Avaliação de Performance de Contratos de Identidade Digital Descentralizada em Redes Blockchain Baseada em Ethereum

Jeffson Celeiro Sousa[1,2], **Bruno Evaristo**[1,2],
Antonio Mateus de Sousa[1], **Ismael Ávila**[1]

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

[2]Universidade Federal do Pará (UFPA)
Belém – PA – Brasil

{jcsousa, elderb, amateus, avila_an}@cpqd.com.br

Resumo. *Este artigo apresenta uma avaliação de performance de contratos inteligentes voltados à gestão de identidades digitais descentralizadas em redes blockchain baseadas em Ethereum. A análise foca em operações fundamentais do ciclo de vida de identidades, como criação, atualização, definição de esquemas de credenciais e controle de revogação, implementadas em contratos Solidity. Foram considerados dois contextos de execução: um ambiente com Hyperledger Besu operando em modo permissionado, e uma referência ao modelo tradicional Hyperledger Indy. Os testes foram conduzidos em rede privada simulando diferentes níveis de carga e configurações de consenso. As métricas avaliadas incluem tempo de resposta, vazão, uso de recursos (CPU e memória) e escalabilidade. Os resultados fornecem subsídios para a escolha de arquiteturas eficientes para soluções de identidade digital baseadas em SSI (Self-Sovereign Identity) e Ethereum, especialmente em cenários corporativos ou regulados.*

Abstract. *This paper presents a performance evaluation of smart contracts designed for managing decentralized digital identities on Ethereum-based blockchain networks. The analysis focuses on core identity lifecycle operations such as creation, update, credential schema definition, and revocation control, all implemented through Solidity smart contracts. Two execution contexts were considered: an environment using Hyperledger Besu operating in permissioned mode, and a reference to the traditional Hyperledger Indy model. The tests were conducted in a private network simulating different load levels and consensus configurations. The evaluated metrics include response time, throughput, resource usage, and scalability. The results provide insights to support the selection of efficient architectures for digital identity solutions based on Self-Sovereign Identity (SSI) and Ethereum, particularly in enterprise or regulated environments.*

1. Introdução

A Identidade Digital Descentralizada (IDD) ou Identidade Autossoberana (SSI) surgiu como uma resposta aos modelos tradicionais de gerenciamento de identidade digital, que frequentemente dependem de autoridades centralizadas e expõem os usuários a riscos de

privacidade, vazamentos de dados e falta de portabilidade [Allen 2016]. Com o apoio de iniciativas como o W3C DID [W3C 2022a] e W3C Verifiable Credentials [W3C 2022b], a SSI busca empoderar indivíduos e organizações por meio de identificadores descentralizados e credenciais verificáveis.

O *Hyperledger Indy* foi uma das primeiras plataformas a implementar os princípios da SSI de forma prática, oferecendo um ledger público permissionado baseado em um protocolo de consenso tolerante a falhas bizantinas (RBFT) [Hyperledger Foundation 2019]. Sua adoção foi impulsionada por redes como Sovrin e frameworks como Hyperledger Aries, que utilizam o Indy como *backend* para resoluções de DIDs, publicação de esquemas de credenciais e controle de revogação [Foundation 2020].

Com o tempo, desafios relacionados à escalabilidade, interoperabilidade e evolução dos padrões motivaram o surgimento de propostas alternativas. Entre elas, destaca-se o *Indy Besu*, uma iniciativa que visa migrar a lógica de identidade do Indy para uma nova infraestrutura baseada no *Hyperledger Besu*, um cliente Ethereum com suporte nativo a contratos inteligentes em Solidity [Community 2024a].

O uso de contratos inteligentes para identidade descentralizada já é explorado em métodos como `did:ethr`, baseado no padrão ERC-1056, permitindo a gestão de DIDs autônomos na Ethereum [Thorstensson 2018]. Contudo, avaliações sistemáticas da performance dos contratos DID em redes permissionadas, especialmente no contexto do Indy Besu, ainda são escassas na literatura. Este artigo busca preencher essa lacuna, propondo um estudo comparativo da execução de contratos DID na rede Besu em diferentes cenários. A avaliação de desempenho foi feita por meio de experimentos numa rede *Hyperledger Besu* permissionada, configurada com diferentes parâmetros de rede e carga transacional.

Este trabalho está organizado da seguinte forma: a Seção 2 apresenta os trabalhos relacionados à proposta. A Seção 3 apresenta a descrição da proposta. A Seção 4 apresenta a metodologia de avaliação do trabalho. A Seção 5 apresenta os resultados. E, por fim, a Seção 6 apresenta a conclusão e trabalhos futuros acerca deste estudo.

