Análise de Desempenho no Hyperledger Besu: Impacto de Parâmetros via Benchmarking utilizando Caliper

Vinicius S. Mendes[1,3], Arthur N. de Freitas[2,3], Gustavo S. Bom[1,3], Jeffson C. Sousa[3]

[1]Faculdade de Ciências Exatas e Tecnologia - Universidade Federal da Grande Dourados (UFGD) – Dourados – MS – Brasil.

[2]Faculdade de Tecnologia em Geoprocessamento – Universidade Federal do Pará (UFPA) – Ananindeua – PA – Brasil

[3]Centro de Pesquisa e Desenvolvimento em Telecomunicações (CPQD) Campinas – SP – Brasil

{stefanesv31, arthur.freitas7723, bomgustavo10}@gmail.com

jcsousa@cpqd.edu.br

Resumo. *Blockchain é uma tecnologia de registro distribuído que permite aplicações descentralizadas com integridade e transparência. Este trabalho tem como objetivo analisar o impacto de parâmetros de configuração no desempenho de redes permissionadas baseadas em Hyperledger Besu. Propõe-se uma arquitetura automatizada de benchmarking que integra criação da rede, deploy de contratos e execução de experimentos de forma reprodutível. A metodologia avalia diferentes configurações de número de nós e tempo de bloco. Os resultados indicam que mais nós aumentam a latência, enquanto blocos maiores reduzem a vazão e elevam o tempo de resposta, evidenciando a influência desses parâmetros no desempenho da rede.*

Abstract. *Blockchain is a distributed ledger technology that enables decentralized applications with integrity and transparency. This work aims to analyze the impact of configuration parameters on the performance of permissioned networks based on Hyperledger Besu. It proposes an automated benchmarking architecture that integrates network creation, smart contract deployment, and experiment execution in a reproducible manner. The methodology evaluates different configurations in terms of the number of nodes and block time. The results indicate that increasing the number of nodes raises latency, while larger block sizes reduce throughput and increase response time, highlighting the significant influence of these parameters on network performance.*

1. Introdução

Blockchain representa uma tecnologia de registro distribuído que permite manter um livro-razão compartilhado entre múltiplos participantes sem necessidade de autoridade central [Kaushal and Kumar 2024]. A tecnologia fundamenta-se em três pilares principais: criptografia de chave pública para identidade e autenticação, função hash criptográfica para integridade de dados, e algoritmo de consenso distribuído para acordo sobre o estado do sistema.

Redes blockchain podem ser classificadas como públicas, nas quais qualquer participante pode ingressar na rede, e privadas, que restringem a participação a entidades autorizadas e que operam assumindo certo nível de confiança entre elas [Fan et al. 2022]. Além dessa distinção, as redes também podem ser categorizadas como permissionadas ou não-permissionadas. Redes não permissionadas permitem que qualquer participante da rede coopere com o mecanismo de consenso sem necessidade de autorização prévia, enquanto redes permissionadas restringem essas funções a entidades previamente autorizadas, que possuem permissões específicas definidas pelo administrador da rede [Crawford 2025]. As redes privadas e permissionadas combinam essas duas características e apresentam propriedades específicas, como maior controle sobre participantes, melhor desempenho devido ao número reduzido de validadores, conformidade regulatória decorrente de identidades conhecidas e maior privacidade por meio de mecanismos de controle de acesso aos dados.

Hyperledger Besu consiste em um cliente Ethereum de código aberto desenvolvido em Java e mantido pela Linux Foundation como parte do projeto Hyperledger [Besu 2023a]. Besu implementa especificações completas do protocolo Ethereum incluindo máquina virtual Ethereum para execução de contratos inteligentes e múltiplos algoritmos de consenso adequados para redes públicas e privadas e APIs JSON-RPC completas para interação com a rede. A plataforma suporta especificamente redes permissionadas através de mecanismos de controle de acesso em nível de nó e conta [Besu 2023b].

[Marchesi et al. 2025] abordam que contratos inteligentes representam programas executados na máquina virtual Ethereum que implementam lógica de negócio de forma determinística e transparente. Os contratos são escritos tipicamente em linguagem Solidity de alto nível, compilados para bytecode da EVM (Ethereum Virtual Machine), e submetidos na blockchain através de transações especiais que criam novos endereços de contrato. A execução de contratos inteligentes consome recursos computacionais medidos em unidades de *gas*. Cada operação na máquina virtual tem custo em gas definido, e transações especificam limite máximo de *gas* disponível além de preço por unidade de *gas*. Este mecanismo econômico previne loops infinitos e ataques.

O *benchmarking* de sistemas blockchain consiste na avaliação sistemática de desempenho sob cargas controladas, fundamentando-se em métricas essenciais como a vazão e latência dada uma taxa de envio. A vazão, também referida como *throughput*, é definida como o número total de transações válidas confirmadas pela rede por segundo (TPS – *Transactions Per Second*), representando a capacidade efetiva do sistema em processar e registrar dados no *ledger* em um intervalo específico. Complementarmente, a latência corresponde ao tempo médio decorrido entre a submissão da transação pelo cliente e sua confirmação final na blockchain, medindo a rapidez com que uma operação torna-se efetivamente disponível na rede.

[Kaushal and Kumar 2024] destacam que o Hyperledger Caliper tem se consolidado como ferramenta padrão para *benchmarking*, oferecendo estrutura padronizada e reprodutível para redes permissionadas. A avaliação requer consideração de múltiplas dimensões incluindo características da rede como número de nós e topologia, configurações de consenso como tempo de bloco, características de carga como taxa de transações e complexidade de contratos, e métricas de sistema como CPU e memória.

