográfico, há uma uniformidade nos dados utilizados pelos trabalhos, dessa forma, cria-se a necessidade de estruturar um conjunto de dados diversificado que garantidamente testem a *blockchain*. Além disso, esse conjunto de dados deve ser próximo de condições reais, de modo a testar a aplicabilidade da *blockchain* nessas condições.

Devido à falta de justificativa dos autores para a escolha de uso de um número limitado de nós para a execução dos experimentos, um dos desafios deste projeto se encontra em investigar

quais são os fatores que levaram a esse cenário. Em conjunto com esse levantamento, verificar se é possível realizar os experimentos com um maior número de nós de forma simular um contexto próximo de uma aplicação real.

Os desafios citados anteriormente resumem a principal dificuldade da realização deste projeto, que é trazer os resultados encontrados para o cenário de aplicações reais. Além dos experimentos e resultados encontrados, é fundamental analisar e apresentar as conclusões encontradas de forma que impacte no desenvolvimento de aplicações *blockchain* em áreas diversas, como na saúde e na educação.

3.5 Considerações Finais

Neste capítulo foi apresentado o levantamento bibliográfico de estudos relacionados a teste, verificação ou validação de plataformas e/ou componentes da *Blockchain*. A relevância dessa investigação da literatura se deve à descrição do cenário das pesquisas realizadas nesse contexto. Através dela foi possível determinar quais são as tendências de pesquisa da área e quais as contribuições dos futuros trabalhos.

Uma das etapas da revisão da literatura é a definição das questões de pesquisa, que determina quais são os objetivos e metas na busca pelos trabalhos relacionados. Nesse sentido, o levantamento bibliográfico se mostra satisfatório por responder os questionamentos apresentados. Dessa forma, foi possível estabelecer quais são os componentes analisados, as plataforma utilizadas e a metodologia de pesquisa utilizada por esses estudos.

Com a conclusão do levantamento de trabalhos relacionados, foi conduzida uma análise dos estudos com a finalidade de averiguar quais são as qualidades e eventuais lacunas não englobadas por eles. Nessa fase, pode ser observado a preocupação da área em se restringir o escopo do que se deseja investigar. Os objetivos dos trabalhos e contexto nos quais eles estão inseridos são pormenorizados de forma a dar validade às pesquisas realizadas. Porém, houve falhas em relação à caracterização dos cenários de teste, o que afeta a reprodutibilidade de alguns trabalhos. Outra questão observada nos trabalhos encontrados é a pouca análise dos resultados, desfavorecendo as conclusões encontradas.

A partir das análises realizadas foi possível encontrar um contexto específico para o projeto de mestrado no uso da *blockchain* para dados heterogêneos. Essa escolha se deve à falta de estudos que validassem a aplicabilidade da *blockchain* em áreas diferentes da financeira. Esse cenário se mostra um desafio, pois, além da dificuldade de configuração da infraestrutura do ambiente de execução é necessário que o mesmo tenha capacidade de simular condições reais, de modo a resultar em contribuições efetivas no desenvolvimento de aplicações *blockchain* em diferentes contextos.

O próximo Capítulo introduz os experimentos realizados neste projeto de mestrado,

definindo-os e apresentando as aspectos em comum de todos.

CAPÍTULO

4

VISÃO GERAL DOS EXPERIMENTOS

4.1 Considerações Iniciais

Este capítulo tem como objetivo apresentar os experimentos realizados para avaliar o desempenho da *blockchain* para a distribuição de dados heterogêneos. Inicialmente, são especificadas as plataformas de desenvolvimento *blockchain* usadas, posteriormente é descrita uma introdução dos experimentos que foram executados e estão descritos nos próximos capítulos desta dissertação.

São descritas a metodologia utilizada para o desenvolvimento dos experimentos e as métricas utilizadas nos mesmos para medir o desempenho. Por último, é realizada uma descrição dos dados considerados nos experimentos.

4.2 Aspectos Norteadores dos Experimentos

De maneira geral, os experimentos deste trabalho têm como foco avaliar o desempenho e a aplicabilidade da *blockchain* para o armazenamento e distribuição de dados heterogêneos. Como já descrito, a *blockchain* garante a distribuição de informações em um ambiente descentralizado, porém seguro. Mesmo que os participantes da rede não se conheçam ou não confiem uns nos outros, podem trocar informações sem a necessidade de um agente centralizador.

Dadas essas características, existe um interesse em seu uso fora do contexto das criptomoedas, onde foi consolidado. Os experimentos realizados utilizaram dados do contexto da saúde, devido à heterogeneidade das informações armazenadas nesse cenário.

Este estudo utiliza a plataforma de desenvolvimento de aplicações *blockchain Hyperledger Fabric* para abstrair questões de infraestrutura da *blockchain* e focar no tipo de aplicação que se deseja produzir, generalizando, inclusive, as métricas de desempenho avaliadas.

A escolha do *Hyperledger Fabric* se deve ao seu foco em aplicações voltadas para empresas, em que a cadeia é distribuída em uma rede privada (caracterizada na Seção 2.3.2). A versão utilizada do *Hyperledger Fabric* é a 1.4.1, pois dispunha de código estável e documentação no período que os experimentos foram executados.

As métricas de desempenho foram coletadas com o uso do *benchmark Hyperledger Caliper*. A versão utilizada foi a 0.3.2, pois além de estar estável na época da realização desses experimentos, também apresentava compatibilidade com a versão do *Hyperledger Fabric* escolhida.

Quatro experimentos foram realizados neste trabalho:

1. O primeiro experimento traz uma avaliação preliminar do desempenho da *blockchain* com o uso de dados médicos. Foram analisadas as medidas de vazão e latência da rede considerando o armazenamento dos registros no banco de dados *CouchDB* (FOUNDATION, 2020). Os fatores utilizados neste experimento foram o volume de transações e a frequência com que as mesmas foram enviadas à rede. Este primeiro experimento também investigou como o tipo de transação (requisições de escrita e leitura) e como as características dos dados armazenados afetam o desempenho da aplicação, em termos de vazão e latência de rede.

2. O segundo experimento permite comparar o uso da *blockchain* sem e com a associação a um banco de dados na rede *blockchain*. O cenário de avaliação é semelhante ao experimento anterior, no sentido em que as mesmas operações de leitura e escrita foram realizadas e os dados utilizados foram os mesmos. Também foram modificados os mesmos fatores (quantidade de transações e frequência de requisições) porém com valores diferentes, com o objetivo de estressar - no sentido de aumentar a carga - ambos os cenários de uso (com e sem o banco de dados *CouchDB*). Os resultados mostram o impacto da sobrecarga imposta pelo banco de dados, quando este é usado em uma rede *blockchain*.

3. O terceiro experimento tem como objetivo avaliar quais fatores da infraestrutura da *blockchain* afetam o desempenho da rede como um todo. Ao contrário dos experimentos anteriores, os fatores quantidade de transações e frequência de requisições foram mantidos inalterados, enquanto a política de validação, a quantidade de nós validadores, tamanho do bloco e tempo de montagem de um bloco foram modificados. As operações de leitura e escrita foram as mesmas dos experimentos anteriores e consideraram o uso do banco de dados *CouchDB*.

4. O quarto experimento apresenta o impacto de alterar parâmetros da rede no desempenho da aplicação *blockchain*. Dessa forma, os fatores relacionados à rede, número de nós participantes e número de organizações foram alterados, enquanto o tamanho do bloco, política de validação, tempo de montagem, quantidade de nós validadores, taxa de che-

gada de requisições, e quantidade de entradas foram mantidas inalteradas. Para avaliar o desempenho, as mesmas operações de escrita e leitura dos experimentos anteriores foram utilizadas, considerando o uso do banco de dados *CouchDB*. A avaliação realizada revela o peso que a quantidade de organizações de uma rede *blockchain* tem no desempenho de uma aplicação quando analisadas as medidas de latência e vazão.

Visando garantir reprodutibilidade e estabelecer um maior rigor científico para condução e análise dos experimentos descritos neste Capítulo, a metodologia de Wohlin *et al.* (2012) foi utilizada para descrever o planejamento, a execução e a discussão dos resultados. A partir dessa proposta, os experimentos são descritos individualmente nos próximos quatro Capítulos de acordo com escopo, planejamento, execução, descrição dos resultados e análise do experimento em particular. No Capítulo 9 uma discussão final dos resultados é apresentada, englobando os resultados de todos os experimentos.

Com a finalidade de sistematizar a avaliação de desempenho da *blockchain*, padrões de terminologias e métricas para avaliar essa plataforma foram definidas pelo projeto *Hyperledger* em Performance H.; Group (2018). A seção de execução de cada experimento é descrita considerando o ambiente de teste, ponto de observação, característica das transações, carga de trabalho e teste de falhas.

O ambiente de teste de cada experimento é definido levando em conta os parâmetros de execução que são modificáveis dentro da infraestrutura do *Hyperledger Fabric* de acordo com a necessidade de cada aplicação desenvolvida na plataforma. Essas variáveis determinam diferentes aspectos da aplicação: os dois primeiros se referem à rede desenvolvida, o terceiro e o quarto em conjunto determinam a quantidade de nós que validam uma transação e os quatro últimos que formam o *BatchSize* que determinam o tamanho do bloco. A descrição desses elementos é feita a seguir:

- Número de máquinas reais: o tamanho da rede utilizada considerando o número de máquinas físicas utilizadas no experimento;
- Número de contêineres: a quantidade de *Docker* contêineres criadas durante a execução do experimento;
- Nós validadores: determina o número de nós que têm a função de validar uma transação assim que ela é conhecida na rede. O processo de validação ocorre em três etapas, essa variável determina quantos nós participam da primeira fase, antes da transação ser inserida no bloco;
- Política de validação: estabelece qual organização participa da primeira etapa do processo de validação. Essa variável pode ter dois valores *AND* em que todas as organizações presentes na rede participam da validação, ou *OR* onde apenas uma organização participa desse processo, não sendo necessário determinar qual;

- MaxMessageCount: essa variável faz parte do *BatchSize* e determina o número máximo de transações permitidas em um bloco;
- AbsoluteMaxBytes: parte do *BatchSize* e especifica o número máximo absoluto de *bytes* permitido para o serialização das transações de um bloco;
- PreferredMaxBytes: também faz parte do *BatchSize* e estipula o número máximo de *bytes* preferencialmente permitido para a serialização de transações em um bloco. Caso uma transação seja maior que esse valor, bloco criado também será;
- BatchTimeout: o montante de tempo em segundos que o sistema espera para criar um bloco. A criação do bloco ocorre quando ou o tamanho do bloco foi atingido ou se o tempo de espera foi esgotado.

Com relação ao protocolo de consenso, a versão 1.4.1 do *Hyperledger Fabric* usada neste estudo não apresenta um algoritmo como o *Bitcoin* com o *Proof-of-Work*. A infraestrutura do *Fabric* acrescenta o conceito de *ordering service* em que um nó ou vários nós da rede chamados de *orderer* são responsáveis por ordenar as transações e criar o bloco. Nos experimentos realizados neste trabalho foi utilizada a implementação *Solo* do *ordering service*, descrita na Seção 2.5.2. A escolha dessa implementação deve-se a seu baixo custo de manutenção, uma vez que apenas um nó *orderer* precisa ser mantido.

Todos os nós estão na mesma rede e pertencem ao *cluster* do *ICMC-Lasdpc*. O *smart-contract* foi desenvolvido em *JavaScript* e está disponível no Apêndice A. Os demais códigos usados neste trabalho estão disponíveis em: <https://github.com/anacspengler/mestrado>.

Como descrito na Seção 2.5.3, o *Caliper* permite avaliar o desempenho de diferentes plataformas *blockchain*. Ele considera dois tipos de transações para o cálculo das métricas de desempenho, *read* e *transaction*, as quais se diferenciam pela primeira não alterar a cadeia e a segunda ser uma atualização da mesma (PERFORMANCE H.; GROUP, 2018). Dentro do *Fabric*, a documentação chama essas diferentes transações de *read* e *update* (como descrita na Seção 2.5.2), porém a semântica é a mesma. Para fim de padronização, a nomenclatura empregada será a do *Fabric*.

Para as transações *read* a latência (Equação 4.1) é calculada considerando o tempo em que a resposta é recebida menos o tempo que ela foi submetida.

$$\text{Latência Read} = \text{Tempo na resposta} - \text{Tempo na submissão} \tag{4.1}$$

A vazão dessas transações é calculada como o total de transações executadas pelo tempo total em segundos, ilustrada na Equação 4.2.

$$\text{Vazão Read} = \frac{\text{Total de transações } \textit{read}}{\text{Tempo total em segundos}} \tag{4.2}$$

A latência das transações *update* é o tempo desde que as transações são submetidas até o resultado ser conhecido pela rede, o que engloba os tempos de validação e propagação da transação de acordo com a política de consenso. A Equação 4.3 demonstra a fórmula utilizada pelo *Caliper*.

$$\text{Latência Update} = (\text{Tempo resposta incluindo val. e prop. na rede}) - \text{Tempo submissão} \quad (4.3)$$

A vazão das transações *update* (Equação 4.4) é a taxa em que as transações válidas são submetidas à *blockchain* pelo tempo total em segundos da validação da transação por toda a rede (PERFORMANCE H.; GROUP, 2018).

$$\text{Vazão Update} = \frac{\text{Total de transações válidas}}{\text{Tempo total em segundos usados na inserção}} \quad (4.4)$$

Os dados usados neste trabalho foram tabulados pela base de dados disponibilizada pelo *MIMIC-III* (JOHNSON *et al.*, 2016). As informações contidas nessa base de dados foram coletadas de pacientes clínicos que foram admitidos no *Centro Médico Beth Israel Deaconess Medical Center* em *Boston*, *Massachusetts*. Os registros foram coletados de diversas fontes durante a rotina do hospital, incluindo arquivos de unidades de terapia intensiva (UTI), dados retirados de registro eletrônico do hospital e o registro de óbitos da previdência social.

Os registros armazenados foram manipulados por Johnson *et al.* (2016) de forma a impedir a identificação das pessoas cujas informações foram coletadas. Segundo os autores, a manipulação ocorreu de acordo com o *HIPAA (Health Insurance Portability and Accountability Act)*. Para garantir o anonimato dos dados coletados os autores removeram os elementos de identificação listados pelo *HIPAA*, como o nome do paciente, número de telefone, endereço e datas de entrada, saída e realização de procedimentos. O *HIPPA* também determina a remoção de informações relacionadas a parentes e membros da família que possam auxiliar na identificação do indivíduo.

Os dados usados neste projeto diferem dos dados usuais encontrados em pesquisas que avaliam a *blockchain* pois estas últimas focam na análise de cenários de aplicações financeiras. Nesses contextos, os dados utilizados são em grande parte formados por conjuntos chave-valor nos quais há um único valor. Os dados considerados neste estudo consideram conjunto chave-valor formados por múltiplos valores (quantidade e tipo de dados são descritos descritos a seguir) e também consideram o uso de índices de busca [1].

Além disso, os dados considerados nos experimentos podem ser definidos como heterogêneos, de acordo com Wang (2017), o qual classifica dados heterogêneos como sendo quaisquer dados com alta variabilidade de tipos e formatos. Singh, Gao e Jain (2012) também reforçam esta

[1] Índice é uma estrutura associada a um arquivo para facilitar a busca por informações, criando ponteiros para campos específicos

definição de heterogeneidade dos dados utilizados, pois segundo a classificação destes autores, os dados manipulados neste projeto são provenientes de fontes diferentes e incluem vários tipos armazenados em arquivos distintos.

A Base de dados possui vinte e seis arquivos, com descrito nas Tabelas 8 a 11, divididos em quatro categorias de acordo com as informações armazenadas por elas. Seis arquivos são usadas para definir a estadia dos pacientes no hospital (Tabela 8).

Nome dos arquivos	**Descrição**
Admissions	Entrada única para cada hospitalização do paciente
Callout	Saída do paciente da UTI e/ou saída do hospital
Icustay	Entrada única para cada admissão na UTI
Patient	Cadastro dos pacientes aceito no hospital
Services	Serviço clínico prestado ao paciente
Transfers	Registra a movimentação do paciente no hospital

Tabela 8 – Arquivos *MIMIC-III* relacionadas a pacientes

Os próximos oito arquivos apresentadas na Tabela 9 englobam os dados coletados da unidade de terapia intensiva (UTI).

Nome dos arquivos	**Descrição**
Caregivers	Cadastro dos cuidadores
Chartevents	Observações com relação ao paciente
Datetimeevents	Observações relacionadas a datas
Inputevents_cv	Ingestão de fluidos por pacientes na UTI
Inputevents_mv	Ingestão de fluidos por pacientes na UTI
Noteevents	Notas relacionadas ao paciente
Outputevents	Informações da saída de pacientes da UTI
Procedureevents_mv	Registro de procedimentos

Tabela 9 – Arquivos *MIMIC-III* relacionadas a UTI

Dois diferentes tipos de sistemas foram usados para coletar dados da UTI. O sistema *Philips CareVue Clinical Information System* e o sistema *iMDSoft Metavision ICU*. Segundo Johnson *et al.* (2016), quando possível os dados dessas duas fontes foram mesclados, porém, na impossibilidade, os arquivos foram separadas e identificadas com o sufixo que marca qual é a fonte de origem dos dados. Os dados sobre a ingestão de fluídos foram divididos em dois arquivos. As entradas coletadas pelo sistema *CareVue* foram armazenadas no arquivo *INPUTEVENTS_CV*, já os dados coletados pelo sistema *iMDSoft Metavision* foram armazenados na *INPUTEVENTS_MV*.

Sete arquivos contêm dados coletados pelo sistema de registro hospitalar (Tabela 10).

Nome dos arquivos	Descrição
Cptevents	Registro de procedimentos armazenados como CPT [2]
Diagnoses_icd	Diagnósticos atribuídos ao hospital, codificados pelo sistema ICD [3]
Drgcodes	Informações sobre DRG [4]
Microbiologyevents	Medições de microbiologia do banco de dados do hospital
Prescriptions	Registro de pedidos de medicamentos
Procedures_icd	Registro de procedimentos de pacientes usando o sistema ICD

Tabela 10 – Arquivos *MIMIC-III* relacionadas ao hospital

Outros cinco arquivos (vide Tabela 11) são dicionários que mostram referências a códigos que definem identificadores presentes em demais arquivos da base.

Nome dos arquivos	Descrição
D_cpt	Dicionário do CPT
D_icd_diagnoses	Dicionário do ICD para códigos de diagnósticos
D_icd_procedures	Dicionário do ICD para códigos de procedimentos
D_items	Dicionário de ITEMID com exceção a teste de laboratório
D_labitems	Dicionário de ITEMID de teste de laboratório

Tabela 11 – Arquivos *MIMIC-III* dicionário

Este estudo considerou apenas quatro destes vinte e seis arquivos disponíveis no *MIMIC-III* para as transações de escrita e leitura. Para a escolha dos arquivos, houve a preocupação de selecionar pelo menos uma de cada categoria definida em Johnson *et al.* (2016). A quantidade de campos e a necessidade de indexação foram as principais características consideradas de forma que as quatro utilizadas foram selecionadas por já permitirem avaliar qual o impacto da quantidade de dados e das indexações no desempenho das inserções e buscas.

As Tabelas numeradas de 12 a 15 descrevem as informações armazenadas nos arquivos do *MIMIC-III* usadas neste estudo, mostrando o nome da variável armazenada e seu tipo.

Nome	Tipo do dado	Descrição
row_id	int4	Identificador de linha único
subject_id	int4	Identificador único do paciente
gender	varchar	Gênero
dob	timestamp	Data de nascimento
dod	timestamp	Data de óbito
dod_hosp	timestamp	Data do óbito registrada no hospital
dod_ssn	timestamp	Data do óbito registrada no SSN
expire_flag	int4	Indica se o paciente faleceu

Tabela 12 – Dados do arquivo *PATIENTS*

Como mencionado, o arquivo *PATIENTS* inclui informações sobre os pacientes atendidos no hospital (Tabela 12). Ele possui oito campos distintos para cada entrada no arquivo e não tem necessidade do uso de índice seus dados na inserção; esta é a única Tabela do *MIMIC-III* com essa característica. O tamanho de cada entrada do arquivo pode chegar a 50 *bytes*. Essa oscilação nos tamanhos das entradas está presente em todos os arquivos utilizados neste estudo por haver colunas que podem ser vazias e também em função da variação de tamanho do tipo de dado *varchar* em cada campo. Ao todo ele, contém 46.520 entradas distintas e também foi utilizado para as transações de leitura (busca).

Nome	**Tipo do dado**	**Descrição**
row_id	int4	Identificador de linha único
itemid	int4	Identificador do item
label	varchar	Nome do item
abbreviation	varchar	Abreviatura do item
dbsource	varchar	Banco de dados de origem do item
linksto	varchar	Tabela dos dados para o ITEMID
category	varchar	Categoria de dados do item
unitname	varchar	Unidade associada ao item
param_type	varchar	Tipo do item
conceptid	int4	Todos os valores NULL

Tabela 13 – Dados do arquivo *D_ITEMS*

A Tabela *D_ITEMS* ilustrada na Tabela 13 é um dicionário de *itemid*, o qual está presente em outras arquivos do *MIMIC-III*, com exceção aos dados relacionados a testes de laboratório. Esse arquivo contém dez campos diferentes e cria-se índices para dois campos específicos para as inserções, sendo selecionado por ter o maior número de campos dentre as arquivos dicionário do *MIMIC-III*. Cada entrada pode chegar a 608 *bytes*. A quantidade de entradas deste arquivo é de 12.487.

Nome	Tipo do dado	Descrição
row_id	int4	Identificador de linha único
subject_id	int4	Identificador único do paciente
hadm_id	int4	Identifica a permanência no hospital
icustay_id	int4	Identifica a permanência na *UTI*
startdate	timestamp	Data do fim da prescrição
enddate	timestamp	Data de início da prescrição
drug_type	varchar	Tipo de medicamento
drug	varchar	Nome do medicamento
drug_name_poe	varchar	Nome do medicamento no *POE*
drug_name_generic	varchar	Nome do genérico do medicamento
formulary_drug_cd	varchar	Código da fórmula do medicamento
gsn	varchar	Número sequencial do genérico
ndc	varchar	Código nacional do medicamento
prod_strength	varchar	Força do medicamento
dose_val_rx	varchar	Dose do medicamento prescrita
dose_unit_rx	varchar	Unidade de medida da dose
form_val_disp	varchar	Quantidade da formulação dispensada
form_unit_disp	varchar	Unidade de medida da fórmula
route	varchar	Forma de administração

Tabela 14 – Dados do arquivo *PRESCRIPTIONS*

O arquivo *PRESCRIPTIONS* descrita na Tabela 14 contém o registro de pedidos de medicamentos realizados no hospital, independe se o medicamento foi aplicado ou não no paciente. As entradas para esse arquivo pode alcançar 1592 *bytes*. O arquivo possui 4.156.450 linhas, utiliza dezenove campos para cada entrada e requer a criação de índices para cinco destes campos do arquivo. Este é o maior número de indexações utilizado pelos arquivos do *MIMIC-III*.

Nome	Tipo do dado	Descrição
row_id	int4	Identificador de linha único
subject_id	int4	Identificador único do paciente
hadm_id	int4	Identifica a permanência no hospital
icustay_id	int4	Identifica a permanência na *UTI*
starttime	timestamp	Horário em que o evento começou
endtime	timestamp	Horário em que o evento terminou
itemid	int4	Identifica o item
amount	float8	Quantidade do item administrada
amountuom	varchar	Unidade de medida
rate	float8	Ritmo de administração do item
rateuom	varchar	Unidade de medida da frequência
storetime	timestamp	Horário em que o evento foi registrado
cgid	int4	Identifica o cuidador
orderid	int4	Identificador de itens em uma solução
linkorderid	int4	Identificador de pedidos
ordercategoryname	varchar	Grupo que o item corresponde
secondaryordercategoryname	varchar	Grupo secundário para o item
ordercomponenttypedescription	varchar	Função do item administrado
ordercategorydescription	varchar	Tipo de item administrado
patientweight	float8	Peso do paciente
totalamount	float8	Valor total na solução do item fornecido
totalamountuom	varchar	Unidade de medida do total do item
isopenbag	int2	Indica se a embalagem está aberta
continueinnextdept	int2	Indica se o item é transferido
cancelreason	int2	Motivo do cancelamento
statusdescription	varchar	Status do pedido
comments_editedby	varchar	Título do cuidador que fez a solicitação
comments_canceledby	varchar	Título do cuidador que cancelou
comments_date	timestamp	Data da edição ou cancelamento
originalamount	float8	Quantidade do item original
originalrate	float8	Ritmo de administração original

Tabela 15 – Dados do arquivo *INPUTEVENTS_MV*

O arquivo *INPUTEVENTS_MV* mostrada na Tabela 15 armazena a ingestão de fluídos por pacientes na UTI monitorados pelo sistema *iMDSoft Metavision*. As colunas para esse arquivo podem somar até 826 *bytes*. Apresenta o maior número de campos por entrada, sendo essa a motivação do uso neste estudo, com um total de 31 campos e requer a criação índices para quatro campos na inserção dos dados. As entradas somam 3.618.991 itens.

4.3 Considerações Finais

Neste Capítulo foram mostradas as características em comum dos experimentos realizados neste trabalho. Em um primeiro momento, a plataforma *blockchain* utilizada foi apresentada

assim como a ferramenta utilizada para coletar as métricas de desempenho consideradas. Os experimentos realizados foram brevemente introduzidos, exibindo objetivo, motivação e qual foi o principal resultado encontrado. Em seguida, a metodologia de cada experimento foi definida da mesma maneira que os parâmetros de execução considerados no estudo proposto. Os dados da base de dados *MIMIC-III* usadas neste estudo foram descritas por último.

O próximo Capítulo disserta sobre o primeiro experimento cujo foco é analisar o desempenho da *blockchain* quando esta é usado para o armazenamento de dados heterogêneos em diferentes cenários de requisição de operações de leitura e escrita.

CAPÍTULO

5

EXPERIMENTO 1 - QUANTIDADE DE TRANSAÇÕES E FREQUÊNCIA DE REQUISIÇÕES

5.1 Considerações Iniciais

Este Capítulo apresenta o experimento para investigar o desempenho do *blockchain Hyperledger Fabric* em diversos cenários de requisição de acesso aos dados. Espera-se realizar uma análise inicial para, em conjunto com os demais experimentos, verificar alguns fatores que impactam no desempenho de uma aplicação *blockchain* com o uso de dados heterogêneos.

A descrição do experimento segue a metodologia de (WOHLIN *et al.*, 2012) e dessa forma, o Capítulo se inicia com a descrição do escopo do experimento, definindo características como objetivo e objeto de estudo. O planejamento apresenta as hipóteses e questões de pesquisa, assim como os fatores e níveis considerados. A Seção de infraestrutura para execução explica a metodologia usada para a realização do experimento, além de ilustrar a rede e as características das máquinas utilizadas. Os resultados encontrados são apresentados em forma de tabelas mostrando a medidas de desempenho encontradas nas execuções e gráficos destacando cada operação explorada. A análise dos resultados deste primeiro experimento explora as suas causas e o impacto dos mesmos no desenvolvimento de aplicações.

5.2 Escopo

O objetivo deste experimento é avaliar quantitativamente as métricas latência e vazão das requisições de inserção e leitura quando a *blockchain* é utilizado com dados heterogêneos armazenados em banco de dados. A plataforma *Hyperledger Fabric* e o banco de dados *CouchDB* foram escolhidos para serem os objetos de avaliação neste experimento. O foco qualitativo é es-

tabelecer o desempenho da *blockchain* neste contexto, quando é usado a *blockchain* em conjunto com um banco de dados. O experimento tem como base a aplicação de diferentes quantidades de requisições de inserção e busca, com diferentes taxas de chegada dessas requisições e em arquivos de dados com diferentes características.

5.3 Planejamento

Este primeiro experimento foi realizado na rede do laboratório de pesquisa (*LaSDPC-ICMC*) usando dados médicos heterogêneos reais, encontrados na base de dados do *MIMIC-III* (JOHNSON *et al.*, 2016). Os formatos das informações utilizadas o *MIMIC-III* estão descritos no Capítulo 4.2.

Este primeiro experimento visa comprovar três hipóteses relacionadas ao desempenho da *blockchain* associado a um banco de dados quando usado para acessos a dados heterogêneos.

1. A primeira hipótese é que a inserção na *blockchain* é mais custosa computacionalmente quando comparada às operações de busca, e isso afeta o desempenho na inclusão de grandes volumes de dados.

2. A segunda hipótese é que a taxa de chegada das requisições e o total de requisições impactam o desempenho das transações na *blockchain*.

3. A terceira hipótese é que a complexidade das tabelas nos bancos de dados, em termos de número de colunas e indexação, influencia no tempo de resposta das transações com a *blockchain*.

A primeira hipótese foi definida com base na maneira como *Hyperledger Fabric* trata os dois tipos requisições: a inserção e a busca de dados. As diferenças arquiteturais dessas operações foram descritas na Seção 2.5.2. A distinção entre as operações se fundamenta na necessidade de validação pela rede *blockchain* antes das inserções de dados, o que não ocorre na busca.

Já a segunda hipótese tem como base as várias etapas, de consenso (para as inserções), criação do bloco e distribuição na rede, de uma transação na *blockchain* antes dessa ser atendida e inserida na rede. Com o aumento do fluxo de dados, seja com o um maior número de requisições ou taxa de acessos à rede espera-se que o desempenho da rede decline com o aumento da vazão e queda na latência, em virtude da sobreposição dessas etapas.

A última hipótese considera que o tipo de informações armazenadas, principalmente a interação com o banco de dados *CouchDB*, são fatores de impacto no desempenho de redes *blockchain*.

Com a finalidade de validar as hipóteses levantadas, foram especificadas as seguintes questões de pesquisa:

QP1: Qual é a diferença da latência e da vazão entre as requisições de inserção e de leitura a dados heterogêneos com a *blockchain*?

QP2: Qual é o impacto nas latências e vazões das requisições de inserção e leitura quando aplicadas diferentes taxas de requisições sobre diferentes quantidades totais de transações por rodada de execução?

QP3: Qual o impacto nas latências e vazões das requisições de inserção quando são usados arquivos com diversas características distintas, com foco no número de informações registradas e necessidade de indexação?

As métricas de avaliação para esse experimento são latência e vazão, coletadas através do *benchmark Caliper* e calculadas como descrito nas Equações 4.1 a 4.4.

Os fatores considerados nesse experimento são as quantidades de transações e a taxa de envio/chegada das transações à *blockchain*, descritas na Tabela 16.

A quantidade de transações, em rajada, é representativa para um hospital do porte do Hospital das Clínicas (HC) da Faculdade de Medicina (FM) da USP, segundo dados de 2014 (IMPRENSA, 2014). Já a faixa de valores para a frequência (taxa) de envio das requisições foi baseada no volume investigado para a *blockchain* no contexto do *Bitcoin* que atinge vazão de sete transações por segundo (CROMAN *et al.*, 2016).

Fatores	**Níveis**				
Quantidade de transações	1.000	2.500	5.000	7.500	10.000
Taxa de envio (Requisições por segundo)	3	5	7	10	

Tabela 16 – Fatores e níveis Experimento 1

A quantidade de transações realizadas neste estudo é fruto do produto entre os cinco níveis de transações (1.000, 2.500, 5.000, 7.500 e 10.000 transações), os quatro níveis de taxa de envio das requisições (3, 5, 7, e 10) e, finalmente, as cinco operações: quatro inserções (dos diferentes arquivos do *MIMIC-III*) e uma leitura do arquivo *Patient*. Ao todo foram feitas 520.000 transações sobre a *blockchain* neste experimento (26.000 transações, quatro diferentes taxas de serviço e cinco operações).

5.4 Infraestrutura para Execução

Os parâmetros da execução estão expostos na Tabela 17, sendo os dois primeiros parâmetros relacionados à rede, determinando número de máquinas utilizadas e a quantidade de nós *Docker* criados para a execução. O terceiro e o quarto definem a quantidade de nós que vão validar uma transação na primeira etapa do processo de validação (neste experimento, os quatro

nós das organizações têm esta função). Os quatro últimos caracterizam o tamanho e o tempo de criação de cada bloco.

Parâmetro	Valor
Número de máquinas reais	4
Número de contêineres	9
Nós validadores	4
Política de validação	AND
MaxMessageCount	10
AbsoluteMaxBytes	128 MB
PreferredMaxBytes	128 MB
BatchTimeout	1s

Tabela 17 – Parâmetros da execução Experimento 1

A rede é composta por treze nós (todos no mesmo *channel*), em que quatro atuam como clientes, gerando a carga de trabalho (workload) através do *Hyperledger Caliper*, outros cinco atuam como nós na rede *blockchain* com o *Hyperledger Fabric* e quatro hospedam o *CouchDB*. Os cinco nós que dão suporte à rede *blockchain* estão divididos em um nó *orderer*, responsável por ordenar as transações, e quatro nós divididos em duas organizações, simulando organizações do mundo real que desejam compartilhar informações (há dois nós para cada organização). A Figura 15 retrata a configuração da rede, considerando os nós clientes, as organizações e o nó *orderer*.

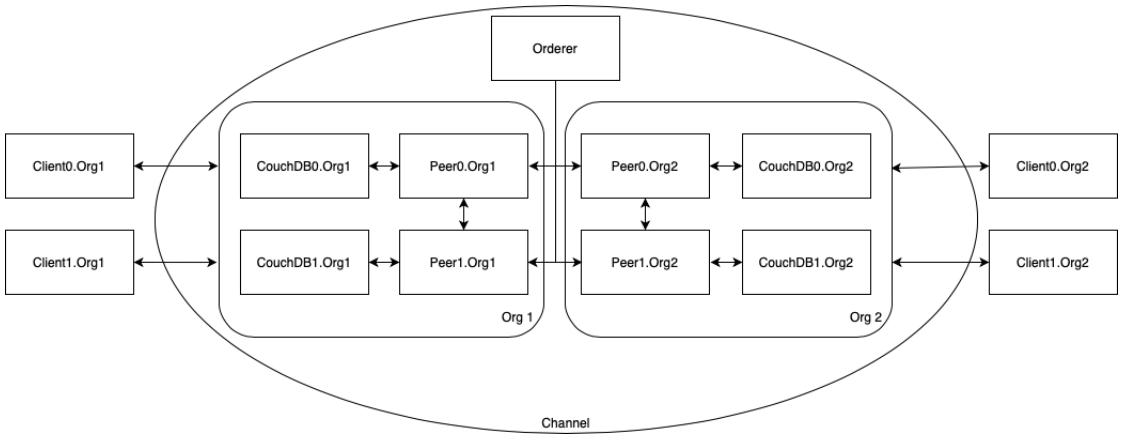


Figura 15 – Configuração da rede Experimento 1

Os sistemas de *software* instalados para a execução deste experimento foram somente os pré-requisitos de instalação encontrados na documentação do *Hyperledger Fabric* e *Hyperledger Caliper*. Os dados heterogêneos reais foram armazenados no banco de dados não-relacional *CouchDB*, ao qual o *Fabric* dá suporte. Foram criados quatro nós *CouchDB* e quatro nós *Fabric*. Cada nó *CouchDB* foi associado a um nó *Fabric* pertencente à sua organização. Por

meio da virtualização com o *Docker*, o nó *Fabric* e o *CouchDB* estão na mesma máquina. As características das máquinas utilizadas estão detalhadas na Tabela 18.

Configurações de *Hardware* (*Cluster Andromeda - LaSDPC*)	
Sistema Operacional	Ubuntu 16.04.7 LTS
CPU	QEMU Virtual CPU (512 KB Cache, 4 GHz)
Memória	16 GB RAM
Disco	Seagate HD 2TB ST2000DM001-1ER1

Tabela 18 – Características das máquinas Experimento 1

O ponto de observação utilizado nesse experimento foi configurado através do *Hyperledger Caliper*. Com o *Caliper* é possível configurar um monitor para coletar as informações sobre as transações e gerar os relatórios de desempenho. Além do relatório, também foi coletado o tempo de execução individual de cada transação com o objetivo de explorar medidas além da média.

A carga de trabalho e a coleta das métricas de avaliação foram geradas pelo *Caliper*. As operações de busca e inserção de dados foram realizadas para avaliar as diferenças na latência e na vazão dos dois tipos de transações executadas pelo *Fabric*, a *read* e *update*, respectivamente. Para cada inserção, os dados são lidos dos arquivos do *MIMIC-III*, distribuídos pela rede e armazenados no *CouchDB*. Durante o processo de distribuição das transações, elas também são validadas pelos nós de maneira individual antes de serem armazenadas. Na busca não ocorre o processo de validação, pois o nó mais próximo a quem fez a requisição é que responde com a informação solicitada. Nesse experimento foram realizadas cinco operações, sendo quatro inserções e uma busca. A escolha dessas operações foi baseada nos arquivos encontrados no *MIMIC-III*, descritos na Seção 4.2 deste Capítulo.

As operações de inserção de dados são as primeiras a serem executadas, sendo que a de leitura dependente da inserção do arquivo *PATIENTS*. As diferentes operações são executadas em blocos, onde uma tarefa só é inicializada após a finalização da anterior. A quantidade total de requisições assim como a quantidade de requisições a serem feitas por segundo são parâmetros determinados na execução, porém a taxa real de requisições executadas por segundo (vazão) será limitada pela capacidade da infraestrutura.

A execução desse experimento não inclui teste de falhas de nós, ataques de *man in middle* e transações maliciosas. Essa última é caracterizada como uma transação que apresenta um comportamento não esperado pela infraestruturada da *blockchain*.

5.5 Resultados

Esta Seção apresenta os principais resultados obtidos com as execuções descritas. As Tabelas de 19 a 38 mostram a latência média, variância e desvio padrão da mesma e a vazão das operações realizadas em cada cenário de execução, combinando a quantidade de transações (1.000, 2.500, 5.000, 7.500 e 10.000) que foram utilizadas e taxa de chegada das requisições (10, 7, 5 e 3) em cada execução.

As Tabelas 19 a 22 ilustram as métricas de desempenho coletadas nas execuções considerando 1.000 transações e, respectivamente, 3, 5, 7 e 10 requisições por segundo.

Operação	Latência Média (s)	Variância	Desvio Padrão	Vazão (TPS)
Inserção *PATIENTS*	1,33	0,22	0,47	3
Inserção *D_ITEMS*	1,42	0,19	0,44	3
Inserção *PRESCRIPTIONS*	1,48	0,2	0,45	3
Inserção *INPUTEVENTS_MV*	1,54	0,22	0,47	3
Busca *PATIENTS*	0,03	9,93E-05	0,00996	3

Tabela 19 – Execução com 1.000 transações e taxa de chegada de 3 req/s

Operação	Latência Média (s)	Variância	Desvio Padrão	Vazão (TPS)
Inserção *PATIENTS*	1,35	0,28	0,53	5
Inserção *D_ITEMS*	1,36	0,17	0,41	5
Inserção *PRESCRIPTIONS*	6,16	18,45	4,3	4,7
Inserção *INPUTEVENTS_MV*	2,08	0,73	0,85	4,9
Busca *PATIENTS*	0,03	2,73E-05	0,00522	5

Tabela 20 – Execução com 1.000 transações e taxa de chegada de 5 req/s

Operação	Latência Média (s)	Variância	Desvio Padrão	Vazão (TPS)
Inserção *PATIENTS*	1,59	0,75	0,87	6,9
Inserção *D_ITEMS*	1,94	1,76	1,32	6,9
Inserção *PRESCRIPTIONS*	11,47	45,61	6,75	6,1
Inserção *INPUTEVENTS_MV*	17,75	107,39	10,36	5,7
Busca *PATIENTS*	0,03	2,56E-05	0,00506	7

Tabela 21 – Execução com 1.000 transações e taxa de chegada de 7 req/s

Operação	Latência Média (s)	Variância	Desvio Padrão	Vazão (TPS)
Inserção *PATIENTS*	6,41	4,51	2,12	9,5
Inserção *D_ITEMS*	16,41	105,13	10,25	7,5
Inserção *PRESCRIPTIONS*	30,39	272,47	16,51	6,4
Inserção *INPUTEVENTS_MV*	39,26	510,13	22,59	5,7
Busca *PATIENTS*	0,03	2,86E-05	0,00535	10

Tabela 22 – Execução com 1.000 transações e taxa de chegada de 10 req/s

As Tabelas 23 a 26 evidenciam a latência média, com a variância e desvio padrão, e a vazão para os testes realizados com 2.500 entradas e com a taxa de chegada entre 3 e 10.

Operação	Latência Média (s)	Variância	Desvio Padrão	Vazão (TPS)
Inserção *PATIENTS*	1,36	0,23	0,48	3
Inserção *D_ITEMS*	1,52	0,25	0,5	3
Inserção *PRESCRIPTIONS*	1,63	0,27	0,52	3
Inserção *INPUTEVENTS_MV*	1,63	0,25	0,5	3
Busca *PATIENTS*	0,03	6,52E-05	0,008076	3

Tabela 23 – Execução com 2.500 transações e taxa de chegada de 3 req/s

Operação	Latência Média (s)	Variância	Desvio Padrão	Vazão (TPS)
Inserção *PATIENTS*	1,33	0,21	0,46	5
Inserção *D_ITEMS*	1,5	0,19	0,44	5
Inserção *PRESCRIPTIONS*	15,7	110,11	10,49	4,7
Inserção *INPUTEVENTS_MV*	33,7	323,02	17,97	4,4
Busca *PATIENTS*	0,03	2,49E-05	0,00499	5

Tabela 24 – Execução com 2.500 transações e taxa de chegada de 5 req/s

Operação	Latência Média (s)	Variância	Desvio Padrão	Vazão (TPS)
Inserção *PATIENTS*	1,4	0,33	0,58	7
Inserção *D_ITEMS*	12,74	40,25	6,34	6,6
Inserção *PRESCRIPTIONS*	40,81	726	26,94	5,6
Inserção *INPUTEVENTS_MV*	58,59	1263,26	35,54	5,3
Busca *PATIENTS*	0,03	0,000034	0,00583	7

Tabela 25 – Execução com 2.500 transações e taxa de chegada de 7 req/s

Operação	Latência Média (s)	Variância	Desvio Padrão	Vazão (TPS)
Inserção *PATIENTS*	13,53	24,35	4,93	9,2
Inserção *D_ITEMS*	63,27	1280,21	35,78	6,7
Inserção *PRESCRIPTIONS*	98,81	3218,4	56,73	5,6
Inserção *INPUTEVENTS_MV*	110,58	4038,26	63,55	5,4
Busca *PATIENTS*	0,03	0,00112	0,0335	10

Tabela 26 – Execução com 2.500 transações e taxa de chegada de 10 req/s

Já as Tabelas de 27 a 30 apresentam os valores alcançados com as execuções dado 5.000 requisições e as taxas de chegada variando entre 3, 5, 7 e 10.