2. Trabalhos Relacionados

Diversos estudos recentes têm se dedicado à avaliação de desempenho de plataformas blockchain, incluindo Ethereum, Hyperledger Fabric e Hyperledger Indy, refletindo sua adoção crescente em aplicações empresariais e institucionais. Uma das ferramentas mais utilizadas nesses estudos é o Hyperledger Caliper, que oferece suporte padronizado e reprodutível para benchmarking de redes permissionadas e públicas.

[Kaushal e Kumar 2024] utilizaram o Hyperledger Caliper para avaliar uma rede Fabric simulando um sistema de monitoração remota de pacientes (RPM). A rede incluía três organizações com suas respectivas autoridades certificadoras e nós ordenadores. O estudo mediu latência e vazão em operações de leitura e gravação, e concluiu que a Fabric apresentou bom desempenho sob diferentes taxas de transação, com mínimas variações.

[Kshirsagar e Pachghare 2022] propuseram um novo algoritmo de consenso chamado Proof of Scope, comparando-o com Raft e PoW-Ethash na plataforma Hyperledger. O mecanismo mostrou ganhos de até 38% na latência e 22% na vazão. O estudo destaca como mecanismos alternativos de consenso podem impactar o desempenho da rede.

[Melo et al. 2024] criaram um modelo de desempenho para a Fabric utilizando *Stochastic Petri Nets*, medindo parâmetros como latência, vazão e utilização. A validação experimental do modelo com alto nível de confiabilidade mostrou que o tamanho do bloco e as políticas de endosso são fatores críticos para o desempenho.

[Choi e Won-Ki Hong 2021] compararam o desempenho de uma Ethereum Private Network com a testnet Ropsten. Utilizando Caliper, mediram latência, vazão e estabilidade da rede. A rede privada obteve desempenho superior, com menor latência e maior taxa de transações por segundo. O estudo demonstrou que transações simples têm melhor desempenho, mesmo com limites de gás equivalentes.

[Bastos et al. 2024] apresentaram o MinIndy, uma ferramenta que automatiza a implantação e gestão de redes Hyperledger Indy. A solução facilita o uso de SSI ao automatizar tarefas repetitivas com Ansible e Docker. Avaliações mostraram que o desempenho da rede Indy mantém-se equivalente à implantação manual tradicional.

No contexto da Hyperledger Besu, estudos ainda são escassos. Em [Fan et al. 2022], foi realizada uma avaliação detalhada utilizando Caliper para analisar os algoritmos de consenso PoA, IBFT 2.0 e QBFT em redes Besu. Os resultados mostraram que o QBFT escala até 14 validadores sem degradação perceptível e que tempo de bloco e tamanho afetam significativamente o desempenho.

[Mostarda et al. 2023] propuseram uma ferramenta personalizada de *benchmarking* para redes Besu, superando limitações do Caliper em redes reais operadas por múltiplas organizações. A ferramenta detectou anomalias em validadores que adicionavam blocos vazios ou com poucas transações, revelando assimetrias na contribuição entre nós.

A Tabela 1 resume os principais aspectos dos trabalhos discutidos. Em geral, este trabalho se diferencia principalmente em dois pontos: (i) por realizar uma avaliação de desempenho focada em contratos inteligentes voltados à identidade descentralizada (DID) na Hyperledger Besu, baseando-se em operações fundamentais do ciclo de vida de identidades digitais; e (ii) por apresentar um cenário reprodutível com configurações completas de rede, contratos e *benchmarking* disponibilizados publicamente.

Trabalho	Blockchain	Caliper	Consenso Avaliado	Modelo Reprodutível	Avaliação DID	Ferramenta Personalizada
[Kaushal e Kumar 2024]	Fabric	✓	Solo	✗	✗	✗
[Kshirsagar e Pachghare 2022]	Fabric	✓	Proof of Scope	✗	✗	✗
[Melo et al. 2024]	Fabric	✓	Solo	✓	✗	✗
[Choi e Won-Ki Hong 2021]	Ethereum	✓	PoW	✗	✗	✗
[Bastos et al. 2024]	Indy	✗	RBFT	✓	✓	✗
[Fan et al. 2022]	Besu	✓	QBFT, IBFT 2.0, PoA	✗	✗	✗
[Mostarda et al. 2023]	Besu	✓	QBFT	✓	✗	✓
Proposta	Besu	✓	QBFT	✓	✓	✗

Tabela 1. Resumo dos trabalhos relacionados e suas características principais.

3. Proposta de arquitetura e modelo de avaliação

Esta seção descreve a arquitetura proposta para a realização dos experimentos de avaliação de desempenho de contratos inteligentes de identidade descentralizada (DID) nas plataformas *Hyperledger Indy* e *Indy Besu*. O objetivo é permitir uma comparação sistemática entre as duas abordagens, considerando aspectos como desempenho, complexidade de operação e escalabilidade.