A avaliação de desempenho de contratos inteligentes em ambientes blockchain representa um desafio técnico significativo, devido à complexidade de configuração, execução e análise de experimentos controlados. Esse processo envolve, tradicionalmente, múltiplas ferramentas desconectadas, incluindo plataforma blockchain para infraestrutura de rede, framework de desenvolvimento para contratos inteligentes, ferramenta de publicação de contratos, framework de *benchmarking* para geração de carga e *scripts* de análise para processamento de resultados. A integração manual desses componentes nos leva a significativos problemas no processo experimental, por meio de configurações manuais propensas a erros, dificuldades de sincronização entre componentes, grande esforço para variação de parâmetros experimentais, que leva a problemas de reprodutibilidade decorrentes da variabilidade de configurações. A análise comparativa entre diferentes abordagens e configurações permite identificar gargalos e otimizar o desempenho, sendo fundamental documentar todo o processo evolutivo.

Embora tecnologias de blockchain permissionadas como o Hyperledger Besu ofereçam extrema flexibilidade, compreender o impacto exato das escolhas arquiteturais no desempenho do sistema ainda é um desafio. Conforme analisado por [Cardoso and et al. 2024] demonstra que o desempenho do Besu é altamente dependente do contexto, ressaltando a escassez de avaliações comparativas que isolem essas variáveis operacionais. Neste trabalho, partimos da hipótese de que pequenas variações no protocolo de consenso (como a escolha entre IBFT 2.0 e QBFT), no tamanho da rede e nos tempos de bloco alteram consideravelmente a escalabilidade (vazão) e a responsividade (latência) da rede sob diferentes níveis de estresse. Portanto, a principal contribuição deste artigo é tanto o desenvolvimento de uma arquitetura automatizada de benchmarking quanto a condução de um estudo exploratório objetivando responder à seguinte questão de pesquisa empírica: como variações no algoritmo de consenso, tempo de bloco e número de nós validadores impactam a vazão e a latência em redes permissionadas? Através desta arquitetura, apresentamos uma avaliação exploratória que visa suprir essa lacuna metodológica.

O restante deste artigo está organizado da seguinte forma. A Seção 2 apresenta os trabalhos relacionados. A Seção 3 descreve a metodologia e a configuração da plataforma experimental. A Seção 4 apresenta e analisa os resultados obtidos. Por fim, a Seção 5 apresenta as conclusões e direções para trabalhos futuros.

2. Trabalhos Relacionados

Apesar do crescente número de implantações de sistemas blockchains atualmente, [Nasrulin et al. 2022] pontuam que ainda existe uma pesquisa limitada sobre comparações das características de desempenho dessas soluções.

O trabalho de [Saraiva et al. 2025] buscou identificar, analisar e categorizar métricas de software utilizadas na avaliação de sistemas baseados em blockchain, com resultados que destacam o desempenho e a segurança como fundamentais para a qualidade nesses sistemas, com foco especial em vazão e latência. Ademais, o Hyperledger Caliper é considerado uma ferramenta frequente para avaliar o desempenho de redes blockchain.

[Kaushal and Kumar 2024] utilizaram o Hyperledger Caliper para medir o desempenho de soluções baseadas em blockchain em um cenário envolvendo múltiplas organizações e peers, configurado com o framework Hyperledger Fabric. O experimento

simulou um sistema de monitoramento remoto de pacientes, avaliando métricas como vazão e taxa de envio de transações (TPS). Os resultados indicaram pouca diferença entre a taxa de envio e o vazão durante operações de escrita em 80 e 100 TPS, enquanto nas operações de leitura esses valores foram praticamente idênticos, demonstrando a eficiência da rede sob diferentes cargas.

Também utilizando Hyperledger Caliper, o trabalho de [Sousa et al. 2025] investiga a avaliação de arquitetura e performance de contratos inteligentes para identidade digital descentralizada em redes blockchain permissionadas. O estudo foca em métricas como tempo de resposta, vazão e uso de recursos, evidenciando a importância da análise detalhada para suportar soluções eficientes de identidade digital baseadas em blockchain, especialmente em ambientes empresariais ou regulados.

De acordo com [Tran et al. 2022], os processos de *deployment* e avaliação de redes blockchain são demorados, propensos a erros e dependem de conhecimentos específicos de plataformas. Desta forma, os autores enfatizam os benefícios da automação como forma de melhorar a eficiência e a eficácia destes processos, além de propor um sistema para realizar instalações e avaliações de forma automática, com tempo médio de 95 milissegundos, consideravelmente inferior ao processo manual tradiciomal.

Quanto a outras ferramentas utilizadas no processo, [Bansod and Ragha 2024] apontam que a automação reduz esforços em testes estáticos e dinâmicos, minimiza a intervenção humana e erros, e, quando combinada aos testes de regressão, diminui o custo total do desenvolvimento de software. Em aplicações baseadas em Blockchain, são necessárias ferramentas de teste específicas para verificação e validação do sistema, que devem ser cuidadosamente selecionadas conforme a finalidade, o ciclo de vida do teste, as competências do testador e a acessibilidade das ferramentas.

Nesse cenário, autores como [Ren et al. 2023] vêm buscando desenvolver modelos padronizados de framework para o benchmark. Estes autores propõem uma estrutura denominada Blockchain Benchmark Standardized Format (BBSF), com intuito de permitir que desenvolvedores comparem plataformas blockchain usando métricas derivadas de cargas de trabalho realistas.