Operação	Latência Média (s)	Variância	Desvio Padrão	Vazão (TPS)
Inserção *PATIENTS*	1,39	0,25	0,5	3
Inserção *D_ITEMS*	1,61	0,27	0,52	3
Inserção *PRESCRIPTIONS*	1,66	0,28	0,53	3
Inserção *INPUTEVENTS_MV*	1,69	0,28	0,53	3
Busca *PATIENTS*	0,03	2,17E-05	0,00466	3

Tabela 27 – Execução com 5.000 transações e taxa de chegada de 3 req/s

Operação	Latência Média (s)	Variância	Desvio Padrão	Vazão (TPS)
Inserção *PATIENTS*	1,36	0,21	0,45	5
Inserção *D_ITEMS*	5,15	13,22	3,64	5
Inserção *PRESCRIPTIONS*	54,03	1105,2	33,24	4,5
Inserção *INPUTEVENTS_MV*	78,99	2025,45	45	4,4
Busca *PATIENTS*	0,03	3,25E-05	0,0057	5

Tabela 28 – Execução com 5.000 transações e taxa de chegada de 5 req/s

Operação	Latência Média (s)	Variância	Desvio Padrão	Vazão (TPS)
Inserção *PATIENTS*	1,42	0,24	0,49	7
Inserção *D_ITEMS*	8,6	34,1	5,84	6,8
Inserção *PRESCRIPTIONS*	101,25	1849,61	43,01	5,4
Inserção *INPUTEVENTS_MV*	88,53	3895,1	62,41	5,5
Busca *PATIENTS*	0,03	2,44E-05	0,00494	7

Tabela 29 – Execução com 5.000 transações e taxa de chegada de 7 req/s

Operação	Latência Média (s)	Variância	Desvio Padrão	Vazão (TPS)
Inserção *PATIENTS*	28,9	360,04	18,97	8,9
Inserção *D_ITEMS*	155,32	7558,31	86,94	6,3
Inserção *PRESCRIPTIONS*	231,23	17728,29	133,15	5,2
Inserção *INPUTEVENTS_MV*	256,38	21475,84	146,55	5
Busca *PATIENTS*	0,03	3,32E-05	0,00576	10

Tabela 30 – Execução com 5.000 transações e taxa de chegada de 10 req/s

Por último, as Tabelas 31 a 34 mostram as métricas de desempenho coletadas nas execuções com 7.500 e taxa de chegada de requisições de 3, 5, 7 e 10 respectivamente.

Operação	Latência Média (s)	Variância	Desvio Padrão	Vazão (TPS)
Inserção *PATIENTS*	1,47	0,26	0,51	3
Inserção *D_ITEMS*	1,55	0,24	0,49	3
Inserção *PRESCRIPTIONS*	1,61	0,26	0,51	3
Inserção *INPUTEVENTS_MV*	1,78	0,59	0,77	3
Busca *PATIENTS*	0,03	0,000031	0,00559	3

Tabela 31 – Execução com 7.500 transações e taxa de chegada de 3 req/s

Operação	Latência Média (s)	Variância	Desvio Padrão	Vazão (TPS)
Inserção *PATIENTS*	1,41	0,2	0,45	5
Inserção *D_ITEMS*	1,58	0,38	0,62	5
Inserção *PRESCRIPTIONS*	57,64	1447,81	38,05	4,7
Inserção *INPUTEVENTS_MV*	116,99	4692,15	68,5	4,3
Busca *PATIENTS*	0,03	2,31E-05	0,0048	5

Tabela 32 – Execução com 7.500 transações e taxa de chegada de 5 req/s

Operação	Latência Média (s)	Variância	Desvio Padrão	Vazão (TPS)
Inserção *PATIENTS*	3,32	16,59	4,07	6,9
Inserção *D_ITEMS*	21,82	332,99	18,25	6,7
Inserção *PRESCRIPTIONS*	188,47	14788,51	121,61	5,1
Inserção *INPUTEVENTS_MV*	275,16	26382,3	162,43	4,6
Busca *PATIENTS*	0,03	0,00002	0,00452	7

Tabela 33 – Execução com 7.500 transações e taxa de chegada de 7 req/s

Operação	Latência Média (s)	Variância	Desvio Padrão	Vazão (TPS)
Inserção *PATIENTS*	57,91	2362,59	48,61	8,3
Inserção *D_ITEMS*	132	6498,77	80,61	7,4
Inserção *PRESCRIPTIONS*	324,15	36532,73	191,14	5,4
Inserção *INPUTEVENTS_MV*	393,76	50223,23	224,11	5
Busca *PATIENTS*	0,03	0,000024	0,00485	10

Tabela 34 – Execução com 7.500 transações e taxa de chegada de 10 req/s

As Tabelas 35 a 38 exibem os valores encontrados nas métricas de desempenho, latência média e vazão das execuções com 10.000 transações e frequência de requisição variando de 3, 5, 7 e 10 por segundo. A disposição das Tabelas permite verificar o impacto da taxa de chegada nas métricas avaliadas, quanto frequências maiores geram uma maior a latência e também aumentam a vazão.

Operação	Latência Média (s)	Variância	Desvio Padrão	Vazão (TPS)
Inserção *PATIENTS*	1,48	0,29	0,54	3
Inserção *D_ITEMS*	1,61	0,27	0,52	3
Inserção *PRESCRIPTIONS*	1,64	0,28	0,53	3
Inserção *INPUTEVENTS_MV*	5,09	46,72	6,84	3
Busca *PATIENTS*	0,03	0,000022	0,00469	3

Tabela 35 – Execução com 10.000 transações e taxa de chegada de 3 req/s

Operação	Latência Média (s)	Variância	Desvio Padrão	Vazão (TPS)
Inserção *PATIENTS*	1,48	0,24	0,49	5
Inserção *D_ITEMS*	1,6	0,26	0,51	5
Inserção *PRESCRIPTIONS*	54,13	2208,87	47	4,7
Inserção *INPUTEVENTS_MV*	87,48	4321,47	65,74	4,6
Busca *PATIENTS*	0,03	0,000029	0,00543	5

Tabela 36 – Execução com 10.000 transações e taxa de chegada de 5 req/s

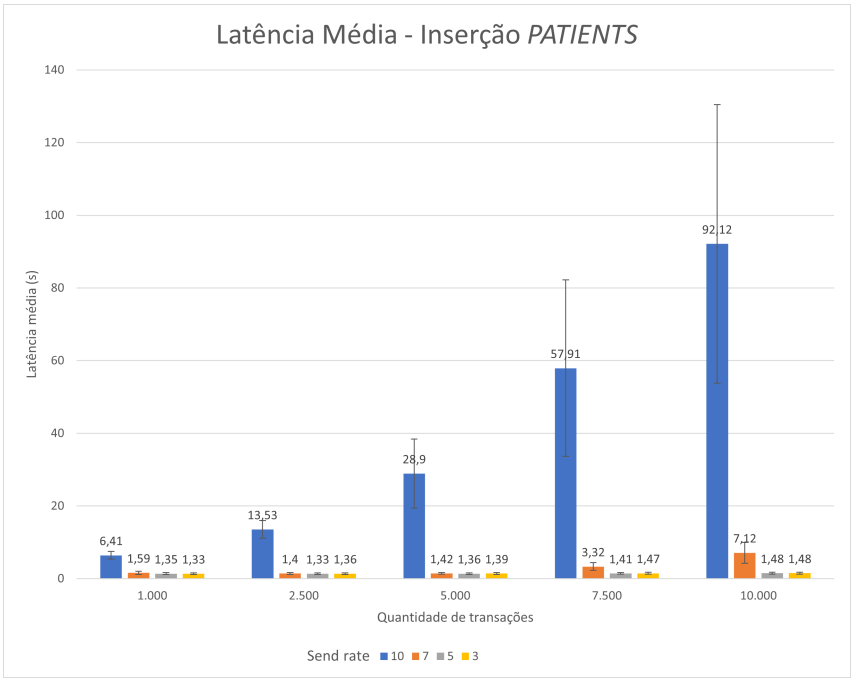
Operação	Latência Média (s)	Variância	Desvio Padrão	Vazão (TPS)
Inserção *PATIENTS*	7,12	76,54	8,75	6,9
Inserção *D_ITEMS*	40,16	905,74	30,1	6,6
Inserção *PRESCRIPTIONS*	287,75	29921,74	172,98	5
Inserção *INPUTEVENTS_MV*	389,65	52480,87	229,09	4,6
Busca *PATIENTS*	0,03	0,000026	0,0051	7

Tabela 37 – Execução com 10.000 transações e taxa de chegada de 7 req/s

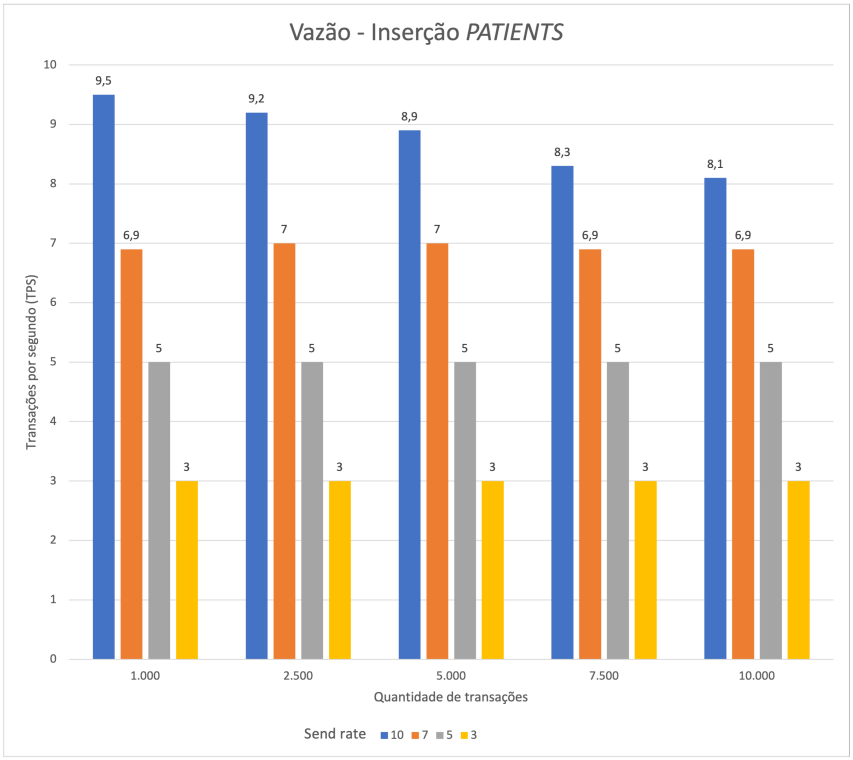
Operação	Latência Média (s)	Variância	Desvio Padrão	Vazão (TPS)
Inserção *PATIENTS*	92,12	5884,52	76,71	8,1
Inserção *D_ITEMS*	179,29	12080	109,91	7,3
Inserção *PRESCRIPTIONS*	438,65	66175,84	257,25	5,3
Inserção *INPUTEVENTS_MV*	410,31	62998,23	250,99	5,4
Busca *PATIENTS*	0,03	0,000025	0,00503	10

Tabela 38 – Execução com 10.000 transações e taxa de chegada de 10 req/s

Os dez gráficos apresentados nas Figuras 16 a 20 estão organizados separando cada operação realizada, as quatro inserções (dos arquivos *PATIENTS*, *D_ITEMS*, *PRESCRIPTIONS* e *INPUTEVENTS_MV*) e a busca (com o arquivo *PATIENTS*). Cada gráfico mostra a variação da quantidade de transações efetuadas (1.000, 2.500, 5.000, 7.500 e 10.000) e as quatro variações de taxa de envio (10, 7, 5 e 3). Cada Figura de 16 a 20 mostra os resultados para latência no gráfico (a), e a vazão no gráfico (b). A barra de erro dos gráficos para as latências média retrata o desvio padrão obtido.



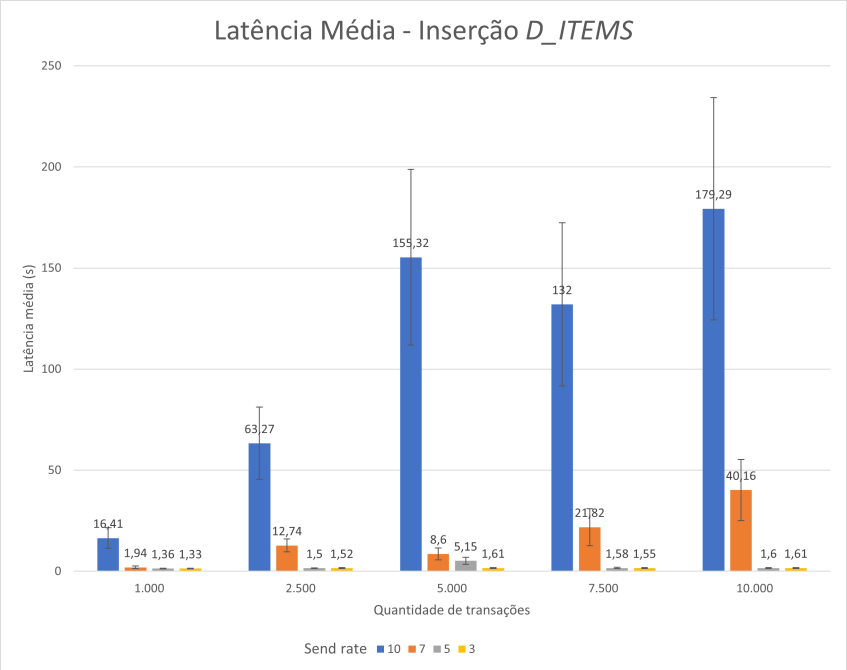
(a) Latência



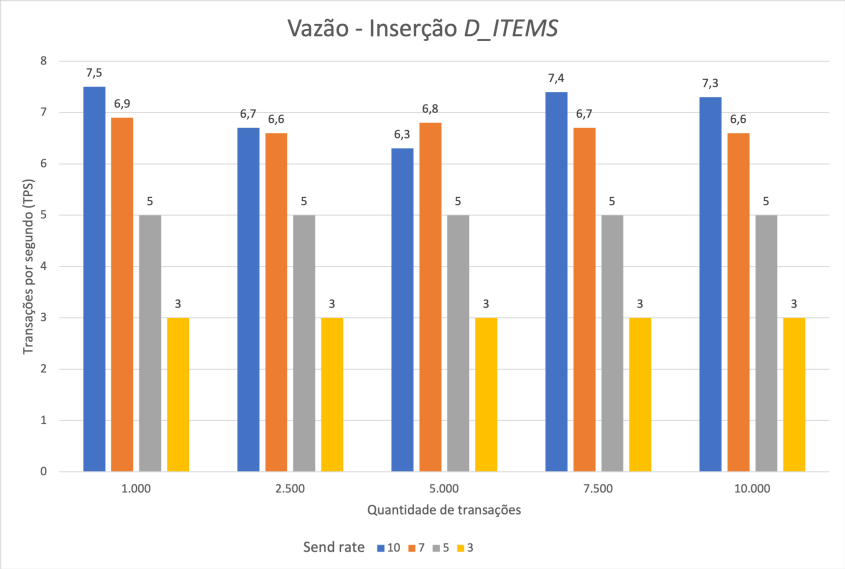
(b) Vazão

Figura 16 – Resultados para inserção de *PATIENTS*

As Figuras 16 a 20 mostram que em todos os cenários testados a latência média variou de maneira similar tanto para a variação das quantidades de transações quanto para a modificação das taxas de envio, com uma diminuição das latências médias de escrita, conforme ambas (quantidade de transações e taxas de envios) diminuem.



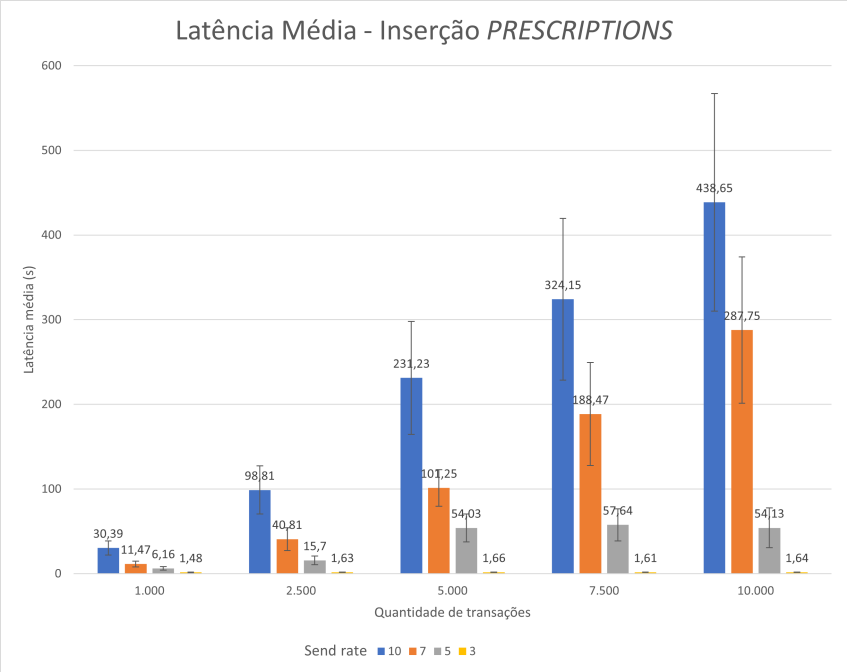
(a) Latência



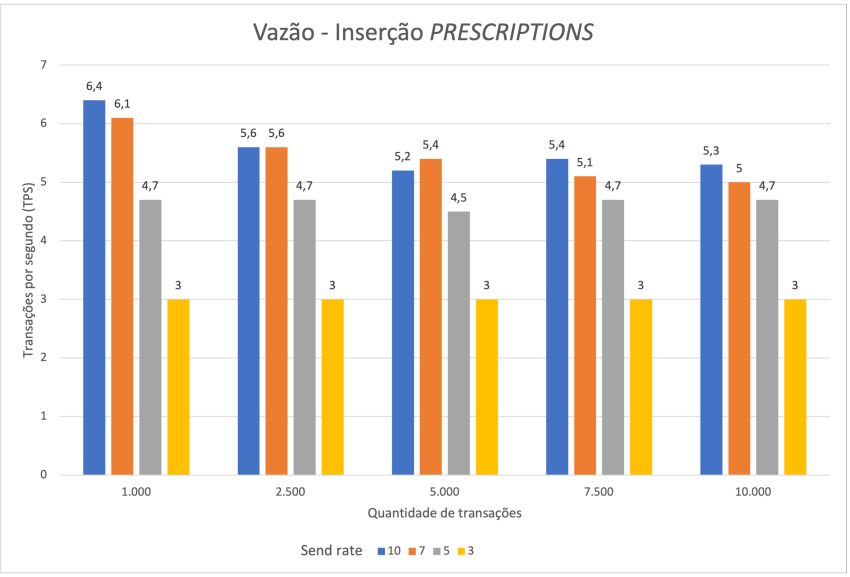
(b) Vazão

Figura 17 – Resultados para inserção de *D_ITEMS*

As latências observadas nas escritas sobre os diferentes arquivos, como esperado, mostram tempos médios maiores para arquivos com mais indexações e maior quantidade de campos, ou seja, mais dados a serem manipulados a cada transação. As latências observadas para a leitura dos dados permaneceram no mesmo patamar de valor (0,03s) em virtude da baixa quantidade de transações e de taxa de envio usadas nos experimentos, frente ao tempo de atendimento dessas requisições.



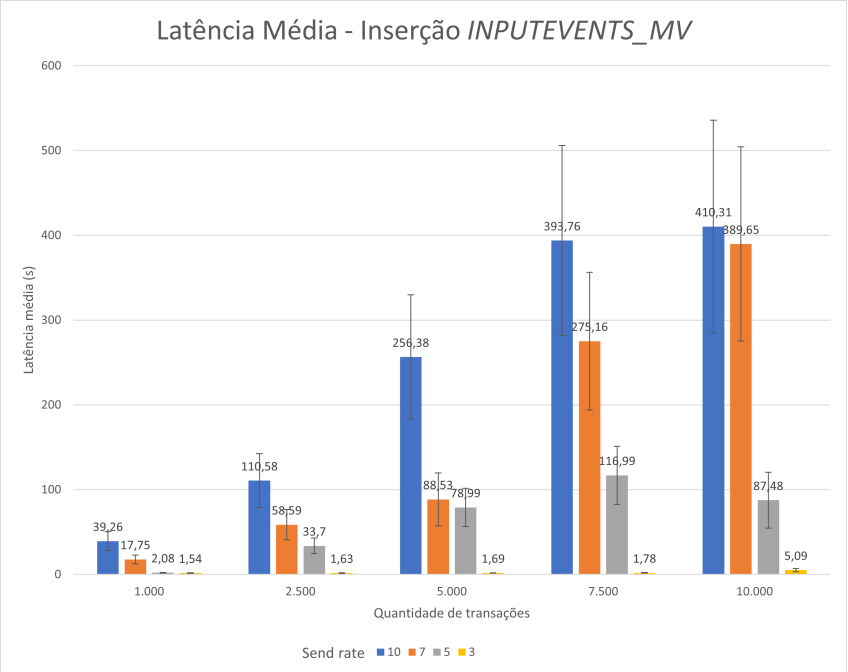
(a) Latência



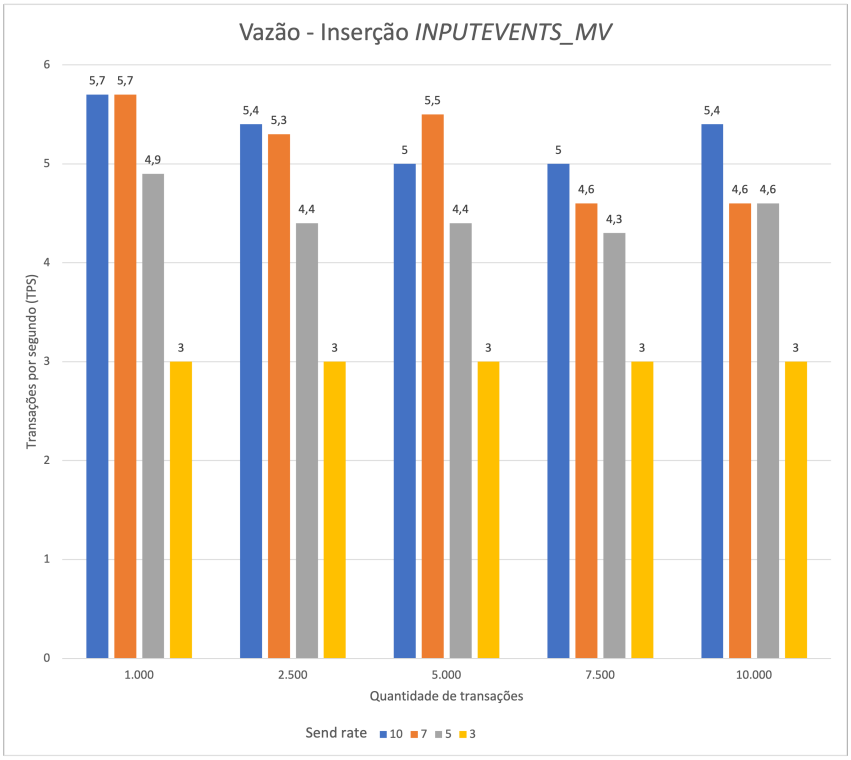
(b) Vazão

Figura 18 – Resultados para inserção de *PRESCRIPTIONS*

Considerando a vazão, observa-se que os resultados para menores taxas de envio (5 e 3) ficaram em vários casos limitados às respectivas taxas. A vazão não atingiu o teto da taxa de envio 5, nas escritas dos arquivos *PRESCRIPTIONS* (Figura 18b) e *INPUTEVENTS_MV* (Figura 19b) por requisitarem mais indexações e/ou manipularem mais dados. Porém, mesmo nesses casos obteve vazões altas de, no mínimo, 4.3 transações por segundo.



(a) Latência

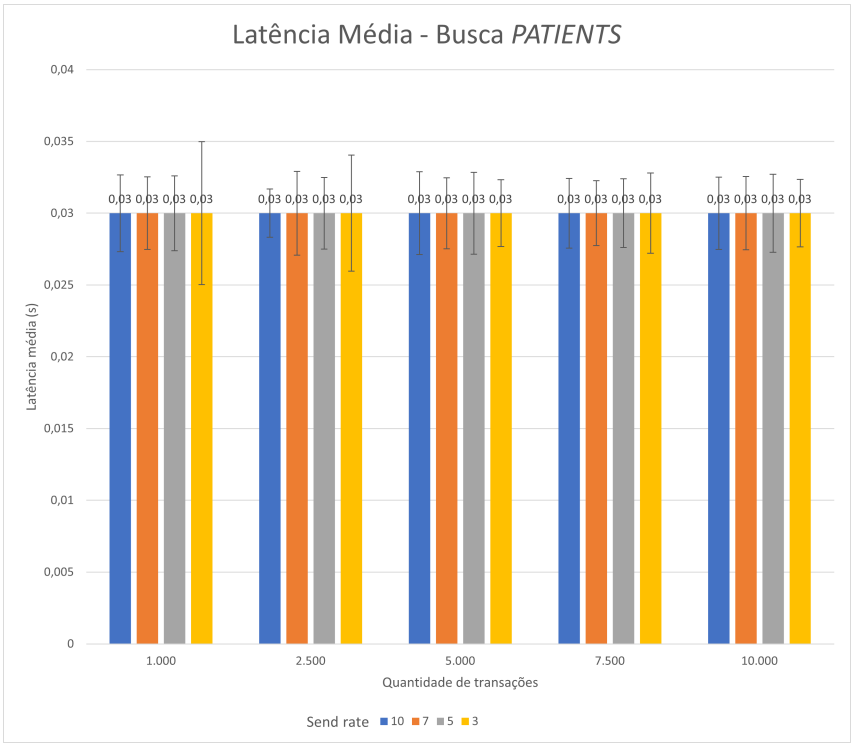


(b) Vazão

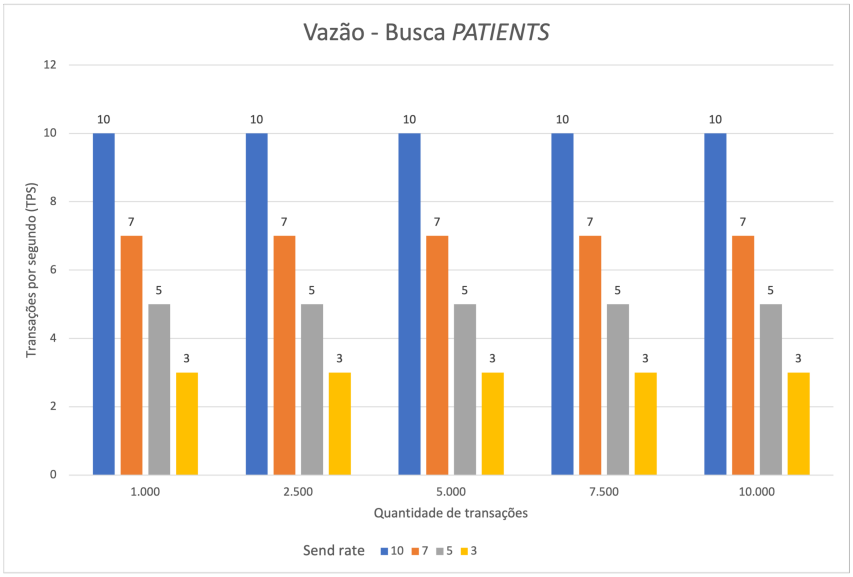
Figura 19 – Resultados para inserção de *INPUTEVENTS_MV*

Analisando-se as taxas maiores (10 e 7), observou-se que, de um modo geral, neste experimento, as vazões aumentaram quando quantidades menores de transações foram usadas nas escritas dos diferentes arquivos, indicando que possivelmente o processo de escrita tem o seu gargalo no processo de criação de blocos. As vazões das leituras foram limitadas pelas taxas

de envio, pelos mesmos motivos da latência média permanecer constante.



(a) Latência



(b) Vazão

Figura 20 – Resultados para busca de *PATIENTS*

5.6 Análise dos Resultados

5.6.1 Hipótese I

A primeira hipótese formulada para os experimentos é sobre a comparação entre o desempenho, em termos das métricas de vazão e latência, das operações de inserção e busca na rede. Essas operações são caracterizadas como tipos de transações diferentes no contexto da *blockchain*, em particular no *Fabric* que especifica essas operações com diferentes etapas, como descrito na Seção 2.5.2.

Para apurar a veracidade desta hipótese a análise realizada compara os valores das métricas coletadas (latência e vazão) obtidos na inserção dos arquivo do *MIMIC-III PATIENTS* e na busca com o mesmo arquivo. Para a latência, foi verificada a normalidade de cada amostra de dados. O teste de *Cramér-von Mises* (THODE, 2002) com nível de significância 95% confirma a não-normalidade do conjunto de dados, os quais apresentam *p-valor* menor do que 0,05.

Dada a não-normalidade, o teste de *Wilcoxon* (WALPOLE *et al.*, 1993) com nível de significância 95% foi aplicado para verificar se as amostras são estatisticamente equivalentes ou não. Os *p-valores* encontrados comparando as latências nas execuções da inserção e da busca de *PATIENTS* é descrita na Tabela 39.

Transações	**3**	**5**	**7**	**10**
1.000	0	0	0	0
2.500	0	0	0	0
5.000	0	0	0	0
7.500	0	0	0	0
10.000	0	0	0	0

Tabela 39 – *p-valores* entre a inserção e a busca de *PATIENTS*

Dado que todos os *p-valores* encontrados são menores que 0,05 pode-se afirmar que as amostras de dados não são equivalentes. Com isso, observa-se as latências médias obtidas nas execuções realizadas neste experimento. Para todos os cenários avaliados neste experimento, a latência das inserções mostrou ser significativamente maior que a da busca (gráficos das Figuras 16 e 20, em que é possível comparar a inserção e a busca do mesmo arquivo *PATIENTS*).

Comparando as vazões alcançadas na inserção e na busca, é possível observar que os valores encontrados na inserção são inferiores aos da busca. Dessa forma, pode-se afirmar que as inserções de dados são mais custosas no desempenho de aplicações *blockchain* com o *Hyperledger Fabric* que a busca de informações.

Observa-se no gráfico da Figura 20a) que a latência das buscas se manteve constante nos cenários avaliados (com uma média de 0.03s) e no gráfico da Figura 20b que a vazão acompanha a taxa de envio das requisições. Isso indica que o atendimento das requisições de busca, nos

experimentos realizados, está limitado pela taxa de envio de tais requisições para a rede, o que mantém a latência média constante.

5.6.2 Hipótese II

A segunda hipótese refere-se ao impacto no desempenho dos fatores taxa de chegada das requisições e total de requisições submetidas para processamento na *blockchain*.

Nas amostras da latência das operações realizadas, quatro inserções e a busca, foi aplicado o teste de normalidade para todos os cenários testados. O teste de *Cramér-von Mises* (THODE, 2002) com nível de significância 95% confirma a não-normalidade do conjunto de dados, os quais apresentam *p-valor* menor do que 0,05.

Considerando a não-normalidade, o teste de *Wilcoxon* (WALPOLE *et al.*, 1993) com nível de significância 95% foi aplicado para verificar se as amostras são estatisticamente equivalentes ou não. Os *p-valores* encontrados comparando as latências nas execuções das inserções de *PATIENTS*, *D_ITEMS*, *PRESCRIPTIONS* e *INPUTEVENTS_MV* e da busca de *PATIENTS* são mostradas nas Tabelas 40 a 44. Para cada quantidade de entrada avaliada foram comparadas as frequências de acesso utilizadas neste experimento.

Transações	10 e 7	10 e 5	10 e 3	7 e 5	7 e 3	5 e 3
1.000	0	0	0	3,58e-09	4,889e-09	0,31
2.500	0	0	0	0,00084	0,34	0,00038
5.000	0	0	0	4,43E-06	0,0378	0,00018
7.500	0	0	0	0	0	1,60e-13
10.000	0	0	0	0	0	0,19

Tabela 40 – *p-valores* entre as inserções de *PATIENTS*

Nas inserções de *PATIENTS*, com 1.000 entradas e taxa de requisições de 5 e 3 são estatisticamente equivalentes, pois *p-valor* maior que 0,05. Outro cenário no qual o teste de equivalência é aceito é com 2.500 transações, comparando os valores de frequência de aceso de 7 e 3. Assim como para 10.000 transações, entre requisições sendo realizadas a cada 5 e 3 segundos.

Transações	10 e 7	10 e 5	10 e 3	7 e 5	7 e 3	5 e 3
1.000	0	0	0	0	1,57e-13	0,00065
2.500	0	0	0	0	0	0,088
5.000	0	0	0	0	0	0
7.500	0	0	0	0	0	0,35
10.000	0	0	0	0	0	0,00922

Tabela 41 – *p-valores* entre as inserções de *D_ITEMS*

O teste de equivalência também é aceito na inserção de *D_ITEMS* nas execuções com 2.500 transações e requisições de 5 e 3 segundos, dado que *-p-valor* maior que 0,05. Também com taxa de 5 e 3 segundos, o teste de equivalência comprova a equivalência estatística das amostras quando considerado 7.500 transações.

Transações	10 e 7	10 e 5	10 e 3	7 e 5	7 e 3	5 e 3
1.000	0	0	0	0	0	0
2.500	0	0	0	0	0	0
5.000	0	0	0	0,00027	0	0
7.500	0	0	0	0	0	0
10.000	0	0	0	0	0	0

Tabela 42 – *p-valores* entre as inserções de *PRESCRIPTIONS*

Transações	10 e 7	10 e 5	10 e 3	7 e 5	7 e 3	5 e 3
1.000	0	0	0	0	0	0
2.500	0	0	0	0	0	0
5.000	0	0	0	0	0	0
7.500	0	0	0	0	0	0
10.000	0	0	0	0	0	0

Tabela 43 – *p-valores* entre as inserções de *INPUTEVENTS_MV*

Transações	10 e 7	10 e 5	10 e 3	7 e 5	7 e 3	5 e 3
1.000	0	7,22E-08	0	5,15e-07	1,42e-05	0
2.500	0	0	0	1,81e-14	0	0
5.000	0	0	0	0,012	2,52e-08	6,77e-15
7.500	0	0	0	0,00029	0	0
10.000	0	0	0	0	0	0

Tabela 44 – *p-valores* entre as buscas de *PATIENTS*

Para as inserções de *PRESCRIPTIONS* e *INPUTEVENTS_MV* e na busca com *PATIENTS* não foi encontrado *p-valor* maior que 0,05, dessa forma, rejeita-se a equivalência das amostras analisadas.

As Figuras 16 a 20 mostram que no experimento realizado, obtiveram-se diferentes desempenhos com a variação desses fatores, sendo as operações de inserção as mais afetadas por essas modificações. O desempenho das inserções nos experimentos foi mais afetado pela taxa de chegada dessas requisições. A latência das inserções neste experimento variou 221% a depender da taxa de chegada escolhida para uma mesma quantidade de entradas, como o caso da inserção do arquivo *INPUTEVENTS_MV* (Figura 19a) para 7.500 entradas, observando a variação entre a latência com taxa de 10 e 3.

Ainda para estas inserções, nota-se que a diferença entre as latências diminuiu e tendeu a valores aproximados para taxas menores de requisições. Essa tendência pode ser observada na Figura 16a que apresenta os resultados da latência média da inserção do arquivo *PATIENTS*; para a taxa de chegada de requisições definida em 3 os valores da latência variaram entre 1,48 e 1,33.

A coleta individual do tempo de latência de cada transação mostra que se forma uma fila de requisições não atendidas, principalmente nos cenários de inserção com 7.000 e 10.000 transações e taxa de requisições de 7 e 10. Neste experimento, assim como nos demais, não foi estabelecido um limite de tempo para as transações serem armazenadas na rede, logo a latência média e o desvio padrão aumentam conforme aumenta o número de requisições esperando serem atendidas. Como esperado, a latência no *Hyperledger Caliper* é calculada com base no tempo total de inserção de uma transação, o qual considera o tempo desde esta transação ser conhecida pela rede até ela ser armazenada como uma cópia em todos os nós. O aumento da latência média e do desvio padrão é ilustrado nos gráficos das Figuras 16a a 19a.

Esse resultado aponta possíveis gargalos do processo de inserção de dados na *blockchain Hyperledger Fabric*, como o processo de criação de blocos que é analisado no terceiro experimento deste trabalho.

Considerando as buscas neste experimento, o cenário é não conclusivo sobre quais dos fatores apresentam maior impacto no desempenho, uma vez que os experimentos realizados não demonstraram variações nas métricas coletadas. Isso se deve ao fato das execuções não terem alcançado os limites superiores de demanda que afetem vazão e latência para buscas.

5.6.3 Hipótese III

A terceira hipótese neste experimento analisa o impacto das diferentes características dos arquivos, com foco no número de colunas (campos) dos arquivos e suas indexações para a inserção no banco de dados *CouchDB*, nas métricas de desempenho coletadas para as inserções. Percebe-se que as inserções do arquivo *PATIENTS* resultaram em um menor tempo de latência e uma maior vazão que os demais arquivos para as escritas, em virtude de não apresentarem a necessidade de indexação. O impacto da indexação aparece é maior quando comparado com a inserção do arquivo *D_ITEMS* que contém duas indexações e um número próximo de campos a cada inserção.

Na maioria dos cenários testados, a latência da inserção do arquivo *INPUTEVENTS_MV* é maior que em *PRESCRIPTIONS*, sugerindo que a quantidade de dados do arquivo influencia mais no tempo de resposta que a quantidade de indexações. Essa conclusão se deve ao arquivo *INPUTEVENTS_MV* conter doze campos a mais que o arquivo *PRESCRIPTION* e uma indexação a menos. Também é possível perceber que há um menor contraste entre as vazões da inserções desses arquivos, quando comparadas com as latências encontradas, dadas as diferentes quantidades de requisições efetuadas.

5.7 Considerações Finais

Este capítulo mostrou as etapas de elaboração e realização do primeiro experimento com a finalidade de avaliar o impacto de diferentes cenários de requisições para operações de escrita e leitura com o *Hyperlegder Fabric*. Os resultados deste experimento destacam como o desempenho dessas operações é afetado pela maneira como o *Fabric* trabalha com as requisições de escrita e leitura na rede.

Além disso, foi possível notar como a modificação da quantidade e taxa de chegada de transações modificam de maneira distinta essas duas operações. As escritas apresentam as maiores variações quando modificados esses fatores nas métricas analisadas. Para finalizar, uma análise das diferenças entre os arquivos usadas nas inserções foi realizada. Arquivos com quantidades maiores de indexação para serem inseridas no *CouchDB* têm as maiores latências e menores vazões quando comparadas a arquivos sem ou menos indexações.

O próximo Capítulo apresenta o segundo experimento, este realizado com o objetivo de comparar o desempenho da *blockchain* sem e o com o uso do banco de dados *CouchDB*.

CAPÍTULO

6

EXPERIMENTO 2 - ARMAZENAMENTO COM E SEM O *COUCHDB*

6.1 Considerações Iniciais

Este Capítulo relata o segundo experimento deste trabalho com o propósito de comparar o desempenho da *blockchain* sem e com o uso de um banco de dados externo, o *CouchDB*. Atráves deste experimento espera-se isolar o armazenamento e analisar o impacto das formas de registro de informações no desempenho de aplicações *blockchain* com dados heterogêneos.

O Capítulo é divido como descrito a seguir. Inicialmente o escopo do experimento é apresentado, em seguida o planejamento com as hipóteses e questões de pesquisa é descrito, além dos fatores e níveis analisados. A infraestrutura utilizada em termos de máquinas e parâmetros da *blockchain* considerados é explicada na sequência, assim como a metodologia para a execução. A Seção de resultados inclui as tabelas e os gráficos que mostram os o desempenho da aplicação em termos de latência média e vazão com as duas formas de armazenamento. Por último, são apresentadas as considerações sobre os principais resultados encontrados e como estes validam ou não as hipóteses de pesquisa levantadas para o experimento.

6.2 Escopo

Este experimento compara o desempenho de uma rede *blockchain* no armazenamento de dados heterogêneos, quando os dados são armazenados na cadeia e quando são associados bancos de dados. As métricas utilizadas neste experimento para quantificar o desempenho são latência e vazão das requisições de acesso aos dados. O *Hyperledger Fabric* na versão 1.4.1 é a plataforma de desenvolvimento *blockchain* escolhida para ser o objeto de avaliação. O foco qualitativo deste experimento é estabelecer o desempenho do *Hyperledger Fabric* com o uso de dados heterogêneos.

A perspectiva deste estudo é quantificar o desempenho do *Fabric* comparando no mesmo cenário duas formas de armazenamento, com e sem o uso de banco de dados.

6.3 Planejamento

Os experimentos tiveram como contexto dados médicos reais obtidos da base de dados *MIMIC-III* (JOHNSON *et al.*, 2016), descrita na Seção 4.2. As execuções foram realizadas na rede do laboratório de pesquisa (*LaSDPC*).

Dado o objetivo do experimento de comparar o desempenho da *blockchain* sob duas perspectivas de armazenamento com ou sem um banco de dados, duas hipóteses foram levantadas.

1. A primeira hipótese afirma que associar a *blockchain* com um de banco de dados externo para o armazenamento de informações impacta negativamente na latência e na vazão das requisições de acesso aos dados (escrita e leitura) em comparação com o armazenamento na cadeia.

2. A segunda hipótese declara que usar um banco de dados com a *blockchain* impacta de maneira distinta os dois tipos de requisições, leitura e escrita.

A primeira hipótese tem como base o custo do acesso ao banco de dados *CouchDB*. Ao acrescentar o banco de dados na infraestrutura do *Hyperledger Fabric*, espera-se que a latência e a vazão das operações testadas (quatro escritas e uma leitura) sejam afetadas negativamente, aumentando a latência e diminuindo a vazão.

Já a segunda hipótese considera que a inserção de dados é mais custosa que a busca, principalmente quando o *CouchDB* está inserido na arquitetura da rede *blockchain*.

Para investigar a validade dessas duas hipóteses, foram elencadas as seguintes questões de pesquisa:

QP1: Quais são as latências e vazões das requisições de escritas e de leituras a dados heterogêneos com a *blockchain*, usando banco de dados para armazenamento externo à cadeia?

QP2: Quais são as latências e vazões das requisições de escritas e de leituras a dados heterogêneos com a *blockchain*, sem usar banco de dados para armazenamento externo à cadeia?

As métricas vazão e latência foram calculadas com o uso do *benchmark Hyperledger Caliper* e os diferentes tipos de transações foram calculados de acordo com as Equações de 4.1 a 4.4.

Os fatores e níveis utilizados neste experimento são descritos na Tabela 45.

Fatores	Níveis			
Quantidade de transações	10.000	12.000	15.000	
Taxa de envio (Requisições por segundo)	35	40	45	50
Armazenamento	sem Banco	com Banco		

Tabela 45 – Fatores e níveis Experimento 2

A quantidade total de transações realizadas nesse estudo é o produto da soma dos cinco níveis da quantidade de transações (37.000), os quatro níveis de taxa de envio, os dois níveis de armazenamento e, por último, as cinco operações realizadas (quatro inserções de diferentes arquivos e uma busca). Para o arquivo *D_ITEMS* não houve execução considerando 15.000 transações, pois esse arquivo tem menos que esse total de entradas. Ao todo foram realizadas 1.030.000 transações sobre a *blockchain*.