3.1. Hyperledger Besu

A *Hyperledger Besu* é um cliente Ethereum de código aberto desenvolvido em Java e mantida como um projeto graduado pela Hyperledger Foundation desde 2020 [Besu 2023b]. Ela é compatível com a Ethereum Virtual Machine (EVM) e permite a operação em redes públicas e privadas, com suporte completo a contratos inteligentes escritos em Solidity. No contexto de redes permissionadas, a Besu oferece suporte a algoritmos de consenso como *Clique* (Proof of Authority), *QBFT* e *IBFT 2.0*, viabilizando governança controlada sobre a validação de blocos [Besu 2023a].

Uma das principais vantagens da Besu é sua arquitetura modular, que permite a integração de *plugins*, gerenciamento de permissões por conta, canais privados de transação, e APIs RPC compatíveis com ferramentas Ethereum existentes [Besu 2023c]. Ela também oferece ferramentas de monitoramento nativas via Prometheus e métricas detalhadas para análise de desempenho. Além disso, sua compatibilidade com bibliotecas como `ethers.js` e *frameworks* como Hardhat facilita o desenvolvimento, teste e automação de contratos inteligentes.

No contexto de identidade descentralizada, a Besu serve como base para a iniciativa *Indy Besu* [Community 2024b], que implementa contratos para operações DID como `createDid`, `updateDid` e `createCredentialDefinition`. Essa abordagem permite a execução de identidades digitais descentralizadas em um ambiente Ethereum permissionado, utilizando endereços como identificadores e mantendo compatibilidade com métodos estabelecidos como `did:ethr` e extensões como `did:indy:besu`. Essa flexibilidade torna a Besu uma alternativa moderna e escalável à *ledger* tradicional da Hyperledger Indy.

3.2. Hyperledger Indy

A *Hyperledger Indy* é um *framework* especializado de identidade digital descentralizada, com foco exclusivo na gestão de DIDs e credenciais verificáveis. Lançado originalmente pela Sovrin Foundation e posteriormente incorporado à Hyperledger como projeto graduado, a Indy provê uma *ledger* permissionada pública baseada no protocolo de consenso RBFT (Redundant Byzantine Fault Tolerance), desenvolvido para suportar confiança distribuída entre nós validadores [Indy 2022].

A arquitetura da Indy é composta por dois componentes principais: a *Indy Ledger*, responsável por armazenar as transações de identidade como `NYM` (para criação de DIDs), `SCHEMA` (para definição de atributos), `CRED_DEF` (para definição de credenciais) e registros de revogação [Indy 2023]; e a *Indy SDK*, um conjunto de bibliotecas escritas em Rust com *wrappers* para diversas linguagens, utilizado por agentes Aries para interagir com a *ledger*, gerir carteiras e estabelecer conexões *peer-to-peer* com segurança criptográfica.

Apesar de sua maturidade e adoção em ambientes como a rede Sovrin, a Hyperledger Indy tem limitações de desempenho, interoperabilidade com padrões modernos (como `did:ethr`) e dificuldades na evolução da base de código devido à sua arquitetura monolítica [Community 2023]. A iniciativa *Indy Besu* surge como uma proposta complementar que busca migrar a lógica de identidade da Indy para contratos inteligentes em Ethereum, preservando a semântica das transações originais e facilitando a integração com ferramentas amplamente utilizadas no ecossistema Ethereum. Essa abordagem também melhora o desempenho da rede e reduz os requisitos computacionais dos nós.

Dessa forma, é oportuno comparar as funções utilizadas nas duas tecnologias a fim de avaliar o desempenho da Indy Besu em termos das operações correspondentes na Hyperledger Indy, conforme especificadas na Indy DID Method Specification. A Tabela 2 apresenta essa comparação:

Tabela 2. Comparação entre funções da Indy Besu e operações da Hyperledger Indy

Função na Indy Besu	Operação Correspondente na Hyperledger Indy	Descrição
`createDid`	`NYM`	Cria um novo Identificador Descentralizado (DID) na rede. Na Indy, a transação `NYM` é utilizada para registrar um novo DID, associando-o a uma chave pública e outros metadados relevantes.
`updateDid`	`NYM`	Atualiza um DID existente. Na Indy, a transação `NYM` também é usada para modificar informações de um DID já registrado, como atualizar chaves públicas ou alterar permissões associadas.
`createRevocation Registry`	`REVOC_REG_DEF`	Cria um Registro de Revogação para gerenciar o status de revogação de credenciais emitidas. Na Indy, a transação `REVOC_REG_DEF` define um novo registro de revogação vinculado a uma definição de credencial específica.
`createOrUpdate Entry`	`REVOC_REG_ENTRY`	Adiciona ou atualiza entradas em um Registro de Revogação existente. A transação `REVOC_REG_ENTRY` na Indy é usada para modificar o estado de revogação de credenciais específicas dentro de um registro de revogação.
`createSchema`	`SCHEMA`	Cria um novo esquema que define a estrutura de atributos para credenciais. Na Indy, a transação `SCHEMA` é utilizada para registrar um esquema contendo os nomes dos atributos que serão incluídos nas credenciais emitidas.
`createCredential Definition`	`CRED_DEF`	Cria uma definição de credencial baseada em um esquema existente. Na Indy, a transação `CRED_DEF` estabelece os parâmetros criptográficos e associa um esquema a um emissor específico para a emissão de credenciais.

As operações na *Hyperledger Indy* são realizadas por meio de transações específicas que interagem com a *ledger* para registrar, atualizar ou consultar os diversos objetos relacionados à identidade descentralizada.

No contexto da *Hyperledger Besu*, essas funções são implementadas como contratos inteligentes na rede *Hyperledger Besu*, aproveitando a flexibilidade e o poder dos contratos em *Solidity* para gerenciar identidades descentralizadas e credenciais verificáveis.

A transação `ATTRIB`, anteriormente utilizada na Indy para adicionar atributos a um DID, foi substituída pelo método `did:indy`, que permite a inclusão direta de

endpoints de serviço e outros dados no documento DID.

Essa comparação destaca como as funcionalidades essenciais para a gestão de identidades descentralizadas e credenciais são implementadas em ambas as plataformas, refletindo a evolução das tecnologias de identidade digital no ecossistema *Hyperledger*.

3.3. Fluxo de operações DID do cenário

O fluxo de operações para ambas as redes segue a mesma lógica funcional, com as seguintes etapas:

1. **Criação de Identificador (createDid):** geração de um par de chaves criptográficas e registro do DID na *ledger*.
2. **Atualização de Identificador (updateDid):** modificação da chave pública ou metadados do DID.
3. **Criação de Esquema (createSchema):** definição de um conjunto de atributos que serão utilizados em credenciais.
4. **Criação de Definição de Credencial (createCredentialDefinition):** associação entre o emissor e o esquema definido, incluindo parâmetros criptográficos.
5. **Criação do Registro de Revogação (createRevocationRegistry):** inicialização de um conjunto de controle de validade para credenciais emitidas.
6. **Entrada de Revogação (createOrUpdateEntry):** atualização do status de uma ou mais credenciais emitidas.

4. Avaliação

Nesta seção, descrevemos a metodologia adotada, a configuração experimental dos ambientes avaliados e os parâmetros de execução utilizados nos testes. Os experimentos foram conduzidos de forma controlada, variando a taxa de envio de transações — definida como o número total de requisições de transações enviadas por segundo por todos os *workers* (req/s) — de 20 até 120 req/s, com incremento de 10 e duração fixa de 10 segundos por rodada. Essa configuração foi definida diretamente nos arquivos YAML utilizados pela ferramenta de *benchmark*.

Todos os artefatos utilizados neste estudo — incluindo códigos-fonte, arquivos de configuração, contratos inteligentes, módulos de carga, logs e resultados experimentais — estão disponíveis publicamente em nosso repositório reprodutível[1].

Para efeito de comparação, avaliamos o desempenho de operações relacionadas à gestão de Identidades Descentralizadas (DID) em dois contextos distintos:

- **Cenário 1 — Hyperledger Indy:** Em [Bastos et al. 2024], foi avaliada uma rede Indy, implantada sobre quatro máquinas virtuais com as seguintes especificações: 4 vCPUs (Intel Xeon E312xx a 2.0 GHz), 4 GB de RAM, rodando Ubuntu 20.04. Esses nós foram configurados como validadores RBFT numa rede permissionada.
- **Cenário 2 — Hyperledger Besu:** Implantada uma rede blockchain baseada no Hyperledger Besu em um servidor dedicado com Ubuntu 22.04, 32 GB de RAM e 12 núcleos de CPU. A rede foi configurada com quatro nós validadores utilizando o consenso QBFT e 2 *bootnodes*, seguindo as recomendações da documentação oficial para ambientes de produção.

[1] `https://github.com/jeffsonsousa/evaluation-contracts-indy-besu`

4.1. Ferramenta de avaliação de desempenho

Para garantir que as medições de desempenho refletissem a lógica dos contratos inteligentes e não fossem afetadas por limitações operacionais de consumo de gás, todas as redes Besu foram configuradas como *gasless*. Especificamente, definiu-se:

- `gasLimit` = `0x1fffffffffff`;
- `contractSizeLimit` = 2147483647;
- `min-gasprice` = 0;
- `difficulty` = `0x1`;
- `txPoolLimit` = 4096 (padrão).