Além disso, estudos recentes também têm investigado comparações experimentais entre diferentes plataformas blockchain por meio de ferramentas de benchmarking. Nesse contexto, [Thakur et al. 2023] realizaram uma análise de desempenho envolvendo Hyperledger Fabric, Ethereum e Hyperledger Besu, utilizando o Hyperledger Caliper para mensuração de métricas como throughput, latência, taxa de sucesso das transações e consumo de recursos computacionais. Os resultados indicaram melhor desempenho geral do Hyperledger Fabric em comparação às demais plataformas avaliadas, reforçando a importância do uso de ferramentas padronizadas de benchmarking para apoiar a escolha de plataformas blockchain em diferentes cenários de aplicação.

A Tabela 1 apresenta uma comparação entre os principais trabalhos relacionados, considerando aspectos como automação do pipeline, integração de ferramentas, execução em lote, controle de parâmetros e reprodutibilidade experimental.

Diferentemente dos trabalhos relacionados, que abordam isoladamente aspectos como *benchmarking* com Hyperledger Caliper, automação de *deployment* ou padronização de métricas experimentais, a arquitetura proposta neste trabalho integra de

Tabela 1. Comparação de Trabalhos Relacionados

Trabalho	Plataforma	Automação do pipeline	Integração Besu + Hardhat + Caliper	Execução em lote	Controle sistemático de parâmetros	Reprodutibilidade experimental	Foco
Kaushal & Kumar (2024)	Hyperledger Fabric	Parcial	Não	Não	Parcial	Parcial	Benchmark com Caliper
Sousa et al. (2025)	Ethereum permissionado	Parcial	Não	Não	Sim	Parcial	Identidade digital
Tran et al. (2022)	Blockchain genérico	Sim	Não	Não	Parcial	Sim	Automação de deployment
Ren et al. (2023)	Multiplataforma	Parcial	Não	Não	Sim	Sim	Padronização de benchmark
Cardoso et al. (2024)	Hyperledger Besu	Parcial	Não	Não	Sim	Parcial	Testes via API
Thakur et al. (2023)	Hyperledger Fabric e BesuEthereum	Não	Parcial	Sim	Sim	Parcial	Benchmark comparativo entre plataformas
Este trabalho	Hyperledger Besu	Sim (pipeline completo)	Sim (integração nativa)	Sim	Sim	Sim (ambiente conteinerizado)	Arquitetura unificada para benchmarking automatizado

forma unificada o ciclo completo de experimentação envolvendo criação da rede, deploy de contratos inteligentes e execução automatizada de benchmarks em ambientes Hyperledger Besu. Além disso, a solução permite execução sistemática de experimentos em lote com variação controlada de parâmetros como número de nós, algoritmo de consenso e tempo de bloco, garantindo maior reprodutibilidade e reduzindo significativamente a intervenção manual no pipeline experimental. Esse nível de integração entre Besu, Hardhat, Caliper e Docker não é explorado de forma conjunta nos trabalhos analisados, constituindo o principal diferencial da abordagem proposta.

3. Metodologia

3.1. Seleção e parametrização da plataforma experimental

A configuração do ambiente experimental foi tratada como uma etapa metodológica central, pois estudos indicam que a escolha de plataforma, ferramentas de benchmark e parâmetros de rede influencia diretamente as métricas de desempenho observadas em blockchains permissionadas [Fan et al. 2020, Dabbagh et al. 2021]. De forma semelhante a processos de seleção tecnológica em ambientes corporativos, foram definidos critérios, analisadas alternativas e verificada sua adequação ao objetivo deste trabalho: avaliar o desempenho de contratos inteligentes executados em uma rede baseada na Máquina Virtual Ethereum. Com base nesses critérios, a seleção das ferramentas da arquitetura experimental seguiu um processo sistemático orientado por recomendações da literatura, contemplando aspectos como compatibilidade, controle experimental e reprodutibilidade.

Os critérios utilizados foram organizados em três grupos principais: critérios da plataforma blockchain, critérios de *benchmarking* e critérios de infraestrutura experimental. Para a seleção da plataforma blockchain (Besu), foram considerados os seguintes critérios: suporte a redes permissionadas com controle explícito de participantes; compatibilidade com a Ethereum Virtual Machine (EVM); suporte a algoritmos de consenso tolerantes a falhas bizantinas (IBFT e QBFT); possibilidade de parametrização do tempo de bloco; possibilidade de variação do número de validadores; disponibilidade de interface JSON-RPC para integração com ferramentas externas; integração com ferramentas

de *benchmarking* compatíveis com EVM; ecossistema consolidado e documentação ativa; estabilidade experimental reportada na literatura; suporte à execução em ambientes conteinerizados.

Para a seleção da infraestrutura de execução experimental baseada em contêineres, foram considerados: isolamento entre execuções experimentais; reprodutibilidade do ambiente; facilidade de orquestração de múltiplos nós blockchain; eliminação de interferências entre dependências externas; suporte à reconstrução rápida da infraestrutura experimental; compatibilidade com pipelines automatizados; portabilidade entre diferentes ambientes computacionais.

Para a seleção do framework de desenvolvimento e instanciação de contratos inteligentes, foram considerados: compatibilidade nativa com EVM; suporte à automação de compilação e instanciação; integração com clientes Ethereum via JSON-RPC; geração determinística de bytecode entre execuções; suporte à execução de *scripts* automatizados; compatibilidade com ambientes Docker; ampla adoção pela comunidade e consolidação do ecossistema.

Como critérios de seleção da ferramenta de *benchmarking*, foram considerados: suporte nativo a plataformas compatíveis com EVM; possibilidade de definição parametrizada de workloads; controle explícito da taxa de envio de transações (send rate); coleta estruturada de métricas de throughput e latência; monitoramento de recursos computacionais (CPU, memória e rede); suporte à execução repetida de experimentos sob condições controladas; geração automática de relatórios experimentais; adoção consolidada na literatura de avaliação de desempenho em blockchain.