6.4 Infraestrutura para Execução

Os parâmetros usados na execução deste experimento estão na Tabela 46. O primeiro parâmetro indica o número de máquinas utilizadas nas execuções. O segundo e o terceiro definem a quantidade de nós que vão validar uma transação na primeira etapa do processo de validação, nesse experimento, os quatro nós pertencentes às organizações atuam nessa função. Os quatro últimos caracterizam o tamanho e o tempo de criação de cada bloco.

O número de contêineres criados neste experimento variou de acordo com a forma de armazenamento utilizado. Para os experimentos executados sem o *CouchDB* foram criados cinco contêineres; já com o banco de dados, como usual, foi adicionado um nó com o *CouchDB* para cada nó pertencente a uma organização, somando mais quatro contêineres à rede (nove contêineres ao todo).

Parâmetro	Valor
Número de máquinas reais	4
Nós validadores	4
Política de validação	AND
MaxMessageCount	10
AbsoluteMaxBytes	128 MB
PreferredMaxBytes	128 MB
BatchTimeout	1s

Tabela 46 – Parâmetros da execução Experimento 2

Foram criadas duas configurações de rede para as execuções realizadas, apesar disso, apenas uma característica as diferenciam: a presença ou não do contêiner *CouchDB* para o armazenamento de informações exterior a cadeia. Ambas as redes criadas contêm cinco nós

atuantes na rede, estes divididos em duas organizações com dois nós cada e um nó atuando como *orderer*.

Os nós estão em um único *channel*, dessa forma, não há restrição de acesso às informações compartilhadas entre os integrantes da rede.

Neste experimento foram utilizadas quatro máquinas, cada uma com um contêiner com um nó pertencente à rede do *Fabric*. Uma das máquinas tem um contêiner a mais com o nó *orderer*. Quando o *CouchDB* foi utilizado, outro contêiner na mesma máquina foi criado. As Figuras 21 e 22 ilustram a rede criada nos dois cenários de execução, com e sem o banco de dados *CouchDB* respectivamente.

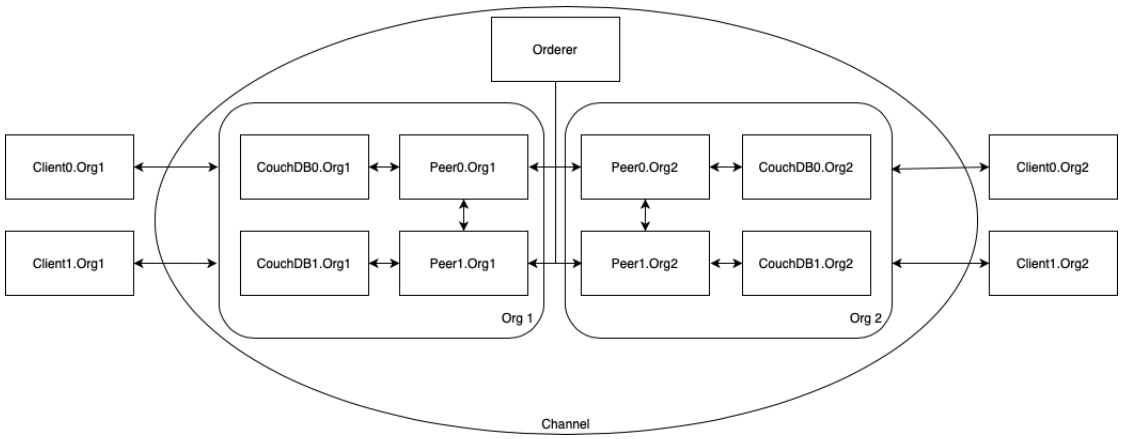


Figura 21 – Configuração da rede com o *CouchDB* Experimento 2

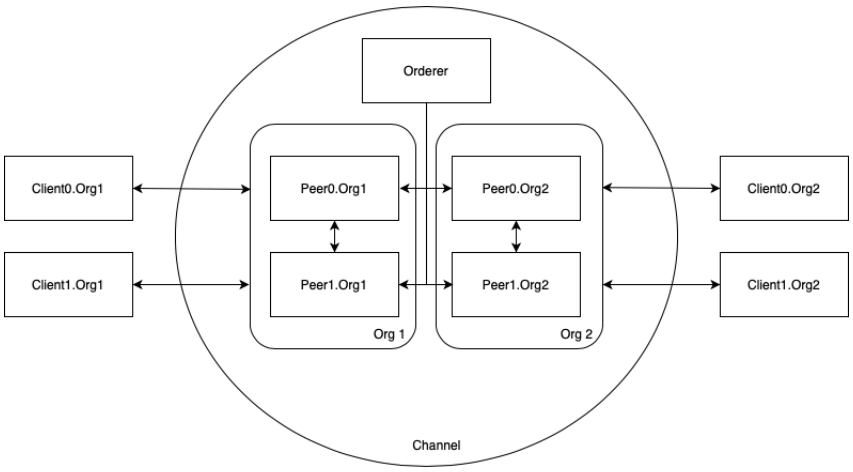


Figura 22 – Configuração da rede sem o *CouchDB* Experimento 2

Os sistemas de software instalados para a execução deste experimento foram somente os pré-requisitos de instalação encontrados na documentação do *Hyperledger Fabric* e *Hyperledger Caliper*. Os nós estão distribuídos na rede (*ICMC-Lasdpc*) em quatro máquinas com a configuração descrita na Tabela 47.

Configurações de *Hardware* (*Cluster Andromeda - LaSDPC*)	
Sistema Operacional	Ubuntu 16.04.7 LTS
CPU	QEMU Virtual CPU (512 KB Cache, 4 GHz)
Memória	16 GB RAM
Disco	Seagate HD 2TB ST2000DM001-1ER1

Tabela 47 – Características das máquinas do Experimento 2

O *Hyperledger Caliper* foi utilizado como ponto de observação, através dele é possível configurar um monitor para coletar as informações sobre as transações e gerar os relatórios de desempenho apontando a latência média e a vazão das transações. Além dos dados trazidos no relatório, também foi coletada a latência individual de cada transação realizada.

As operações efetuadas têm como objetivo avaliar as diferenças de desempenho entre as diferentes transações executadas pelo *Hyperledger Fabric*. A inserção de dados analisa a transação de *update* e a busca a de *read*. Para cada inserção realizada, os dados são lidos dos arquivos do *MIMIC-III*, distribuídos pela rede e armazenados ou na cadeia ou no *CouchDB*. A validação das transações pelos nós ocorre antes destas de serem distribuídas na rede (armazenadas na cadeia ou no *CouchDB*). Tal processo de validação das transações de *update* ocorre em três etapas, como descrito na Seção 2.5.2. Já na busca, o processo de validação não ocorre, pois o nó mais próximo atende a informação solicitada. Neste experimento foram realizadas cinco operações, quatro inserções e uma busca. A escolha dessas operações foi baseada nos arquivos encontradas no *MIMIC-III* (descritas na Seção 4.2).

Em cada rodada de execução, a primeira inserção é do arquivo *PATIENTS*, a segunda inserção é do arquivo *D_ITEMS*, a próxima inserção é do arquivo *PRESCRIPTIONS* e por último é realizada a inserção da *INPUTEVENTS_MV*. A busca se baseia no arquivo *PATIENTS*. Essa operação diferencia-se das demais por ter dependência de outra operação, a inserção da arquivo *PATIENTS* na rede. Tanto a inserção quanto a busca são realizadas em blocos, ou seja, uma execução só é inciada ao término da anterior.

Este experimento não inclui testes de falhas tanto de nós quanto da rede e ataque do tipo *man in middle*. Além disso, não apresenta testes de execução com transações que não apresentam as características esperadas pela *blockchain*.

6.5 Resultados

Os resultados encontrados nas execuções descritas são mostrados nas Tabelas de 48 a 71 e nos gráficos das Figuras 23 a 32.

As Tabelas apresentadas são divididas pelos cenários de testes considerados, as primeiras de 48 a 59 mostram a latência média, variância e desvio padrão dessa métrica e além disso, a

vazão obtida no cenário avaliado sem o uso do *CouchDB*. Para as execuções com 10.000 entradas, variando a frequência de acesso aos dados de 35, 40, 45 e 50, as Tabelas de 48 a 51 exibem os resultados das métricas de desempenho obtidos nos testes realizados sem o banco de dados.

Operação	Latência Média (s)	Variância	Desvio Padrão	Vazão (TPS)
Inserção *PATIENTS*	0,65	0,36	0,6	35
Inserção *D_ITEMS*	0,46	0,01	0,1	35
Inserção *PRESCRIPTIONS*	0,52	0,09	0,09	35
Inserção *INPUTEVENTS_MV*	0,5	0,01	0,11	35
Busca *PATIENTS*	0,02	0,00002	0,0045	35

Tabela 48 – Resultados para 10.000 transações e 35 req/s sem o *CouchDB*

Operação	Latência Média (s)	Variância	Desvio Padrão	Vazão (TPS)
Inserção *PATIENTS*	2,85	0,29	0,54	39,5
Inserção *D_ITEMS*	1,63	0,21	0,46	39,7
Inserção *PRESCRIPTIONS*	2,19	1,78	1,34	39,4
Inserção *INPUTEVENTS_MV*	2,84	2,76	1,66	39,3
Busca *PATIENTS*	0,02	0,000021	0,0046	40

Tabela 49 – Resultados para 10.000 transações e 40 req/s sem o *CouchDB*

Operação	Latência Média (s)	Variância	Desvio Padrão	Vazão (TPS)
Inserção *PATIENTS*	19,94	108,62	10,42	39,6
Inserção *D_ITEMS*	18,07	108,95	10,44	39,8
Inserção *PRESCRIPTIONS*	17,87	85,59	9,25	40
Inserção *INPUTEVENTS_MV*	26,04	164,11	12,81	38,3
Busca *PATIENTS*	0,01	0,000022	0,0047	45

Tabela 50 – Resultados para 10.000 transações e 45 req/s sem o *CouchDB*

Operação	Latência Média (s)	Variância	Desvio Padrão	Vazão (TPS)
Inserção *PATIENTS*	43,52	493,43	22,21	37,2
Inserção *D_ITEMS*	39,2	394,07	19,85	38,4
Inserção *PRESCRIPTIONS*	39,11	421,51	20,53	38,2
Inserção *INPUTEVENTS_MV*	43,15	514,62	22,69	37,1
Busca *PATIENTS*	0,02	0,000026	0,0051	50

Tabela 51 – Resultados para 10.000 transações e 50 req/s sem o *CouchDB*

Já as Tabelas de 52 a 55 mostram as métricas de desempenho coletadas nas execuções com 12.000 transações e alterando a taxa de acesso aos dados das operações consideradas em 35, 40, 45 e 50, respectivamente.

Operação	Latência Média (s)	Variância	Desvio Padrão	Vazão (TPS)
Inserção *PATIENTS*	0,51	0,09	0,29	35
Inserção *D_ITEMS*	0,53	0,13	0,36	35
Inserção *PRESCRIPTIONS*	0,47	0,01	0,1	35
Inserção *INPUTEVENTS_MV*	0,48	0,01	0,1	35
Busca *PATIENTS*	0,02	0,000019	0,0044	35

Tabela 52 – Resultados para 12.000 transações e 35 req/s sem o *CouchDB*

Operação	Latência Média (s)	Variância	Desvio Padrão	Vazão (TPS)
Inserção *PATIENTS*	4,27	5,06	2,25	39,1
Inserção *D_ITEMS*	3,6	1,7	1,3	39,4
Inserção *PRESCRIPTIONS*	4	6,13	2,48	39
Inserção *INPUTEVENTS_MV*	10,24	32,57	5,71	38
Busca *PATIENTS*	0,02	0,000021	0,0046	40

Tabela 53 – Resultados para 12.000 transações e 40 req/s sem o *CouchDB*

Operação	Latência Média (s)	Variância	Desvio Padrão	Vazão (TPS)
Inserção *PATIENTS*	28,52	218,01	14,77	38,6
Inserção *D_ITEMS*	34,1	314,87	17,74	37,5
Inserção *PRESCRIPTIONS*	38,53	442,73	21,04	36,2
Inserção *INPUTEVENTS_MV*	40,52	472,08	21,73	36,5
Busca *PATIENTS*	0,02	0,000021	0,0046	45

Tabela 54 – Resultados para 12.000 transações e 45 req/s sem o *CouchDB*

Operação	Latência Média (s)	Variância	Desvio Padrão	Vazão (TPS)
Inserção *PATIENTS*	41,14	428,91	20,71	39,7
Inserção *D_ITEMS*	40,79	423,64	20,58	39,5
Inserção *PRESCRIPTIONS*	41,87	446,84	21,14	39,5
Inserção *INPUTEVENTS_MV*	41,55	493,46	22,214	39,4
Busca *PATIENTS*	0,01	0,000024	0,0049	50

Tabela 55 – Resultados para 12.000 transações e 50 req/s sem o *CouchDB*

Os resultados encontrados nas execuções com 15.000 entradas e considerando 35, 40, 45 e 50 requisições por segundo são mostradas nas Tabelas definidas em 56 a 59. A inserção do arquivo *MIMIC-III D_ITEMS* não foi executada para 15.000 entradas, como definido na Seção 6.3.

Operação	Latência Média (s)	Variância	Desvio Padrão	Vazão (TPS)
Inserção *PATIENTS*	0,53	0,071	0,27	35
Inserção *PRESCRIPTIONS*	0,51	0,06	0,24	35
Inserção *INPUTEVENTS_MV*	0,48	0,01	0,1	35
Busca *PATIENTS*	0,02	0,00002	0,0044	35

Tabela 56 – Resultados para 15.000 transações e 35 req/s sem o *CouchDB*

Operação	Latência Média (s)	Variância	Desvio Padrão	Vazão (TPS)
Inserção *PATIENTS*	20,4	148,6	12,19	36,8
Inserção *PRESCRIPTIONS*	10,69	33,62	5,8	38,1
Inserção *INPUTEVENTS_MV*	20,99	147,09	12,13	36,7
Busca *PATIENTS*	0,02	0,000025	0,005	40

Tabela 57 – Resultados para 15.000 transações e 40 req/s sem o *CouchDB*

Operação	Latência Média (s)	Variância	Desvio Padrão	Vazão (TPS)
Inserção *PATIENTS*	38,14	436,21	20,89	38,1
Inserção *PRESCRIPTIONS*	34,85	344,35	18,56	38,8
Inserção *INPUTEVENTS_MV*	33,47	315,91	17,77	39
Busca *PATIENTS*	0,02	0,000025	0,005	45

Tabela 58 – Resultados para 15.000 transações e 45 req/ssem o *CouchDB*

Operação	Latência Média (s)	Variância	Desvio Padrão	Vazão (TPS)
Inserção *PATIENTS*	67,31	1091,85	33,04	37,3
Inserção *PRESCRIPTIONS*	60,18	977,14	31,26	37,9
Inserção *INPUTEVENTS_MV*	68,77	1266,55	35,59	36,7
Busca *PATIENTS*	0,01	0,000023	0,0048	50

Tabela 59 – Resultados para 15.000 transações e 50 req/s sem o *CouchDB*

As próximas Tabelas apresentam os resultados encontrados nas execuções considerando a *blockchain Hyperldger Fabric* com o uso do banco de dados *CouchDB*. De 60 a 63 as Tabelas destacam as medidas de desempenho encontradas nos cenários de 10.000 transações, com 35, 40, 45 e 50 requisições por segundo, com o banco de dados.

Operação	Latência Média (s)	Variância	Desvio Padrão	Vazão (TPS)
Inserção *PATIENTS*	432,89	51710,07	227,4	8,9
Inserção *D_ITEMS*	538,44	78498,1	280,18	7,7
Inserção *PRESCRIPTIONS*	798,57	182445,6	427,14	5,5
Inserção *INPUTEVENTS_MV*	862,19	217830,3	466,72	5,1
Busca *PATIENTS*	0,02	0,000061	0,0078	35

Tabela 60 – Resultados para 10.000 transações e 35 req/s com o *CouchDB*

Operação	Latência Média (s)	Variância	Desvio Padrão	Vazão (TPS)
Inserção *PATIENTS*	443,06	53114,5	230,47	9,1
Inserção *D_ITEMS*	552,91	79710,38	282,33	7,8
Inserção *PRESCRIPTIONS*	771,19	166727,2	408,32	5,8
Inserção *INPUTEVENTS_MV*	864,38	213469,6	462,03	5,2
Busca *PATIENTS*	0,02	0,000045	0,0067	40

Tabela 61 – Resultados para 10.000 transações e 40 req/s com o *CouchDB*

Operação	Latência Média (s)	Variância	Desvio Padrão	Vazão (TPS)
Inserção *PATIENTS*	470,14	59411,98	243,75	8,9
Inserção *D_ITEMS*	572,9	87864,11	296,42	7,7
Inserção *PRESCRIPTIONS*	796,42	181664,2	426,22	5,7
Inserção *INPUTEVENTS_MV*	899,64	230508,3	480,11	5,1
Busca *PATIENTS*	0,02	0,000034	0,0059	45

Tabela 62 – Resultados para 10.000 transações e 45 req/s com o *CouchDB*

Operação	Latência Média (s)	Variância	Desvio Padrão	Vazão (TPS)
Inserção *PATIENTS*	505,3	63133,11	251,26	8,7
Inserção *D_ITEMS*	695,92	125999,6	354,96	6,7
Inserção *PRESCRIPTIONS*	854,51	204474,3	452,19	5,4
Inserção *INPUTEVENTS_MV*	899,18	222980,6	472,21	5,2
Busca *PATIENTS*	0,02	0,000195	0,014	50

Tabela 63 – Resultados para 10.000 transações e 50 req/s com o *CouchDB*

Os valores obtidos nas execuções com 12.000 entradas são mostradas nas Tabelas 64, 65, 66 e 67 para taxa de chegada de 35, 40, 45 e 50 respectivamente.

Operação	Latência Média (s)	Variância	Desvio Padrão	Vazão (TPS)
Inserção *PATIENTS*	544,07	83703,61	289,32	8,6
Inserção *D_ITEMS*	654,09	114891,4	338,96	7,7
Inserção *PRESCRIPTIONS*	869,91	224828,1	474,16	5,9
Inserção *INPUTEVENTS_MV*	882,83	232305,7	481,98	5,8
Busca *PATIENTS*	0,02	0,0053	0,0044	35

Tabela 64 – Resultados para 12.000 transações e 35 req/s com o *CouchDB*

Operação	Latência Média (s)	Variância	Desvio Padrão	Vazão (TPS)
Inserção *PATIENTS*	550,2	83162,32	288,38	8,9
Inserção *D_ITEMS*	677,34	123448,5	351,35	7,6
Inserção *PRESCRIPTIONS*	883,4	234796,6	484,56	5,9
Inserção *INPUTEVENTS_MV*	922,3	256467,6	506,43	5,7
Busca *PATIENTS*	0,02	0,000078	0,0088	40

Tabela 65 – Resultados para 12.000 transações e 40 req/s com o *CouchDB*

Operação	Latência Média (s)	Variância	Desvio Padrão	Vazão (TPS)
Inserção *PATIENTS*	582,96	94889,55	308,04	8,6
Inserção *D_ITEMS*	704,67	131499,8	362,63	7,5
Inserção *PRESCRIPTIONS*	891,47	237897,7	487,75	5,9
Inserção *INPUTEVENTS_MV*	911,94	247385,8	497,38	5,9
Busca *PATIENTS*	0,02	0,000405	0,0201	45

Tabela 66 – Resultados para 12.000 transações e 45 req/s com o *CouchDB*

Operação	Latência Média (s)	Variância	Desvio Padrão	Vazão (TPS)
Inserção *PATIENTS*	610,33	97814,58	312,75	8,6
Inserção *D_ITEMS*	695,16	126176,1	355,21	7,8
Inserção *PRESCRIPTIONS*	895,18	234668,6	484,43	6
Inserção *INPUTEVENTS_MV*	926,82	256983	506,93	5,8
Busca *PATIENTS*	0,02	0,000074	0,0086	50

Tabela 67 – Resultados para 12.000 transações e 50 req/s com o *CouchDB*

Por último, as Tabelas enumeradas de 68 a 71 apresentam os resultados da latência e vazão das operações realizadas com 15.000 entradas e as frequências de requisições avaliadas neste experimento.

Operação	Latência Média (s)	Variância	Desvio Padrão	Vazão (TPS)
Inserção *PATIENTS*	661,11	125257,1	353,92	8,8
Inserção *PRESCRIPTIONS*	1081,64	346908,5	588,99	5,9
Inserção *INPUTEVENTS_MV*	1255,3	453421,7	673,37	5,3
Busca *PATIENTS*	0,02	0,000033	0,0058	35

Tabela 68 – Resultados para 15.000 transações e 35 req/s com o *CouchDB*

Operação	Latência Média (s)	Variância	Desvio Padrão	Vazão (TPS)
Inserção *PATIENTS*	738,56	155590,4	394,45	8,3
Inserção *PRESCRIPTIONS*	1116,3	368996,9	607,45	5,9
Inserção *INPUTEVENTS_MV*	1275,34	457819,2	676,62	5,3
Busca *PATIENTS*	0,02	0,000029	0,0054	40

Tabela 69 – Resultados para 15.000 transações e 40 req/s com o *CouchDB*

Operação	Latência Média (s)	Variância	Desvio Padrão	Vazão (TPS)
Inserção *PATIENTS*	764,17	169853,3	412,13	8,2
Inserção *PRESCRIPTIONS*	1137,24	377394,8	614,32	5,9
Inserção *INPUTEVENTS_MV*	1294,71	470347,6	685,82	5,3
Busca *PATIENTS*	0,02	0,000034	0,0059	45

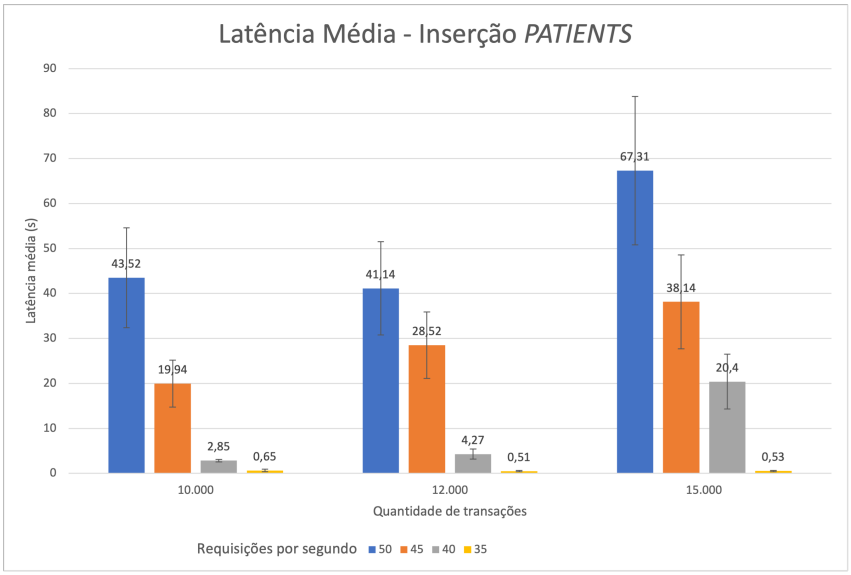
Tabela 70 – Resultados para 15.000 transações e 45 req/s com o *CouchDB*

Operação	Latência Média (s)	Variância	Desvio Padrão	Vazão (TPS)
Inserção *PATIENTS*	762,28	161992,7	402,48	8,5
Inserção *PRESCRIPTIONS*	1169,47	400041,4	632,49	5,8
Inserção *INPUTEVENTS_MV*	1175,05	403003	634,83	5,8
Busca *PATIENTS*	0,02	0,000048	0,0069	50

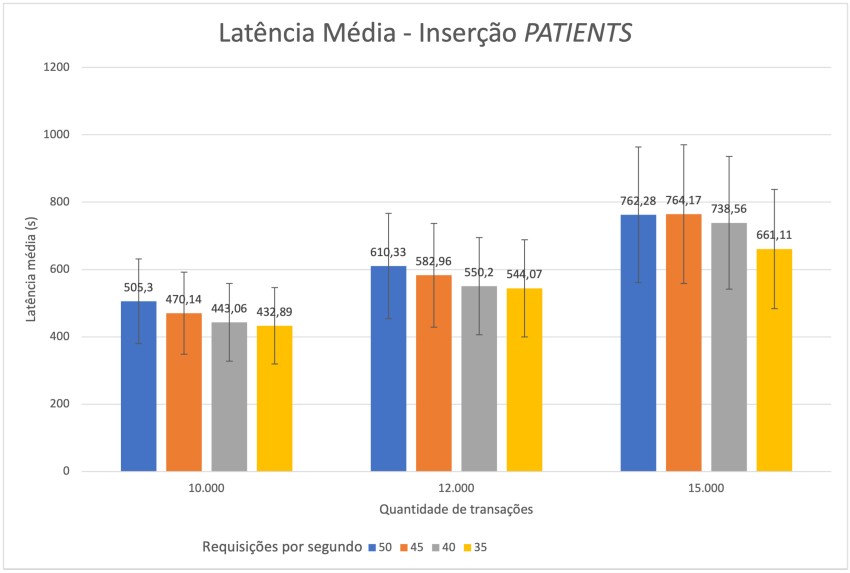
Tabela 71 – Resultados para 15.000 transações e 50 req/s com o *CouchDB*

Os gráficos apresentados nesta Seção mostram o desempenho dz *blockchain* na inserção dos quatro arquivos de dados (*PATIENTS*, *D_ITEMS*, *PRESCRIPTIONS* e *INPUTEVENTS_MV*) e na busca com o arquivo *PATIENTS*.

Os gráficos foram colocados lado a lado para facilitar a comparação dos valores das latências médias e da vazão considerando os mesmos valores de frequência de requisições e quantidade de entradas. As Figuras 23 a 27 mostram as comparações das latências médias entre as inserções e a busca sem e com o uso do banco de dados *CouchDB*. A barra de erro desses gráficos mostram o desvio padrão encontrado. Já as Figuras 28 a 32 têm como objetivo mostrar a vazão das mesmas operações.



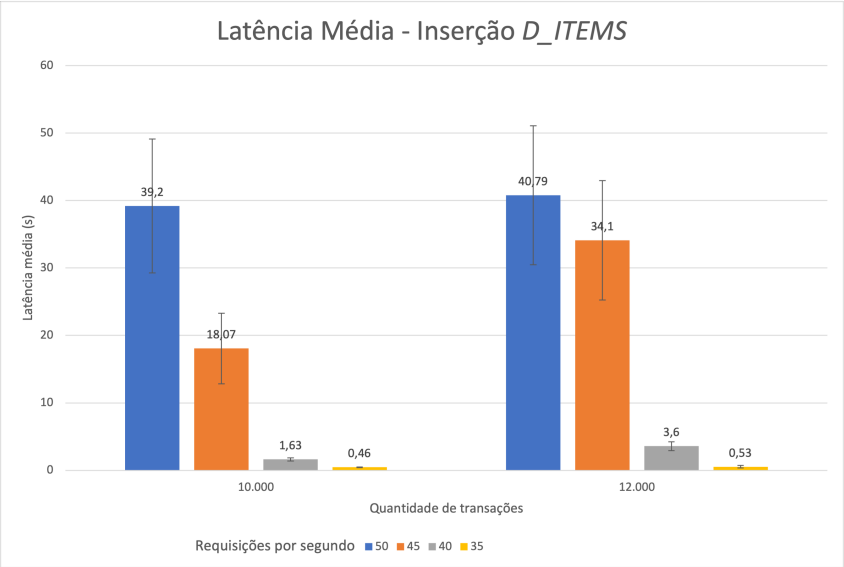
(a) Latência sem o *CouchDB*



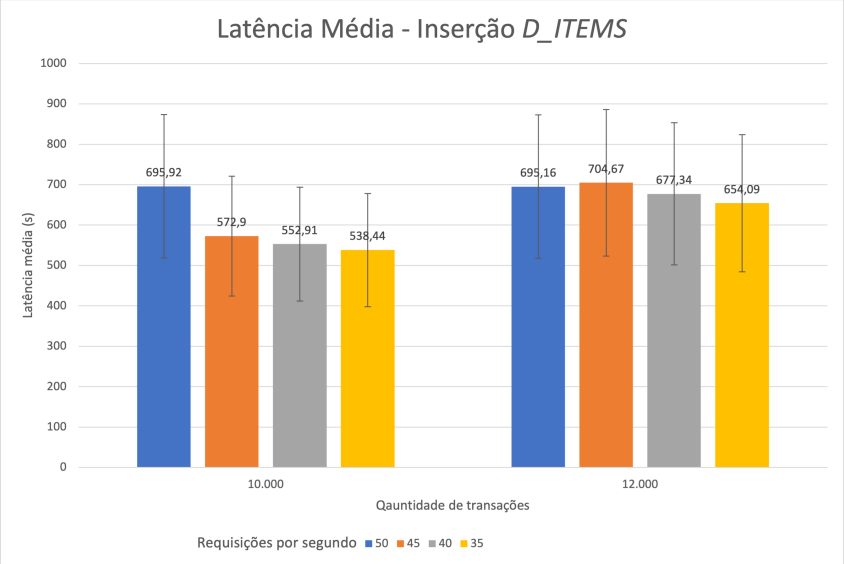
(b) Latência com o *CouchDB*

Figura 23 – Resultados para inserção de *PATIENTS*

A Figura 23a mostra os resultados encontrados para a latência sem o uso do banco de dados da inserção de *PATIENTS* e a Figura 23b evidencia os resultados com a inserção dos dados na *blockchain* associado ao *CouchDB*. Todos os gráficos desta Seção seguem esse formato. A comparação das Figuras (a e b) para cada inserção de dados na rede, neste experimento, evidencia que o uso do banco de dados impactou o desempenho da *blockchain* nessas operações de inserção, influenciando fortemente o aumento da latência.



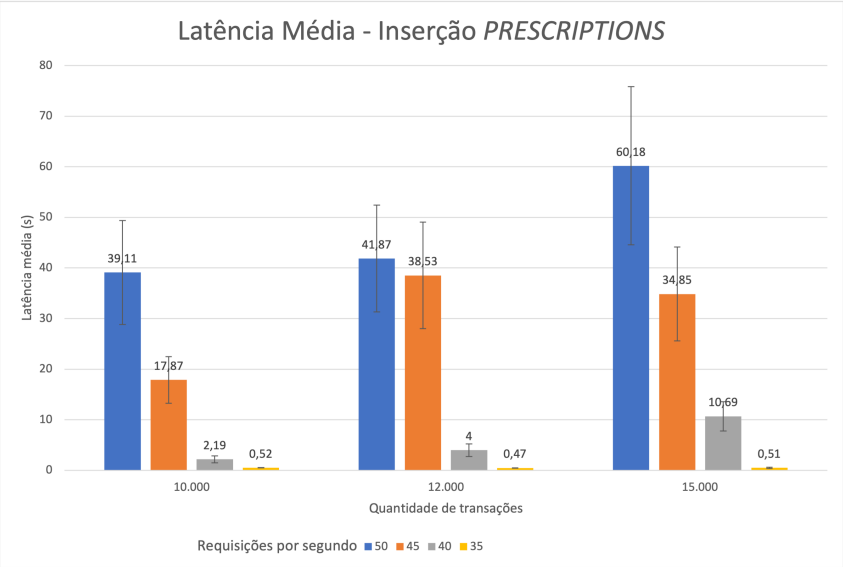
(a) Latência sem o *CouchDB*



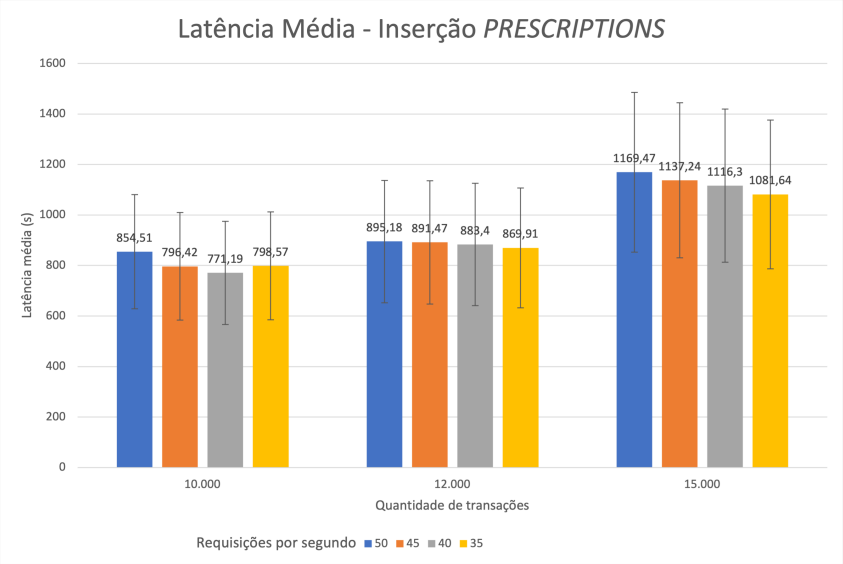
(b) Latência com o *CouchDB*

Figura 24 – Resultados para inserção de *D_ITEMS*

Sem o uso do banco de dados, a taxa de chegada das requisições mostrou ter influência direta na latência média de atendimento das requisições de inserção, ou seja, o aumento desse fator gerou uma alta no tempo de resposta. Para o armazenamento associado com o banco de dados, a taxa de chegada das requisições de inserção tem um peso menor na variação das latências médias. Pode-se observar neste experimento que em alguns casos, frequências menores apresentam maiores latências para a mesma quantidade de transações. Isso ocorre, por exemplo no gráfico de inserção do arquivo *D_ITEMS* (Figura 24b) em que a latência para taxa de 45 com 12.000 entradas é maior que a taxa de 50. Estes resultados sugerem a dominância que acesso ao banco de dados tem no desempenho do sistema em comparação com os demais fatores avaliados.



(a) Latência sem o *CouchDB*

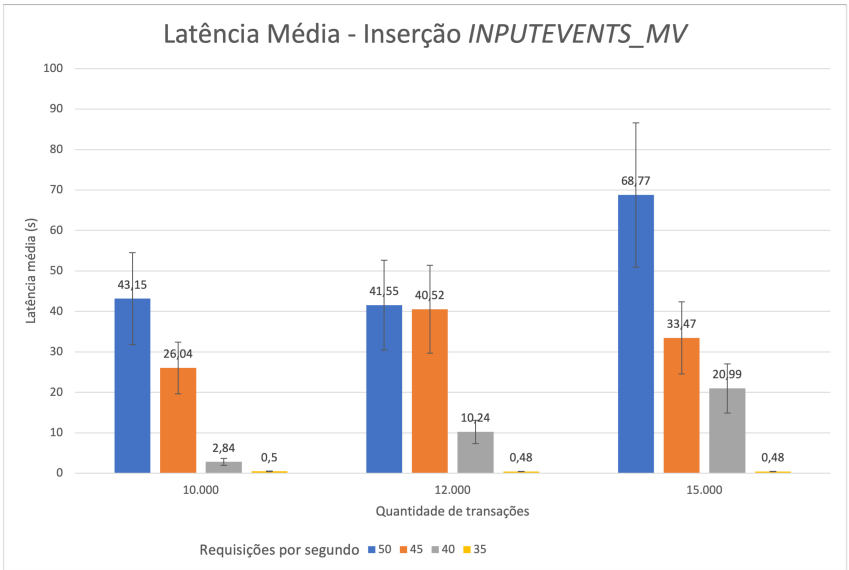


(b) Latência com o *CouchDB*

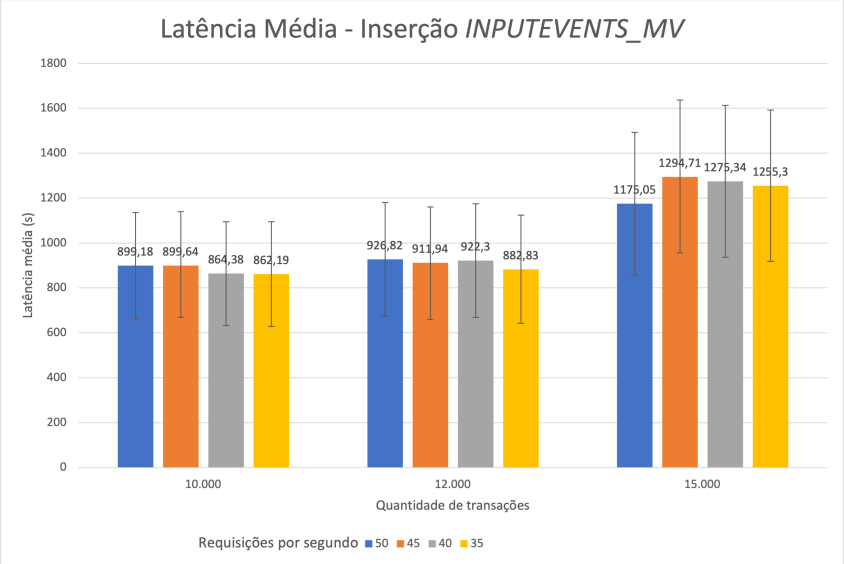
Figura 25 – Resultados para inserção de *PRESCRIPTIONS*

O acesso ao banco de dados nas inserções impactam diretamente a latência das inserções nas quais são realizadas indexações. As Figuras 23a e 25a mostram os resultados das inserções sem o *CouchDB*, respectivamente, em arquivos sem indexação (como *PATIENTS*) e com cinco indexações (como *PRESCRIPTIONS*). Nessas Figuras, observa-se que há pouca variação na latência média, pois as indexações não são realizadas em virtude dos dados não estarem sendo tratados pelo *CouchDB*.

Para as inserções com a *blockchain* associado ao *CouchDB* (Figuras 23b e 25b) a indexação das informações é realizada e a diferença entre os valores encontrados fica evidente, comprovando o impacto dessa etapa da inserção na latência do sistema.

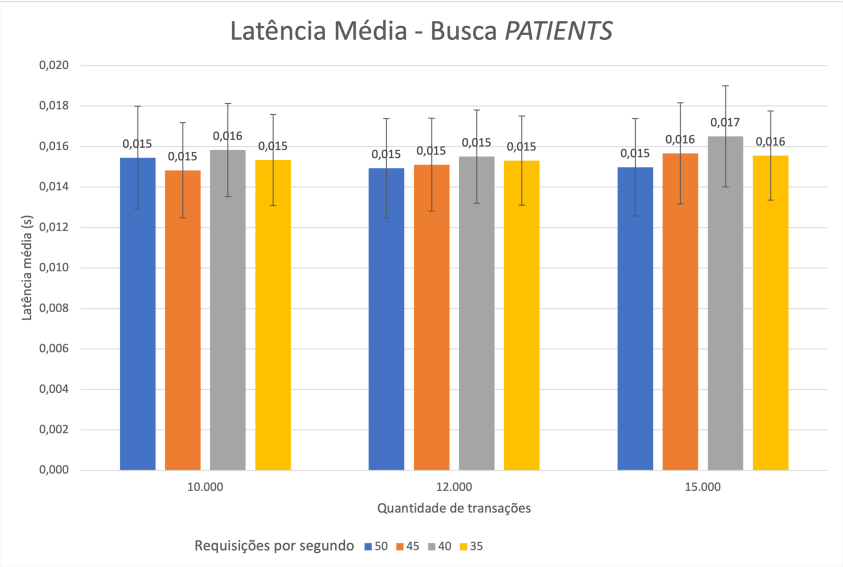


(a) Latência sem o *CouchDB*

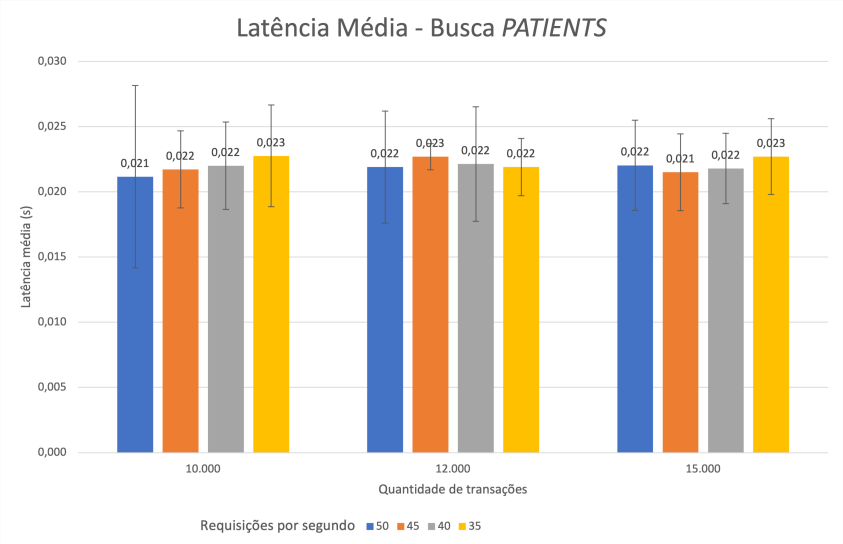


(b) Latência com o *CouchDB*

Figura 26 – Resultados para inserção de *INPUTEVENTS_MV*



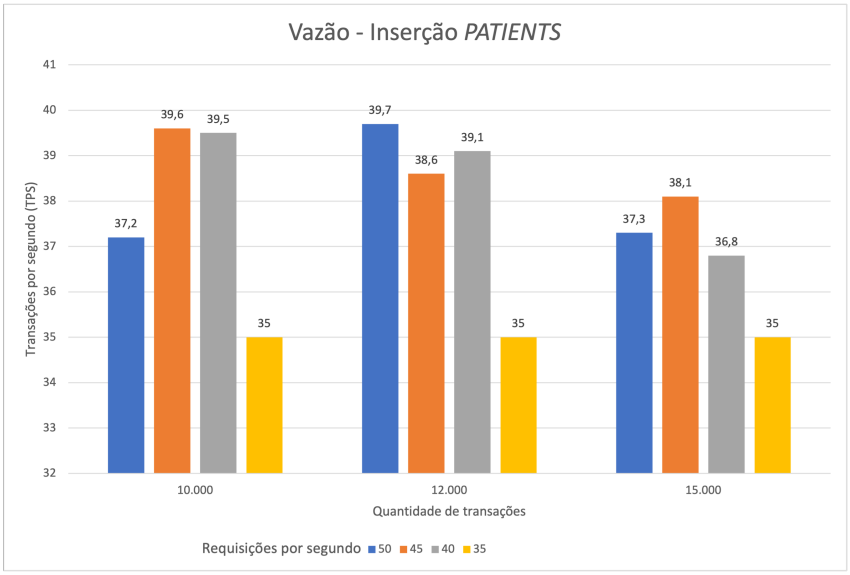
(a) Latência sem o *CouchDB*



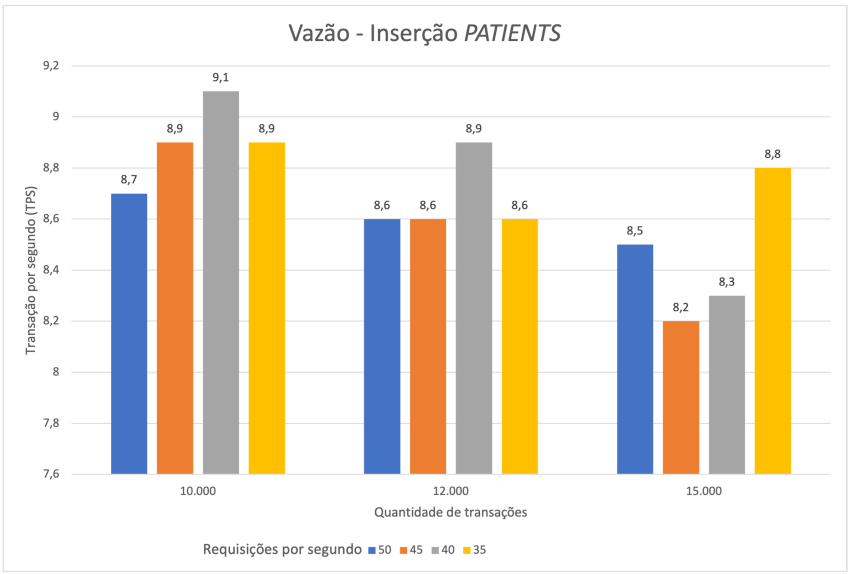
(b) Latência com o *CouchDB*

Figura 27 – Resultados para busca de *PATIENTS*

Ao contrário das inserções, a busca do arquivo *PATIENTS* (Figura 27) obteve pouca variação da latência comparando as duas formas de armazenamento. A pequena variação na latência, em ambos cenários de armazenamento, não é permite estabelecer se algum dos fatores (quantidade de entradas e taxa de chegada de requisições) foi determinante para essa diferença.