A ferramenta utilizada para execução dos testes foi o *Hyperledger Caliper*[2] na versão 0.5.0, utilizando o Ethereum SDK v1.4. O Caliper permitiu a orquestração de chamadas aos contratos inteligentes implementados em Solidity para simular operações como `createDid`, `createSchema`, entre outras.

Ademais, configurou-se a monitoração com Docker para coletar métricas de uso de CPU e memória dos contêineres dos nós da rede Besu durante cada rodada de teste. Os resultados foram processados automaticamente e armazenados em relatórios HTML e CSV, que foram utilizados para gerar os gráficos e análises apresentados neste trabalho.

4.2. Cenários de teste

A Figura 1 apresenta a arquitetura utilizada para a execução do cenário de avaliação de desempenho dos contratos inteligentes de identidade digital descentralizada implantados na rede Hyperledger Besu. O processo é conduzido com o auxílio da ferramenta *Hyperledger Caliper*, que permite simular múltiplos clientes (*workers*) enviando transações para o ledger por meio de chamadas aos contratos Solidity responsáveis pelas funções de identidade. A seguir, é detalhada cada etapa numerada da figura.

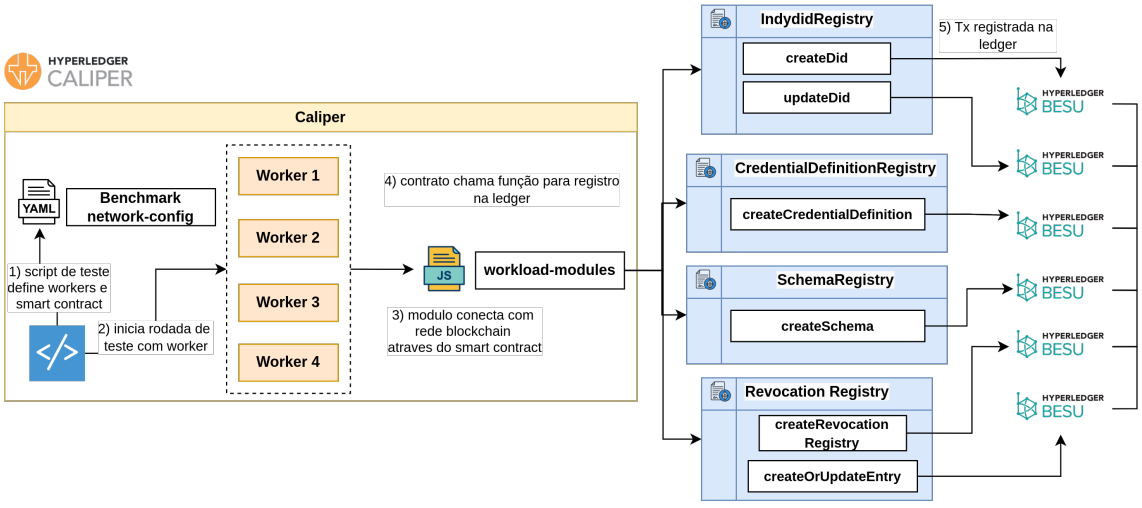

Figura 1. Arquitetura de avaliação de performance dos contratos de identidade na rede Hyperledger Besu com Hyperledger Caliper.

Na primeira etapa, o *benchmark* é configurado por meio de arquivos no formato YAML, os quais especificam: a topologia da rede blockchain (nós, validadores,

[2] `https://www.lfdecentralizedtrust.org/projects/caliper`

consenso), os contratos inteligentes a serem avaliados, os módulos de carga (*workload-modules*) e o número de *workers* que irão simular clientes concorrentes. Cada *worker* será responsável por executar chamadas independentes e paralelas aos contratos definidos, simulando um ambiente de uso realista.

Em seguida, com os arquivos de configuração prontos, um orquestrador simples foi criado para gerenciar no Caliper as instâncias dos *workers* conforme especificado, ativando as rotinas de teste definidas no plano de carga. Cada *worker* executa chamadas específicas às funções do contrato inteligente.

No passo 3, os módulos de carga, escritos em JavaScript, encapsulam a lógica de envio de transações à rede. Cada *worker* carrega seu módulo, que realiza a conexão com a rede Besu, invocando diretamente funções específicas dos contratos inteligentes. No passo 4, cada módulo conecta-se à rede Besu e chama funções dos contratos, de acordo com o tipo de operação avaliada.