Além dos critérios individuais de cada ferramenta, foi adotado como critério transversal a possibilidade de integração entre os componentes em um pipeline experimental automatizado. Esse requisito foi determinante para a escolha conjunta de Hyperledger Besu, Hardhat, Hyperledger Caliper e Docker, permitindo a construção de uma arquitetura unificada capaz de executar experimentos completos com mínima intervenção manual e alto grau de reprodutibilidade.

3.2. Implementação do sistema automatizado

O desenvolvimento deste trabalho teve início a partir da análise de três repositórios independentes originalmente propostos por [Sousa 2024a, Sousa 2024b, Sousa 2024c]. Contudo, os passos para execução do benchmark eram manuais e fragmentados, dificultando a integração entre as ferramentas e a reprodutibilidade dos experimentos.

A Figura 1 apresenta uma visão geral dos passos manuais necessários para executar benchmarks com as tecnologias selecionadas, destacando que cada etapa está suscetível a falhas humanas, ilustradas como “fail”. Para superar essas limitações, os repositórios foram integrados em uma arquitetura unificada [Mendes 2025], baseada em *scripts* que automatizam todo o ciclo de execução dos experimentos.

O elemento central consiste em um *script* em *shell* (*run-experiment.sh*), responsável por unificar e automatizar essas etapas manuais por meio de um conjunto de fases sequenciais em cada experimento, incluindo: instalação de dependências, interrupção da rede atual, atualização do *genesis.json*, geração de chaves do Besu, configuração dos nós (incluindo chaves e permissões), criação dinâmica do *docker-compose*, *build* da ima-

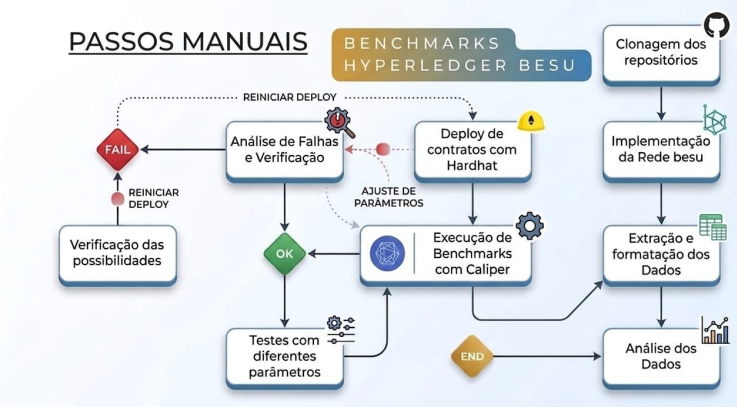


Figura 1. Pipeline manual para benchmarks utilizando Caliper

gem Docker, criação e inicialização da rede, espera pela estabilização, *deploy* automático do contrato, atualização dos arquivos YAML do Caliper, execução dos *benchmarks* e extração dos dados. Essa abordagem elimina a necessidade de intervenções manuais entre ferramentas, reduz erros de configuração e viabiliza a execução de testes em lote com variação sistemática de parâmetros.

A base funcional dos *benchmarks* é o contrato `Simple`, derivado do repositório oficial de exemplos do Hyperledger Caliper[1]. Esse contrato define três operações fundamentais:

- **open:** criação ou inicialização de uma conta;
- **query:** leitura do saldo associado a uma conta;
- **transfer:** transferência de valores entre contas.

Algorithm 1 Contrato Inteligente *Simple*

Variável de Estado:
  $accounts$: mapeamento de *string* para *int*

function OPEN($acc\_id, amount$)
  $accounts[acc\_id] \leftarrow amount$
end function

function QUERY(acc_id) *int*
  return $accounts[acc\_id]$
end function

function TRANSFER($acc\_from, acc\_to, amount$)
  $accounts[acc\_from] \leftarrow accounts[acc\_from] - amount$
  $accounts[acc\_to] \leftarrow accounts[acc\_to] + amount$
end function

Embora o contrato Simple apresente uma estrutura baseada em transferências genéricas, seu conjunto reduzido de operações isola comportamentos críticos que são

[1]Disponível em: `https://github.com/hyperledger-caliper/caliper-benchmarks/blob/v0.6.0/src/ethereum/simple/simple.sol`

diretamente transponíveis para sistemas críticos governamentais ou de saúde digital. Por exemplo, a operação de escrita de estado ('open') reflete os custos de processamento da criação de um registro ou prontuário, a leitura ('query') reflete a validação de uma credencial verificável por um profissional, e a atualização ('transfer') espelha a delegação de acesso a dados sensíveis.

Esse conjunto reduzido, porém completo, de operações torna o contrato `Simple` particularmente adequado para testes de desempenho e comportamento da rede, pois apresenta lógica determinística e de baixa complexidade, permitindo atribuir variações de métricas principalmente à configuração da rede, além de ser amplamente utilizado em estudos de *benchmarking*, o que favorece a reprodutibilidade e a comparabilidade com trabalhos relacionados.

Dentro desse *pipeline* unificado, os parâmetros experimentais são tratados de forma declarativa e podem ser customizados para cada cenário de teste. Entre os principais parâmetros configuráveis estão:

- **Configuração da rede Besu:** número de nós validadores, algoritmo de consenso (por exemplo, QBFT ou IBFT), tempo de geração de blocos (*block time*) e versão do cliente Besu;
- **Perfil de carga do Caliper:** lista de taxas de transação por segundo (TPS) a serem aplicadas para cada função do contrato (`open`, `query`, `transfer`), número de *rounds*, duração de cada fase e distribuição entre tipos de operação

A produção desse ambiente experimental permitem, portanto, que um único comando dispare uma série de experimentos em lote, nos quais diferentes combinações de parâmetros de rede e de carga são testadas sobre o mesmo contrato de referência. Esse desenho metodológico garante maior controle sobre variáveis, facilita a repetição de experimentos e fornece uma base sólida para análise comparativa de desempenho em redes Hyperledger Besu.