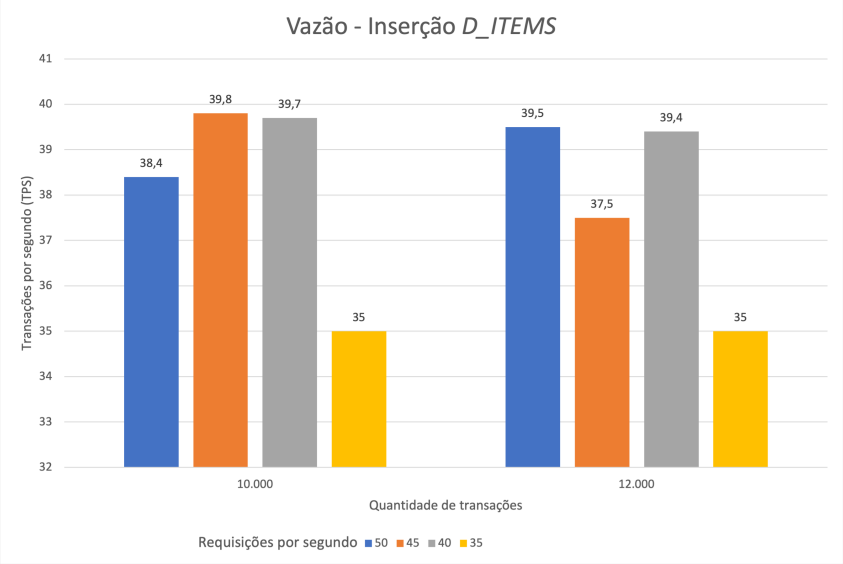
(a) Vazão sem o *CouchDB*



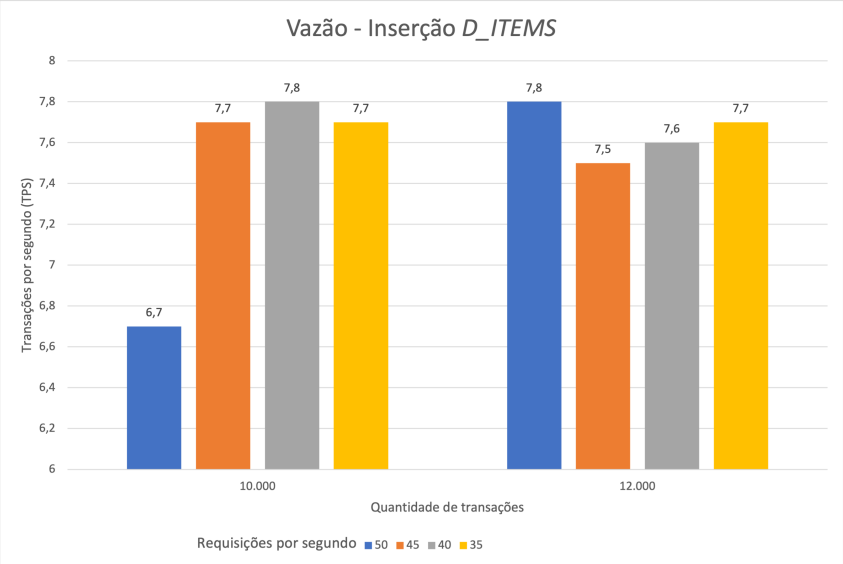
(b) Vazão com o *CouchDB*

Figura 28 – Resultados para inserção de *PATIENTS*

Os gráficos das Figuras 28 a 32 mostram os valores encontrados para a vazão das operações de inserção e busca realizadas. Para as inserções sem o *CouchDB* os gráficos das Figuras 28a a 31a mostram que vazão das requisições de inserção, com as quantidades de entrada testada, têm um limite superior para o atendimento das requisições próximo ou no máximo de 40 transações por segundo. O aumento da frequência de chegada das requisições de 40 para 45 ou 50 não melhora a vazão, mas a latência dessa operação é afetada, diminuindo o desempenho total da rede.



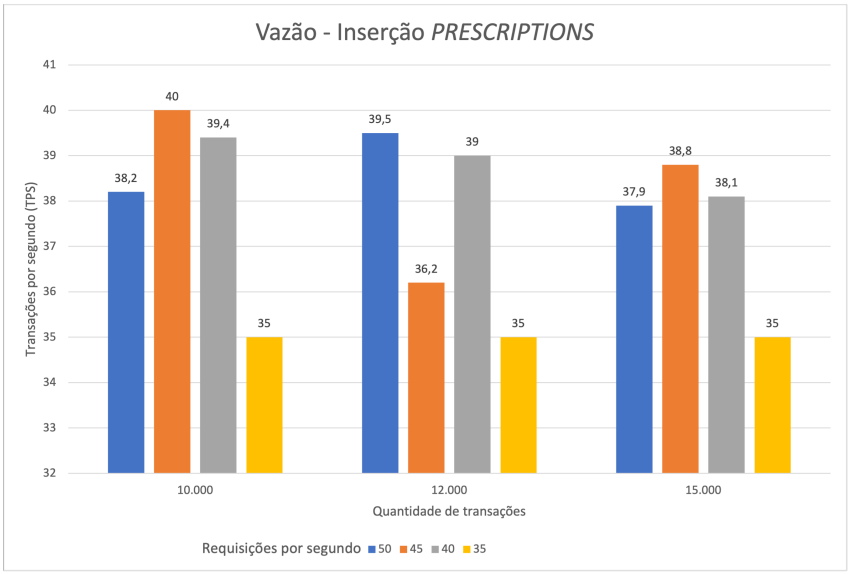
(a) Vazão sem o *CouchDB*



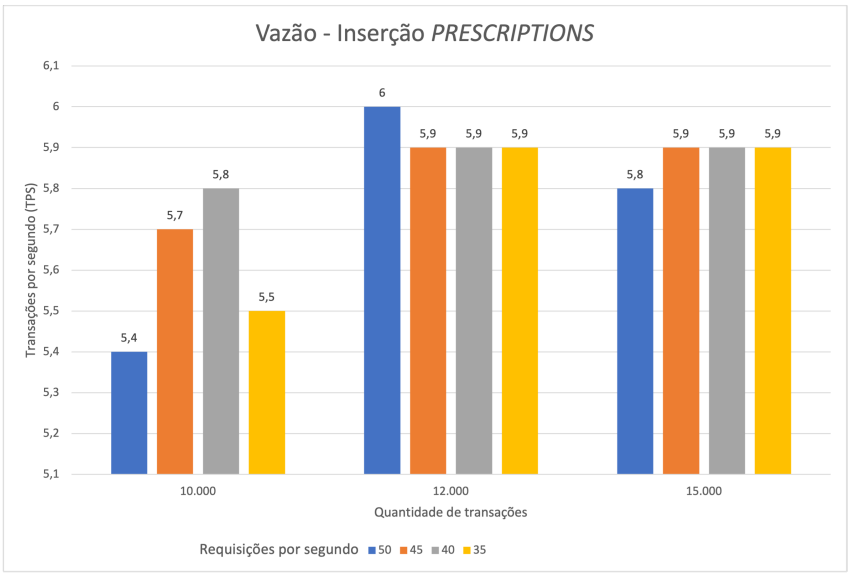
(b) Vazão com o *CouchDB*

Figura 29 – Resultados para inserção de *D_ITEMS*

A vazão das inserções com o banco de dados são ilustradas nas Figuras 28b a 31b. Através dos gráficos é possível notar que o limite superior dessa métrica para as inserções é próximo de nove, um valor abaixo das frequências de requisições utilizadas neste experimento.

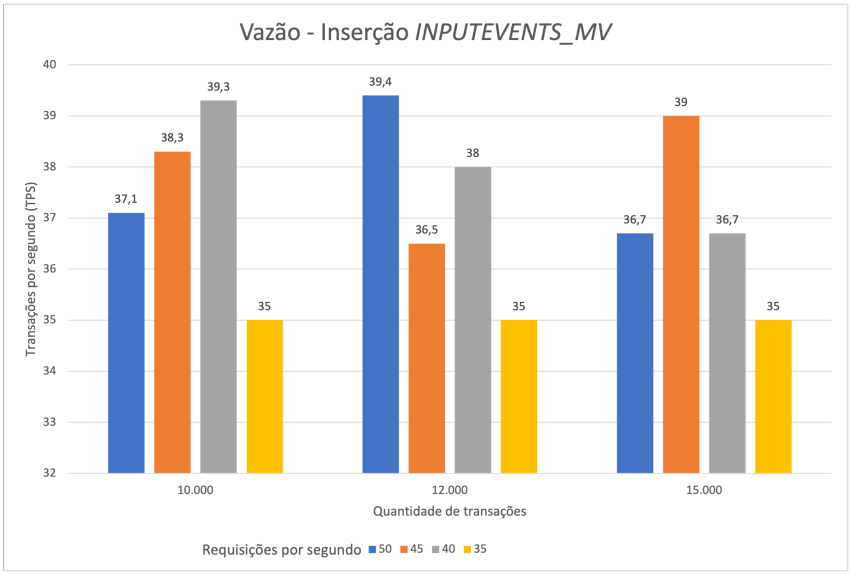


(a) Vazão sem o *CouchDB*

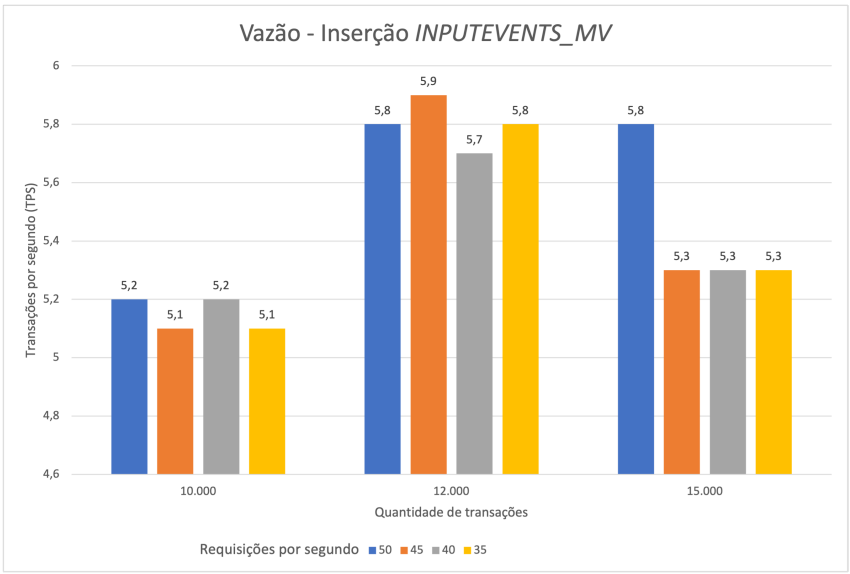


(b) Vazão com o *CouchDB*

Figura 30 – Resultados para inserção de *PRESCRIPTIONS*

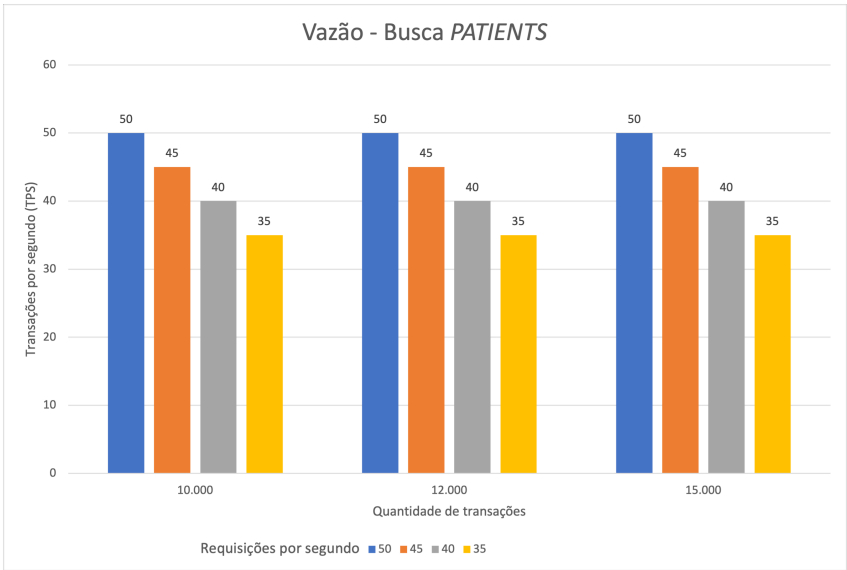


(a) Vazão sem o *CouchDB*

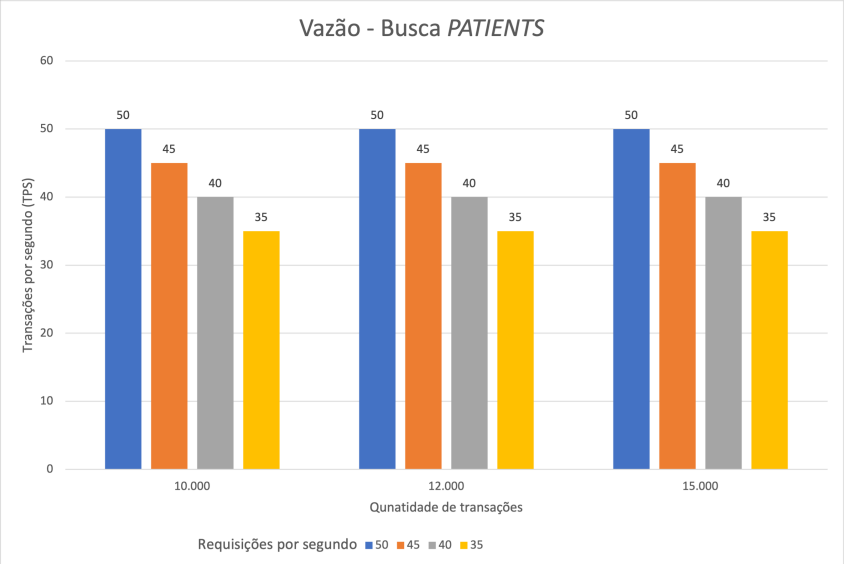


(b) Vazão com o *CouchDB*

Figura 31 – Resultados para inserção de *INPUTEVENTS_MV*



(a) Vazão sem o *CouchDB*



(b) Vazão com o *CouchDB*

Figura 32 – Resultados para busca de *PATIENTS*

A operação de busca não sofreu alterações comparando sem e com o *CouchDB*. Os valores encontrados são iguais às taxas de chegada das requisições de leitura, mostrando que as requisições são atendidas na mesma frequência com que chegam na rede, independente da forma de armazenamento.

6.6 Análise dos Resultados

6.6.1 Hipótese I

A primeira hipótese se refere ao desempenho da *blockchain* considerando as diferentes formas de armazenamento, com e sem a associação ao banco de dados externo. A hipótese afirma que o uso do banco de dados impacta negativamente o desempenho da rede, sob as métricas de latência e vazão.

A veracidade desta hipótese é investigada comparando as latências e vazões encontradas nos cenários de execução com e sem o *CouchDB* para cada operação realizada. Para a latência, foi verificada a normalidade de cada amostra de dados. O teste de *Cramér-von Mises* (THODE, 2002) com nível de significância 95% confirma a não-normalidade do conjunto de dados, os quais apresentam *p-valor* menor do que 0,05.

Dada a não-normalidade, o teste de *Wilcoxon* (WALPOLE *et al.*, 1993) com nível de significância 95% foi aplicado para verificar se as amostras são estatisticamente equivalentes ou não. Os *p-valores* encontrados comparando as latências nas execuções das inserções de *PATIENTS*, *D_ITEMS*, *PRESCRIPTIONS* e *INPUTEVENTS_MV*, e na busca com *PATIENTS* são apresentados nas Tabelas 72, 73, 74, 75 e 76, respectivamente.

Transações	**35**	**40**	**45**	**50**
10.000	0	0	0	0
12.000	0	0	0	0
15.000	0	0	0	0

Tabela 72 – *p-valores* entre a inserção *PATIENTS* com e sem *CouchDB*

Transações	**35**	**40**	**45**	**50**
10.000	0	0	0	0
12.000	0	0	0	0
15.000	0	0	0	0

Tabela 73 – *p-valores* entre a inserção *D_ITEMS* com e sem *CouchDB*

Transações	**35**	**40**	**45**	**50**
10.000	0	0	0	0
12.000	0	0	0	0
15.000	0	0	0	0

Tabela 74 – *p-valores* entre a inserção *PRESCRIPTIONS* com e sem *CouchDB*

Transações	35	40	45	50
10.000	0	0	0	0
12.000	0	0	0	0
15.000	0	0	0	0

Tabela 75 – *p-valores* entre a inserção *INPUTEVENTS_MV* com e sem *CouchDB*

Transações	35	40	45	50
10.000	0	0	0	0
12.000	0	0	0	0
15.000	0	0	0	0

Tabela 76 – *p-valores* entre a busca *PATIENTS* com e sem *CouchDB*

Uma vez que todos os *p-valores* encontrados são menores que 0,05 pode-se afirmar que as amostras de dados não são equivalentes. Com isso, observa-se as latências médias obtidas nas execuções realizadas neste experimento. Como pode-se observar com os gráficos apresentados na Seção anterior, as latência médias das inserções com o banco de dados são superiores aos valores obtidos sem o banco de dados. A vazão, por outro lado, diminui comparando os cenários testados. Dessa forma, valida-se essa hipótese para a inserção de dados na rede *blockchain*.

A menor diferença na latência sem e com o uso do *CouchDB* foi de 1.032,49% para a inserção de *PATIENTS* com 15.000 entradas e frequência de 50 requisições por segundo (comparando os gráficos das Figuras 23 (a) e (b)). A maior diferença chegou a 261.420,83% para a inserção de *INPUTEVENTS_MV* com 15.000 e taxa de chegada de 35 (gráficos das Figuras 26 (a) e (b)).

Já para a vazão, a maior queda, comparando o armazenamento sem o banco de dados e com o banco de dados, é da inserção de *INPUTEVENTS_MV* (Figura 31) para 10.000 entradas e taxa de requisição de 40 com redução em 86,77%. A menor redução da vazão é na inserção de *PATIENTS* (Figura 28) em que a vazão foi reduzida em 74,57% para execução com 10.000 entradas e frequência de 35.

É necessário destacar quais arquivos apresentam as maiores e menores diferenças entre a inserção considerando o banco de dados ou não. O arquivo *PATIENTS* além de ser a menor, não demanda indexações de informações antes de ser armazenada no banco de dados, pois apresenta uma variável que permite identificação única de cada entrada do arquivo. Já o arquivo *INPUTEVENTS_MV* foi escolhido justamente por ter o maior número de colunas dentre os arquivos disponibilizadas pelo *MIMIC-III*, totalizando 31. As particularidades desses arquivos permitem afirmar que a quantidade de dados na inserção impacta diretamente o desempenho da rede *blockchain* quando considerado o banco de dados *CouchDB*.

Essa diferença na latência e vazão nas inserções entre as formas de armazenamento se deve ao acesso externo do *CouchDB*. Como mostrado nas Figuras 21 e 22 o uso do banco de dados acrescenta um *Docker* contêiner à rede do *Hyperledger Fabric*. A cada bloco distribuído pela rede, além da informação na *blockchain*, dados também são armazenadas no *CouchDB* (como descrito na Seção 2.5.2).

Diferentemente das inserções, a busca não demonstra uma grande variação na latência média entre as duas formas de armazenamento das latências médias nos ambientes de teste executados, apesar das amostras serem estatisticamente não equivalentes. As vazões da busca sem e com o banco de dados permanecem as mesmas, sendo limitadas pela frequência de chegada das requisições. Com esse resultado é possível concluir que a escolha arquitetural do *Fabric* em usar dois componentes de armazenamento na cadeia, o *world state database* e a *blockchain*, melhora o desempenho da busca. Isso se deve às informações serem armazenadas como um retrato da cadeia no *world state*. Dessa forma não há necessidade de percorrer a cadeia ou banco de dados para recuperar informações, tornando possível o resultado encontrado de atendimento das requisições na mesma frequência com que elas ocorrem independente da forma de armazenamento.

6.6.2 Hipótese II

A segunda hipótese proposta neste experimento afirma que o uso do banco de dados como forma de armazenamento na *blockchain* influencia distintamente as duas operações consideradas na arquitetura do *Hyperledger Fabric*, a *update* e a *read*. A primeira se caracteriza por modificar a *blockchain*, que são as inserções realizadas. Já a segunda não modifica as informações, apenas busca uma informação contida na *blockchain*, como a busca executada.

A veracidade desta hipótese é apurada comparando os valores das métricas coletadas (latência e vazão) obtidos na inserção dos arquivo do *MIMIC-III PATIENTS* e na busca com o mesmo arquivo. A normalidade das amostras da latência foram testadas para o experimento anterior, no qual foi comprovada a não-normalidade do conjunto de dados através do teste de *Cramér-von Mises* (THODE, 2002), os quais apresentam *p-valor* menor do que 0,05.

Dado a não-normalidade, o teste de *Wilcoxon* (WALPOLE *et al.*, 1993) com nível de significância 95% foi aplicado para verificar se as amostras são estatisticamente equivalentes ou não. Os *p-valores* encontrados comparando as latências nas execuções da inserção e da busca de *PATIENTS* é descrita na Tabela 77.

Transações	**35**	**40**	**45**	**50**
10.000	0	0	0	0
12.000	0	0	0	0
15.000	0	0	0	0

Tabela 77 – *p-valores* entre a inserção e a busca de *PATIENTS* com o *CouchDB*

Dado que todos os *p-valores* encontrados são menores que 0,05 pode-se afirmar que as amostras de dados não são equivalentes. Com isso, observa-se as latências médias obtidas nas execuções realizadas neste experimento.

Os resultados deste experimento mostram que, para todos os cenários executados, o tempo de latência das operações de inserção são superiores aos encontrados na busca. Isso era esperado, pois a inserção necessita de um processo de validação de três etapas para ser aceita pela rede e inserida na *blockchain*. A busca, por sua vez, não necessita de validação dos nós da rede dentro da arquitetura do *Fabric*.

Além da forma de armazenamento, este experimento também variou a quantidade de requisições realizadas por rodada e taxa com as quais essas transações chegavam na rede. A análise dos resultados mostra que a latência média das operações de inserção sem o banco de dados é influenciada diretamente pela taxa de chegada dessas requisições. Embora também impacte na latência das operações, a quantidade de transações teve, neste experimento, um peso menor na variação dessa métrica. A vazão das inserções nesse cenário também não demonstra relação direta com o aumento da quantidade de requisições, mas sim com a taxa de chegada dessas requisições. Nos cenários executados, a vazão das inserções sem o banco de dados se limita a 40 requisições por segundo mesmo com taxas de chegada maiores e tem o limite inferior de 35 requisições por segundo, indicando que a frequência de requisições ideal neste experimento se encontra neste intervalo.

A vazão das inserções com banco de dados tem pouca variação nos valores encontrados, não evidenciando relação direta com a quantidade de requisições ou com frequência de chegada de requisições. Esse resultado pode ser justificado pelo limite máximo dessa métrica para cada inserção ser inferior às taxas de chegada testadas, ou seja, as taxas de chegada de requisições são superiores a taxa de atendimento da rede. Com o uso do *CouchDB* pode-se observar que a latência foi influenciada pelo o número total de transações em cada rodada de teste, e a frequência das requisições tendo pouco impacto. Esse resultado também pode ser justificado pela sobrecarga no sistema devido às altas taxas de chegada utilizadas, de forma que o aumento da quantidade de transações evidencia esse cenário.

6.7 Considerações Finais

Este Capítulo descreve o experimento realizado para comparar as duas formas de armazenamento com a infraestrtura da *blockchain Hyperledger Fabric*. Os valores encontrados nas métricas selecionadas (latência média e vazão), revelam o impacto do uso do banco de dados *CouchDB* em conjunto com a *blockchain*. É possível afirmar que a inserção de informações na rede foi a operação mais afetada neste experimento, principalmente em termos de latência média, que alcançou diferenças percentuais consideráveis entre as formas de armazenamento.

O Capítulo a seguir descreve o experimento que analisa o desempenho da aplicação

blockchain quando modificados os parâmetros da infraestrutura do *Hyperlegder Fabric*. Dois dos fatores analisados foram a política de validação e a quantidade de nós validadores, que em conjunto determinam quais e quantos nós validam uma transação antes desta ser inserida na *blockchain*. Também foi verificado o impacto do tamanho do bloco e seu tempo de criação nas medidas de desempenho.

CAPÍTULO

7

EXPERIMENTO 3 - VALIDAÇÃO DE TRANSAÇÕES E CRIAÇÃO DO BLOCO

7.1 Considerações Iniciais

O terceiro experimento é apresentado neste Capítulo. Esse experimento visa analisar características do processo de validação de transações e criação de blocos. A maneira como o *Hyperledger Fabric* resolve o consenso e criação de blocos, questões intrínsecas das redes *blockchain*, o diferencia de outras soluções, como o *Bitcoin* e o *Etheruem*. Essa distinção ocorre pois o *Fabric* introduz uma proposta diferente da *blockchain* com o uso de redes privadas.

O experimento é descrito em cinco partes, que também são Seções neste Capítulo. A primeira apresenta o escopo do experimento, é nela que os objetivo do estudo é definido. A segunda é o planejamento e mostra as hipóteses levantadas para este experimento e os fatores do *Fabric* analisados. A terceira descreve a infraestrutura utilizada de rede e máquinas para a execução do experimento. As tabelas e s gráficos com os resultados encontrados são demonstrados na quarta. Na quinta uma análise com base nas hipóteses de pesquisa é apresentada.

7.2 Escopo

Este experimento avalia como a quantidade de nós validadores, em conjunto com a política de validação, impacta o desempenho da rede *blockchain* usada em conjunto com um banco de dados. Outras características do *Hyperlegder Fabric* investigadas nesse experimento são as variáveis que determinam o tamanho e o tempo de criação de um bloco. Neste experimento, o objeto de estudo continua sendo o *Fabric* na versão 1.4.1 associado ao banco de dados *CouchDB*.

O foco qualitativo deste experimento é analisar o comportamento da *blockchain* com o *Hyperlegder Fabric* ao se alterar características do processo de validação e criação de blocos em

um cenário com o uso de dados médicos reais.

7.3 Planejamento

Assim como nos experimentos anteriores, as execuções ocorreram com as máquinas e a rede do laboratório *LaSDPC-ICMC*. Os dados utilizados são os disponibilizados pela base de dados do *MIMIC-III* (JOHNSON *et al.*, 2016), na mesma formatação descrita no Capítulo 4.

Os experimentos executados pretendem analisar três hipóteses relacionadas ao desempenho da *blockchain* junto com banco de dados para a manipulação de dados heterogêneos.

1. A primeira hipótese afirma que diminuir a quantidade de nós necessários para validar uma transação melhora o desempenho das inserções de dados em termos de latência e vazão.
2. A segunda hipótese declara que criar blocos maiores piora o desempenho da rede quanto à latência e melhora o desempenho em relação à vazão do sistema.
3. A terceira hipótese assume que aumentar o tempo de criação de um bloco piora o desempenho do sistema no que tange a latência e melhora o desempenho quanto a vazão.

Essas hipóteses foram definidas com base na arquitetura do *Fabric*. A política de validação e a quantidade de nós validadores determinam quantos nós participam do processo de consenso durante a inserção de dados na cadeia e posteriormente no *CouchDB*, assim, a análise desses dois fatores foi agrupada em uma hipótese de pesquisa. Na Seção 7.4 é descrita a disposição desses nós nas execuções realizadas.

A segunda e a terceira hipótese são relacionadas ao processo de criação do bloco na rede do *Hyperledger Fabric*. Nessa plataforma a criação dos blocos é feita de modo determinístico pelo *ordering service* e depende do tamanho do bloco estabelecido e o tempo disponibilizado para a criação.

Para validar as hipóteses levantadas, foram especificadas as seguintes questões de pesquisa:

QP1: Qual é o desempenho da rede *blockchain* ao diminuir a quantidade de nós validadores?

QP2: Qual é o desempenho da rede *blockchain* ao aumentar o tamanho dos blocos?

QP3: Qual é o desempenho da rede *blockchain* ao aumentar o tempo de criação dos blocos?

Latência e vazão são as métricas utilizadas para avaliar o desempenho da *blockchain*. Essas foram calculadas com o uso do *benchmark Hyperledger Caliper* considerando os diferentes tipos de transações, como ilustrados nas Equações de 4.1 a 4.4.

Os fatores e níveis utilizados neste experimento são descritos na Tabela 78. Os níveis relacionados ao tamanho do bloco foram estabelecidos com base nos códigos exemplos encontrados na arquitetura do *Caliper* e do *Fabric*. O primeiro sugere o uso do tamanho do bloco definido com 10 transações, já o segundo em 500 transações pertencentes ao bloco.

Fatores	**Níveis**	
Política de validação	AND	OR
Quantidade de nós validadores	2	4
Tamanho do bloco	10	500
Tempo de criação do bloco	1s	10s

Tabela 78 – Fatores e níveis Experimento 3

Para este experimento, a quantidade de transações foi fixada em 10.000 entradas e taxa de chegada de 10 requisições por segundo para cada rodada de execução. O total de transações realizadas é a multiplicação dos níveis de cada fator examinado (16 ao todo) com a quantidade de operações (4 inserções e uma busca), gerando 800.000 transações sobre a *blockchain*.

7.4 Infraestrutura para Execução

O *Hyperledger Fabric* adiciona o conceito de *ordering service*, responsável por ordenar as transações, criar o bloco e transmitir para a rede. O *ordering service* cria o bloco quando uma de duas condições é alcançada, o tempo limite de espera é atingido (*BatchTimeout*) ou o bloco já atingiu o tamanho estipulado (*MaxMessageCount*). Essas variáveis são fatores desse experimento nomeadas na Seção anterior como tempo de criação do bloco e tamanho do bloco, respectivamente.

A política de validação também é analisada, essa variável pode apresentar dois valores: *OR* e *AND*. A política *OR* define que a transação deve ser validada por pelo menos uma organização, assim, necessitando consumir o recurso de apenas um nó. Já a política *AND* estabelece que uma transação deve ser validada por todas organizações da rede, consumindo o recurso de um nó de cada organização pertencente a rede.

O número de nós validadores também é avaliado e quantifica quantos nós são necessários para validar uma transação. Esse valor é referente à primeira etapa do processo de validação das transações do tipo *update* (detalhada na Seção 2.5.2). Para esta análise foram considerados dois e quatro nós validadores nas execuções realizadas.

A Tabela 79 apresenta os parâmetros que permaneceram inalteráveis durante a execução desse experimento. Os dois primeiros parâmetros são relacionados ao tamanho da rede e a quantidade de contêineres *Docker* utilizados. O terceiro define o tamanho máximo em *byte* do

bloco criado pelo *ordering service* e o quarto o tamanho preferencial do bloco. Essas últimas variáveis são explicadas com mais detalhes na Seção 4.2.

Parâmetro	Valor
Número de máquinas reais	4
Número de contêineres	9
AbsoluteMaxBytes	128 MB
PreferredMaxBytes	128 MB

Tabela 79 – Parâmetros da execução Experimento 3

A rede criada é formada por nove contêineres *Docker*, com cinco atuando como nós na rede *blockchain* com o *Hyperledger Fabric* e quatro hospedando o *CouchDB*. Um dos nós responsáveis pela *blockchain* é o nó *orderer* e os outros quatro são participantes da rede divididos em duas organizações. A carga de trabalho é gerada pelos clientes com o *benchmark Hyperledger Caliper*. A Figura 33 representa a configuração da rede, considerando os clientes, as organizações, o nó *orderer* e o *channel*.

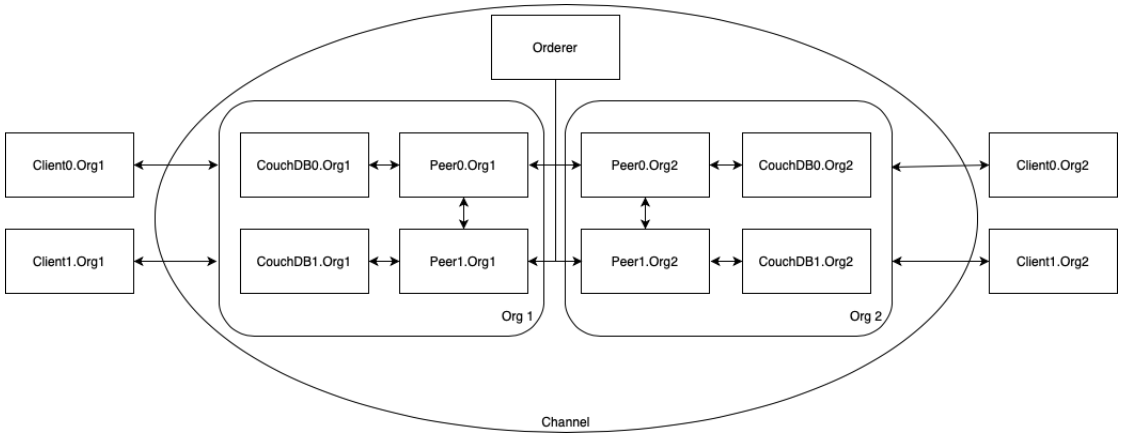


Figura 33 – Configuração da rede Experimento 3

Nas execuções examinadas, quando a política de validação é *AND* e são selecionados quatro nós para validar uma transação, todos os nós participantes da rede (*Peer0.Org1*, *Peer1.Org1*, *Peer0.Org2* e *Peer1.Org2*) são necessários para garantir o consenso. Já quando a política *AND* é usada com dois nós validadores, um dos nós de cada organização é indicado para validação. Nas execuções deste experimento, esta função foi exercida pelos nós *Peer0.Org1* e *Peer0.Org2*.

Para as execuções considerando a política *OR*, quando considerados quatro nós validadores, todos os nós da rede têm a possibilidade de validar a transação e quando configurado com dois, apenas os *Peer0.Org1* e *Peer0.Org2* podem retornar sobre a validade da transação. Nesse cenário é necessário que apenas um dos nós indique para o cliente solicitante se a transação está valida ou não, para assim dar sequência ao processo de inserção da transação na rede *blockchain*.

As máquinas utilizadas estão na mesma rede, do laboratório de pesquisa *LaSDPC*. Os sistemas de *software* extra usados na execução deste experimento são os requisitos de instalação propostos nas documentações do *Hyperldger Fabric* e do *Hyperledger Caliper*.

As características das máquinas usadas estão detalhadas na Tabela 80.

Configurações de *Hardware* (*Cluster Andromeda - LaSDPC*)	
Sistema Operacional	Ubuntu versões 16.04.7 LTS e 20.04 LTS
CPU	QEMU Virtual CPU (512 KB Cache, 4 GHz)
Memória	16 GB RAM
Disco	Seagate HD 2TB ST2000DM001-1ER1

Tabela 80 – Características das máquinas Experimento 3

O ponto de observação utilizado neste experimento foi configurado com o *Hyperledger Caliper*. Esse *benchmark* permite a configuração de um monitor para coleta de informações sobre a situação das transações e na rede além de produzir relatórios de desempenho. Com o *Caliper* também é possível coletar o tempo de execução de cada transação na rede.

Nesse experimento foram reutilizadas as operações adotadas nos experimentos anteriores, isto é quatro inserções e uma busca. A escolha dessas operações foi baseada nas características dos dados encontradas no *MIMIC-III*, explicados na Seção 4.2. Essas operações foram realizadas para avaliar as diferenças na latência e na vazão dos dois tipos de transações executadas pelo *Fabric* (*read* e *update*).

Para cada inserção, os dados são lidos dos arquivos do *MIMIC-III* e entram na primeira etapa do processo de consenso. Nessa fase, cada transação deve ser avaliada pela quantidade de nós selecionada, de acordo com a política de validação e quantidade de nós validadores de cada execução. Após validadas, as requisições são encaminhadas para o nó *orderer* que monta o bloco, considerando o tempo de criação do bloco e o tamanho do mesmo. Posteriormente, o bloco é distribuído pela rede e os dados são armazenados no *CouchDB*. Na busca, o cliente faz a requisição da informação que deseja acessar e um nó da rede responde com o dado solicitado, não havendo as etapas de validação para essa transação.

As operações de inserção de dados são as primeiras a serem executadas, sendo a operação de leitura dependente da inserção prévia do arquivo *PATIENTS*. As diferentes operações são executadas em blocos, em que uma tarefa só é inicializada após a finalização da anterior.

Assim como ocorreu nos demais experimentos, a execução deste não inclui testes de falhas tanto de nós quanto da rede ou ataques do tipo *man in middle*. Todas as transações executadas têm comportamento esperado pela infraestruturada da *blockchain*, não havendo testes para verificar o suporte da rede às transações fora do padrão.

7.5 Resultados

Nesta Seção, os resultados encontrados nas execuções realizadas são apresentados de duas formas, com o uso de tabelas e gráficos. As Tabelas isolam os cenários testados e mostram a latência média, seguida pela sua respectiva variância e desvio padrão, e a vazão para cada operação realizada. Os gráficos destacam os resultados de latência média e vazão encontrados nas execuções separando pela operação, alterando os parâmetros relacionados aos processos de validação e criação dos blocos.

Nas tabelas de 81 a 84 são demostrados as medidas de desempenho coletadas quando o tamanho do bloco é definido em dez e o tempo de criação é estabelecido em um segundo. A política de validação é modificada em *AND* e *OR* e o número de nós validadores é alternado entre dois e quatro.

Operação	Latência Média (s)	Variância	Desvio Padrão	Vazão (TPS)
Inserção *PATIENTS*	67,54	3718,01	60,98	8,4
Inserção *D_ITEMS*	311,88	30748,37	175,35	6,3
Inserção *PRESCRIPTIONS*	346,36	47660,79	218,31	5,8
Inserção *INPUTEVENTS_MV*	490,07	79628,87	282,19	5,1
Busca *PATIENTS*	0,03	0,000026	0,0051	10

Tabela 81 – *MaxMessageCount*:10 *BatchTimeout*:1s Validadores:2 Política:*AND*

Operação	Latência Média (s)	Variância	Desvio Padrão	Vazão (TPS)
Inserção *PATIENTS*	59,6	3062,88	55,34	8,5
Inserção *D_ITEMS*	187,45	14051,24	118,54	7,3
Inserção *PRESCRIPTIONS*	340,17	45594,74	213,53	5,8
Inserção *INPUTEVENTS_MV*	400,95	62380,92	249,76	5,5
Busca *PATIENTS*	0,03	0,000034	0,0058	10

Tabela 82 – *MaxMessageCount*:10 *BatchTimeout*:1s Validadores:2 Política:*OR*

Operação	Latência Média (s)	Variância	Desvio Padrão	Vazão (TPS)
Inserção *PATIENTS*	64,82	3820,34	61,81	8,4
Inserção *D_ITEMS*	321,84	33751	183,71	6,1
Inserção *PRESCRIPTIONS*	357,43	49377,08	222,21	5,8
Inserção *INPUTEVENTS_MV*	485,61	79092,66	281,23	5,1
Busca *PATIENTS*	0,03	0,000029	0,0053	10

Tabela 83 – *MaxMessageCount*:10 *BatchTimeout*:1s Validadores:4 Política:*AND*

Operação	Latência Média (s)	Variância	Desvio Padrão	Vazão (TPS)
Inserção *PATIENTS*	58,39	3003,74	54,81	8,5
Inserção *D_ITEMS*	304,22	26869,54	163,92	6,4
Inserção *PRESCRIPTIONS*	377,69	54064,73	232,52	5,6
Inserção *INPUTEVENTS_MV*	501,06	80988,06	284,58	5,1
Busca *PATIENTS*	0,03	0,000022	0,0047	10

Tabela 84 – *MaxMessageCount*:10 *BatchTimeout*:1s Validadores:4 Política:*OR*

As Tabelas 85 a 88 exibem as métricas de desempenho encontradas para o tamanho do bloco estabelecido com o tamanho do bloco com dez transações e tempo de criação estipulado em dez segundos. Os fatores alterados nessas Tabelas seguem sendo a política de validação e a quantidade de nós validadores.