Após a execução da função, a transação é propagada e registrada no *ledger* da rede Besu. Com o uso do mecanismo de consenso *QBFT (Quorum Byzantine Fault Tolerance)*, os blocos contendo as transações são validados e adicionados à blockchain.

4.3. Métricas de coleta

O principal objetivo de uma aplicação blockchain é tratar uma série de transações enviadas pelos participantes e proceder ao processo de verificação e requisição, levando à geração de um bloco e o resultado da transação sendo registrado no livro-razão distribuído [Hang e Kim 2019, Afraz e Ruffini 2020]. Portanto, o desempenho de nosso aplicativo blockchain é medido pelas métricas vazão e tempo de resposta da transação.

Vazão da transação (Vt): refere-se ao número de transações cujo resultado a blockchain pode processar e registrar na *ledger* num dado segundo. Equação 1:

$$Vt = \frac{TotalValidTransactions}{TotalTime(s)} \tag{1}$$

Tempo de resposta da transação (Trt): refere-se ao tempo que uma transação leva do momento em que é chamada pelo cliente até ser salva na *ledger*. Equação 2:

$$Trt = ConfirmationTime * Threshold - SendTime \tag{2}$$

5. Resultados

Com base nos gráficos de vazão e latência, as Figuras 3 e 4 mostram que as operações de escrita na rede Besu, como `createDid`, `createSchema` e `create Credential Definition`, atingiram picos de vazão de até 37,5 TPS, 58 TPS e 49 TPS, respectivamente. Em contraste a Figura 2 mostra os testes realizados na rede Indy reportaram tetos significativamente mais baixos: 14 TPS para `createDid`, 9 TPS para `createSchema` e 8 TPS para `createOrUpdateEntry` (atributo). Isso representa um aumento de aproximadamente 168%, 600% e 511% na vazão para as respectivas funções quando executadas no ambiente da Besu, evidenciando o ganho substancial de desempenho viabilizado pelo modelo baseado em contratos inteligentes.

As operações de leitura (e.g., `getDid`, `getSchema`) apresentaram latência inferior a 5 segundos em ambos os contextos, sendo um indicativo de que o gargalo está principalmente nas transações de escrita e consenso. Durante os experimentos, observou-se que as leituras mantiveram comportamento estável mesmo sob aumento da carga transacional, sem variações significativas na vazão ou no consumo de recursos computacionais.

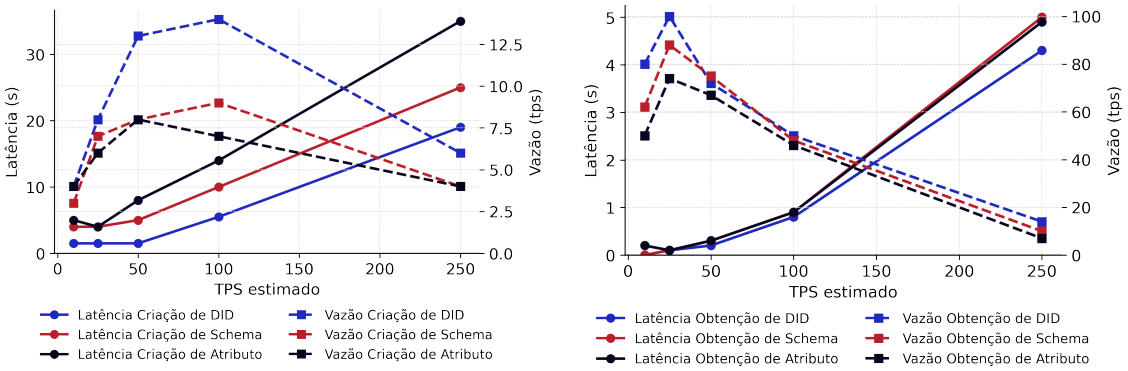

Figura 2. Comparação entre latência e vazão das operações de criação e obtenção na rede Indy. Adaptado de [Bastos et al. 2024] e [Veloso et al. 2024].

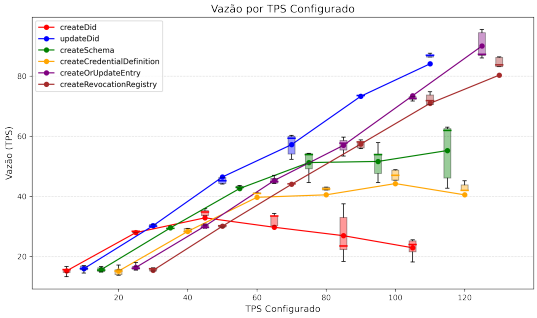

Figura 3. Comparação de vazão entre funções na rede Besu.