3.3. Ambiente de execução

Os experimentos foram executados em uma máquina dedicada configurada com o sistema operacional Zorin OS 17.3 Pro, 32 GB de memória RAM, processador Intel® Xeon® E5-2670 v3 (12 núcleos, 24 threads, 2,30 GHz), GPU NVIDIA RTX 2070 Super e unidade de armazenamento SSD NVMe PCIe 3.0x4 com 512 GB de capacidade. Foram utilizados nós locais, mantendo-se a máquina dedicada exclusivamente à execução dos experimentos durante todo o período de avaliação, sem a execução de outras cargas de trabalho concorrentes.

3.4. Parametrização Experimental

As combinações de parâmetros de infraestrutura de rede avaliadas estão sintetizadas na Tabela 2. A análise contempla a variação do número de nós validadores, o tempo de geração de bloco e o algoritmo de consenso (QBFT e IBFT), além de comparar o desempenho entre diferentes versões do cliente Hyperledger Besu. No total, foram definidos 18 cenários de configuração de rede distintos para cobrir o impacto dessas variáveis na estabilidade e performance do sistema. A delimitação do escopo para uma variação de 4 a 12 nós validadores não foi arbitrária. Esse intervalo reflete de forma realista o dimensionamento típico de consórcios permissionados institucionais, como federações de

pesquisa ou redes integradas de instituições de saúde colaborativas, permitindo avaliar o comportamento do consenso BFT em topologias que não visam a descentralização global pública, mas sim a governança restrita.

Tabela 2. Configurações da Rede

Nós	T. Bloco (s)	Consenso	Versão Besu
4, 6, 8, 10, 12	5	QBFT, IBFT	26.2.0
6	10, 15, 20, 25	QBFT, IBFT	26.2.0

Para avaliar de forma mais abrangente o comportamento desses cenários sob condições de estresse, cada função do contrato *Simple* (`open`, `query` e `transfer`) foi submetida a diferentes níveis de carga de trabalho, especificamente 125, 150, 175, 200, 225, 250, 275 e 300 TPS, com uma quantidade fixa de 2000 transações por benchmark. Cada experimento resulta da combinação única entre um cenário de rede e um perfil de carga, totalizando $18\,(cenários) \times 8\,(TPS) \times 3\,(funções) = 432$ relatórios gerados.

4. Resultados experimentais e análise

Visando a transparência e a reprodutibilidade dos resultados, todos os relatórios e scripts de análise foram estruturados em um repositório público no GitHub[2]. Tal organização permite a rastreabilidade dos artefatos empregados na geração dos gráficos e análises deste estudo, constituindo uma base sólida para a validação das métricas e o aprofundamento das investigações de forma mais específica e detalhada.

4.1. Condução dos resultados

As análises foram conduzidas com o objetivo de identificar padrões de comportamento e métricas de desempenho em cenários de rede variando a quantidade de nós validadores e o tempo de geração de blocos. Para garantir a representatividade estatística e a comparabilidade entre as tecnologias, cada cenário de teste foi dividido equitativamente: metade dos experimentos foi executada utilizando o algoritmo de consenso QBFT e a outra metade utilizando o IBFT.

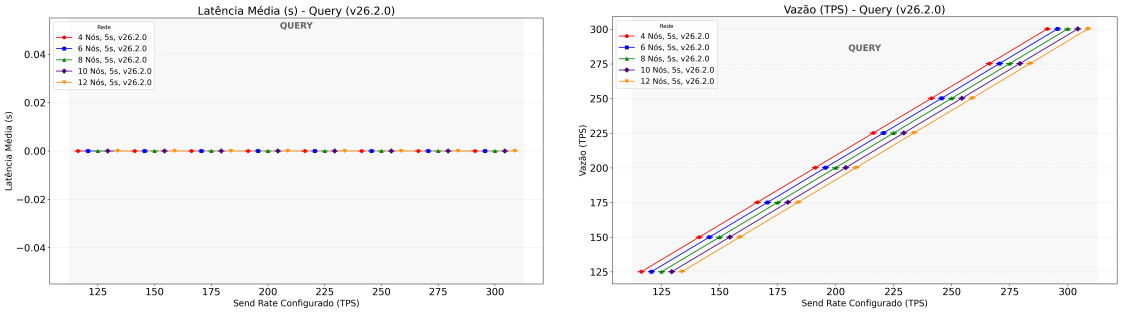
Dessa forma, a dispersão ilustrada nos gráficos de *boxplot* subsequentes não representa apenas variações aleatórias, mas evidencia a variância de resultados entre esses dois algoritmos em cada ponto específico de carga (TPS). Observou-se que, em termos de médias globais, ambos os algoritmos apresentam desempenho similar (vazão aproximada de 126 TPS para a função `open`), com variações pontuais que justificam o uso de *boxplots* para evidenciar a estabilidade de cada um.