Operação	Latência Média (s)	Variância	Desvio Padrão	Vazão (TPS)
Inserção *PATIENTS*	41,99	1348,28	36,72	9
Inserção *D_ITEMS*	134,44	7840,36	88,55	7,8
Inserção *PRESCRIPTIONS*	354,4	45286,32	212,81	5,8
Inserção *INPUTEVENTS_MV*	338,51	43865,97	209,44	5,9
Busca *PATIENTS*	0,03	0,000025	0,005	10

Tabela 85 – *MaxMessageCount*:10 *BatchTimeout*:10s Validadores:2 Política:*AND*

Operação	Latência Média (s)	Variância	Desvio Padrão	Vazão (TPS)
Inserção *PATIENTS*	51,09	2258,92	47,53	8,7
Inserção *D_ITEMS*	121,8	6844,39	82,73	7,9
Inserção *PRESCRIPTIONS*	306,18	35330,43	187,96	6,2
Inserção *INPUTEVENTS_MV*	322	38909,99	197,26	6
Busca *PATIENTS*	0,03	0,000035	0,0059	10

Tabela 86 – *MaxMessageCount*:10 *BatchTimeout*:10s Validadores:2 Política:*OR*

Operação	Latência Média (s)	Variância	Desvio Padrão	Vazão (TPS)
Inserção *PATIENTS*	45,43	1747,87	41,81	8,9
Inserção *D_ITEMS*	124,7	5870,6	76,62	8,1
Inserção *PRESCRIPTIONS*	353,01	43998,67	209,76	5,9
Inserção *INPUTEVENTS_MV*	436,35	63745,8	252,48	5,4
Busca *PATIENTS*	0,03	0,000027	0,0052	10

Tabela 87 – *MaxMessageCount*:10 *BatchTimeout*:10s Validadores:4 Política:*AND*

Operação	Latência Média (s)	Variância	Desvio Padrão	Vazão (TPS)
Inserção *PATIENTS*	49,12	2065,28	45,45	8,8
Inserção *D_ITEMS*	122,18	5498,23	74,15	8
Inserção *PRESCRIPTIONS*	296,23	34425,08	185,54	6,2
Inserção *INPUTEVENTS_MV*	390,63	50795,12	225,38	5,7
Busca *PATIENTS*	0,03	0,000028	0,0053	10

Tabela 88 – *MaxMessageCount*:10 *BatchTimeout*:10s Validadores:4 Política:*OR*

Os valores encontrados para a latência média, variância e desvio padrão desta, e vazão quando testado o cenário com bloco composto por 500 transações e intervalo de criação de um segundo são mostrados nas Tabelas 89 a 92

Operação	Latência Média (s)	Variância	Desvio Padrão	Vazão (TPS)
Inserção *PATIENTS*	8,15	89,79	9,48	9,7
Inserção *D_ITEMS*	59,68	1485,44	38,54	8,9
Inserção *PRESCRIPTIONS*	129,66	6573,91	81,08	7,9
Inserção *INPUTEVENTS_MV*	182,53	11905,35	109,11	7,4
Busca *PATIENTS*	0,03	0,000033	0,0057	10

Tabela 89 – *MaxMessageCount*:500 *BatchTimeout*:1s Validadores:2 Política:*AND*

Operação	Latência Média (s)	Variância	Desvio Padrão	Vazão (TPS)
Inserção *PATIENTS*	6,45	50,94	7,14	9,8
Inserção *D_ITEMS*	51,27	1112,66	33,36	9,1
Inserção *PRESCRIPTIONS*	164,28	9352,55	96,71	7,6
Inserção *INPUTEVENTS_MV*	157,34	10094,11	100,47	7,5
Busca *PATIENTS*	0,03	0,000037	0,006	10

Tabela 90 – *MaxMessageCount*:500 *BatchTimeout*:1s Validadores:2 Política:*OR*

Operação	Latência Média (s)	Variância	Desvio Padrão	Vazão (TPS)
Inserção *PATIENTS*	9,08	103,31	10,16	9,7
Inserção *D_ITEMS*	54,39	1047,37	32,36	9
Inserção *PRESCRIPTIONS*	159,77	9203,12	95,93	7,6
Inserção *INPUTEVENTS_MV*	198,91	14911,82	122,11	7,2
Busca *PATIENTS*	0,03	0,000041	0,0064	10

Tabela 91 – *MaxMessageCount*:500 *BatchTimeout*:1s Validadores:4 Política:*AND*

Operação	Latência Média (s)	Variância	Desvio Padrão	Vazão (TPS)
Inserção *PATIENTS*	9,54	127,87	11,31	9,6
Inserção *D_ITEMS*	40,69	663,53	25,76	9,2
Inserção *PRESCRIPTIONS*	153,61	7985,45	89,36	7,7
Inserção *INPUTEVENTS_MV*	151,05	8514,16	92,27	7,7
Busca *PATIENTS*	0,03	0,00003	0,0059	10

Tabela 92 – *MaxMessageCount*:500 *BatchTimeout*:1s Validadores:4 Política:*OR*

Por último, as Tabelas de 93 a 96 exibem os resultados obtidos com as execuções considerando o tempo de criação do bloco de dez segundos e o bloco contendo 500 transações. Entre as Tabelas, os níveis que determinam a política de validação e a quantidade de nós validadores são alterados.

Operação	Latência Média (s)	Variância	Desvio Padrão	Vazão (TPS)
Inserção *PATIENTS*	6,3	8,6	2,93	9,9
Inserção *D_ITEMS*	6,38	8,69	2,95	9,9
Inserção *PRESCRIPTIONS*	6,6	8,58	2,93	9,9
Inserção *INPUTEVENTS_MV*	6,71	8,57	2,93	9,9
Busca *PATIENTS*	0,03	0,00003	0,0052	10

Tabela 93 – *MaxMessageCount*:500 *BatchTimeout*:10s Validadores:2 Política:*AND*

Operação	Latência Média (s)	Variância	Desvio Padrão	Vazão (TPS)
Inserção *PATIENTS*	6,37	8,63	2,94	9,9
Inserção *D_ITEMS*	6,43	8,78	2,96	9,9
Inserção *PRESCRIPTIONS*	6,55	8,57	2,93	9,9
Inserção *INPUTEVENTS_MV*	6,65	8,57	2,93	9,9
Busca *PATIENTS*	0,03	0,000024	0,0049	10

Tabela 94 – *MaxMessageCount*:500 *BatchTimeout*:10s Validadores:2 Política:*OR*

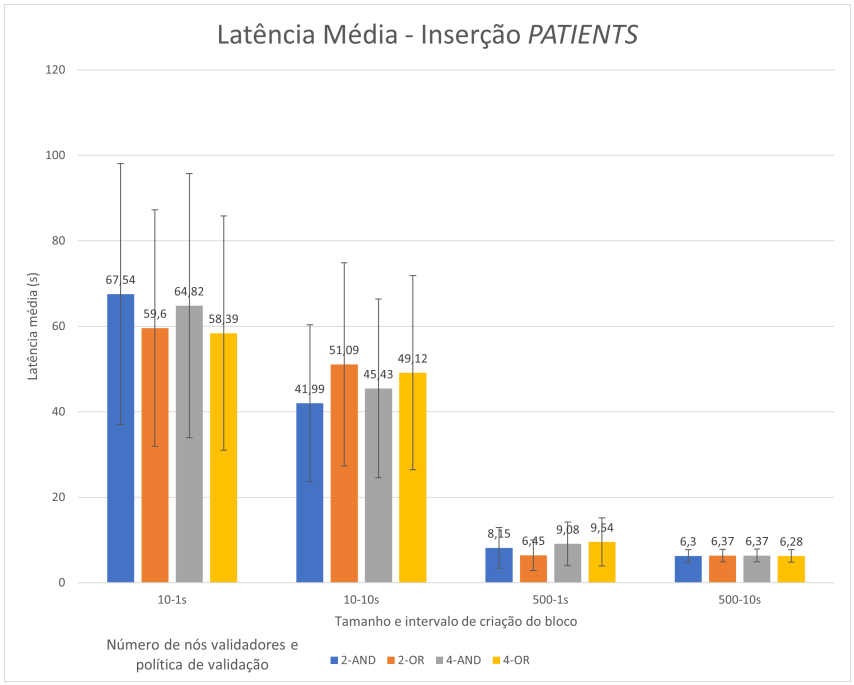
Operação	Latência Média (s)	Variância	Desvio Padrão	Vazão (TPS)
Inserção *PATIENTS*	6,37	8,68	2,95	9,9
Inserção *D_ITEMS*	6,35	8,66	2,94	9,9
Inserção *PRESCRIPTIONS*	6,58	8,61	2,93	9,9
Inserção *INPUTEVENTS_MV*	6,67	8,61	2,93	9,9
Busca *PATIENTS*	0,03	0,00002	0,0048	10

Tabela 95 – *MaxMessageCount*:500 *BatchTimeout*:10s Validadores:4 Política:*AND*

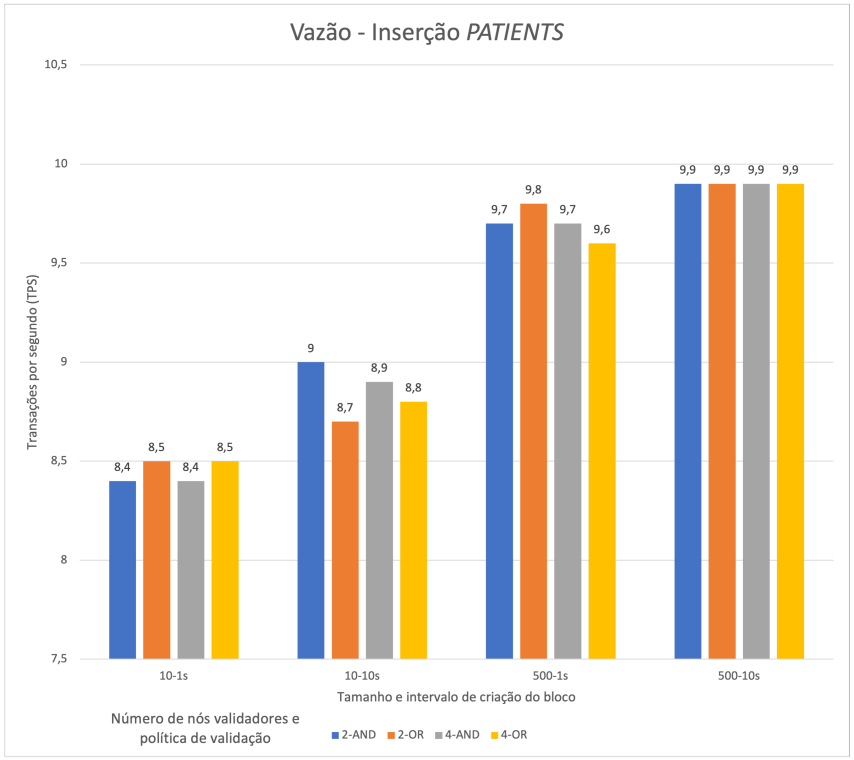
Operação	Latência Média (s)	Variância	Desvio Padrão	Vazão (TPS)
Inserção *PATIENTS*	6,28	8,65	2,94	9,9
Inserção *D_ITEMS*	6,33	8,62	2,94	9,9
Inserção *PRESCRIPTIONS*	6,79	8,78	2,96	9,9
Inserção *INPUTEVENTS_MV*	6,85	8,92	2,99	9,9
Busca *PATIENTS*	0,03	0,000153	0,0124	10

Tabela 96 – *MaxMessageCount*:500 *BatchTimeout*:10s Validadores:4 Política:*OR*

Entre as Figuras 34 e 38 são mostrados os valores encontrados para a latência e vazão das execuções. Nos gráficos da latência média, a barra de erro considera o desvio padrão das operações.



(a) Latência

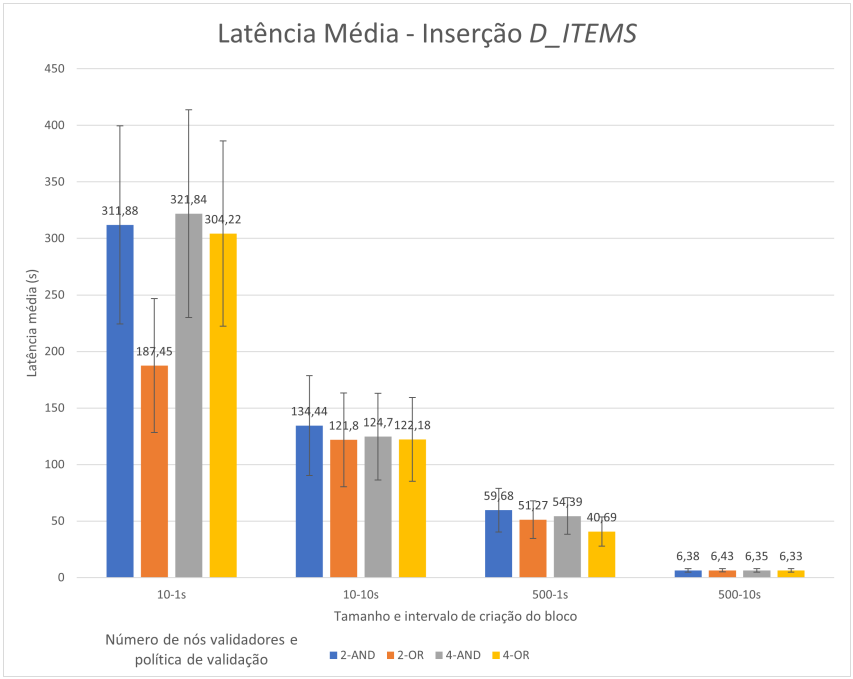


(b) Vazão

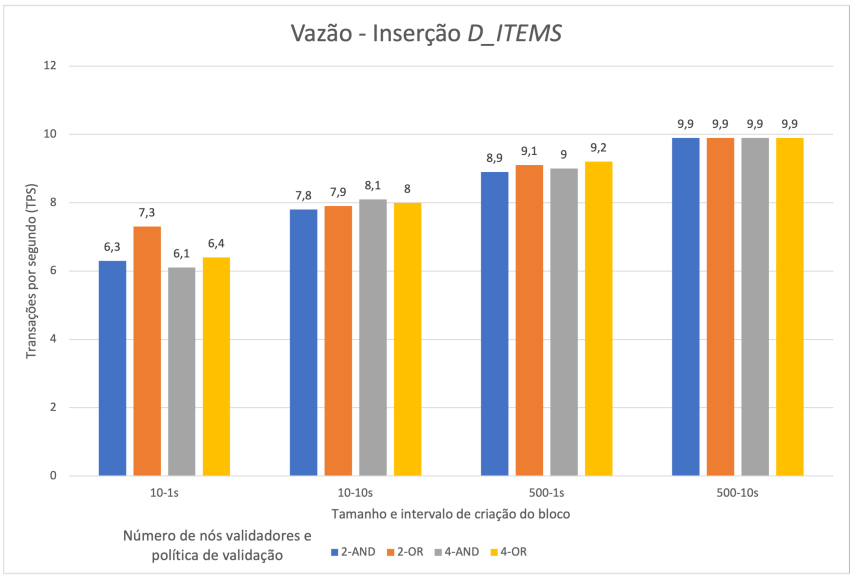
Figura 34 – Resultados para inserção de *PATIENTS*

Os gráficos das Figuras 34 a 37 mostram que para a quantidade de transações e frequência executadas neste experimento, tanto a política de validação quanto a quantidade de nós validadores têm pouca influência no desempenho da aplicação. Os resultados encontrados não são coesos, com execuções em que a política *AND* obteve uma menor latência e uma maior vazão

que quando comparado com a política *OR* com o mesmo número de nós validadores e condições iguais de criação do bloco, resultado contrário do esperado. Um exemplo desse resultado ocorre na inserção de *PATIENTS* (Figura 34), quando observada a política *AND* e *OR* associado a 4 nós validadores e tamanho de bloco 10 e tempo de criação de 10 segundos.



(a) Latência

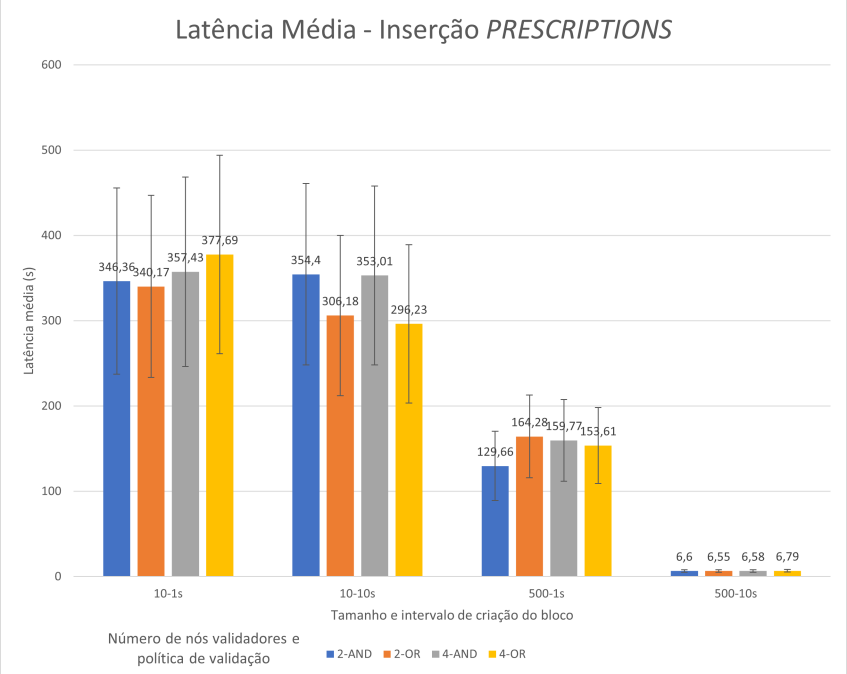


(b) Vazão

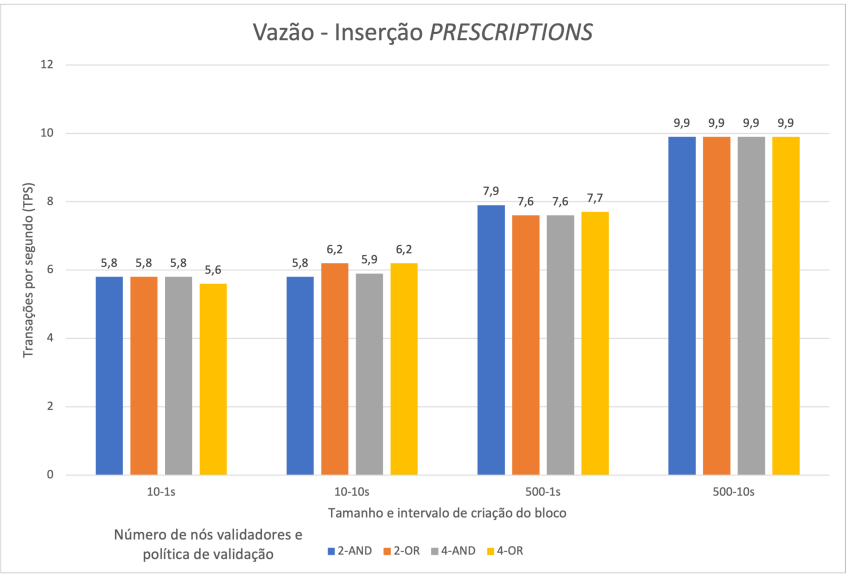
Figura 35 – Resultados para inserção de *D_ITEMS*

Por outro lado, os resultados encontrados nos cenários avaliados mostram o impacto da alteração do tamanho do bloco e tempo de criação. O gráfico da Figura 35 por exemplo, mostra como a cada alteração desses parâmetros o desempenho da aplicação melhora de forma a diminuir a latência e aumentar a vazão. Com *MaxMessageCount* e *BatchTimeout* definidos em

500 e 10 segundos, os melhores resultados foram obtidos, principalmente na vazão que atingiu 9,9 transações por segundo com uma frequência de submissão de 10 transações por segundo.



(a) Latência

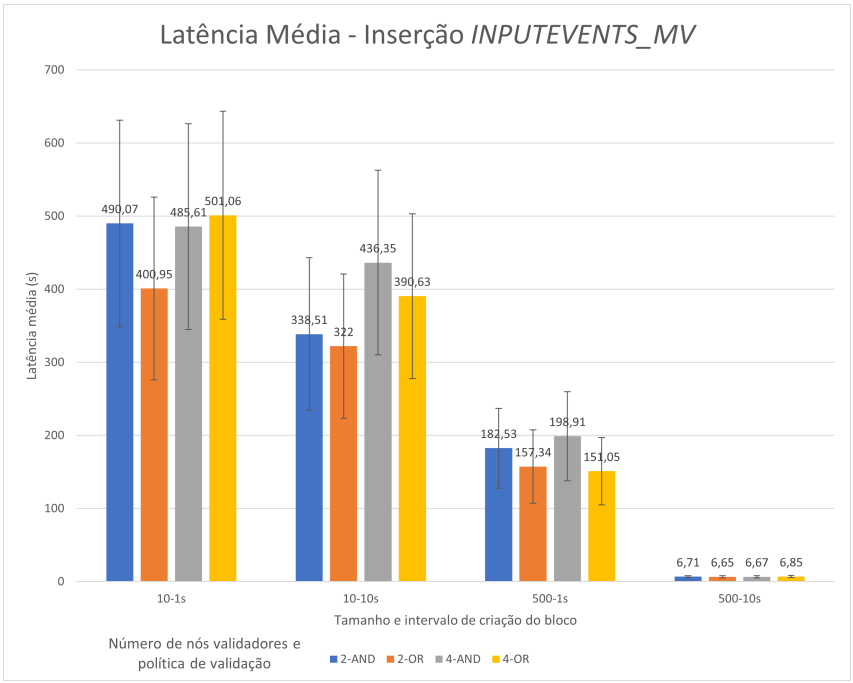


(b) Vazão

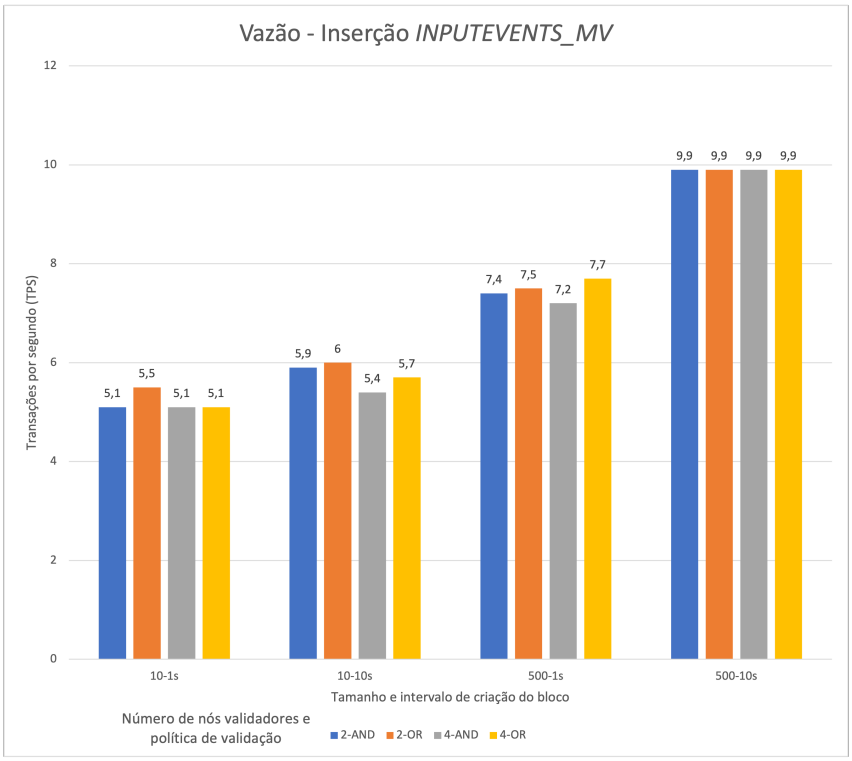
Figura 36 – Resultados para inserção de *PRESCRIPTIONS*

Essa melhora no desempenho é mais evidente na inserções de Tabelas com quantidades maiores de indexações para a inserção dos dados no banco de dados *CouchDB*. A Figura 36 mostra os resultados com a inserção da Tabela *PRESCRIPTIONS* na rede, que demanda o maior número de indexações. Neste gráfico é possível notar uma queda significativa na latência quando o tamanho do bloco é alterado de 10 para 500, mesmo considerando o tempo de criação de 1

segundo. Quando combinado o *MaxMessageCount* em 500 e *BatchTimeout* em 10 segundos, os valores da latência média e vazão da inserção de todas os arquivos testadas se equiparam.

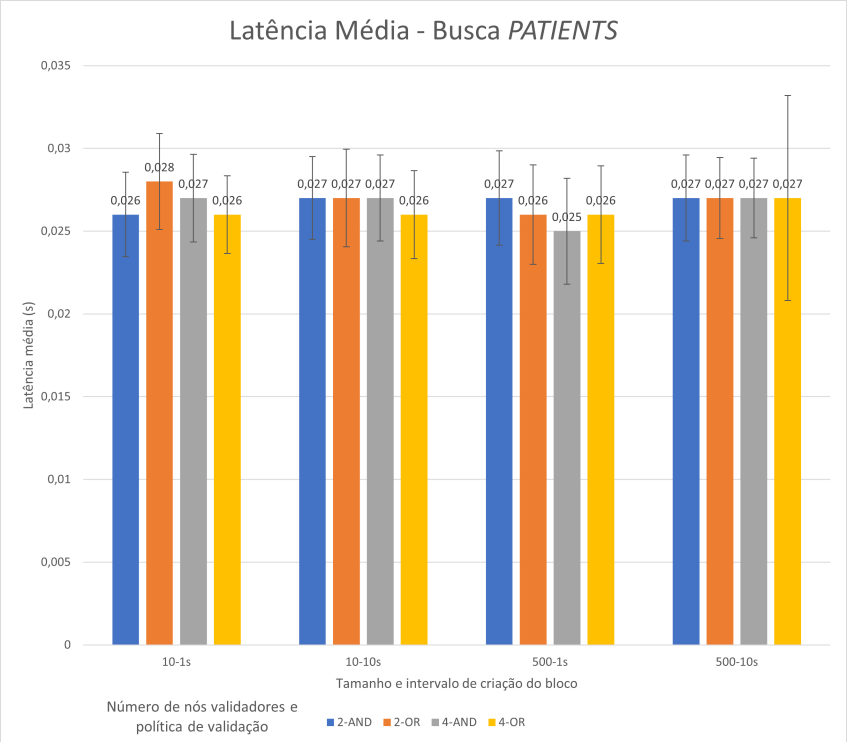


(a) Latência

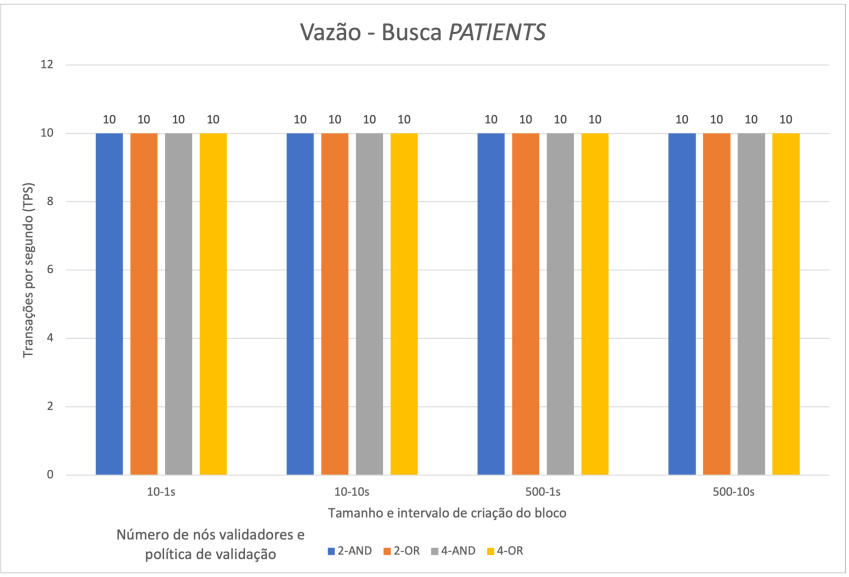


(b) Vazão

Figura 37 – Resultados para inserção de *INPUTEVENTS_MV*



(a) Latência



(b) Vazão

Figura 38 – Resultados para busca de *PATIENTS*

Como mostrado no gráfico da Figura 38 os valores tanto da latência média quanto da vazão permaneceram inalterados em todos os cenários testados. Era esperado que a modificação da política de validação e do número de nós validadores não alterasse as métricas avaliadas na busca devido à natureza dessa transação dentro do *Hyperledger Fabric* de não haver a necessidade da validação dos nós para ocorrer a transação.

7.6 Análise dos Resultados

Esta Seção apresenta a análise dos resultados encontrados e é desenvolvida com base nas hipóteses de pesquisa levantadas para este experimento.

7.6.1 Hipótese I

A primeira hipótese levantada relaciona a quantidade de nós necessários para validar uma transação e a política de validação que, em conjunto, determinam quantos nós da rede participam do processo de validação das transações *update*. O comportamento esperado, era que com a diminuição dos nós validadores, haveria melhoria no desempenho da aplicação *blockchain* para as inserções. Na arquitetura do *Fabric* a busca não apresenta as etapas de validação, como já definido.

Nesses experimentos, duas formas de reduzir o número de nós validadores em uma rede *blockchain Hyperledger Fabric* foram utilizadas. Uma maneira é aplicando a política de validação *OR*, que independente do número de nós validadores da rede, demanda apenas uma resposta sobre a transação. Outro modo é diminuir a quantidade de nós que participam desse processo na política *AND*, nesse experimento foram considerados todos os nós validando a transação em uma rede com quatro nós ou somente dois.

Com a finalidade de validar a hipótese levantada foram comparadas as medidas de desempenho (latência e vazão) considerando as diferentes configurações no processo de validação. Para a latência, inicialmente, as amostras coletadas foram investigas para averiguar a normalidade ou não do conjunto de dados. O teste de *Cramér-von Mises* (THODE, 2002) com nível de significância 95% foi aplicado em todos os conjuntos de dados, que apresentam *p-valor* menor do que 0,05. Dessa forma, foi determinado a não normalidade das amostras analisadas.

Dada a não-normalidade, o teste de *Wilcoxon* (WALPOLE *et al.*, 1993) com nível de significância 95% foi aplicado para verificar se as amostras são estatisticamente equivalentes ou não. As Tabelas de 97 a 100 apresentam os *p-valores* encontrados quando comparados os diferentes cenários de execução com relação a quantidade de nós validadores e política de validação de transações na rede.

Operação	10-1s	10-10s	500-1s	500-10s
Inserção *PATIENTS*	0	0	0	0
Inserção *D_ITEMS*	0	0	0	0
Inserção *PRESCRIPTIONS*	0	0	0	0
Inserção *INPUTEVENTS_MV*	0	0	0	0

Tabela 97 – *p-valores* entre as políticas *AND* e *OR* com quatro nós validadores

Operação	10-1s	10-10s	500-1s	500-10s
Inserção *PATIENTS*	0	0	0	0
Inserção *D_ITEMS*	0	0	0	1,11e-16
Inserção *PRESCRIPTIONS*	0	0	0	0
Inserção *INPUTEVENTS_MV*	0	0	0	0

Tabela 98 – *p-valores* entre as políticas *AND* e *OR* com dois nós validadores

Operação	10-1s	10-10s	500-1s	500-10s
Inserção *PATIENTS*	0	0	0	0
Inserção *D_ITEMS*	0	0	0	0
Inserção *PRESCRIPTIONS*	0	0	0	0
Inserção *INPUTEVENTS_MV*	0	0	0	0

Tabela 99 – *p-valores* entre 2 e 4 nós validadores e a política *AND*

Operação	10-1s	10-10s	500-1s	500-10s
Inserção *PATIENTS*	0	0	0	0
Inserção *D_ITEMS*	0	1,01e-11	0	0
Inserção *PRESCRIPTIONS*	0	0	0	0
Inserção *INPUTEVENTS_MV*	0	0	0	0

Tabela 100 – *p-valores* entre 2 e 4 nós validadores e a política *OR*

Como os *p-valores* encontrados foram menores que 0,05 pode-se afirmar que as amostras comparadas não são estatisticamente equivalentes. Com isso, a análise da hipótese é voltada para as latências obtidas nas execuções.

O impacto referente ao processo de validação das transações realizadas não tem uma influência considerável sobre a performance das operações avaliadas, dado que o processo de criação dos blocos e a inserção de dados no banco de dados se mostram fatores determinantes no desempenho da aplicação *blockchain*.

Pode-se observar nos gráficos das Figuras 34a e 35a (inserção de *PATIENTS* e *D_ITEM*, respectivamente) que ao diminuir o número de nós validando uma transação também ocorre uma diminuição da latência média nos cenários analisados. Essa tendência é observada principalmente quando o tamanho do bloco se manteve em 10. Com o aumento do tamanho do bloco, a latência diminui e o processo de inserção não se mostra como gargalo para essa métrica.

Já nas inserções de *PRESCRIPTIONS* e *INPUTEVENTS_MV* (Figuras 36a e 37a), para os cenários com *MaxMessageCount* em dez e *BatchTimeout* de um segundo, o processo de criação do bloco e inserção no banco de dados são fatores dominantes na latência. Ao aumentar o intervalo de criação do bloco, esse processo deixa de ser dominante na latência das inserções e

é possível verificar a queda na latência com a diminuição do número de nós validando a inserção de dados.

Nesses cenários de execução, os fatores analisados não exercem influência considerável sobre a vazão da rede *blockchain*, visto que em todos os resultados obtidos a métrica avaliada sofre pouca ou nenhuma alteração.

7.6.2 Hipótese II

A segunda hipótese de pesquisa afirma que a criação de blocos maiores aumenta a latência média das requisições, porém também acarreta em aumento da vazão de transações de inclusão na *blockchain*.

A verificação da normalidade dos dados em função da aplicação do teste de *Cramér-von Mises* (THODE, 2002) com nível de significância 95% foi apresentada para a hipótese anterior para a latência e permanece a mesma. O teste comprova a distribuição não-normal do conjunto de dados.

O teste de *Wilcoxon* (WALPOLE *et al.*, 1993) com nível de significância 95% foi utilizado para verificar a equivalência das amostras coletadas. Os *p-valores* resultantes da aplicação do teste de hipótese comparando os dois tamanhos de bloco considerados nas execuções são mostrados nas Tabelas 101 e 102.

Operação	4-OR	4-AND	2-OR	2-AND
Inserção *PATIENTS*	0	0	0	0
Inserção *D_ITEMS*	0	0	0	0
Inserção *PRESCRIPTIONS*	0	0	0	0
Inserção *INPUTEVENTS_MV*	0	0	0	0
Busca *PATIENTS*	0	0	0	0

Tabela 101 – *p-valores* entre *MaxMessageCount* de 10 e 500 para *BatchTimeout* de 1s

Operação	4-OR	4-AND	2-OR	2-AND
Inserção *PATIENTS*	0	0	0	0
Inserção *D_ITEMS*	0	0	0	0
Inserção *PRESCRIPTIONS*	0	0	0	0
Inserção *INPUTEVENTS_MV*	0	0	0	0
Busca *PATIENTS*	0	0,237	0,013	8,39e-14

Tabela 102 – *p-valores* entre *MaxMessageCount* de 10 e 500 para *BatchTimeout* de 10s

A busca de *PATIENTS* entre *MaxMessageCount* de 500 e 10 para a política de validação *AND*, com dois nós validadores e considerando *BatchTimeout* de dez segundos as amostras são estatisticamente similares, como mostrado na Figura 38a. As demais amostras são consideradas

não equivalentes, por apresentarem *p-valor* menor que 0,05. Tendo em vista a não equivalência das amostras, a latência média é observada e aponta a diminuição desta métrica quando a rede é executada com um tamanho de bloco maior, contrário a hipótese levantada. A vazão segue o cenário esperado, de aumento quando utilizada o *MaxMessageCount* maior.

Dessa forma, os resultados mostram que o tamanho do bloco é um fator de grande impacto no desempenho de uma aplicação *blockchain* desenvolvida com o *Hyperledger Fabric* e por isso deve ser cuidadosamente selecionado principalmente considerando o tipo de dado utilizado e o armazenamento no banco de dados *CouchDB*.

Nos cenários testados, criar blocos maiores melhorou o desempenho do sistema em até 6441,98% na redução da latência média na inserção de *INPUTEVENTS_MV* com quatro nós validadores, política de validação *AND* e intervalo de criação de dez segundos. A menor redução foi na inserção de *PRESCRIPTIONS* de 107,07% na execução com *BatchTimeout* em 1 segundo, quantidade de nós validadores estabelecida em dois e política *OR*.

Para a vazão, o caso mais significativo da diferença percentual também é na inserção de *INPUTEVENTS_MV* com 4 nós validadores, política de validação *AND* e intervalo de criação de 10 segundos, com aumento de 83,33%. Já a menor alteração ocorreu na inserção de *PATIENTS* com 10% de diferença percentual considerando o intervalo de criação de 10 segundos, 2 nós validadores e política *AND*. Essa diferença percentual na diminuição da vazão entre os arquivos é justificada pelo acesso ao banco de dados *CouchDB* e o número de indexações de cada arquivo. Mesmo com o tamanho do bloco menor, a vazão da inserção de *PATIENTS* apresenta resultados próximos a dez (frequência de requisição utilizada nas execuções), pois não há a criação de índice na sua inserção.

O comportamento esperado era o aumento da latência média com a criação de blocos maiores, pois criar blocos maiores demandaria mais tempo. Porém, uma vez prontos, os blocos distribuiriam um maior número de transações na rede, dessa forma aumentando a vazão. No entanto esses resultados mostram que criar blocos menores gera uma sobrecarga na rede, fazendo as requisições de inserção terem que esperar mais tempo para serem atendidas visto que é necessário validar e inserir na *blockchain* uma quantidade maior de pequenos blocos.

7.6.3 Hipótese III

A terceira hipótese se refere ao tempo de criação do bloco e sugere que aumentar esse intervalo aumenta a latência do sistema, porém, aumenta também a vazão.

Novamente, para validar esse hipótese, o teste de *Wilcoxon* (WALPOLE *et al.*, 1993) com nível de significância 95% foi aplicado para verificar a equivalência das amostras analisadas. Esse teste foi realizado devido a não-normalidade dos dados, estabelecido com o teste de *Cramér-von Mises* (THODE, 2002) com nível de significância 95%. As Tabelas 103 e 104 mostram os *p-valores* encontrados quando aplicado o teste de equivalência.

Operação	4-OR	4-AND	2-OR	2-AND
Inserção *PATIENTS*	0	0	0	0
Inserção *D_ITEMS*	0	0	0	0
Inserção *PRESCRIPTIONS*	0	0	0	0
Inserção *INPUTEVENTS_MV*	0	0	0	0
Busca *PATIENTS*	0	0	0	1,35e-7

Tabela 103 – *p-valores* entre *BatchTimeout* de 1s e 10s para *MaxMessageCount* de 10

Operação	4-OR	4-AND	2-OR	2-AND
Inserção *PATIENTS*	1,25e-14	0	0	0,156
Inserção *D_ITEMS*	0	0	0	0
Inserção *PRESCRIPTIONS*	0	0	0	0
Inserção *INPUTEVENTS_MV*	0	0	0	0
Busca *PATIENTS*	0	0	0	0,28

Tabela 104 – *p-valores* entre *BatchTimeout* de 1s e 10s para *MaxMessageCount* de 500

A comparação das amostras da busca realizada com *PATIENTS* entre o tempo de criação do bloco de 10 segundos e 1 segundo para a política de validação *AND*, com dois nós validadores e *MaxMessageCount* de 500, mostra que as amostas são equivalentes (Figura 38a). Esse cenário é o único no qual as amostras são consideradas equivalentes, para as restantes o *p-valor* obtido foi menor que 0,05 de forma que essas são consideradas não equivalentes.

Como os gráficos apresentados na Seção 7.5 mostram, o aumento do tempo limite para a criação de um bloco impacta diretamente a latência, porém, oposto ao que era esperado, diminui a latência do sistema. O aumento dessa variável melhora a vazão, como esperado. Na infraestrutura do *Fabric*, o bloco é criado quando ou o tamanho do bloco atinge seu limite ou o tempo de criação é alcançado. O fato da latência diminuir uma vez que aumentado o intervalo de criação do bloco indica que o tempo de um segundo não é suficiente para a montagem dos blocos, gerando blocos menores que o tamanho do bloco indicado pela variável *MaxMessageCount* para ambos valores testados.

7.7 Considerações Finais

Este Capítulo apresentou o experimento que analisa os parâmetros de execução do *Hyperledger Fabric* relacionados ao processo de validação das transações e a criação dos blocos na rede. As execuções variam a política de validação, quantidade de nós validadores, tamanho e tempo de criação do bloco.

Com relação ao processo de validação, os resultados encontrados indicam que nos cenários avaliados, diminuir o número de nós necessários para validar uma transação, seja com

aplicação de uma política *OR* ou alterando a quantidade de nós validadores, tem um impacto no desempenho da aplicação, porém não é o fator de maior impacto na rede quando comparado ao processo de criação de blocos. No que se refere à criação de blocos, os valores obtidos mostram que, nos experimentos realizados, aumentar o tamanho do bloco e o tempo para a criação do bloco melhora o desempenho da aplicação *blockchain* com o *CouchDB* na inserção de dados, em relação às métricas latência e vazão.

O Capítulo seguinte relata o experimento que avalia a implicação no desempenho da aplicação de se alterar a infraestrutura da rede do *Hyperledger Fabric*. Os fatores examinados são o tamanho da rede *blockchain* e a quantidade de organizações da rede.

CAPÍTULO

8

EXPERIMENTO 4 - TAMANHO DA REDE E DISTRIBUIÇÃO DOS NÓS

8.1 Considerações Inicias

O experimento descrito nesse Capítulo analisa o efeito de diferentes configurações de rede no desempenho de aplicações *blockchain* com o uso do *CouchDB*. Para isso, os fatores modificados foram o número de nós participantes da rede e sua distribuição na rede dentro das organizações do *Hyperledger Fabric*.

Assim como feito nos experimentos já descritos, a descrição deste se inicia com a introdução do escopo do estudo, determinando características como objetivo, objeto de estudo e foco qualitativo. O planejamento é descrito na sequência com o foco em apresentar as hipóteses e questões de pesquisa. Em um terceiro momento, a infraestrutura de execução é detalhada em termos de máquinas e disposição dos nós na rede em cada rodada do experimento. Os resultados encontrados são mostrados logo após. Por fim, a análise com base nas hipóteses levantadas destacam os principais resultados.

8.2 Escopo

O experimento descrito nesta Seção tem como objetivo avaliar quais características da rede *blockchain* impactam o desempenho de uma aplicação quando usado banco de dados. Como os demais, neste experimento o *Hyperledger Fabric* na versão 1.4.1 com banco de dados *CouchDB* é o objeto de estudo. O foco qualitativo final ainda é determinar o desempenho do *Hyperledger Fabric* quando este é usado com dados heterogêneos.

A perspectiva deste estudo é quantificar o desempenho da *blockchain* quando é aumentado o número de nós participantes da rede, consequentemente o número de máquinas e alterando a distribuição dos nós com diferentes quantidades de organizações.

8.3 Planejamento

Para este experimento, as máquinas utilizadas estão na rede do laboratório de pesquisa *LaSDPC/ICMC*. Os dados heterogêneos utilizados foram obtidos da base de dados médicos reais *MIMIC-III* cuja a descrição dos dados foi realizada na Seção 4.2

Considerando o objetivo deste experimento de verificar como a configuração da rede pode afetar o desempenho de uma aplicação *blockchain*, as seguintes hipóteses foram propostas:

1. A primeira hipótese afirma que aumentar o número de nós e, dessa forma, o número de máquinas, deteriora o desempenho da rede *blockchain*;

2. A segunda hipótese declara que a forma como os nós estão distribuídos na rede, considerando o número de organizações, impacta negativamente no desempenho da rede, isto é, um maior número de organizações gera uma queda no desempenho, considerando as métricas latência de atendimento e vazão das requisições atendidas;

A primeira hipótese é formulada considerando que o cálculo das métricas de desempenho, latência e vazão, são dependentes da propagação da rede. Dessa forma, espera-se que com uma maior quantidade de nós de rede, ocorra uma piora dessas métricas, de maneira a aumentar a latência e diminuir a vazão.

Já a segunda hipótese tem como base a arquitetura do *Hyperledger Fabric* que limita a comunicação de nós não pertencentes à mesma organização. A comunicação inter-organizações é realizada apenas por nós específicos, chamados nós âncoras. Uma organização necessita de no mínimo um nó âncora.

A validade dessas duas hipóteses foram investigadas através das seguintes questões de pesquisa:

QP1: Qual é o impacto no desempenho da rede quando modificado o número de nós integrantes da rede *blockchain*?