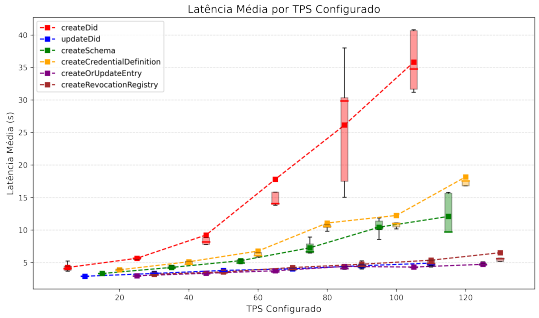

Figura 4. Comparação de latência entre funções na rede Besu.

As Figuras 5 a 8 apresentam a média de uso de CPU e memória para cada função crítica (e.g., `createDid`, `createOrUpdateEntry`, `createSchema`). Observa-se que, na rede Besu, a carga computacional é mais bem distribuída entre os nós, com variações menores de latência mesmo sob aumento gradual de carga.

Podemos observar que as funções `create Did`, `create Schema` e `create Credential Definition` são as que mais consomem recursos de CPU. Essas funções são responsáveis por operações críticas de escrita e, como era esperado, impõem maior carga computacional sobre a rede. Observando especificamente os dados de vazão, percebemos um ponto de inflexão no consumo de CPU entre 40 e 60 TPS, que coincide com o momento em que a rede atinge sua vazão máxima para essas operações. A função `createDid`, por exemplo, atinge **37,5 TPS** no cenário com 100 TPS configurados, enquanto `createCredentialDefinition` chega a **49 TPS** e `createSchema` atinge **58 TPS**. Isso indica que `createSchema` foi a operação com maior capacidade de escalabilidade horizontal entre as três.

A partir do momento em que essas funções se aproximam do pico de vazão, observa-se que o uso de CPU por nó se estabiliza próximo a **100%**, ou até mesmo ul-

trapassa essa marca virtualmente, considerando múltiplos núcleos. Esse comportamento é indicativo de saturação da capacidade de processamento, o que reforça a importância de estratégias de balanceamento ou otimização contratual em cenários de produção.

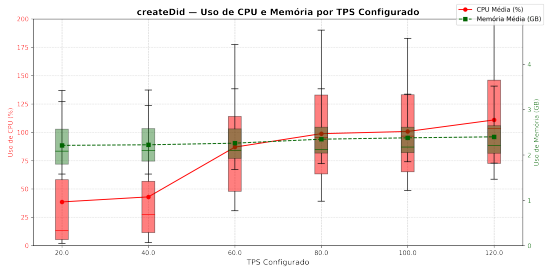

Figura 5. Média de CPU e memória — operação `createDid`.

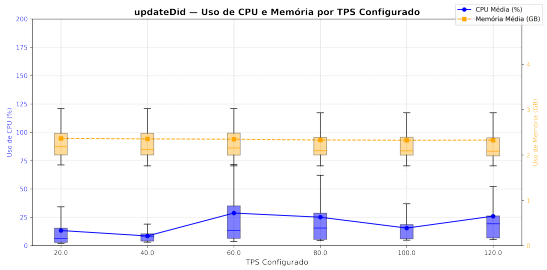

Figura 6. Média de CPU e memória — operação `updateDid`.

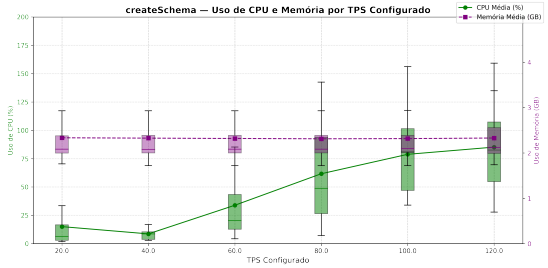

Figura 7. Média de CPU e memória — operação `createSchema`.

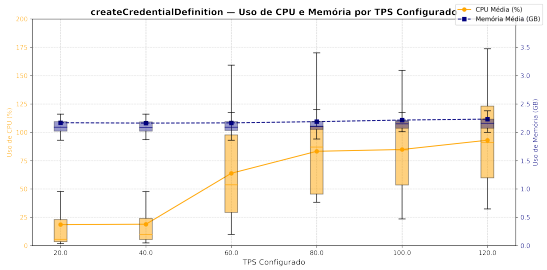

Figura 8. Média de CPU e memória — operação `createCredentialDefinition`.

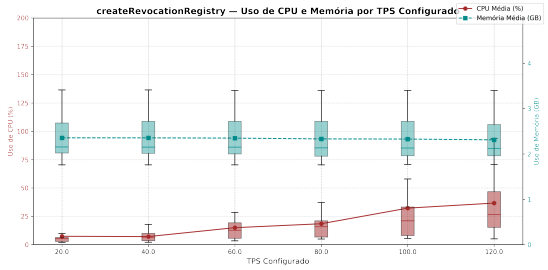

Figura 9. Média de CPU e memória — operação `createRevocationRegistry`.