4.2. Impacto no número de nós

Para avaliar o escalonamento da rede, foram analisados 10 cenários mantendo o tempo de bloco fixo em 5 segundos e variando o número de nós validadores entre 4, 6, 8, 10 e 12. Conforme ilustrado na Figura 2, a performance da função `query` representa o

[2]`https://github.com/viniciusSt1/Resultados-Benchmarks/tree/main/ViniciusXeonE5/432experimentos/graficos_artigo`

limite superior teórico do ambiente. Por tratar-se de uma operação de leitura que não exige consenso global nem altera o estado da blockchain, a latência observada foi desprezível (valores próximos ou iguais a zero) em todos os testes. Consequentemente, a vazão acompanhou linearmente a taxa de envio, atingindo o máximo de 300 TPS sem degradação, confirmando que o gargalo do sistema não reside na camada de interface ou rede básica, mas sim no processamento de transações de escrita.

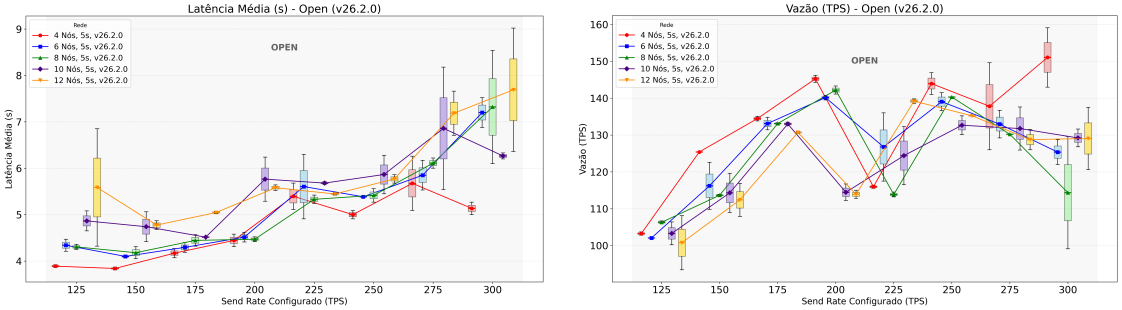


(a) Latência da query (b) Vazão query

Figura 2. Comportamento da função query com aumento do TPS

Ao analisar a função `open` (Figura 3), observa-se que o aumento no número de nós validadores impacta diretamente a latência média, assim como o incremento da taxa de envio força a subida tanto da latência quanto da vazão até o ponto de saturação. Para uma carga de 125 TPS, a latência cresce de $3,89$ s (4 nós) para $5,58$ s (12 nós), um aumento de aproximadamente 43%, padrão que se repete de forma assintótica conforme a carga de trabalho escala. Esse fenômeno é inerente aos algoritmos de consenso baseados em PBFT (como QBFT e IBFT), nos quais a complexidade de comunicação e o tempo de propagação de mensagens aumentam com o número de participantes.

A vazão para a função `open` demonstrou saturação a partir de 175 TPS, mantendo-se estável na faixa de 130–150 TPS, independentemente do aumento da taxa de envio. Notamos ainda uma maior variância nos resultados entre os algoritmos de consenso, especialmente nos pontos de carga elevada (acima de 250 TPS), onde a dispersão dos dados indica uma instabilidade na manutenção da performance sob estresse severo.



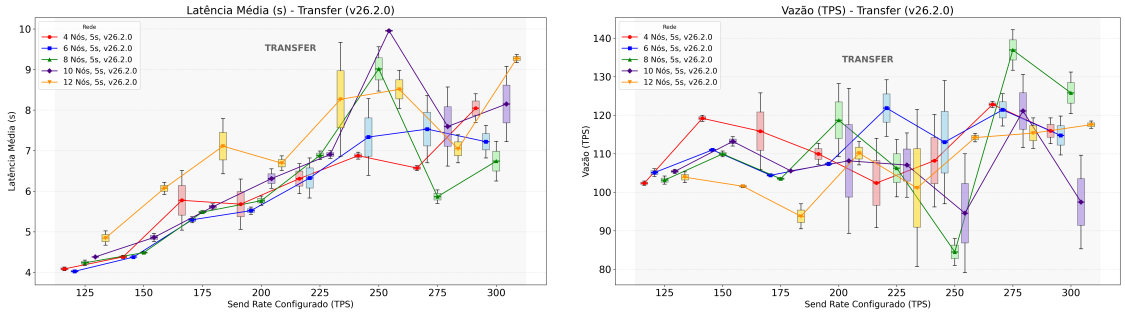
(a) Latência de open (b) Vazão de open

Figura 3. Desempenho da função open para diferentes números de nós

A função `transfer` (Figura 4) apresentou um desempenho ligeiramente inferior

ao da função `open`, com a vazão média estabilizando em torno de 110 TPS. A latência média para 12 nós sob carga máxima (300 TPS) atingiu o pico de $9,27$ s. Essa diferença decorre da maior complexidade computacional da operação de transferência, que exige verificações adicionais de saldo e estado no *Smart Contract*. Notavelmente, observa-se no gráfico 4(b) uma menor correlação entre o número de nós e as cargas de trabalho em relação à vazão na função `transfer`, revelando um comportamento menos previsível e controlado que o da função `open`.

Essa oscilação é visível tanto com o aumento do número de nós quanto com o incremento da taxa de envio, sugerindo que a lógica de negócio mais densa da transferência torna o sistema mais sensível a micro-atrasos de rede e processamento.