QP2: Qual é a variação no desempenho quando alterado o número de organizações distribuídas pela rede *blockchain*?

O desempenho da rede *blockchain* foi calculado por intermédio do *benchmark Hyperledger Caliper* considerando as métricas de latência e vazão. Os valores foram obtidos considerando as Equações descritas de 4.1 a 4.4 para os dois tipos de transações consideradas, inserção e busca.

Os fatores e níveis utilizados neste experimento são descritos na Tabela 105.

Fatores	Níveis		
Quantidade de nós	4	8	12
Número de organizações	1	2	4
Quantidade de transações	10.000	15.000	
Taxa de envio	10	50	

Tabela 105 – Fatores e níveis Experimento 4

O total de transações executadas é o produto da soma dos dois níveis da quantidade de transações (25.000), os três níveis da quantidade de nós da rede e os três níveis do número de organizações e as cinco operações realizadas (quatro inserções de diferentes arquivos e uma busca). A taxa de envio de requisições não foi variada, para as execuções considerando 10.000 entradas o valor considerado foi 10 e para 15.000 a frequência utilizada foi 50. Para o arquivo *D_ITEMS* não houve execução considerando 15.000 transações, pois essa arquivo tem menos que esse total de entradas. Ao todo foram realizadas 990.000 transações sobre a *blockchain* neste experimento.

8.4 Infraestrutura para Execução

Os parâmetros usados na execução deste experimento estão na Tabela 106. O primeiro e o segundo parâmetros definem a quantidade de nós que vão validar uma transação na primeira etapa do processo de validação. Para este experimento, todos os nós integrantes da rede, independentemente do tamanho, vão ter essa função. Os últimos parâmetros da Tabela caracterizam o tamanho e o tempo de criação de cada bloco. Os valores dessas variáveis foram escolhidos por apresentarem o melhor desempenho no experimento anterior para assim, reduzir o impacto do processo de criação dos resultados encontrados nesse experimento em questão.

Parâmetro	Valor
Nós validadores	Todos
Política de validação	AND
MaxMessageCount	500
AbsoluteMaxBytes	128 MB
PreferredMaxBytes	128 MB
BatchTimeout	10s

Tabela 106 – Parâmetros da execução Experimento 4

Para este experimento foram criadas nove configurações de rede para realizar os testes propostos. Para cada número de nós testados (4, 8 e 12) foram desenvolvidas três redes considerando 1, 2 e 4 organizações. Além dos nós participantes da rede, cada rede apresenta um nó do tipo *orderer*. Em todos os cenários de execução, os nós estão em um único *channel*, sendo assim, não há restrições de acesso às informações compartilhadas.

O número de máquinas utilizadas neste experimento equivale ao número de nós integrantes da rede, com uma das máquinas contendo um contêiner a mais com o nó *orderer*. O uso do *CouchDB* foi considerado em todas as execuções realizadas, dessa forma, todas as máquinas tinham pelo menos dois contêineres, um representando o nó da rede e outro com o *CouchDB* associado ao nó.

Na infraestrutura do *Hyperledger Fabric*, os clientes estão associados às organizações e as requisições de escrita e leitura são direcionadas aos nós pertencentes à mesma organização. Neste experimento, o número de clientes na rede equivale ao número de nós dentro de cada organização.

As Figuras 39, 40 e 41 ilustram a rede criada com 4 nós, considerando 1, 2 e 4 organizações, respectivamente. Essas duas Figuras também mostram a interação das organizações com os clientes e como são divididos entre as organizações nos cenários testados.

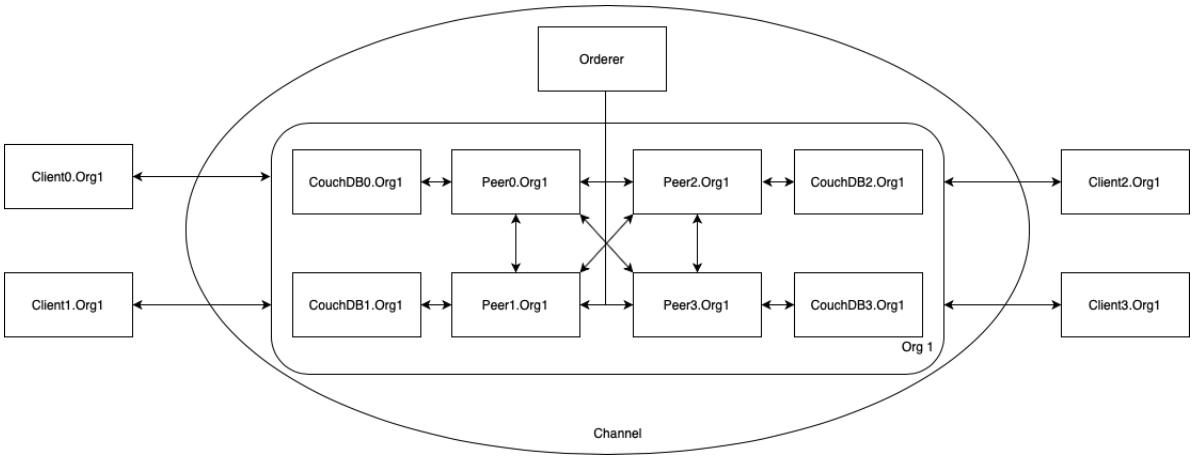


Figura 39 – Configuração da rede com 4 nós e 1 organização Experimento 4

O cliente notifica organização e esta organização distribui a transação entre os nós sob sua responsabilidade e demais organizações da rede. Assim, uma transação é recebida pelos nós e validadas por eles, finalizando a primeira etapa do processo de validação de uma transação, descrita em 2.5.2. Se for uma transação de *query* não há outras etapas. Para requisições do tipo *update*, o cliente notifica o nó *orderer* que recebe a transação e cria o bloco que depois é distribuído para todos os nós da rede.

Para redes como a da Figura 39, os clientes interagem com apenas uma organização, que comunica com os seus integrantes e notifica sobre a validade da transação. O bloco é criado pelo nó *orderer* e distribuído por este.

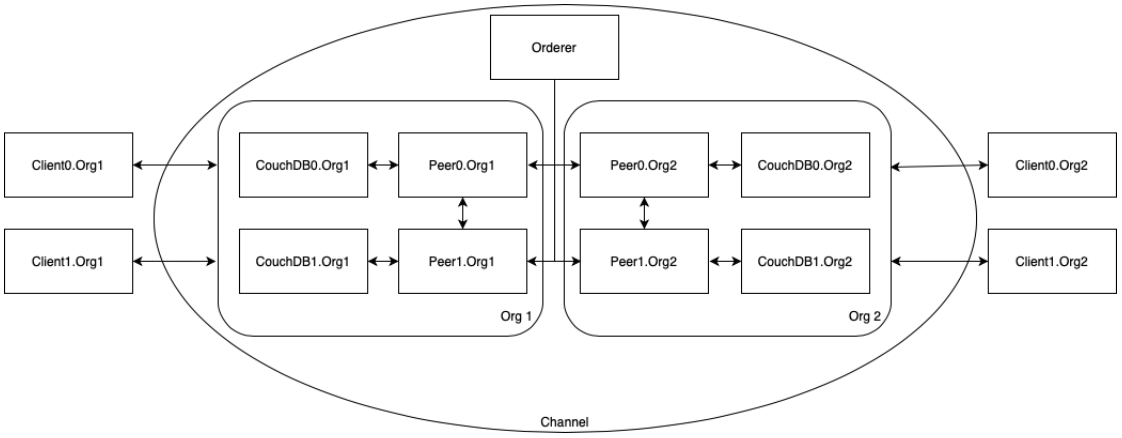


Figura 40 – Configuração da rede com 4 nós e 2 organizações Experimento 4

Para um número maior de organizações, como nas Figuras 40 e 41, as organizações têm que se comunicar para distribuir as requisições de modo que todos os nós validadores possam dar seu parecer sobre a validade da transação. Da mesma forma, depois desta etapa, os clientes notificam o nó *orderer* que cria o bloco com as requisições de todos os clientes, independente da organização, e distribui pela rede. Assim, todos os nós têm em sua cadeia o mesmo conjunto de informações.

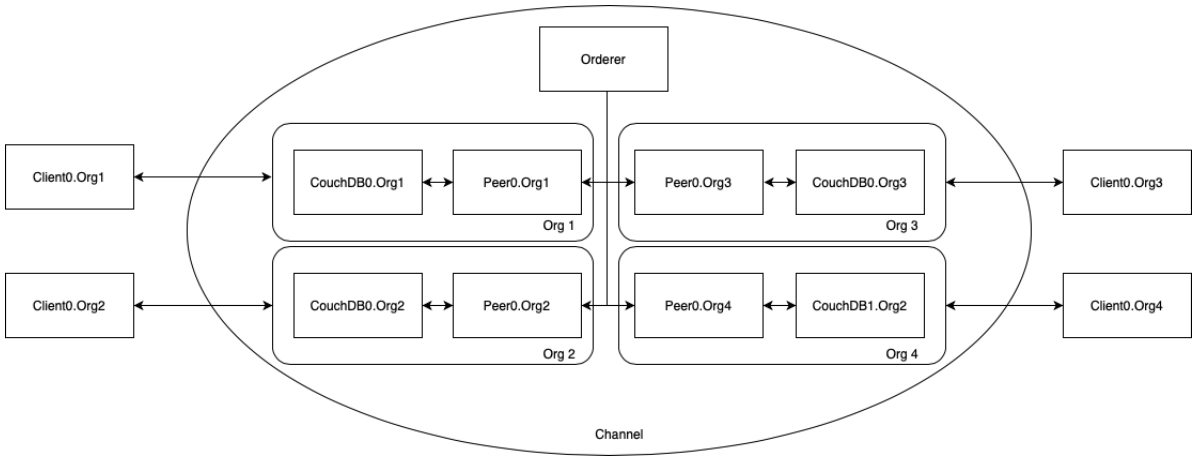


Figura 41 – Configuração da rede com 4 nós e 4 organizações Experimento 4

Os sistemas de software instalados para a execução deste experimento foram os pré-requisitos de instalação encontrados na documentação do *Hyperledger Fabric* e *Hyperledger Caliper* nas versões recomendas. Os nós estão distribuídos na rede (*ICMC-Lasdpc*) com a configuração descrita na Tabela 107.

Configurações de *Hardware* (*Cluster Andromeda - LaSDPC*)	
Sistema Operacional	Ubuntu versões 16.04.7 LTS e 20.04 LTS
CPU	QEMU Virtual CPU (512 KB Cache, 4 GHz)
Memória	16 GB RAM
Disco	Seagate HD 2TB ST2000DM001-1ER1

Tabela 107 – Características das máquinas do Experimento 4

Como nos demais experimentos já descritos nesta dissertação, o *Hyperledger Caliper* foi utilizado como ponto de observação. Além dos dados trazidos no relatório gerado pelo *benchmark*, também foi coletada a latência individual de cada transação realizada.

Em cada rodada de execução, a primeira inserção é de *PATIENTS*, a segunda inserção é de *D_ITEMS*, a próxima inserção é de *PRESCRIPTIONS* e por último é realizada a inserção de *INPUTEVENTS_MV*. A busca se baseia no arquivo *PATIENTS*, que se diferencia das operações de inserção pela dependência da inserção do mesmo arquivo na rede. Inicialmente as operações foram executadas considerando 10.000 transações com frequência de submissão de 10 transações por segundo. Posteriormente foi realizada uma segunda rodada com 15.000 transações e taxa de chegada de 50 requisições por segundo, com a finalidade de aumentar a demanda da a aplicação.

Assim como nos outros experimentos, este não inclui testes de falhas de nós ou da rede e ataques a rede. Todas as transações executadas respeitaram as características esperadas pela rede *blockchain*.

8.5 Resultados

Os resultados encontrados são demostradas através das tabelas e dos gráficos desta Seção. Os primeiros resultados apresentados (nas Tabelas de 108 a 116) focam nas operações realizadas com 10.000 transações e taxa de chegada de requisições de dez transações por segundo. Em seguida, os gráficos dessas execuções, nesse mesmo cenário, são ilustrados.

As Tabelas 108 a 110 mostram as execuções com 4 nós, e variando o número de organizações de quatro, dois e uma.

Operação	Latência Média (s)	Variância	Desvio Padrão	Vazão (TPS)
Inserção *PATIENTS*	8,05	9,74	3,12	9,9
Inserção *D_ITEMS*	8,4	10,14	3,18	9,9
Inserção *PRESCRIPTIONS*	8,8	9,43	3,07	9,9
Inserção *INPUTEVENTS_MV*	9,16	9,93	3,15	9,9
Busca *PATIENTS*	0,03	0,000242	0,0156	10

Tabela 108 – Quantidade de transações: 10.000 Frequência: 10 Nós: 4 Organização: 4

Operação	Latência Média (s)	Variância	Desvio Padrão	Vazão (TPS)
Inserção *PATIENTS*	7,99	9,61	3,1	9,9
Inserção *D_ITEMS*	8,22	10,08	3,08	9,9
Inserção *PRESCRIPTIONS*	8,62	9,49	3,14	9,9
Inserção *INPUTEVENTS_MV*	8,79	9,75	3,12	9,9
Busca *PATIENTS*	0,03	0,000134	0,0116	10

Tabela 109 – Quantidade de transações: 10.000 Frequência: 10 Nós: 4 Organização: 2

Operação	Latência Média (s)	Variância	Desvio Padrão	Vazão (TPS)
Inserção *PATIENTS*	7,91	9,75	3,12	9,9
Inserção *D_ITEMS*	8,17	9,75	3,12	9,9
Inserção *PRESCRIPTIONS*	8,47	9,4	3,07	9,9
Inserção *INPUTEVENTS_MV*	8,72	9,85	3,14	9,9
Busca *PATIENTS*	0,03	0,000511	0,0226	10

Tabela 110 – Quantidade de transações: 10.000 Frequência: 10 Nós: 4 Organização: 1

As Tabelas definidas em 111, 112 e 113 demostram as métricas de desempenho coletadas nas execuções com oito nós e quatro, duas e uma organização dentro da rede *blockchain Fabric*.

Operação	Latência Média (s)	Variância	Desvio Padrão	Vazão (TPS)
Inserção *PATIENTS*	8,14	9,68	3,11	9,9
Inserção *D_ITEMS*	8,43	9,98	3,16	9,9
Inserção *PRESCRIPTIONS*	8,79	9,76	3,12	9,9
Inserção *INPUTEVENTS_MV*	9,16	9,8	3,13	9,9
Busca *PATIENTS*	0,03	0,000034	0,0058	10

Tabela 111 – Quantidade de transações: 10.000 Frequência: 10 Nós: 8 Organização: 4

Operação	Latência Média (s)	Variância	Desvio Padrão	Vazão (TPS)
Inserção *PATIENTS*	8,01	9,25	3,04	9,9
Inserção *D_ITEMS*	8,23	9,97	3,16	9,9
Inserção *PRESCRIPTIONS*	8,63	9,06	3,01	9,9
Inserção *INPUTEVENTS_MV*	9,11	9,84	3,14	9,9
Busca *PATIENTS*	0,03	0,000024	0,0049	10

Tabela 112 – Quantidade de transações: 10.000 Frequência: 10 Nós: 8 Organização: 2

Operação	Latência Média (s)	Variância	Desvio Padrão	Vazão (TPS)
Inserção *PATIENTS*	7.96	9,36	3,06	9,9
Inserção *D_ITEMS*	8,18	10,56	3,25	9,9
Inserção *PRESCRIPTIONS*	8,53	9,24	3,04	9,9
Inserção *INPUTEVENTS_MV*	8,97	9,81	3,13	9,9
Busca *PATIENTS*	0,02	0,000045	0,0067	10

Tabela 113 – Quantidade de transações: 10.000 Frequência: 10 Nós: 8 Organização: 1

Nas Tabelas de 114 a 116 são apresentados os resultados obtidos com 10.000 transações e frequência de dez transações por segundo quando usado doze nós e modificando entre quatro, dois e uma organização na rede.

Operação	Latência Média (s)	Variância	Desvio Padrão	Vazão (TPS)
Inserção *PATIENTS*	8,78	9,5	3,08	9,9
Inserção *D_ITEMS*	9,29	10,52	3,24	9,9
Inserção *PRESCRIPTIONS*	10,04	11,51	3,39	9,9
Inserção *INPUTEVENTS_MV*	9,64	9,92	3,15	9,9
Busca *PATIENTS*	0,03	0,000062	0,0079	10

Tabela 114 – Quantidade de transações: 10.000 Frequência: 10 Nós: 12 Organização: 4

Operação	Latência Média (s)	Variância	Desvio Padrão	Vazão (TPS)
Inserção *PATIENTS*	8,56	9,6	3,1	9,9
Inserção *D_ITEMS*	9,2	10,01	3,16	9,9
Inserção *PRESCRIPTIONS*	9,35	9,92	3,15	9,9
Inserção *INPUTEVENTS_MV*	9,5	9,9	3,15	9,9
Busca *PATIENTS*	0,03	0,000024	0,0049	10

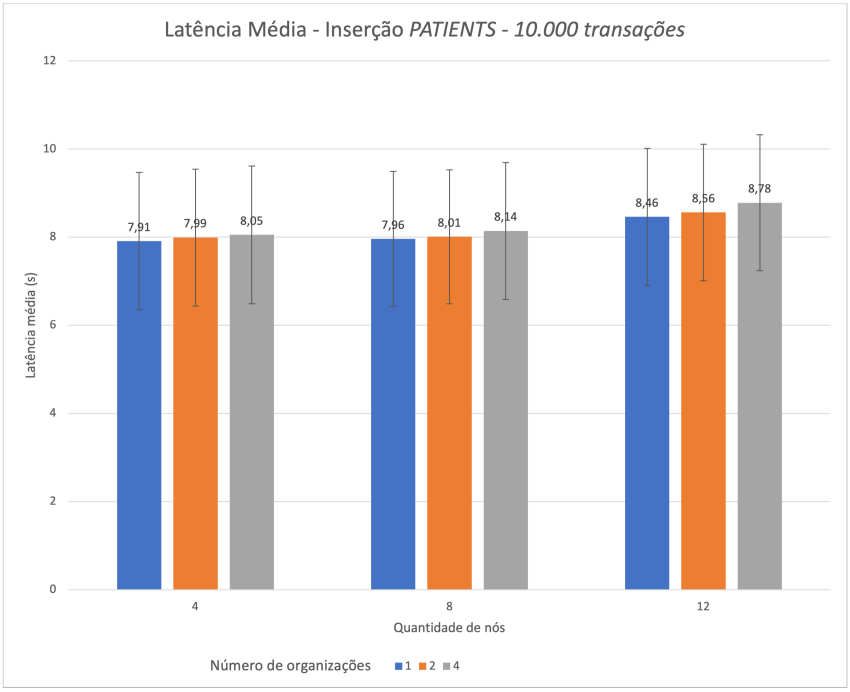
Tabela 115 – Quantidade de transações: 10.000 Frequência: 10 Nós: 12 Organização: 2

Operação	Latência Média (s)	Variância	Desvio Padrão	Vazão (TPS)
Inserção *PATIENTS*	8,46	9,64	3,11	9,9
Inserção *D_ITEMS*	9,13	10,26	3,2	9,9
Inserção *PRESCRIPTIONS*	9,37	9,79	3,13	9,9
Inserção *INPUTEVENTS_MV*	9,62	9,56	3,09	9,9
Busca *PATIENTS*	0,02	0,000037	0,0061	10

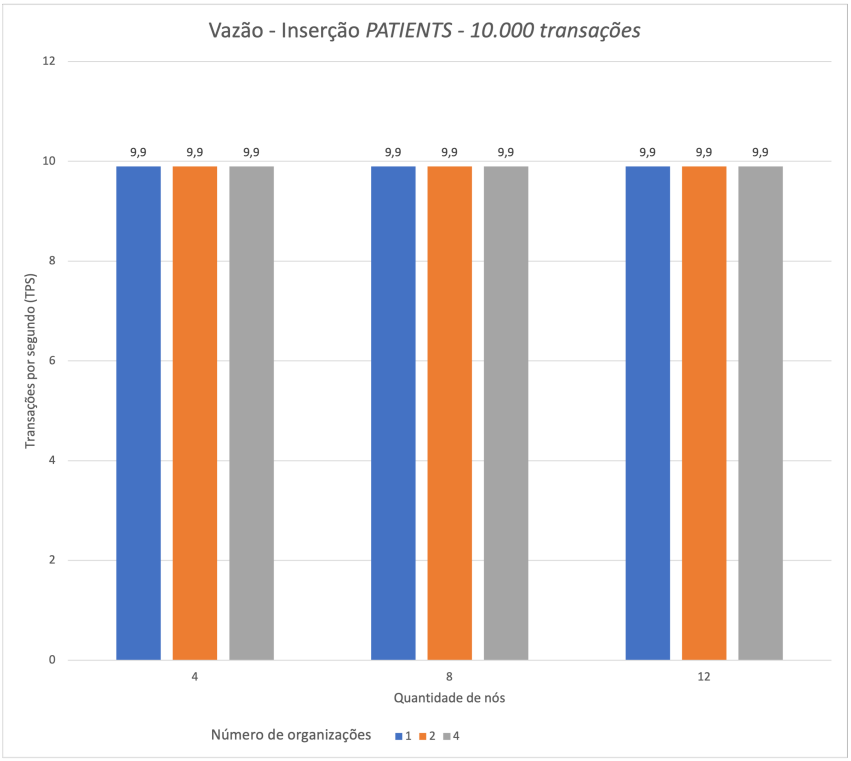
Tabela 116 – Quantidade de transações: 10.000 Frequência: 10 Nós: 12 Organização: 1

Os gráficos apresentados nesta Seção mostram os resultados encontrados variando os fatores quantidade de nós pertencentes à rede *blockchain* e o número de organizações nas quais esses nós estão distribuídos. Como nos demais experimentos, os gráficos da latência média

contém o desvio padrão. Os cinco primeiros gráficos apresentam os resultados de vazão e latência média considerando as execuções com 10.000 entradas e taxa de requisição de dez transações por segundo (Figuras 42 a 46).



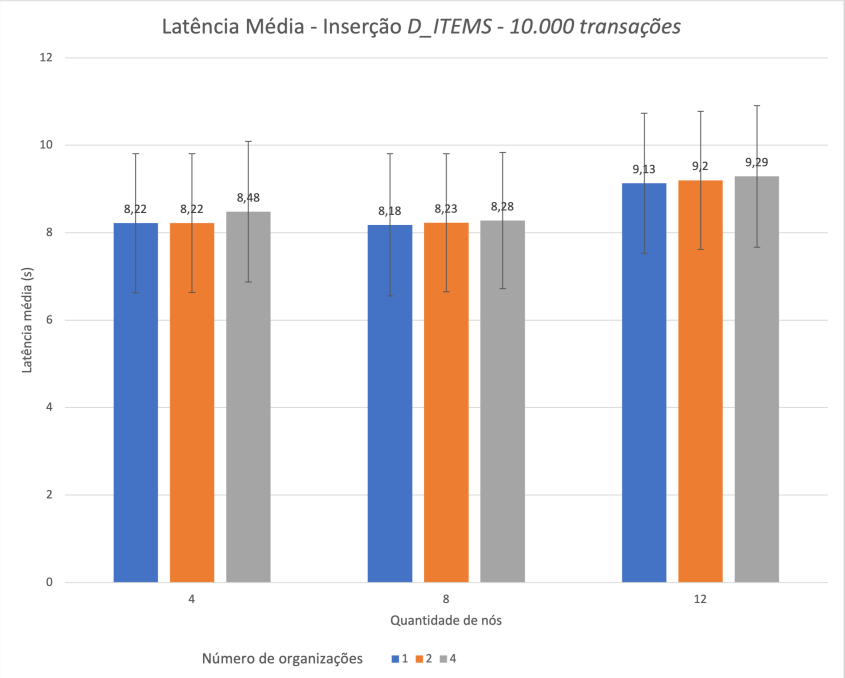
(a) Latência 10.000 entradas e frequência 10



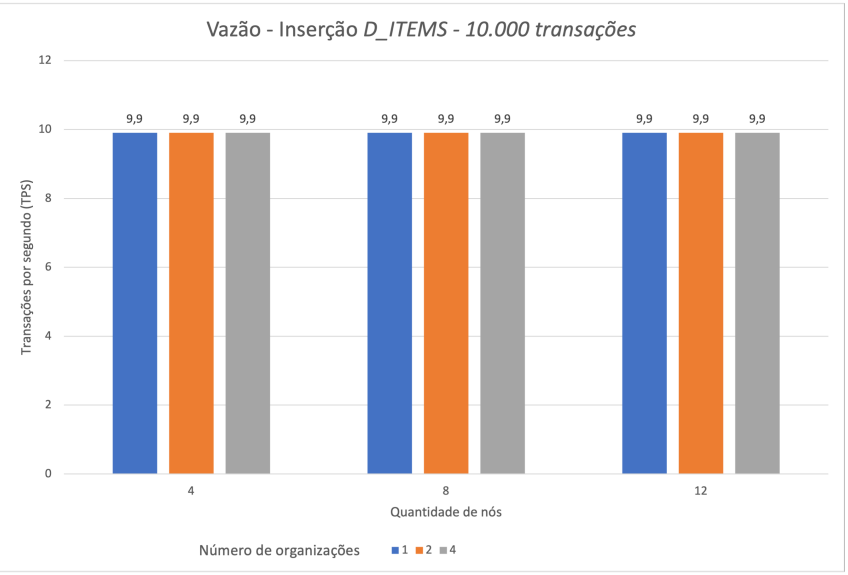
(b) Vazão 10.000 entradas e frequência 10

Figura 42 – Resultados para inserção de *PATIENTS*

Em um aspecto geral, as execuções em que foram utilizadas 10.000 entradas e frequência de requisição de dez transações por segundo apresentam pouca variação da latência média na inserção da mesmo arquivo *MIMIC-III* na rede *blockchain*, quando modificados os fatores analisados neste experimento. Neste cenário de execução, a vazão se manteve constante em 9,9 para todas os arquivos, o que indica a necessidade de outros testes que estressem a aplicação, aumentando o número de requisições e a taxa de chegada, para verificar o comportamento das métricas avaliadas

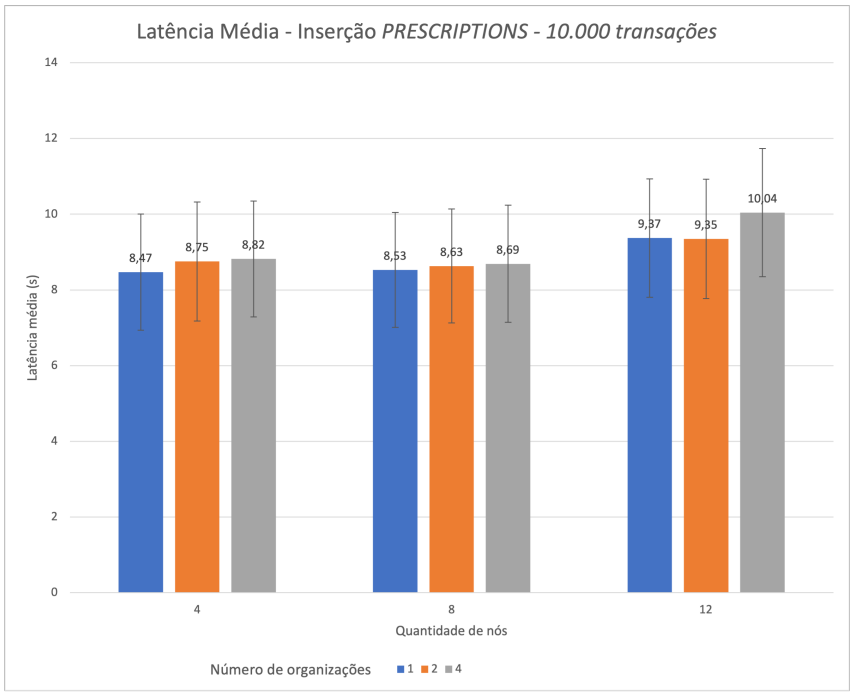


(a) Latência 10.000 entradas e frequência 10

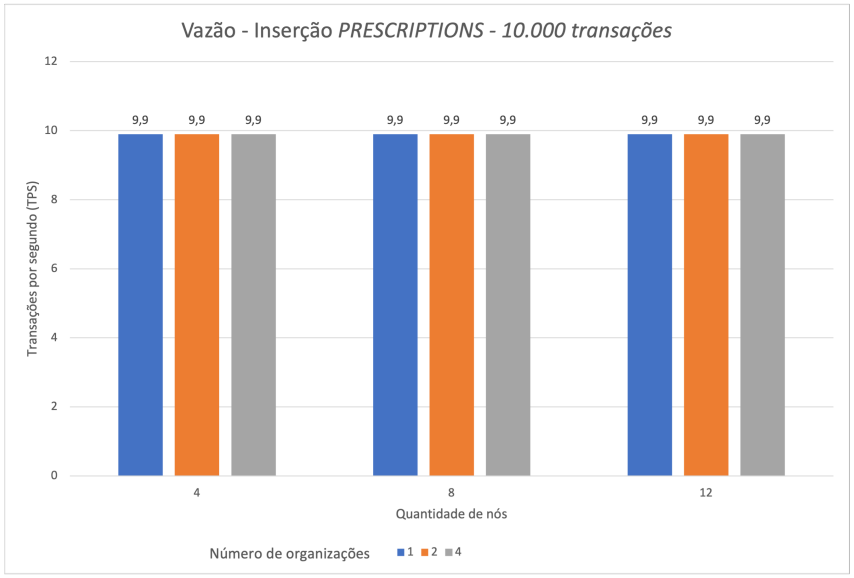


(b) Vazão 10.000 entradas e frequência 10

Figura 43 – Resultados para inserção de *D_ITEMS*



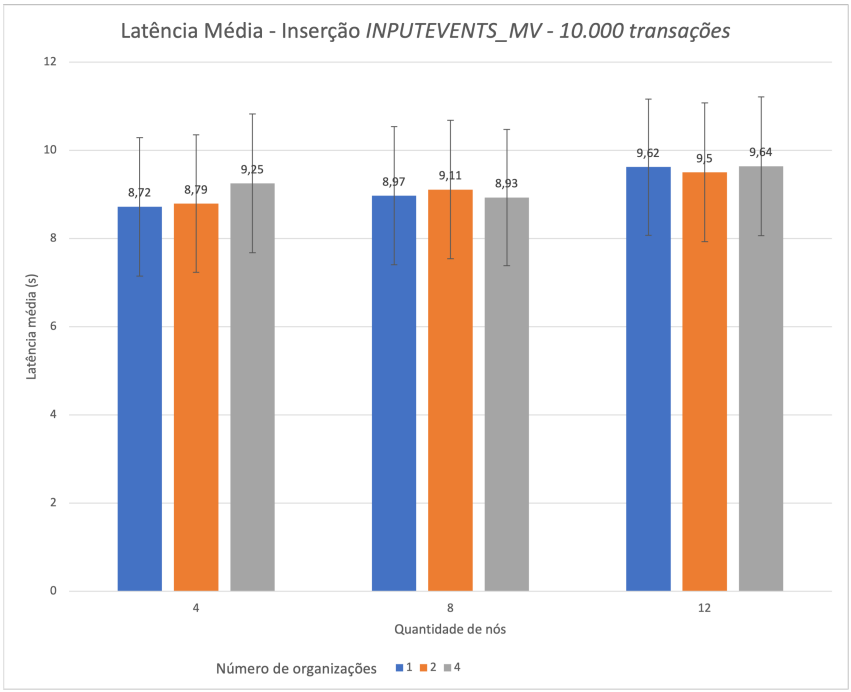
(a) Latência 10.000 entradas e frequência 10



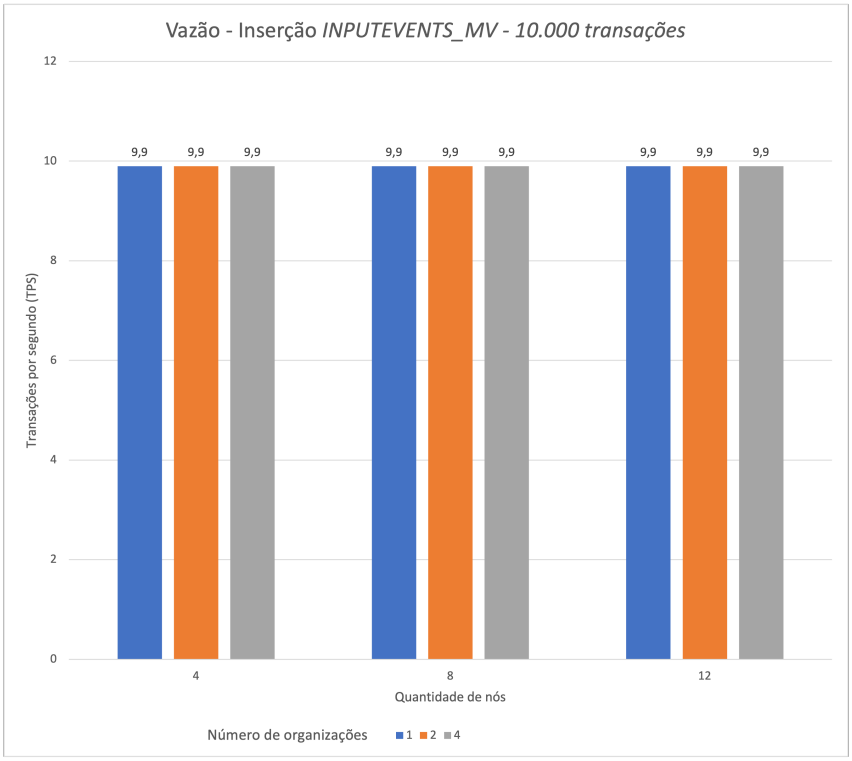
(b) Vazão 10.000 entradas e frequência 10

Figura 44 – Resultados para inserção de *PRESCRIPTIONS*

Nessas execuções, de modo geral, as maiores latências média para as inserções de dados na rede são obtidas quando esta é formada por doze nós e distribuídas em quatro organizações.



(a) Latência 10.000 entradas e frequência 10

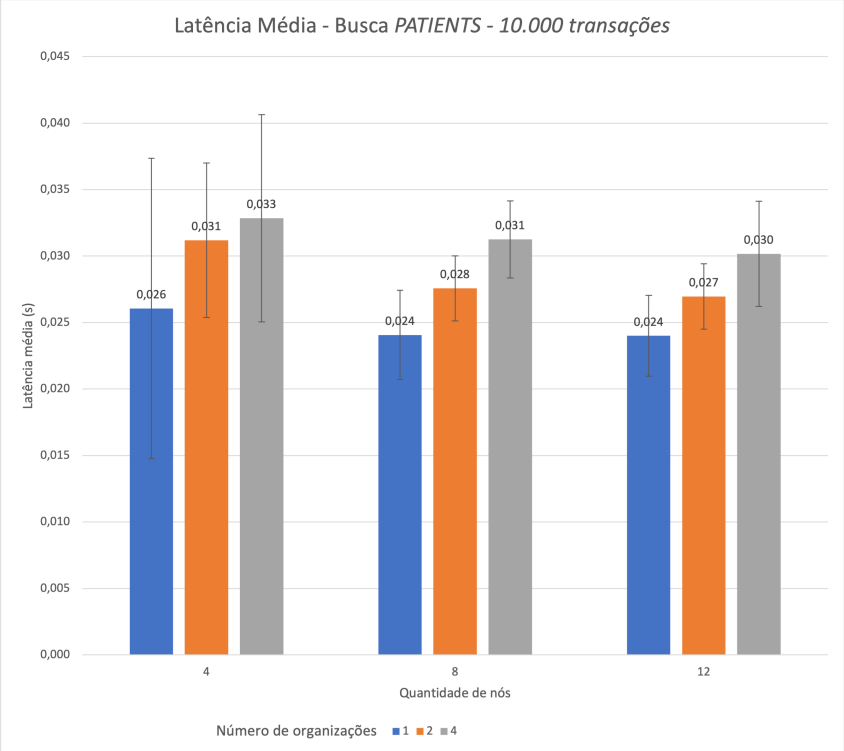


(b) Vazão 10.000 entradas e frequência 10

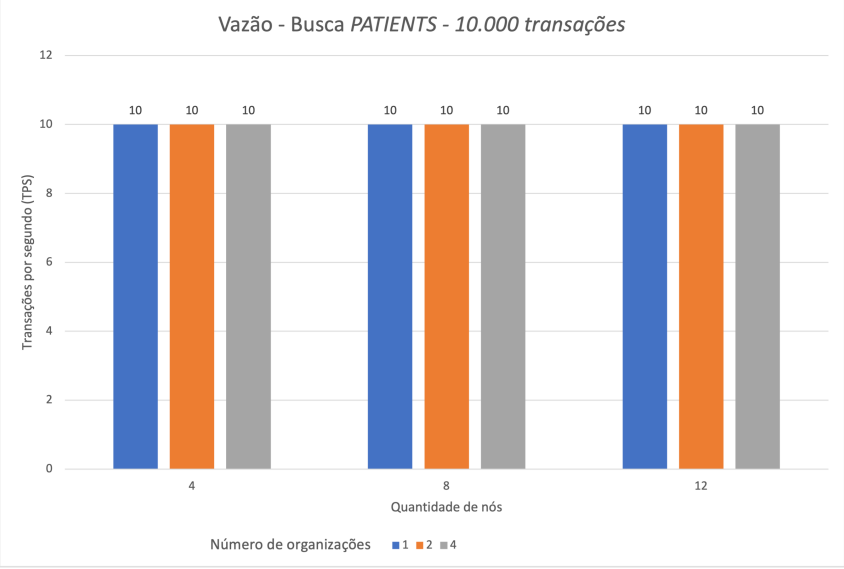
Figura 45 – Resultados para inserção de *INPUTEVENTS_MV*

Os resultados das inserções de *INPUTEVENTS_MV* estão apresentados na Figura 45. Para os valores de entradas e taxa de requisições testados, esta foi a única execução na qual houve uma variação na tendência esperada de aumento da latência média com maior número de organizações. Considerando a execução com 8 nós houve uma pequena queda na latência

de 9,11 para 8,93 quando comparada à execução com 2 e 4 organizações. Também houve uma pequena queda da latência de 9,62 na execução com doze máquinas e duas organizações para 9,5 com apenas uma organização. Apesar dessas diferenças, o desvio padrão mostra que as amostras coletadas se encontram no mesmo intervalo, o que reforça a necessidade de execuções em outros cenários para determinar se essa diferença se mantém na inserção desse arquivo.



(a) Latência 10.000 entradas e frequência 10



(b) Vazão 10.000 entradas e frequência 10

Figura 46 – Resultados para busca de *PATIENTS*

Como ilustrado na Figura 46a, latência média para a operação de busca obteve valores

similares dada a variação dos fatores analisados. A única tendência observada é o aumento da latência média com um maior número de organizações para a mesma quantidade de nós. As requisições de busca são atendidas pelos nós pertencentes à mesma organização do cliente que fez a solicitação. Assim, com apenas uma organização na rede é possível responder à demanda mais rapidamente uma vez que há mais nós disponíveis para essa função. Como demonstrado na Seção 8.4, quando aumentado o número de organizações, o número de nós que atendem a requisição de determinado cliente diminui na mesma proporção.

Os resultados encontrados nas execuções que consideraram 15.000 transações e frequência de acesso de 50 requisições por segundo são apresentados nas tabelas e gráficos a seguir. As Tabelas de 117 a 125 isolam os cenários testados, evidenciando as operações realizadas, as três inserções dos arquivos *MIMIC-III* (com *PATIENTS*, *PRESCRIPTIONS* e *INPUTEVENTS_MV*) e a busca com *PATIENTS*, indicando para cada uma a latência média, variância e desvio padrão em conjunto com a vazão. Como definido anteriormente, não houve execução da inserção de *D_ITEMS* para 15.000 transações.

As Tabelas de 117, 118 e 119 exibem as métricas de desempenho obtidas nas execuções com quatro nós na rede, sendo estes distribuídos em quatro, duas e uma organização.

Operação	Latência Média (s)	Variância	Desvio Padrão	Vazão (TPS)
Inserção *PATIENTS*	113,46	2475,2	49,75	35,4
Inserção *PRESCRIPTIONS*	265,64	11135,11	105,52	21,8
Inserção *INPUTEVENTS_MV*	288,41	13376,41	115,66	20,8
Busca *PATIENTS*	0,04	0,000284	0,0169	50

Tabela 117 – Quantidade de transações: 15.000 Frequência: 50 Nós: 4 Organização: 4

Operação	Latência Média (s)	Variância	Desvio Padrão	Vazão (TPS)
Inserção *PATIENTS*	61,45	631,48	25,13	40,8
Inserção *PRESCRIPTIONS*	205,01	8215,56	90,64	24,2
Inserção *INPUTEVENTS_MV*	228,12	9247,28	96,16	23,4
Busca *PATIENTS*	0,03	0,000055	0,0074	50

Tabela 118 – Quantidade de transações: 15.000 Frequência: 50 Nós: 4 Organização: 2

Operação	Latência Média (s)	Variância	Desvio Padrão	Vazão (TPS)
Inserção *PATIENTS*	40,39	173,34	13,17	44,1
Inserção *PRESCRIPTIONS*	176,73	7017,36	83,77	25,4
Inserção *INPUTEVENTS_MV*	207,64	8754,86	93,57	23,8
Busca *PATIENTS*	0,03	0,000042	0,0065	50

Tabela 119 – Quantidade de transações: 15.000 Frequência: 50 Nós: 4 Organização: 1

Os valores alcançados nas execuções com oito nós pertencentes a rede *blockchain*, com quatro, duas e uma organização são mostrados nas Tabelas 120 a 122.

Operação	Latência Média (s)	Variância	Desvio Padrão	Vazão (TPS)
Inserção *PATIENTS*	114,87	2376,18	48,75	35
Inserção *PRESCRIPTIONS*	266,03	10710,13	103,49	21,8
Inserção *INPUTEVENTS_MV*	301,85	13726,63	117,16	20,1
Busca *PATIENTS*	0,03	0,001489	0,0386	50

Tabela 120 – Quantidade de transações: 15.000 Frequência: 50 Nós: 8 Organização: 4

Operação	Latência Média (s)	Variância	Desvio Padrão	Vazão (TPS)
Inserção *PATIENTS*	76,92	1029,9	32,09	39,5
Inserção *PRESCRIPTIONS*	222,37	8369,46	91,48	23,8
Inserção *INPUTEVENTS_MV*	261,23	11966,41	109,39	21,4
Busca *PATIENTS*	0,03	0,000036	0,006	50

Tabela 121 – Quantidade de transações: 15.000 Frequência: 50 Nós: 8 Organização: 2

Operação	Latência Média (s)	Variância	Desvio Padrão	Vazão (TPS)
Inserção *PATIENTS*	61,59	597,74	24,45	41,4
Inserção *PRESCRIPTIONS*	209,51	7898,85	88,88	24,1
Inserção *INPUTEVENTS_MV*	238,38	11049,59	105,12	22,4
Busca *PATIENTS*	0,02	0,000039	0,0063	50

Tabela 122 – Quantidade de transações: 15.000 Frequência: 50 Nós: 8 Organização: 1

Já as Tabelas caracterizadas em 123 a 125 demonstram os valores obtidos nas métricas de desempenho com as execuções com doze nós e variando a quantidade de organizações em quatro, duas e uma, respectivamente.

Operação	Latência Média (s)	Variância	Desvio Padrão	Vazão (TPS)
Inserção *PATIENTS*	115,26	2379,74	48,78	34,7
Inserção *PRESCRIPTIONS*	268,6	11494,51	107,21	21
Inserção *INPUTEVENTS_MV*	297,59	13869,33	117,77	19,7
Busca *PATIENTS*	0,03	0,000501	0,0224	50

Tabela 123 – Quantidade de transações: 15.000 Frequência: 50 Nós: 12 Organização: 4

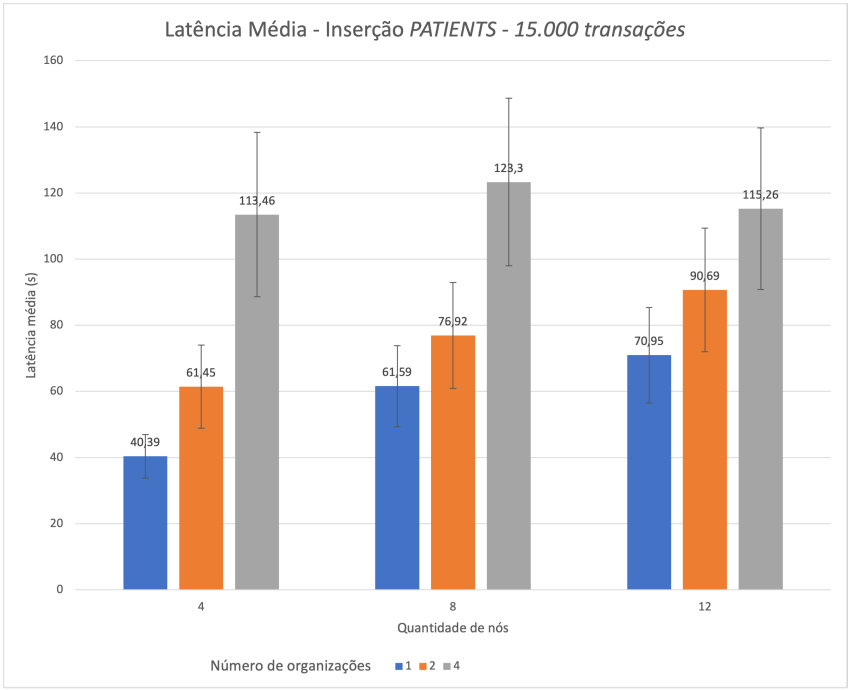
Operação	Latência Média (s)	Variância	Desvio Padrão	Vazão (TPS)
Inserção *PATIENTS*	90,69	1394,18	37,34	37,7
Inserção *PRESCRIPTIONS*	231,68	10086	100,43	22,7
Inserção *INPUTEVENTS_MV*	294,54	14567,79	120,7	19,8
Busca *PATIENTS*	0,03	0,000066	0,0081	50

Tabela 124 – Quantidade de transações: 15.000 Frequência: 50 Nós: 12 Organização: 2

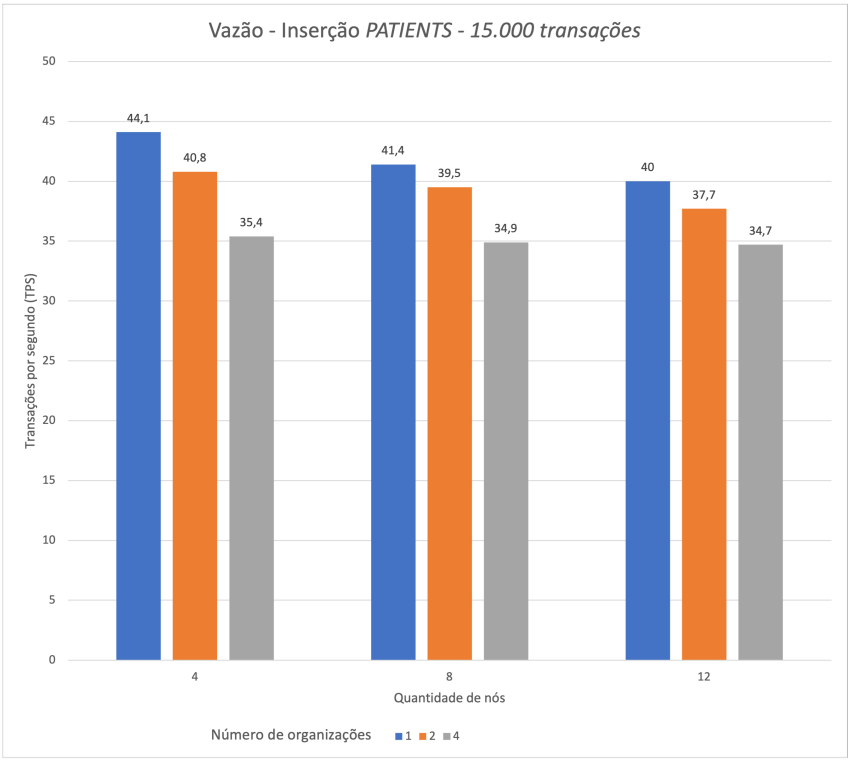
Operação	Latência Média (s)	Variância	Desvio Padrão	Vazão (TPS)
Inserção *PATIENTS*	70,95	828,96	28,79	40
Inserção *PRESCRIPTIONS*	216,37	8283,8	91,02	23,8
Inserção *INPUTEVENTS_MV*	271,43	13701,08	117,05	20,4
Busca *PATIENTS*	0,02	0,000032	0,0057	50

Tabela 125 – Quantidade de transações: 15.000 Frequência: 50 Nós: 12 Organização: 1

Os gráficos das Figuras 47 a 50 apresentam os resultados de latência média e vazão das inserções e busca realizadas com 15.000 entradas e frequência de requisição de 50 transações por segundo para as inserções dos arquivos *PATIENTS*, *PRESCRIPTIONS*, *INPUTEVENTS_MV* e a busca com *PATIENTS*. Como já mencionado, para o arquivo *D_ITEMS* não houve inserções considerando 15.000 entradas. Os resultados encontrados neste cenário têm maior variação com a modificação dos níveis dos fatores a serem avaliados neste experimento (número de organização e quantidade de nós em uma rede *blockchain*).



(a) Latência 15.000 entradas e frequência 50

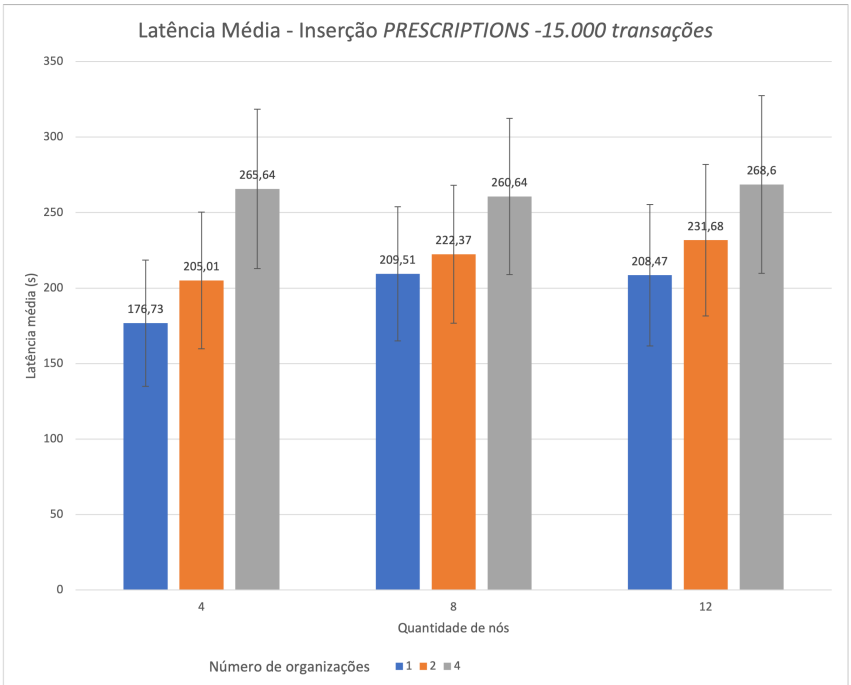


(b) Vazão 15.000 entradas e frequência 50

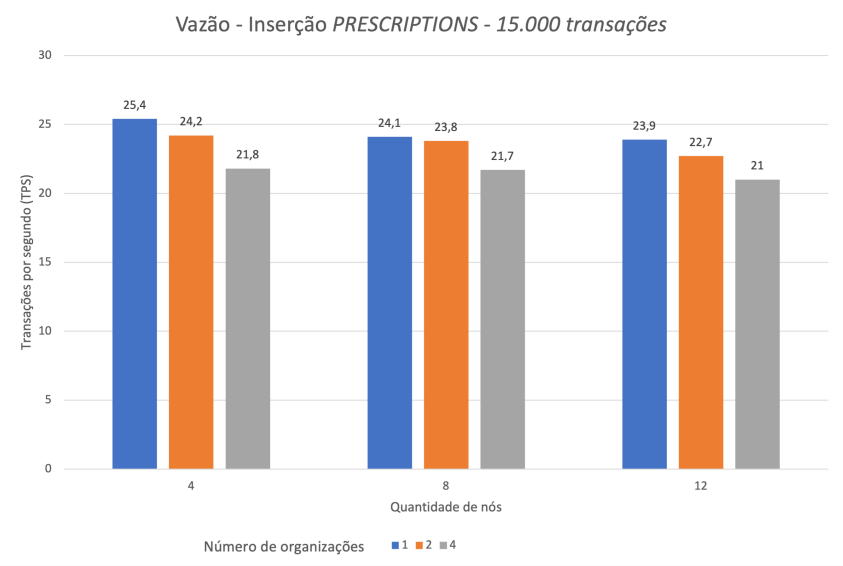
Figura 47 – Resultados para inserção de *PATIENTS*

O aumento do número de organizações para cada quantidade de nós testada (quatro, oito e doze nós na rede) levou a latências médias maiores e vazões menores. O maior número de nós também gerou, na maioria das execuções, maiores latências e menores vazões quando comparado isoladamente às redes com uma, duas e quatro organizações. Uma exceção ocorreu

na inserção de *PATIENTS* (Figura 47a), para oito nós e quatro organizações a latência média foi maior que em doze nós com a mesma quantidade de organizações.



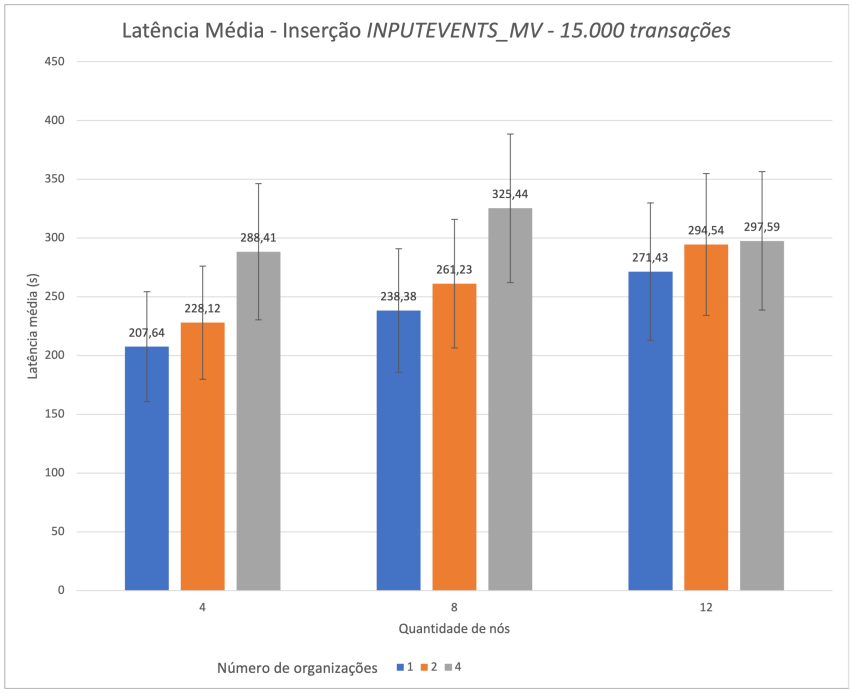
(a) Latência 15.000 entradas e frequência 50



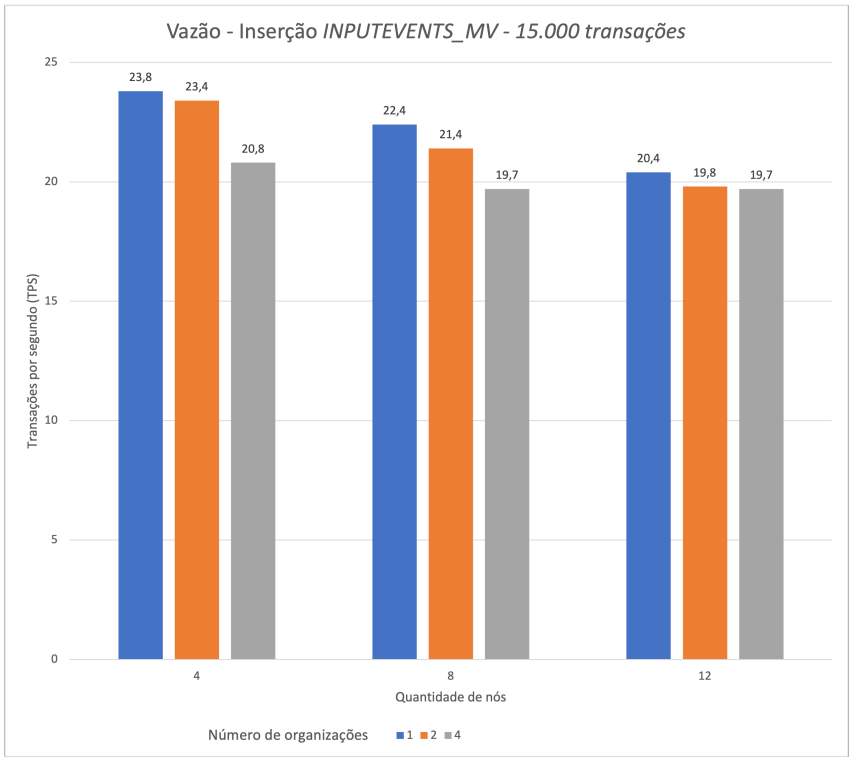
(b) Vazão 15.000 entradas e frequência 50

Figura 48 – Resultados para inserção de *PRESCRIPTIONS*

Ao contrário as execuções considerando 10.000 entradas e requisições chegando com taxa de dez por segundo, neste cenário de teste, as vazões das inserções se diferenciam ao modificar os fatores analisados. Isso permite a análise do impacto da quantidade de organizações e número de nós na métrica em questão. Além disso, é possível destacar como as diferenças dos arquivos utilizadas refletem na vazão para a inserção de dados na rede.

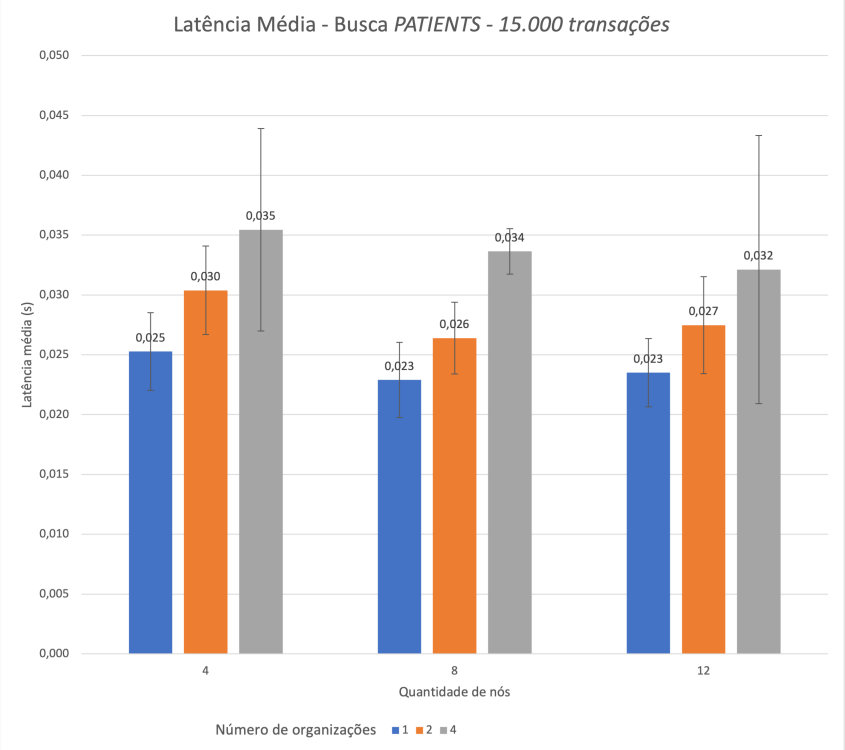


(a) Latência 15.000 entradas e frequência 50

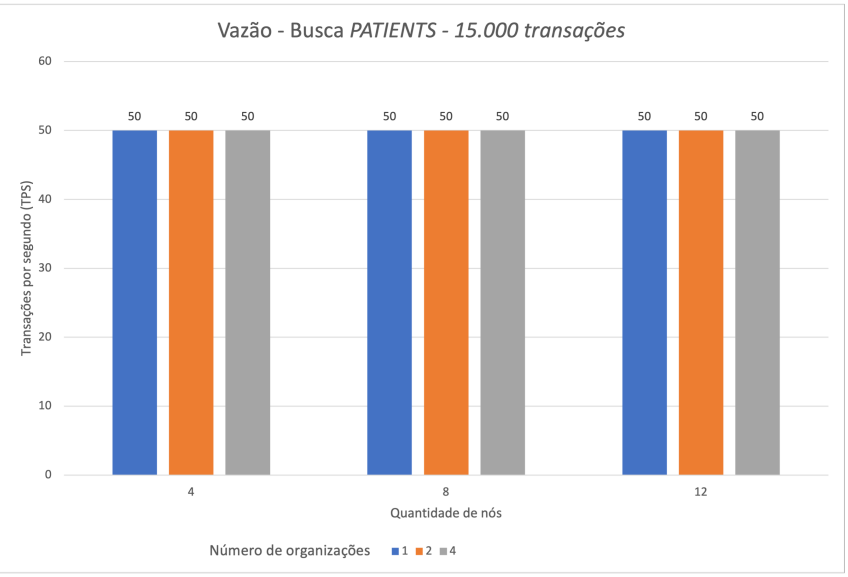


(b) Vazão 15.000 entradas e frequência 50

Figura 49 – Resultados para inserção de *INPUTEVENTS_MV*



(a) Latência 15.000 entradas e frequência 50



(b) Vazão 15.000 entradas e frequência 50

Figura 50 – Resultados para busca de *PATIENTS*

Mesmo com o aumento de requisições e taxa de chegada, a busca manteve resultados semelhantes aos encontrados na execução com 10.000 entradas e frequência de 10 requisições por segundo. A vazão foi de 10 para 50, acompanhando o aumento da taxa de requisições, como esperado.

8.6 Análise dos Resultados

Esta Seção tem como objetivo validar as hipóteses estabelecidas em função dos resultados obtidos por meio dos experimentos realizados.

8.6.1 Hipótese I

A primeira hipótese estabelece a relação entre o aumento do número de nós e a queda no desempenho da rede *blockchain*. Para apurar a veracidade desta hipótese considera-se os valores obtidos para latência. Em função dessa análise, foi verificada a normalidade de cada amostra de dados. O teste de *Cramér-von Mises* (THODE, 2002) com nível de significância 95% confirma a não-normalidade do conjunto de dados, os quais apresentam *p-valor* menor do que 0,05.

Dada a não-normalidade, o teste de *Wilcoxon* (WALPOLE *et al.*, 1993) com nível de significância 95% foi aplicado para verificar se as amostras são estatisticamente equivalentes ou não. Os *p-valores* encontrados comparando as latências nas execuções com 10.000 entradas e frequência de 10 requisições por segundo são mostrados nas Tabelas 126, 127 e 128, para uma, duas e quatro organizações na rede, respectivamente.

Operação	4 nós e 8 nós	4 nós e 12 nós	8 nós e 12 nós
Inserção *PATIENTS*	0	0	0
Inserção *D_ITEMS*	0	0	0
Inserção *PRESCRIPTIONS*	0	0	0
Inserção *INPUTEVENTS_MV*	0	0	0
Busca *PATIENTS*	0	0	0,221

Tabela 126 – *p-valores* considerando 1 organização

Para a busca na tabela *MIMIC-III PATIENTS* com 8 e 12 nós, ambas com uma organização, as amostras são estatisticamente similares, como mostrado na Figura 46a.

Operação	4 nós e 8 nós	4 nós e 12 nós	8 nós e 12 nós
Inserção *PATIENTS*	0,045	0	0
Inserção *D_ITEMS*	0	0	0
Inserção *PRESCRIPTIONS*	0	0	0
Inserção *INPUTEVENTS_MV*	0	0	0
Busca *PATIENTS*	0	0	0,051

Tabela 127 – *p-valores* considerando 2 organizações

Operação	4 nós e 8 nós	4 nós e 12 nós	8 nós e 12 nós
Inserção *PATIENTS*	0	0	0
Inserção *D_ITEMS*	4,07e-10	0	0
Inserção *PRESCRIPTIONS*	3,91e-12	0	0
Inserção *INPUTEVENTS_MV*	0,006	0	0
Busca *PATIENTS*	0	0	0

Tabela 128 – *p-valores* considerando 4 organizações

As Tabelas apresentadas anteriormente mostram que, com exceção de um cenário de teste (busca no arquivo *PATIENTS*), rejeita-se a hipótese de que as amostras são equivalentes. Desta forma, compara-se as médias amostrais obtidas no intuito de avaliar qual o impacto do número de nós na latência da rede *blockchain*.

Em apenas quatro cenários a latência média da rede com um menor número de nós é superior ao valor encontrado quando comparado a rede com mais organizações. São elas:

- Para o arquivo *D_ITEMS* quando comparada a latência de quatro e oito nós com uma organização, de 8,22 para 8,18 com o aumento do número de nós;
- Para o arquivo *PRESCRIPTIONS* em dois cenários. Com a rede composta por 4 nós e 2 organizações a latência média encontrada é de 8,75 e para oito nós é de 8,63 com duas organizações. E também na rede com quatro nós e oito nós com quatro organizações em ambas, uma queda de 8,82 para 8,69;
- Para *INPUTEVENTS_MV* a diminuição da latência média ocorreu entre as execuções com a rede de quatro nós e oito nós para quatro organizações, de 9,25 para 8,93;

Apesar dessas diferenças destacadas, os gráficos das Figuras 43a, 44a e 45a mostram que o desvio padrão dessas inserções coloca os resultados no mesmo intervalo.

A vazão apresenta o valor constante de 9,9 para as inserções de todas os arquivos *MIMIC-III* e o valor constante 10 para a busca em *PATIENTS*. Em função dessas restrições, não é possível analisar qual o impacto do aumento do número de nós na rede *blockchain* para essa métrica.

Em vista dos resultados encontrados nessas execuções não serem constantes, e o aumento da latência com mais nós e a vazão não variarem com a alteração desses fatores, outro cenário foi proposto para avaliar o desempenho da rede *blockchain* quando aumentados o número e a frequência das requisições realizadas. Novamente, o teste de *Cramér-von Mises* com nível de significância de 95% foi aplicado nas amostras da latência e a normalidade foi rejeitada. Dessa forma, o teste de *Wilcoxon* com nível de significância de 95% foi aplicado nas amostras para validar, ou não a hipótese de que as amostras são equivalentes. Os *p-valores* resultantes da aplicação do teste de hipótese nas execuções com 15.000 entradas e frequência de 50 requisições por segundo são mostrados nas Tabelas 129 , 130 e 131.

Operação	4 nós e 8 nós	4 nós e 12 nós	8 nós e 12 nós
Inserção *PATIENTS*	0	0	0
Inserção *PRESCRIPTIONS*	0	0	0,0001
Inserção *INPUTEVENTS_MV*	0	0	0
Busca *PATIENTS*	0	0	0

Tabela 129 – *p-valores* considerando 1 organização

Operação	4 nós e 8 nós	4 nós e 12 nós	8 nós e 12 nós
Inserção *PATIENTS*	0	0	0
Inserção *PRESCRIPTIONS*	0	0	0
Inserção *INPUTEVENTS_MV*	0	0	0
Busca *PATIENTS*	0	0	0

Tabela 130 – *p-valores* considerando 2 organizações

Operação	4 nós e 8 nós	4 nós e 12 nós	8 nós e 12 nós
Inserção *PATIENTS*	0	0	0
Inserção *PRESCRIPTIONS*	0	6,10e-15	0
Inserção *INPUTEVENTS_MV*	0	0	0
Busca *PATIENTS*	0	0	0

Tabela 131 – *p-valores* considerando 4 organizações

Em todos os testes de hipótese realizados para este cenário, rejeita-se a hipótese de que as amostras são equivalentes. Outra vez, a latência média é examinada para verificar qual o impacto do aumento do número de nós na rede *blockchain* para essa métrica.

Há pouco cenários nos quais a latência média de uma rede com menos nós é maior que redes com mais nós para a mesma quantidade de organizações, são esses:

- Para o arquivo *PATIENTS* a latência caiu da rede de oito nós e para doze, com ambas divididas em quatro organização, de 128,3 para 115,26;
- Para o arquivo *PRESCRIPTIONS* essa diminuição ocorreu em dois cenários. Para a rede com oito nós e doze nós e uma organização cada, a latência média foi de 209,51 para 208,47. Outro cenário foi entre a rede de quatro nós e oito nós com quatro organizações, de 265,64 para 260,64;
- Para *INPUTEVENTS_MV* a queda na latência de 325,44 para 297,59 entre as execuções com oito nós e doze com quatro organizações cada;

Os gráficos das Figuras 47a, 48a e 49a ilustram os cenários destacados. Para esses cenários destacados, a vazão ou diminuiu com o aumento do número de nós ou se manteve o mesmo, como mostram os gráficos das Figuras 47b, 48b e 49b.

A queda da latência pode indicar que uma número de nós dentro de uma organização tem um impacto positivo nessa métrica. Outro ponto a destacar é a característica dos dados inseridos na rede. Como já citado, os arquivos apresentam quantidades diferentes de colunas e indexações, o que influência na latência na inserção dos dados. Nessas execuções, a latência das inserções de *PATIENTS* e *INPUTEVENTS_MV* tiveram uma queda quando a rede era composta por doze nós em comparação com oito nós considerando quatro organizações. A inserção de *PRESCRIPTIONS* nesses mesmos cenário seguiu a tendencia de aumento da latência. A maior quantidade de indexações provavelmente impede que a inserção desse arquivo tenha a mesma variação.

A vazão das inserções mostrou-se proporcional à quantidade de nós da rede, diminuindo conforme aumentam-se os nós da rede, como mostra os gráficos da Seção 8.5. Para as operações de busca, a vazão se manteve constante em todos cenários avaliados.

A partir desses resultados, pode-se afirmar que há uma tendência de deterioração do desempenho da rede, em função do aumento da latência e diminuição da vazão, quando aumentam-se o número de nós da rede. Porém é necessário realizar mais experimentos para isolar os elemento da rede *blockchain*, como a quantidade de nós dentro da organização, os tipos de dados armazenados, considerando dados com o mesmo número de colunas e diferente de indexações e o contrário para assim determinar como a rede impacta, principalmente, na latência da aplicação *blockchain*.

8.6.2 Hipótese II

A segunda hipótese estabelece que o aumento das organizações impacta negativamente no desempenho da rede *blockchain*, aumentando a latência e diminuindo a vazão. Foram avaliados três cenários distintos com 1, 2 e 4 organizações e com diferentes números de nós da rede.

A verificação da normalidade dos dados em função da aplicação do teste de *Cramér-von Mises* (THODE, 2002) com nível de significância 95% já foi apresentada para a hipótese anterior e, por se tratar do mesmo conjunto de dados, permanece a mesma. A aplicação do teste comprova a distribuição não-normal do conjunto de dados.

O teste de *Wilcoxon* (WALPOLE *et al.*, 1993) com nível de significância 95% foi utilizado para verificar a equivalência das amostras coletadas. As Tabelas 132, 133 e 134 mostram os *p-valores* obtidos comparando as quantidade de organizações consideradas neste estudo para 10.000 entradas e taxa de chegada de requisições em 10 por segundo.

Operação	1 e 2 orgs	1 e 4 orgs	2 e 4 orgs
Inserção *PATIENTS*	0,012	0	0
Inserção *D_ITEMS*	0	0	0
Inserção *PRESCRIPTIONS*	1,1e-16	9,99e-16	0,003
Inserção *INPUTEVENTS_MV*	0	0	0
Busca *PATIENTS*	0	0	0

Tabela 132 – *p-valores* considerando 4 nós

Operação	1 e 2 orgs	1 e 4 orgs	2 e 4 orgs
Inserção *PATIENTS*	0	0,002	0
Inserção *D_ITEMS*	0	0	4,56e-6
Inserção *PRESCRIPTIONS*	6,43e-6	0	0
Inserção *INPUTEVENTS_MV*	0	0	0,0003
Busca *PATIENTS*	0	0	0

Tabela 133 – *p-valores* considerando 8 nós

Operação	1 e 2 orgs	1 e 4 orgs	2 e 4 orgs
Inserção *PATIENTS*	0	0	0
Inserção *D_ITEMS*	0,0007	0	0
Inserção *PRESCRIPTIONS*	0	0	0
Inserção *INPUTEVENTS_MV*	0,064	0	0
Busca *PATIENTS*	0	0	0

Tabela 134 – *p-valores* considerando 12 nós

As inserções de *INPUTEVENTS_MV*, contrariando as demais amostras, mostram-se estatisticamente equivalentes quando consideradas uma e duas organizações para doze nós na rede.

As Tabelas 135, 136 e 137 apresentam os *p-valores* provenientes da realização dos testes de hipóteses para os execuções realizadas que consideraram 15.000 entradas e 50 requisições por segundos.

Operação	1 e 2 orgs	1 e 4 orgs	2 e 4 orgs
Inserção *PATIENTS*	0	0	0
Inserção *PRESCRIPTIONS*	0	0	0
Inserção *INPUTEVENTS_MV*	0	0	0
Busca *PATIENTS*	0	0	0

Tabela 135 – *p-valores* considerando 4 nós

Operação	1 e 2 orgs	1 e 4 orgs	2 e 4 orgs
Inserção *PATIENTS*	0	0	0
Inserção *D_ITEMS*	0	0	0
Inserção *PRESCRIPTIONS*	0	0	0
Inserção *INPUTEVENTS_MV*	0	0	0
Busca *PATIENTS*	0	0	0

Tabela 136 – *p-valores* considerando 8 nós

Operação	1 e 2 orgs	1 e 4 orgs	2 e 4 orgs
Inserção *PATIENTS*	0	0	0
Inserção *D_ITEMS*	0	0	0
Inserção *PRESCRIPTIONS*	0	0	0
Inserção *INPUTEVENTS_MV*	0	0	0
Busca *PATIENTS*	0	0	0

Tabela 137 – *p-valores* considerando 12 nós

Foi possível perceber, por meio da análise das tabelas anteriores, que os valores obtidos para *p-valor*, são menores do que 0,05, em todas as circunstâncias consideradas. Desta forma, afirma-se que as amostras não são estatisticamente equivalentes.

Dessa forma, a partir da ponderação das médias obtidas para a métrica latência é possível estabelecer que o aumento do número de organizações também aumenta a latência da rede *blockchain*, resultando em um impacto negativo no desempenho do ambiente da rede *blockchain*.

Para a métrica vazão, tanto para as inserções que consideram 10.000 transações e taxa de chegada de dez quanto para as execuções com 15.000 e frequência de 50 requisições por segundo, houve uma queda na métrica com o aumento do número de organizações na rede. No cenários avaliados da busca, a vazão se manteve constante, se equiparando a chegada de requisições em ambos. Assim, para a busca não é possível afirmar o impacto da distribuição dos nós dentro das organizações nessa métrica em questão com as execuções avaliadas.

Através dessa análise é possível afirmar que o aumento do número de organizações gera uma piora no desempenho da rede *blockchain* para as inserções de dados, com a queda da vazão e aumento da latência média.

Os gráficos mostram que a latência é a métrica mais afetada pelo uso de mais organizações na rede. O maior aumento dessa métrica ocorreu na inserção de *PATIENTS* (Figura 47a) com a rede composta por quatro nós. Entre quatro e uma organização houve um aumento de 180,09% na latência média. A menor diferença percentual foi a inserção de *INPUTEVENTS_MV* (Figura 49a) nas redes considerando doze nós e duas e quatro organizações com um aumento de 1,04% na latência.

8.7 Considerações Finais

Este Capítulo apresentou o experimento realizado para analisar o desempenho da *blockchain Hyperledger Fabric* em diferentes configurações de rede, em termos de número de nós integrantes da rede e quantidade de organizações. A análise dos resultados permite verificar que o aumento do número de máquinas não piora o desempenho na mesma proporção em que são acrescentados nós na rede *blockchain*, porém influencia as métricas testadas, aumentando a latência e diminuindo a vazão. Além disso os resultados encontrados mostram que o aumento do número de organizações impacta negativamente no desempenho da aplicação *blockchain* nas mesmas métricas.

O próximo Capítulo propõe uma análise conjunta dos resultados encontrados nos experimento. O objetivo é destacar quais são as principais contribuições de cada experimento individualmente para a discussão geral deste trabalho, a usabilidade da *blockchain* como ferramenta de armazenamento e distribuição de dados heterogêneos.

CAPÍTULO

9

CONCLUSÕES

9.1 Considerações Iniciais

Este Capítulo apresenta as principais conclusões obtidas com a realização deste trabalho. Inicialmente, é apresentada uma discussão dos resultados encontrados, retomando as execuções realizadas. As principais contribuições da realização dos experimentos descritos nos Capítulos anteriores, são destacadas em seguida.

As produções científicas elaboradas e publicadas durante o período de desenvolvimento do projeto, estão ilustradas na sequência. O Capítulo encerra com a indicação das limitações e restrições do trabalho realizado e indicações de possíveis trabalhos futuros com a partir deste.

9.2 Discussão dos Resultados

O objetivo geral deste trabalho foi avaliar o desempenho da *blockchain* como ferramenta para o armazenamento e distribuição de dados heterogêneos. Para tanto, foi adotado o *Hyperlegder Fabric* e foram realizados quatro experimentos com a finalidade de examinar, não só o desempenho dessa aplicação, mas também possíveis desafios e possibilidades ao se usar a *blockchain* para registro de dados heterogêneos.

O primeiro experimento analisa como o fluxo de requisições na rede *blockchain* pode afetar o desempenho da aplicação. O foco deste primeiro experimento foi analisar como a quantidade de transações e a sua frequência de chegada podem influenciar a latência e a vazão do sistema para as operações de inserção e busca de dados. Os cenários de teste nos nossos experimentos variaram a quantidade de transações na faixa de 1.000 a 10.000, e a frequência de submissões (taxa de envio) de transações de leitura ou escrita de 3 a 10 requisições por segundo. A faixa de valores para a quantidade de transações representa diferentes volumes de transações a serem processadas.

Os resultados deste experimento mostraram que a *blockchain* tem um alto custo para a inclusão dos dados heterogêneos na rede, em função da quantidade e natureza dos dados a cada transação e também pelo processo de consenso e criação dos blocos pelos nós da rede. Os experimentos posteriores a este primeiro, mostraram que a rede *blockchain* pode ser estruturada adequadamente para tentar minimizar esse alto custo de inclusão dos blocos, diminuindo o número de nós validadores e criando-se blocos maiores. Tais configurações podem melhorar a vazão de tais requisições de inserção de dados, se estas chegarem com uma alta frequência.

Dentro do objetivo deste trabalho, este primeiro experimento foi uma investigação inicial e levantou questionamentos sobre como os elementos da *blockchain Hyperledger Fabric* tais quais armazenamento, consenso, criação de blocos e rede influenciam nos valores encontrados na latência e vazão. Os demais experimentos deste trabalho foram definidos em função dos resultados obtidos no primeiro experimento.

O segundo experimento focou no armazenamento e avaliou duas formas de registro de informações da *blockchain* do *Fabric*. A primeira forma de armazenamento não considerou o uso de um banco de dados externo e a segunda associou à cadeia o banco de dados *CouchDB*. Os resultados mostram que o banco de dados externo influencia fortemente o desempenho da *blockchain*, provocando uma perda de vazão de até 86,77% e um impressionante aumento da latência de até 261.420,83% da escrita de dados na rede. Novamente, o ponto crítico mostrou ser a escrita na rede quando há uma alta taxa de chegada das requisições.

Estes valores são importantes para qualificar o desempenho da *blockchain* com as duas formas de armazenamento; porém não deve ser o único fator a ser considerado na escolha da configuração no desenvolvimento de uma aplicação *blockchain*.

O uso do banco de dados *CouchDB* com o *Fabric*, embora represente uma sobrecarga significativa à vazão no atendimento das requisições de escrita, possibilita a realização posterior de buscas mais complexas (*rich-queries*), dessa forma facilitando os acessos aos dados na rede. O uso das funcionalidades disponibilizados pelo *CouchDB* e similares permitem a agregação, redução (*map-reduce*), união de conjuntos de dados, além de facilitarem a busca utilizando critérios complexos, como buscar somente as transações que contêm um critério definido (HAN *et al.*, 2011). Além disso, a associação com um banco de dados permite que na cadeia seja especificado qual participante e quando uma transação foi criada; já no banco de dados apenas os dados são armazenados. Dessa forma, o uso combinado de *blockchain* com banco de dados garante o registro das transações de maneira imutável, sem a necessidade de um intermediário, e com os benefícios de separar os dados da cadeia. Outra vantagem do uso de um banco de dados é a possibilidade de integração, caso já exista um banco de dados em uso. Neste caso, a depender da situação, a *blockchain* pode ser acrescentada à infraestrutura já existente, utilizada por uma aplicação baseada em banco de dados.

O uso do banco de dados na *blockchain* é vantajoso em cenários em que não há grande fluxo de requisições de inserção de dados em função do tempo de acesso ao mesmo. A busca,

por ser uma operação menos custosa em termos de operação, não tem perda significativa de desempenho quando comparada ao não uso do banco de dados. Caso haja a necessidade de grande fluxo de inserções na rede, os desenvolvedores devem considerar uma análise criteriosa para projetar a estrutura da *blockchain*, pois as perdas de desempenho podem afetar a escalabilidade do sistema.

É importante destacar que o segundo experimento mostra como o *Fabric* trabalha com dados heterogêneos em ambas as formas de armazenamento às quais oferece suporte. O *Ethereum*, por exemplo, não oferece estrutura nativa para o armazenamento de dados fora da cadeia. Esta plataforma não foi desenvolvida para o armazenamento de grande volume de dados, justamente por oferecer somente suporte para o registro de informações na cadeia. Os nós participantes da rede armazenam toda a cadeia e caso esta se expanda para grandes quantidades de dados, é inviável que todos dos nós continuem em execução (COMMUNITY, 2021).

O terceiro experimento destaca o processo de validação de transações e a criação de blocos, sendo que ambos são gargalos de desempenho conhecidos de redes *blockchain*. Os resultados obtidos com o segundo experimento, mostram que o uso do *CouchDB* para armazenamento das transações tem grande impacto no desempenho da rede *blockchain*. Neste terceiro experimento, investiga-se se a mudança dos parâmetros na criação de blocos ou como o processo de consenso pode impactar positivamente no desempenho da rede, mesmo com o uso do banco de dados em questão.

Os cenários testados mostram o impacto do processo de validação das transações antes destas transações serem inseridas na cadeia. No entanto, é possível perceber que o impacto deste fator não é tão representativo quanto o processo de criação de blocos em si.

Futuros arquitetos e desenvolvedores da *blockchain* devem ser criteriosos com relação ao tamanho e tempo limite de criação do bloco com o *Hyperledger Fabric*. Este experimento aponta a relevância desses fatores na rede do *Fabric* quando usados dados heterogêneos, com diminuição da latência em até 6441,98% e aumento da vazão em até 83,33% entre os tamanhos de bloco comparados, quando se aumentam o tamanho do bloco em bytes e o tempo para a criação do bloco em segundos.

Vale destacar que os resultados neste trabalho espelham a realidade do *Fabric*, o qual possui características específicas. O *Fabric* possui estratégias próprias para o processo de consenso e criação de blocos, os quais são considerados problemas nativos de redes *blockchain*. Dentro do *Fabric*, o processo de criação de um bloco é determinístico, ou seja, uma vez criado e inserido na cadeia, o bloco é imutável e sua posição é garantida, independentemente da configuração do *ordering service* (como descrita em 2.5.2). Para o processo de consenso há, por exemplo, a possibilidade de determinar quais e quantos nós são necessários para validar uma transação. No *Bitcoin*, por exemplo, a criação de blocos é realizada com o algoritmo do *Proof-of-Work* (explicado na Seção 2.3.3) e precisa de seis blocos criados posteriormente para garantir sua posição na cadeia. A estratégia adotada pelo *Bitcoin* deriva da necessidade de se

garantir a criação de blocos válidos em um ambiente no qual a rede é pública e os participantes não são conhecidos, diferentemente do contexto previsto para o *Fabric*, onde a rede é privada e os participantes conhecidos.

Uma alternativa ao *Fabric* no *Hyperlegder* poderia ser o *Sawthooth*, o qual tem como característica a concentração das funcionalidades de processamento de transações em nós denominados *validators*. Ao contrário do *Fabric* que distribui as funcionalidades entre os nós validadores e nós do tipo *orderer*, o *Sawtooth* centraliza o processo de validação de transações e blocos, além da criação de blocos, nos nós *validators*, armazenando nos mesmos uma cópia da cadeia.

O quarto e último experimento realizado neste trabalho teve como objetivo investigar o impacto do tamanho da rede e a divisão da mesma em organizações, no desempenho do *Fabric*. Nas execuções realizadas é possível perceber que a quantidade de nós na rede *blockchain* impacta o desempenho da mesma, piorando o desempenho da vazão e latência quando se aumenta a quantidade de nós participantes que compartilham informações na *blockchain*.

Criar mais organizações gera barreiras na comunicação entre os participantes da rede, influenciando na distribuição das transações na *blockchain*. Os resultados obtidos neste experimento mostram que dividir a rede *blockchain* em organizações causa deterioração do desempenho, o que se traduziu, no nosso caso, em um aumento de latência de até 180,09% e diminuição da vazão em até 24,58% entre os cenários avaliados.

As execuções realizadas no experimento quatro foram configuradas considerando aspectos apresentados nos experimentos anteriores. O fluxo de dados analisado por este experimento foi definido com base nos resultados do primeiro e segundo experimento realizados, os quais mostraram que 10.000 requisições e taxa de chegada de dez requisições por segundo atingem o pico de atendimento da rede por nós definida, quando considerado o uso do banco de dados. O uso do banco de dados é estabelecido devido ao interesse em investigar o impacto do seu acesso no desempenho quando aumentado o número de nós na rede. O processo de criação do bloco foi estabelecido em função do melhor cenário, em termos de latência e vazão, identificado no terceiro experimento, que utiliza tamanho do bloco com 500 transações e tempo de criação de dez segundos.