(a) Latência de transfer

(b) Vazão de transfer

Figura 4. Desempenho da função transfer para diferentes números de nós

4.3. Impacto do tempo de bloco

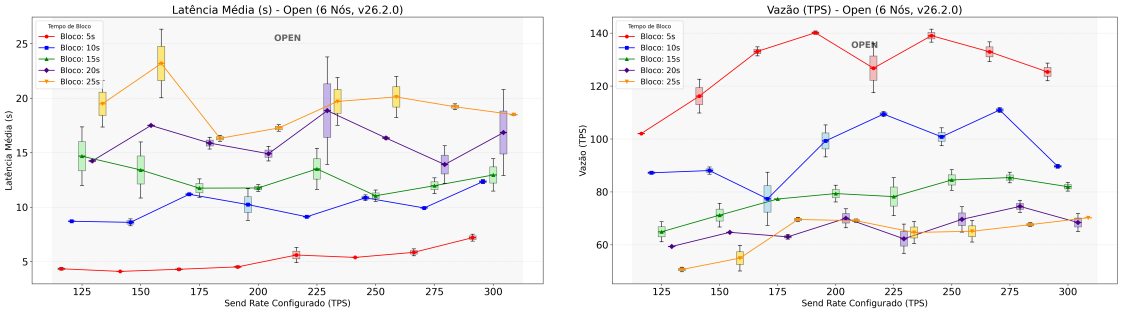
Para analisar a influência da frequência de geração de blocos da rede na performance, fixou-se a rede com 6 nós, variando o tempo de bloco de 5 a 25 segundos. Os resultados para a função `query` mantiveram-se estáveis, reforçando sua independência da camada de consenso.

Para a função `open` (Figura 5), identifica-se que o aumento do tempo de bloco resulta invariavelmente no aumento da latência e na diminuição da vazão. Há uma correlação linear positiva entre o tempo de bloco e o tempo de resposta: com o bloco em 25 s, a latência média saltou para aproximadamente 20 s.

Esse atraso ocorre porque as transações permanecem retidas no *mempool* por períodos maiores. A vazão sofreu degradação severa, caindo de uma média de 102 TPS (bloco de 5 s) para apenas 50 TPS (bloco de 25 s) sob carga inicial de 125 TPS.

Por fim, a função `transfer` (Figura 6) demonstrou ser a operação mais sensível às variações de infraestrutura, apresentando o maior impacto negativo e um comportamento substancialmente menos controlado sob tempos de bloco elevados. Em cenários de estresse com blocos de 25 segundos, a latência média superou a marca de 26 s.

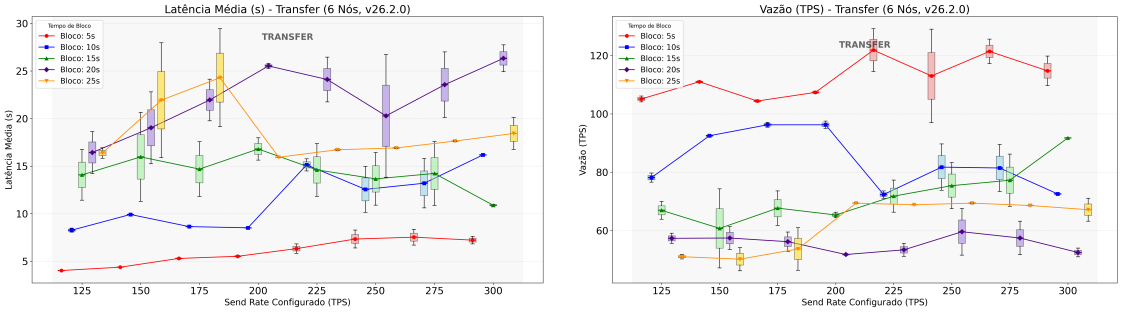
Esse comportamento errático sugere que a combinação de uma lógica de contrato mais complexa com janelas de fechamento de bloco mais esparsas cria um efeito de enfileiramento não linear, onde pequenos atrasos na propagação do consenso resultam em exclusões de transações que precisam aguardar ciclos subsequentes, aumentando exponencialmente o tempo de resposta final.Dessa forma, conclui-se que, para aplicações



(a) Latência de open (b) Vazão de open

Figura 5. Desempenho da função open para diferentes durações de bloco

industriais ou financeiras que demandam alta vazão e determinismo no Hyperledger Besu, a configuração de tempos de bloco reduzidos (próximos a 5 s) é fundamental. O aumento deste parâmetro não atua apenas como um limitador proporcional, mas impõe um ”teto físico”rígido à capacidade de vazão da rede. À medida que as janelas de oportunidade para inclusão de transações diminuem em frequência, o sistema perde previsibilidade, tornando-se suscetível a gargalos onde o *mempool* não consegue ser esvaziado na mesma taxa em que novas requisições são recebidas, degradando a confiabilidade do serviço sob carga constante.



(a) Latência de transfer (b) Vazão de transfer

Figura 6. Desempenho da função transfer para diferentes durações de bloco

5. Conclusões e trabalhos futuros

O trabalho apresentou o desenvolvimento de uma arquitetura integralmente automatizada para execução sistemática de experimentos em redes Hyperledger Besu, unificando processos antes distribuídos em múltiplas ferramentas e sujeitos a falhas de configuração. A solução proposta reduz significativamente o esforço operacional, fornece maior previsibilidade entre execuções e viabiliza a condução eficiente de grandes baterias de testes, contribuindo para maior rigor metodológico em estudos de desempenho de blockchains permissionadas. A consolidação e estabilização dos repositórios existentes, aliada à automação do ciclo experimental, resultou em um ambiente reprodutível, escalável e adequado a diferentes combinações de parâmetros de rede, além da análise de resultados obtidos com sua execução em um ambiente controlado.

Como perspectivas futuras, destaca-se a expansão da arquitetura para abranger novos algoritmos de consenso, diferentes perfis de contratos inteligentes e ferramentas avançadas de análise automática de resultados. O desenvolvimento de visualizações em tempo real e mecanismos de recomendação permitirá identificar gargalos e sugerir ajustes de configuração de forma proativa. Além disso, faz-se relevante aprofundar a análise empírica por meio da reprodução dos experimentos em diferentes infraestruturas de hardware e de conectividade. A integração desta arquitetura de avaliação com redes de telecomunicação avançadas, como redes 5G privativas, permitirá avaliar não apenas a latência da camada de consenso, mas a latência fim-a-fim, agregando o impacto da interface aérea em aplicações de telemetria contínua.