No primeiro fluxo de dados utilizado no quarto experimento, é possível observar que a latência média variou pouco, se mantendo no mesmo intervalo quando analisado com o desvio padrão, mesmo com o dobro ou o triplo de nós usados nos experimentos anteriores. Estes resultados validam os resultados obtidos no terceiro experimento deste trabalho que evidencia o impacto do processo de criação do bloco no desempenho de uma rede *blockchain Fabric*. Também é importante apontar que para esse fluxo de requisições o banco de dados não se mostra um gargalo na rede, frente ao tempo para a criação dos blocos.

Os experimentos realizados neste trabalho indicam que, sob a ótica de desempenho e

não funcionalidade, futuros desenvolvedores de soluções sobre o *Hyperledger Fabric* devem considerar com cuidado aspectos como a frequência de requisições (principalmente), o tamanho dos blocos criados (blocos grandes apresentam melhores desempenho) e intervalo de criação suficiente de tais blocos. O aumento do número de nós na rede e o uso do banco de dados, passam a ser secundários quando comparados a estes aspectos acima, e não se mostram gargalos representativos nesse cenário.

9.3 Principais Contribuições

A partir da condução deste trabalho, obteve-se um conjunto de contribuições que buscam apoiar a comunidade de interesse e desenvolvedores a compreender melhor o desempenho da *blockchain* frente a diferentes condições. Dentre as principais contribuições tem-se:

- Levantamento bibliográfico que apresenta uma síntese sobre as lacunas e oportunidades de pesquisa na área de avaliação de desempenho de redes *blockchain* até a data de realização desta pesquisa;
- Identificação e uso de dados hospitalares reais em rede *blockchain*;
- Avaliação experimental de fatores característicos de redes *blockchain*, tais quais o processo de criação de blocos e validação de transações;
- Avaliação experimental de fatores relacionados ao uso de dados heterogêneos em redes *blockchain*, sendo eles o estudo do desempenho com diferentes fluxos de dados e o uso do banco de dados associado a rede;
- Avaliação experimental de fatores relacionados ao uso da plataforma *Hyperledger Fabric*, como a análise da quantidade de organizações na rede;
- Indicação de fatores de configuração e uso das redes *blockchain* com o *Hyperledger Fabric* que afetam o desempenho das aplicações desenvolvidas sobre tais tecnologias, o que pode nortear futuras otimizações em projetos desta natureza.

9.4 Produção Científica

Durante a condução do trabalho, ao passo que os estudos experimentais foram desenvolvidos, artigos científicos foram desenvolvidos e publicados em conferências especializadas na área.

Trabalhos completos publicados em anais de congressos

1. SPENGLER, ANA CAROLINE FERNANDES; SOUZA, PAULO SÉRGIO LOPES DE . **Avaliação de desempenho do Hyperledger Fabric com banco de dados para o**

armazenamento de grandes volumes de dados médicos. In: Workshop em Desempenho de Sistemas Computacionais e de Comunicação, 2021, Brasil. Anais do XX Workshop em Desempenho de Sistemas Computacionais e de Comunicação (WPerformance 2021), 2021. p. 61-73.

2. SPENGLER, ANA CAROLINE FERNANDES; SOUZA, PAULO SERGIO LOPES DE . **The impact of using CouchDB on Hyperledger Fabric performance for heterogeneous medical data storage**. In: 2021 XLVII Latin American Computing Conference (CLEI), 2021, Cartago. 2021 XLVII Latin American Computing Conference (CLEI), 2021. p. 1.

Resumos expandidos publicados em anais de congressos

1. SPENGLER, ANA CAROLINE FERNANDES; SOUZA, P. S. L. . **Rumo à Avaliação do Ethereum e do Hyperlegder Fabric com Dados Heterogêneos**. In: Escola Regional de Arquitetura de Computadores e Processamento de Alto Desempenho (ERAD-SP 2020), 2020. Anais da Escola Regional de Arquitetura de Computadores e Processamento de Alto Desempenho (ERAD-SP 2020), 2020. p. 1-4.

9.5 Limitações

Os experimentos realizados neste trabalho tiveram como objetivo avaliar um conjunto abrangente de aspectos relacionados ao uso da *blockchain* com o *Hyperledger Fabric* no contexto de armazenamento e distribuição de dados heterogêneos. No entanto, houve aspectos da plataforma utilizada que não foram analisados.

Uma limitação, por exemplo, foi a utilização de um único *ordering service* na implementação "*Solo*", em que apenas um nó é responsável pela ordenação das transações e criação do bloco. O *Fabric*, em sua versão avaliada neste trabalho, oferece suporte a outras implementações que possibilitam ter um grupo de nós com essa função.

Outro aspecto da plataforma não explorado foi a avaliação da rede com mais de um *channel*. Tais *channels* representam uma funcionalidade do *Fabric* que permite que, em uma rede, nós sejam agrupados (dentro de um mesmo *channel*), e compartilhem a administração da rede e dados exclusivamente entre si. Um nó pode ser incluído em múltiplos *channels*.

Foram utilizados um máximo de 12 nós participantes da rede *blockchain* no presente trabalho, estando estes localizados na mesma infraestrutura de rede. Além disso, os experimentos desenvolvidos durante a realização deste projeto foram avaliados em função de métricas de desempenho descritas na Seção 4.2. Não foram feitas avaliações considerando consumo de energia e nem aspectos referentes ao uso de recursos computacionais, tais quais uso de CPU, memória, disco, dentre outros.

Por fim, uma das limitações deste trabalho é não apresentar um modelo para construção de redes e arquiteturas no *Hyperledger Fabric*. A análise do material serve como inspiração e motivação para novos projetos e experimentos ainda que o trabalho, em si, não constitua em um protótipo para implementação de uma rede blockchain *Hyperledger Fabric*.

9.6 Trabalhos Futuros

Considerando as contribuições e limitações do projeto realizado, alguns estudos futuros podem ser direcionados. Um possível estudo futuro é a utilização de outros tipos de formato de dados, tais como imagens, vídeos, áudio e outros tipos de formatos multimídia. Outra possibilidade é extrapolar as análises realizadas para outras plataformas *blockchain*, tais como *Ethereum*, *Hyperledger Sawtooth*, *Hyperldger Iroha*, entre outras.

Pode-se também utilizar outros métodos de armazenamento e processamento de dados distribuídos. O objetivo aqui seria avaliar o custo de desenvolvimento, implantação e manutenção de sistemas *blockchain*, quando comparados a sistemas baseados em bancos de dados como o *CouchDB* e o *MongoDB* ou ferramentas de armazenamento distribuídos como *Apache Spark*, *Apache Hadoop* e outros.

Uma possível linha de investigação inclui a realização de uma análise com fatorial simples sobre os fatores considerados nesse trabalho, para averiguar a influência de um fator sobre o outro. Ao isolar esses fatores em dois níveis, é possível fazer comparações mais precisas para melhor generalizar os resultados.

O *Hyperledger Caliper* também pode ser instrumentado para coletar informações do desempenho de cada etapa de uma transação na rede *blockchain*. Assim, é possível verificar possíveis o gargalos de desempenho da rede de forma precisa.

Por fim, conforme indicado nas limitações, podem ser avaliadas as características do *Hyperledger Fabric* não exploradas neste trabalho, como o uso de outras implementações do *ordering service* e a análise do impacto dos *channels* na rede. Além disso, pode ser investigada uma rede com maior número de nós que foram estudados no contexto desta dissertação.

REFERÊNCIAS

ABDELLATIF, T.; BROUSMICHE, B. Formal verification of smart contracts based on users and blockchain behaviors models. In: **2018 9th IFIP International Conference on New Technologies, Mobility and Security (NTMS)**. [S.l.: s.n.], 2018. p. 1–5. ISSN 2157-4960. Citado na página 70.

ANDROULAKI, E.; BARGER, A.; BORTNIKOV, V.; CACHIN, C.; CHRISTIDIS, K.; CARO, A. D.; ENYEART, D.; FERRIS, C.; LAVENTMAN, G.; MANEVICH, Y. *et al.* Hyperledger fabric: a distributed operating system for permissioned blockchains. In: ACM. **Proceedings of the Thirteenth EuroSys Conference**. [S.l.], 2018. p. 30. Citado nas páginas 54 e 56.

ANTONOPOULOS, A. M. **Mastering Bitcoin: Programming the Open Blockchain**. 2nd. ed. [S.l.]: O'Reilly Media, Inc., 2017. ISBN 1491954388, 9781491954386. Citado nas páginas 13, 33, 34, 35, 39, 41, 42, 43, 45, 46, 50 e 51.

ATTRIBUTION, C. C. **Hyperledger Cello**. 2019. Acessado em 23/05/2021. Disponível em: <https://hyperledger-cello.readthedocs.io/en/latest/getting-started>. Citado na página 54.

AZARIA, A.; EKBLAW, A.; VIEIRA, T.; LIPPMAN, A. **MedRec: Using Blockchain for Medical Data Access and Permission Management**. 2016. 25-30 p. Disponível em: <doi.ieeecomputersociety.org/10.1109/OBD.2016.11>. Citado na página 27.

BAI, X.; CHENG, Z.; DUAN, Z.; HU, K. Formal modeling and verification of smart contracts. In: **Proceedings of the 2018 7th International Conference on Software and Computer Applications**. New York, NY, USA: ACM, 2018. (ICSCA 2018), p. 322–326. ISBN 978-1-4503-5414-1. Disponível em: <http://doi.acm.org/10.1145/3185089.3185138>. Citado na página 70.

BALIGA, A.; SOLANKI, N.; VEREKAR, S.; PEDNEKAR, A.; KAMAT, P.; CHATTERJEE, S. Performance characterization of hyperledger fabric. In: **2018 Crypto Valley Conference on Blockchain Technology (CVCBT)**. [S.l.: s.n.], 2018. p. 65–74. Citado nas páginas 76, 77 e 79.

BHARGAVAN, K.; DELIGNAT-LAVAUD, A.; FOURNET, C.; GOLLAMUDI, A.; GONTHIER, G.; KOBEISSI, N.; KULATOVA, N.; RASTOGI, A.; SIBUT-PINOTE, T.; SWAMY, N.; ZANELLA-BÉGUELIN, S. Formal verification of smart contracts: Short paper. In: **Proceedings of the 2016 ACM Workshop on Programming Languages and Analysis for Security**. New York, NY, USA: ACM, 2016. (PLAS '16), p. 91–96. ISBN 978-1-4503-4574-3. Disponível em: <http://doi.acm.org/10.1145/2993600.2993611>. Citado na página 71.

BLOCKCHAINS, . **Apache CouchDB**. 2020. Acessado em 18/07/2020, às 23:06. Disponível em: <https://101blockchains.com/permissioned-vs-permissionless-blockchains/>. Citado na página 41.

BORE, N.; KARUMBA, S.; MUTAHI, J.; DARNELL, S. S.; WAYUA, C.; WELDEMARIAM, K. **Towards Blockchain-enabled School Information Hub**. New York, NY, USA: ACM, 2017. 19:1–19:4 p. (ICTD '17). Disponível em: <http://doi.acm.org/10.1145/3136560.3136584>. Citado na página 26.

BUTERIN, V. **Ethereum: A next-generation smart contract and decentralized application platform**. 2014. Acessado em 02/07/2019, às 19:17. Disponível em: <https://cryptorating.eu/whitepapers/Ethereum/Ethereum_white_paper.pdf>. Citado nas páginas 51 e 52.

____. **On Public and Private Blockchains**. 2015. Acessado em 02/07/2019, às 19:17. Disponível em: <https://blog.ethereum.org/2015/08/07/on-public-and-private-blockchains/>. Citado nas páginas 34, 40 e 41.

____. **Ethereum Whitepaper**. 2021. Acessado em 26/08/2021, às 14:14. Disponível em: <https://ethereum.org/en/whitepaper/>. Citado na página 50.

CACHIN, C. Architecture of the hyperledger blockchain fabric. In: **Workshop on distributed cryptocurrencies and consensus ledgers**. [S.l.: s.n.], 2016. v. 310. Citado na página 54.

CHEN, Y.; MENG, L.; ZHOU, H.; XUE, G. A blockchain-based medical data sharing mechanism with attribute-based access control and privacy protection. **Wireless Communications and Mobile Computing**, Hindawi, v. 2021, 2021. Citado na página 82.

CHEPURNOY, A.; RATHEE, M. Checking laws of the blockchain with property-based testing. In: **2018 International Workshop on Blockchain Oriented Software Engineering (IWBOSE)**. [S.l.: s.n.], 2018. p. 40–47. Citado na página 78.

COMMUNITY, E. **Decentralized Storage**. 2021. Acessado em 07/02/2022. Disponível em: <https://ethereum.org/en/developers/docs/storage/>. Citado na página 201.

CONDON, M. **Getting Up to Speed on Ethereum**. 2017. Acessado em 02/07/2019, às 19:17. Disponível em: <https://medium.com/@mattcondon/getting-up-to-speed-on-ethereum-63ed28821bbe>. Citado na página 52.

CROMAN, K.; DECKER, C.; EYAL, I.; GENCER, A. E.; JUELS, A.; KOSBA, A.; MILLER, A.; SAXENA, P.; SHI, E.; SIRER, E. G.; SONG, D.; WATTENHOFER, R. On scaling decentralized blockchains. In: . [S.l.: s.n.], 2016. v. 9604, p. 106–125. ISBN 978-3-662-53356-7. Citado nas páginas 28, 78 e 103.

D'ALTO, L. **IBM Announces Major Blockchain Solution to Speed Global Payments**. 2017. Acessado em 02/07/2019, às 19:17. Disponível em: <https://www-03.ibm.com/press/us/en/pressrelease/53290.wss>. Citado na página 25.

DEAN, J.; GHEMAWAT, S. **Leveldb**. 2020. Acessado em 22/10/2020, às 19:15. Disponível em: <https://github.com/google/leveldb/blob/master/doc/index.md>. Citado na página 56.

DHILLON, V.; METCALF, D.; HOOPER, M. The hyperledger project. In: **Blockchain enabled applications**. [S.l.]: Springer, 2017. p. 139–149. Citado nas páginas 51, 53 e 62.

DINH, T. T. A.; WANG, J.; CHEN, G.; LIU, R.; OOI, B. C.; TAN, K.-L. Blockbench: A framework for analyzing private blockchains. In: **Proceedings of the 2017 ACM International Conference on Management of Data**. New York, NY, USA: ACM, 2017. (SIGMOD '17), p. 1085–1100. ISBN 978-1-4503-4197-4. Disponível em: <http://doi.acm.org/10.1145/3035918.3064033>. Citado nas páginas 74 e 80.

ELLUL, J.; PACE, G. J. Runtime verification of ethereum smart contracts. In: **2018 14th European Dependable Computing Conference (EDCC)**. [S.l.: s.n.], 2018. p. 158–163. Citado na página 71.

Esposito, C.; De Santis, A.; Tortora, G.; Chang, H.; Choo, K. R. Blockchain: A panacea for healthcare cloud-based data security and privacy? **IEEE Cloud Computing**, v. 5, n. 1, p. 31–37, Jan 2018. ISSN 2325-6095. Citado na página 26.

EXAME, R. **Comprar bitcoin e outras criptomoedas é para você?** 2022. Acessado em 29/06/2022. Disponível em: <https://exame.com/future-of-money/comprar-bitcoin-e-outras-criptomoedas-e-para-voce/>. Citado na página 25.

FOUNDATION, A. S. **Apache Kafka**. 2017. Acessado em 07/07/2021, às 15:30. Disponível em: <https://kafka.apache.org>. Citado na página 58.

____. **Apache CouchDB**. 2020. Acessado em 22/10/2020, às 19:20. Disponível em: <https://docs.couchdb.org/en/stable/>. Citado nas páginas 49, 56 e 90.

GRAHAM, R. **Ethereum/TheDAO hack simplified**. 2016. Acessado em 02/07/2019, às 19:17. Disponível em: <https://blog.erratasec.com/2016/06/etheriumdao-hack-similfied.html#.XREKjC3OpQK>. Citado nas páginas 52 e 53.

GRISHCHENKO, I.; MAFFEI, M.; SCHNEIDEWIND, C. Foundations and tools for the static analysis of ethereum smart contracts. In: CHOCKLER, H.; WEISSENBACHER, G. (Ed.). **Computer Aided Verification**. Cham: Springer International Publishing, 2018. p. 51–78. Citado na página 71.

____. A semantic framework for the security analysis of ethereum smart contracts. In: BAUER, L.; KÜSTERS, R. (Ed.). **Principles of Security and Trust**. Cham: Springer International Publishing, 2018. p. 243–269. Citado na página 71.

GUPTA, M. **Blockchain for dummies: IBM Limited Edition**. 1nd. ed. [S.l.]: John Wiley & Sons, Inc., 2017. ISBN 9781119371236. Citado nas páginas 25, 26, 27, 32, 40, 46 e 47.

GUPTA, V. **A Brief History of Blockchain**. 2017. Acessado em 02/07/2019, às 19:17. Disponível em: <https://hbr.org/2017/02/a-brief-history-of-blockchain>. Citado nas páginas 33 e 34.

HAN, J.; HAIHONG, E.; LE, G.; DU, J. Survey on nosql database. In: IEEE. **2011 6th international conference on pervasive computing and applications**. [S.l.], 2011. p. 363–366. Citado na página 200.

HYPERLEDGER. **hyperledger-fabricdocs Documentation**. 2019. Acessado em 22/10/2020, às 18:58. Disponível em: <https://hyperledger-fabric.readthedocs.io/en/release-1.4/whatis.html>. Citado nas páginas 13, 47, 48, 54, 55, 56, 57, 58, 59, 60, 61 e 62.

HöLBL, M.; KOMPARA, M.; KAMISALIC, A.; ZLATOLAS, L. N. **A Systematic Review of the Use of Blockchain in Healthcare**. 2018. Citado na página 26.

IMPRENSA, A. de. **Hospital das Clínicas de SP completa 70 anos em franca expansão**. 2014. Acessado em 04/06/2021. Disponível em: <http://www.saude.sp.gov.br/ses/noticias/2014/abril/hospital-das-clinicas-de-sp-completa-70-anos-em-franca-expansao>. Citado nas páginas 28 e 103.

INC, D. **Docker overview**. 2021. Acessado em 02/12/2021, às 19:20. Disponível em: <https://docs.docker.com/get-started/overview/>. Citado na página 48.

INTEL. **Hyperledger Sawtooth documentation**. 2017. Acessado em 23/05/2021. Disponível em: <https://sawtooth.hyperledger.org/docs/core/releases/latest/introduction.html>. Citado na página 54.

ISMAIL, L.; HAMEED, H.; ALSHAMSI, M.; ALHAMMADI, M.; ALDHANHANI, N. Towards a blockchain deployment at uae university: Performance evaluation and blockchain taxonomy. In: **Proceedings of the 2019 International Conference on Blockchain Technology**. New York, NY, USA: Association for Computing Machinery, 2019. (ICBCT 2019), p. 30–38. ISBN 9781450362689. Disponível em: <https://doi.org/10.1145/3320154.3320156>. Citado na página 81.

JENTZSCH, C. **The History of the DAO and Lessons Learned**. 2016. Acessado em 02/07/2019, às 19:17. Disponível em: <https://blog.slock.it/the-history-of-the-dao-and-lessons-learned-d06740f8cfa5>. Citado na página 52.

JOHNSON, A. E.; POLLARD, T. J.; SHEN, L.; LEHMAN, L. H.; FENG, M.; GHASSEMI, M.; MOODY, B.; SZOLOVITS, P.; CELI, L. A.; MARK, R. G. Mimic-iii, a freely accessible critical care database. **Sci Data 3**, May 2016. Citado nas páginas 93, 94, 95, 102, 124 e 150.

KITCHENHAM, B. **Procedures for Performing Systematic Reviews**. Department of Computer Science, Keele University, UK, 2004. Citado nas páginas 67, 68 e 70.

KUROSE, J.; ROSS, K. **Redes de computadores e a Internet: uma abordagem top-down**. [S.l.]: ADDISON WESLEY BRA, 2007. ISBN 9788588639188. Citado nas páginas 34 e 40.

LINUXFOUNDATION. **Hyperledger Architecture, Volume II:Smart Contracts**. 2019. Acessado em 02/08/2021, às 17:17. Disponível em: <https://www.hyperledger.org/wp-content/uploads/2018/04/Hyperledger_Arch_WG_Paper_2_SmartContracts.pdf>. Citado nas páginas 13 e 53.

LISK. **Proof of Stake**. 2019. Acessado em 02/07/2019, às 19:17. Disponível em: <https://lisk.io/academy/blockchain-basics/how-does-blockchain-work/proof-of-stake>. Citado nas páginas 33, 44 e 45.

MELO, C.; DANTAS, J.; DANTAS, R.; Fé, I.; OLIVEIRA, D.; MACIEL, R. D.; MATOS, R.; MACIEL, P. Dependability evaluation of a blockchain-as-a-service environment. In: **2018 IEEE Symposium on Computers and Communications (ISCC)**. [S.l.: s.n.], 2018. Citado na página 72.

MERCANTI, I.; BISTARELLI, S.; SANTINI, F. An analysis of non-standard bitcoin transactions. In: **2018 Crypto Valley Conference on Blockchain Technology (CVCBT)**. [S.l.: s.n.], 2018. p. 93–96. Citado na página 72.

NAKAMOTO, S. **Bitcoin: A peer-to-peer electronic cash system**. 2009. Acessado em 02/07/2019, às 19:17. Disponível em: <http://www.bitcoin.org/bitcoin.pdf>. Citado nas páginas 25, 31, 32, 33, 35, 37, 39, 40, 42, 43 e 46.

NARAYANAN, A.; BONNEAU, J.; FELTEN, E.; MILLER, A.; GOLDFEDER, S. **Bitcoin and Cryptocurrency Technologies: A Comprehensive Introduction**. Princeton, NJ, USA: Princeton University Press, 2016. ISBN 0691171696, 9780691171692. Citado nas páginas 33, 34, 36, 39, 42, 43, 44, 45, 46 e 50.

OLIVEIRA, M. T.; CARRARA, G. R.; FERNANDES, N. C.; ALBUQUERQUE, C. V. N.; CARRANO, R. C.; MEDEIROS, D. S. V.; MATTOS, D. M. F. Towards a performance evaluation of private blockchain frameworks using a realistic workload. In: **2019 22nd Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN)**. [S.l.: s.n.], 2019. p. 180–187. Citado na página 81.

ONGARO, D.; OUSTERHOUT, J. In search of an understandable consensus algorithm. In: **Proceedings of the 2014 USENIX Conference on USENIX Annual Technical Conference**. USA: USENIX Association, 2014. (USENIX ATC'14), p. 305–320. ISBN 9781931971102. Citado na página 57.

PERFORMANCE H.; GROUP, S. **Hyperledger Blockchain Performance Metrics White Paper**. 2018. Acessado em 12/04/2021. Disponível em: <https://www.hyperledger.org/learn/publications/blockchain-performance-metrics#>. Citado nas páginas 91, 92 e 93.

PONGNUMKUL, S.; SIRIPANPORNCHANA, C.; THAJCHAYAPONG, S. Performance analysis of private blockchain platforms in varying workloads. In: **2017 26th International Conference on Computer Communication and Networks (ICCCN)**. [S.l.: s.n.], 2017. p. 1–6. Citado na página 73.

PROJECT, C. **Hyperledger Caliper Documentation**. 2020. Acessado em 22/10/2020, às 19:00. Disponível em: <https://hyperledger.github.io/caliper/v0.3.2/getting-started/>. Citado nas páginas 13, 54, 62, 63 e 64.

ROEHRS, A.; da Costa, C. A.; da Rosa Righi, R.; da Silva, V. F.; GOLDIM, J. R.; SCHMIDT, D. C. Analyzing the performance of a blockchain-based personal health record implementation. **Journal of Biomedical Informatics**, v. 92, p. 103140, 2019. ISSN 1532-0464. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1532046419300589>. Citado na página 80.

ROUHANI, S.; DETERS, R. Performance analysis of ethereum transactions in private blockchain. In: **2017 8th IEEE International Conference on Software Engineering and Service Science (ICSESS)**. [S.l.: s.n.], 2017. p. 70–74. ISSN 2327-0594. Citado na página 75.

SERGEY, I.; HOBOR, A. A concurrent perspective on smart contracts. **CoRR**, abs/1702.05511, 2017. Disponível em: <http://arxiv.org/abs/1702.05511>. Citado na página 71.

SHEN, B.; GUO, J.; YANG, Y. Medchain: Efficient healthcare data sharing via blockchain. **Applied Sciences**, v. 9, n. 6, 2019. ISSN 2076-3417. Disponível em: <https://www.mdpi.com/2076-3417/9/6/1207>. Citado na página 80.

SINGH, V. K.; GAO, M.; JAIN, R. Situation recognition: an evolving problem for heterogeneous dynamic big multimedia data. In: **Proceedings of the 20th ACM international conference on Multimedia**. [S.l.: s.n.], 2012. p. 1209–1218. Citado na página 93.

SORAMITSU. **Hyperledger Iroha documentation**. 2017. Acessado em 23/05/2021. Disponível em: <https://iroha.readthedocs.io/en/main/overview.html>. Citado na página 54.

SWAN, M. **Blockchain : blueprint for a new economy**. Sebastopol, Calif.: O'Reilly Media, 2015. ISBN 9781491920473 1491920475 9781491920459 1491920459 1491920491 9781491920497. Citado nas páginas 25, 32 e 34.

THAKKAR, P.; NATHAN, S.; VISHWANATHAN, B. Performance benchmarking and optimizing hyperledger fabric blockchain platform. **CoRR**, abs/1805.11390, 2018. Disponível em: <http://arxiv.org/abs/1805.11390>. Citado nas páginas 77 e 79.

THIN, W. Y. M. M.; DONG, N.; BAI, G.; DONG, J. S. Formal analysis of a proof-of-stake blockchain. In: **2018 23rd International Conference on Engineering of Complex Computer Systems (ICECCS)**. [S.l.: s.n.], 2018. p. 197–200. Citado na página 72.

THODE, H. Testing for normality marcel dekker. **Inc. New York**, p. 99–123, 2002. Citado nas páginas 117, 118, 144, 146, 164, 166, 167, 191 e 194.

WALPOLE, R. E.; MYERS, R. H.; MYERS, S. L.; YE, K. **Probability and statistics for engineers and scientists**. [S.l.]: Macmillan New York, 1993. v. 5. Citado nas páginas 117, 118, 144, 146, 164, 166, 167, 191 e 194.

WANG, L. Heterogeneous data and big data analytics. **Automatic Control and Information Sciences**, Science and Education Publishing Co., Ltd., v. 3, n. 1, p. 8–15, 2017. Citado na página 93.

WEBER, I.; GRAMOLI, V.; PONOMAREV, A.; STAPLES, M.; HOLZ, R.; TRAN, A. B.; RIMBA, P. On availability for blockchain-based systems. In: **2017 IEEE 36th Symposium on Reliable Distributed Systems (SRDS)**. [S.l.: s.n.], 2017. p. 64–73. Citado nas páginas 72 e 73.

WOHLIN, C.; RUNESON, P.; HöST, M.; OHLSSON, M.; REGNELL, B.; WESSLéN, A. Experimentation in software engineering. In: ____. [S.l.: s.n.], 2012. p. 123–151. ISBN 978-3-642-29043-5. Citado nas páginas 91 e 101.

WüST, K.; GERVAIS, A. Do you need a blockchain? In: **2018 Crypto Valley Conference on Blockchain Technology (CVCBT)**. [S.l.: s.n.], 2018. p. 45–54. Citado na página 41.

YUE, X.; WANG, H.; JIN, D.; LI, M.; JIANG, W. Healthcare data gateways: Found healthcare intelligence on blockchain with novel privacy risk control. **Journal of medical systems**, v. 40, p. 218, 10 2016. Citado na página 27.

ZOU, R.; LV, X.; ZHAO, J. Spchain: Blockchain-based medical data sharing and privacy-preserving ehealth system. **Information Processing Management**, v. 58, n. 4, p. 102604, 2021. ISSN 0306-4573. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0306457321001011>. Citado na página 82.

APÊNDICE

A

SMART-CONTRACT

```
/**
 * Marble asset management chaincode written in node.js, implementing {@link
↪  ChaincodeInterface}.
 * @type {SimpleChaincode}
 * @extends {ChaincodeInterface}
 */
let Chaincode = class {
    /**
     * Called during chaincode instantiate and upgrade. This method can be
↪  used
     * to initialize asset states.
     * @async
     * @param {ChaincodeStub} stub The chaincode stub is implemented by the
↪  fabric-shim
     * library and passed to the {@link ChaincodeInterface} calls by the
↪  Hyperledger Fabric platform. The stub
     * encapsulates the APIs between the chaincode implementation and the
↪  Fabric peer.
     * @return {Promise<SuccessResponse>} Returns a promise of a response
↪  indicating the result of the invocation.
     */
    async Init(stub) {
        let ret = stub.getFunctionAndParameters();
        console.info(ret);
        console.info('=========== Instantiated Marbles Chaincode ===========');
        return shim.success();
    }
```

```
    [...]

    /**
     * Creates a new patient with the given attributes.
     * @async
     * @param {ChaincodeStub} stub The chaincode stub.
     * @param {String[]} args The arguments of the function. Index 0: rowId.
↪   Index 1: subjectId.
     * Index 2: gender. Index 3: dob. Index 4: dod. Index 5: dodHosp. Index 6:
↪   dodSsn.
     * Index 7: expireFlag.
     */
    async insertPatient(stub, args) {
        if (args.length !== 8) {
            throw new Error('Incorrect number of arguments. Expecting 8');
        }
        // ==== Input sanitation ====
        console.log('--- start init patient ---');
        if (args[0].length <= 0) {
            throw new Error('1st argument must be a non-empty string');
        }
        if (args[1].length <= 0) {
            throw new Error('2nd argument must be a non-empty string');
        }
        if (args[2].length <= 0) {
            throw new Error('3rd argument must be a non-empty string');
        }
        if (args[3].length <= 0) {
            throw new Error('4rd argument must be a non-empty string');
        }
        if (args[7].length <= 0) {
            throw new Error('8rd argument must be a non-empty string');
        }

        let rowId = args[0];
        let subjectId = args[1];
        let gender = args[2];
        let dob = args[3];
        let dod = args[4];
        let dodHosp = args[5];
        let dodSsn = args[6];
        let expireFlag = parseInt(args[7]);
```

```
        // // ==== Check if an input already exists ====
        // let inputState = await stub.getState(subjectId);
        // if (inputState.toString()) {
        //     throw new Error('This input already exists: ' + subjectId);
        // }

        // ==== Create an object and marshal to JSON ====
        let input = {};
        input.docType = 'patients';
        input.rowId = rowId;
        input.subjectId = subjectId;
        input.gender = gender;
        input.dob = dob;
        input.dod = dod;
        input.dodHosp = dodHosp;
        input.dodSsn = dodSsn;
        input.expireFlag = expireFlag;

        // === Save an input to state ===
        await stub.putState(subjectId, Buffer.from(JSON.stringify(input)));
        let indexName = 'docType~subjectId';
        let subjectNameIndexKey = await stub.createCompositeKey(indexName,
        ↪  [input.docType, input.subjectId]);
        console.info(subjectNameIndexKey);
        //  Save index entry to state. Only the key name is needed, no need to
        ↪  store a duplicate copy of the marble.
        //  Note - passing a 'nil' value will effectively delete the key from
        ↪  state, therefore we pass null character as value
        await stub.putState(indexName, Buffer.from(subjectNameIndexKey));
        // ==== Marble saved and indexed. Return success ====
        console.info('- end init patient');

        return Buffer.from(JSON.stringify(input));
    };

    [...]

    /**
     * Creates a dictionary of items with the given attributes.
     * @async
     * @param {ChaincodeStub} stub The chaincode stub.
```

```
     * @param {String[]} args The arguments of the function. Index 0: rowId.
↪  Index 1: itemid.
     * Index 2: label. Index 3: abbreviation. Index 4: dbsource. Index 5:
↪  linksto.
     * Index 6: category. Index 7: unitname. Index 8: paramType. Index 9:
↪  conceptid
     */
    async insertDitem(stub, args) {
        if (args.length !== 10) {
            throw new Error('Incorrect number of arguments. Expecting 10');
        }
        // ==== Input sanitation ====
        console.log('--- start init item dictionary ---');
        if (args[0].length <= 0) {
            throw new Error('1st argument must be a non-empty string');
        }
        if (args[1].length <= 0) {
            throw new Error('2nd argument must be a non-empty string');
        }

        let rowId = args[0];
        let itemid = args[1];
        let label = args[2];
        let abbreviation = args[3];
        let dbsource = args[4];
        let linksto = args[5];
        let category = args[6];
        let unitname = args[7];
        let paramType = args[8];
        let conceptid = args[9];

        // // ==== Check if an input already exists ====
        // let inputState = await stub.getState(itemid);
        // if (inputState.toString()) {
        //     throw new Error('This input already exists: ' + itemid);
        // }

        // ==== Create an object and marshal to JSON ====
        let input = {};
        input.docType = 'd_items';
        input.rowId = rowId;
        input.itemid = itemid;
```

```
        input.label = label;
        input.abbreviation = abbreviation;
        input.dbsource = dbsource;
        input.linksto = linksto;
        input.category  = category;
        input.unitname = unitname;
        input.paramType = paramType;
        input.conceptid = conceptid;

        // === Save an input to state ===
        await stub.putState(itemid, Buffer.from(JSON.stringify(input)));
        let indexName = 'itemid';
        let subjectNameIndexKey = await stub.createCompositeKey(indexName,
        ↪  [input.itemid]);
        console.info(subjectNameIndexKey);
        //  Save index entry to state. Only the key name is needed, no need to
        ↪  store a duplicate copy of the marble.
        //  Note - passing a 'nil' value will effectively delete the key from
        ↪  state, therefore we pass null character as value
        await stub.putState(subjectNameIndexKey, Buffer.from('\u0000'));
        // ==== Marble saved and indexed. Return success ====
        console.info('- end init item dictionary');

        return Buffer.from(JSON.stringify(input));
    };

    [...]

    /**
     * Creates a new prescription with the given attributes.
     * @async
     * @param {ChaincodeStub} stub The chaincode stub.
     * @param {String[]} args The arguments of the function. Index 0: rowId.
↪  Index 1: subjectId.
     * Index 2: hadmId. Index 3: icustayId. Index 4: startdate. Index 5:
↪  enddate. Index 6: drugType.
     * Index 7: drug. Index 8: drugNamePoe. Index 9: drugNameGeneric. Index
↪  10: formularyDrugCd.
     * Index 11: gsn. Index 12: ndc. Index 13: prodStrength. Index 14:
↪  doseValRx.
     * Index 15: doseUnitRx. Index 16: formValDisp. Index 17: formUnitDisp.
     * Index 18: route.
```

```
     */
    async insertPrescription(stub, args) {
        if (args.length !== 19) {
            throw new Error('Incorrect number of arguments. Expecting 19');
        }
        // ==== Input sanitation ====
        console.log('--- start init prescription ---');
        if (args[0].length <= 0) {
            throw new Error('1st argument must be a non-empty string');
        }
        if (args[1].length <= 0) {
            throw new Error('2nd argument must be a non-empty string');
        }
        if (args[2].length <= 0) {
            throw new Error('3rd argument must be a non-empty string');
        }
        if (args[6].length <= 0) {
            throw new Error('7rd argument must be a non-empty string');
        }
        if (args[7].length <= 0) {
            throw new Error('8rd argument must be a non-empty string');
        }

        let rowId = args[0];
        let subjectId = args[1];
        let hadmId = args[2];
        let icustayId = args[3];
        let startdate = args[4];
        let enddate = args[5];
        let drugType = args[6];
        let drug = args[7];
        let drugNamePoe = args[8];
        let drugNameGeneric = args[9];
        let formularyDrugCd = args[10];
        let gsn = args[11];
        let ndc = args[12];
        let prodStrength = args[13];
        let doseValRx = args[14];
        let doseUnitRx = args[15];
        let formValDisp = args[16];
        let formUnitDisp = args[17];
        let route = args[18];
```

```
// // ==== Check if an input already exists ====
// let inputState = await stub.getState(rowId);
// if (inputState.toString()) {
//     throw new Error('This input already exists: ' + rowId);
// }

// ==== Create an object and marshal to JSON ====
let input = {};
input.docType = 'prescription';
input.rowId = rowId;
input.subjectId = subjectId;
input.hadmId = hadmId;
input.icustayId = icustayId;
input.startdate = startdate;
input.enddate = enddate;
input.drugType = drugType;
input.drug = drug;
input.drugNamePoe = drugNamePoe;
input.drugNameGeneric = drugNameGeneric;
input.formularyDrugCd = formularyDrugCd;
input.gsn = gsn;
input.ndc = ndc;
input.prodStrength = prodStrength;
input.doseValRx = doseValRx;
input.doseUnitRx = doseUnitRx;
input.formValDisp = formValDisp;
input.formUnitDisp = formUnitDisp;
input.route = route;

// === Save an input to state ===
await stub.putState(rowId, Buffer.from(JSON.stringify(input)));
let indexName = 'rowId';
let subjectNameIndexKey = await stub.createCompositeKey(indexName,
↪  [input.rowId]);
console.info(subjectNameIndexKey);
//  Save index entry to state. Only the key name is needed, no need to
↪  store a duplicate copy of the marble.
//  Note - passing a 'nil' value will effectively delete the key from
↪  state, therefore we pass null character as value
await stub.putState(subjectNameIndexKey, Buffer.from('\u0000'));
// ==== Marble saved and indexed. Return success ====
```

```
        console.info('- end init prescription');

        return Buffer.from(JSON.stringify(input));
    };

    [...]

    /**
     * Creates a new input event mv with the given attributes.
     * @async
     * @param {ChaincodeStub} stub The chaincode stub.
     * @param {String[]} args The arguments of the function. Index 0: rowId.
↪   Index 1: subjectId.
     * Index 2: hadmId. Index 3: icustayId. Index 4: starttime. Index 5:
↪   endtime. Index 6: itemid.
     * Index 7: amount. Index 8: amountuom. Index 9: rate. Index 10: rateuom.
     * Index 11: storetime. Index 12: cgid. Index 13: orderid. Index 14:
↪   linkorderid.
     * Index 15: ordercategoryname. Index 16: secondarycategoryname. Index 17:
↪   ordercomponenttypedescription.
     * Index 18: ordercategorydescription. Index 19: patientweight. Index 20:
↪   totalamount. Index 21: totalamountuom.
     * Index 22: isopenbag. Index 23: continueinnextdept. Index 24:
↪   cancelreason. Index 25: statusdescription.
     * Index 26: commentsEditedby. Index 27: commentsCanceledby. Index 28:
↪   commentsDate.
     * Index 29: originalamount. Index 30: originalrate.
     */
    async insertInputeventMv(stub, args) {
        if (args.length !== 31) {
            throw new Error('Incorrect number of arguments. Expecting 31');
        }
        // ==== Input sanitation ====
        console.log('--- start init input event mv ---');
        if (args[0].length <= 0) {
            throw new Error('1st argument must be a non-empty string');
        }
        if (args[1].length <= 0) {
            throw new Error('2nd argument must be a non-empty string');
        }

        let rowId = args[0];
```

```
let subjectId = args[1];
let hadmId = args[2];
let icustayId = args[3];
let starttime = args[4]
let endtime = args[5];
let itemid = args[6];
let amount = args[7];
let amountuom = args[8];
let rate = args[9];
let rateuom = args[10];
let storetime = args[11];
let cgid = args[12];
let orderid = args[13];
let linkorderid = args[14];
let ordercategoryname = args[15];
let secondarycategoryname = args[16];
let ordercomponenttypedescription = args[17];
let ordercategorydescription = args[18];
let patientweight = args[19];
let totalamount = args[20];
let totalamountuom = args[21];
let isopenbag = args[22];
let continueinnextdept = args[23];
let cancelreason = args[24];
let statusdescription = args[25];
let commentsEditedby = args[26];
let commentsCanceledby = args[27];
let commentsDate = args[28];
let originalamount = args[29];
let originalrate = args[30];

// // ==== Check if an input already exists ====
// let inputState = await stub.getState(rowId);
// if (inputState.toString()) {
//     throw new Error('This input already exists: ' + rowId);
// }

// ==== Create an object and marshal to JSON ====
let input = {};
input.docType = 'inputeventmv';
input.rowId = rowId;
input.subjectId = subjectId;
```

```
input.hadmId = hadmId;
input.icustayId = icustayId;
input.starttime = starttime;
input.endtime = endtime;
input.itemid = itemid;
input.amount = amount;
input.amountuom = amountuom;
input.rate = rate;
input.rateuom = rateuom;
input.storetime = storetime;
input.cgid = cgid;
input.orderid = orderid;
input.linkorderid = linkorderid;
input.ordercategoryname = ordercategoryname;
input.secondarycategoryname = secondarycategoryname;
input.ordercomponenttypedescription = ordercomponenttypedescription;
input.ordercategorydescription = ordercategorydescription;
input.patientweight = patientweight;
input.totalamount = totalamount;
input.totalamountuom = totalamountuom;
input.isopenbag = isopenbag;
input.continueinnextdept = continueinnextdept;
input.cancelreason = cancelreason;
input.statusdescription = statusdescription;
input.commentsEditedby = commentsEditedby;
input.commentsCanceledby = commentsCanceledby;
input.commentsDate = commentsDate;
input.originalamount = originalamount;
input.originalrate = originalrate;

// === Save an input to state ===
await stub.putState(rowId, Buffer.from(JSON.stringify(input)));
let indexName = 'rowId';
let subjectNameIndexKey = await stub.createCompositeKey(indexName,
↪  [input.rowId]);
console.info(subjectNameIndexKey);
//  Save index entry to state. Only the key name is needed, no need to
↪  store a duplicate copy of the marble.
//  Note - passing a 'nil' value will effectively delete the key from
↪  state, therefore we pass null character as value
await stub.putState(subjectNameIndexKey, Buffer.from('\u0000'));
// ==== Marble saved and indexed. Return success ====
```

```
        console.info('- end init input event mv');

        return Buffer.from(JSON.stringify(input));
    };

    [...]

    /**
     * Queries for patient based on a passed id.
     * @async
     * @param {ChaincodeStub} stub The chaincode stub.
     * @param {String[]} args The arguments of the function. Index 0: patient
↪   id.
     * @param {Chaincode} thisObject The chaincode object context.
     * @return {Promise<Buffer>} The patient of the specified id.
     */
    async queryPatientById(stub, args, thisClass) {
        if (args.length !== 1) {
            throw new Error('Incorrect number of arguments. Expecting patient
            ↪  id.');
        }

        let subjectId  = args[0];
        let queryString = {};
        queryString.selector = {};
        queryString.selector.docType = 'patients';
        queryString.selector.subjectId = subjectId;
        let method = thisClass['getQueryResultForQueryString'];
        let queryResults = await method(stub, JSON.stringify(queryString),
        ↪  thisClass);

        return queryResults;
    }

    [...]
};

shim.start(new Chaincode());
```


ICMC
USP
SÃO CARLOS