Neste contexto, a investigação do impacto de desempenho entre múltiplas versões do *Hyperledger Besu* possibilitará o mapeamento de evoluções, regressões e efeitos de alterações internas do cliente. Por fim, análises comparativas e pontuais entre os algoritmos QBFT e IBFT podem ser aprofundadas em estudos posteriores para isolar variáveis de estabilidade e eficiência. Tal abordagem permitirá identificar com precisão os cenários que recomendam o uso de cada protocolo, estabelecendo critérios de escolha baseados no equilíbrio ideal entre tolerância a falhas, latência e vazão exigida por diferentes perfis de aplicação.

Agradecimentos

Os autores agradecem o apoio concedido pelo Ministério da Ciência, Tecnologia e Inovação (MCTI) por meio de duas fontes: recursos da Lei nº 8.248, de 23 de outubro de 1991, no âmbito do PPI SOFTEX (coordenado pela Softex e publicado como Residência em TIC 11, DOU 01245.011733/2022-83), e recursos financeiros do FUNTTEL, administrados pela FINEP, especificamente no âmbito do projeto ”Saúde 5G - Segurança, Privacidade, Inclusão e Qualidade em Telemedicina no Contexto da Web 3.0”(Convênio nº 01.23.0468.00, Referência 0844/23).

Referências

Bansod, S. and Ragha, L. (2024). Intelligent systems and applications in engineering. *IJISAE*. Accessed: 2026-02-04.

Besu, H. (2023a). Consensus protocols overview. Acesso em: 04 fev. 2026.

Besu, H. (2023b). Hyperledger besu documentation. Acesso em: 04 fev. 2026.

Cardoso, J. and et al. (2024). An API-driven framework for performance testing of Hyperledger Besu blockchain networks. In *Anais do Simpósio Brasileiro de Computação Confiável, Segura e Privacidade (LADC/SBC)*, Porto Alegre. SBC.

Crawford, M. (2025). Permissioned vs. permissionless blockchains: Key differences. Alchemy Learn – Blockchain Overviews. Accessed: 2025-12-03.

Dabbagh, M., Choo, K., Beheshti, A., Tahir, M., and Safa, N. (2021). A survey of empirical performance evaluation of permissioned blockchain platforms: Challenges and opportunities. *Computers & Security*, 100:102078.

Fan, C. et al. (2022). Performance analysis of hyperledger besu in private blockchain. In *2022 IEEE International Conference on Decentralized Applications and Infrastructures (DAPPS)*. IEEE.

Fan, C., Ghaemi, S., Khazaei, H., and Musílek, P. (2020). Performance evaluation of blockchain systems: A systematic survey. *IEEE Access*, 8:126927–126950.

Kaushal, R. K. and Kumar, N. (2024). Exploring hyperledger caliper benchmarking tool to measure the performance of blockchain based solutions. In *2024 11th International Conference on Reliability, Infocom Technologies and Optimization (ICRITO)*. IEEE.

Marchesi, L., Pompianu, L., and Tonelli, R. (2025). Security checklists for ethereum smart contract development: Patterns and best practices. *Blockchain: Research and Applications*, page 100367. Accessed: 2025-12-03.

Mendes, V. S. (2025). Besubenchmarking: Projeto de automatização de benchmarks utilizando hyperledger besu, hyperledger caliper e hardhat. GitHub repository, https://github.com/viniciusSt1/BesuBenchmarking. Accessed: 2025-12-03.

Nasrulin, B. et al. (2022). Gromit: Benchmarking the performance and scalability of blockchain systems. *arXiv*.

Ren, K. et al. (2023). Bbsf: Blockchain benchmarking standardized framework. In *Proceedings of the 1st Workshop on Verifiable Database Systems*, pages 10–18. ACM.

Saraiva, R., Araújo, A. A., Soares, P., Pontes, J. C., and Souza, J. (2025). Metrics for quality assessment in blockchain-based systems: A systematic mapping study. In *Anais do XXI Simpósio Brasileiro de Sistemas de Informação (SBSI)*, pages 545–554. SBC.

Sousa, J. (2024a). besu-production-docker: Hyperledger besu for private networks. GitHub repository, https://github.com/jeffsonsousa/besu-production-docker. Accessed: 2025-12-03.

Sousa, J. (2024b). contracts-node-health-monitor. GitHub repository, https://github.com/jeffsonsousa/contracts-node-health-monitor. Accessed: 2025-12-03.

Sousa, J. (2024c). evaluation-contracts-indy-besu. GitHub repository, https://github.com/jeffsonsousa/evaluation-contracts-indy-besu. Accessed: 2025-12-03.

Sousa, J., Evaristo, B., Mateus, A., Ávila, I., Veloso, A., and Abelém, A. (2025). Performance evaluation of decentralized digital identity contracts on ethereum-based blockchain networks. In *2025 7th International Conference on Blockchain Computing and Applications (BCCA)*, pages 285–292.

Thakur, A., Ranga, V., and Agarwal, R. (2023). Performance benchmarking and analysis of blockchain platforms. In *2023 International Conference on Computational Intelligence, Data Analytics and Information Sciences (ICCDI)*, pages 1–7.

Tran, N. K., Babar, M. A., and Walters, A. (2022). A framework for automating deployment and evaluation of blockchain network. *arXiv*. Accessed: 2026-02-04.