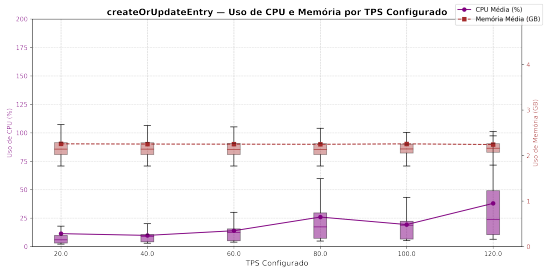

Figura 10. Média de CPU e memória — operação `createOrUpdateEntry`.

Por outro lado, as demais funções analisadas — como `updateDid`, `createOrUpdateEntry` e `createRevocationRegistry` — mantêm seu uso de CPU em níveis significativamente mais baixos, não ultrapassando 50% de uso de CPU mesmo nas maiores taxas de transações configuradas. Isso demonstra que essas funções são menos exigentes computacionalmente e podem ser processadas com maior previsibilidade.

Quanto à **memória RAM**, todas as funções apresentaram um comportamento consistente, com alocação média entre **2 e 3 GB por nó**, mesmo sob diferentes níveis de carga. Esse aspecto é bastante positivo, pois sugere que o gerenciamento interno de *heap* e *threads* do Hyperledger Besu é eficiente, permitindo previsibilidade no consumo de memória, o que é essencial para provisionamento em ambientes com múltiplos validadores.

A rede Indy, por outro lado, apresentou picos mais acentuados de latência, especialmente nas operações de criação de atributos e esquemas. Esse comportamento está relacionado à sua arquitetura monolítica e ao overhead do protocolo RBFT, que impõe maior custo computacional para alcançar consenso mesmo em cenários com baixa concorrência. Apesar disso, a memória média manteve-se estável em ambos os ambientes. No caso do Besu, no entanto, observou-se que o consumo de CPU cresce de maneira mais previsível e linear conforme a carga de transações aumenta, o que é altamente desejável em aplicações com múltiplos validadores ou em redes corporativas que demandam previsibilidade de desempenho.

É importante observar que, devido às limitações das ferramentas de avaliação disponíveis para a rede Indy, não foi possível obter métricas precisas de uso de CPU e memória. Por outro lado, a rede Besu foi monitorada com o Hyperledger Caliper, que fornece essas informações detalhadas. Essa disparidade evidencia uma vantagem metodológica na escolha do Besu também em termos de observabilidade e controle operacional.

A escalabilidade das plataformas foi testada com cargas crescentes de clientes simultâneos. A partir de 50 clientes, observou-se degradação progressiva do desempenho em ambas as redes. Porém, enquanto a Indy teve quedas de até 60% na vazão e picos de latência que quadruplicaram, a Besu manteve uma redução inferior a 30% em vazão e uma elevação de latência mais controlada, inferior a 50%. Esse resultado sugere que a arquitetura baseada em contratos inteligentes da Indy Besu é mais adequada para ambientes com maiores requisitos de escalabilidade horizontal.

Esses dados reforçam que a modularidade, paralelismo e otimizações no nível da EVM oferecidas pela rede Besu proporcionam não apenas maior escalabilidade, mas também maior previsibilidade no uso de recursos computacionais. A arquitetura desacoplada dos contratos facilita ajustes finos, reaproveitamento de componentes e aderência a padrões emergentes como `did:ethr`, o que torna o Indy Besu uma alternativa tecnicamente superior para soluções de identidade digital descentralizada em cenários com requisitos de desempenho mais exigentes.

6. Conclusão e Trabalhos Futuros

Este artigo apresentou uma análise comparativa entre duas abordagens para operações de identidade digital descentralizada (DID): a rede tradicional Hyperledger Indy, e a arquitetura baseada em contratos inteligentes na Hyperledger Besu. Os testes realizados abrangeram métricas de consumo de recursos, vazão e latência, utilizando um conjunto comum de funções-chave do ciclo de vida de identidades e credenciais verificáveis.

Os resultados demonstraram que a implementação baseada na Besu apresenta vantagens significativas em termos de desempenho e escalabilidade. As operações de escrita, que exigem consenso distribuído, foram executadas com menor latência e maior taxa de transações por segundo, enquanto que o uso de recursos como CPU e memória permaneceu estável e previsível. A modularidade proporcionada pela execução das regras de negócio em contratos Solidity também facilita a integração com ecossistemas Ethereum e a adoção de métodos DID compatíveis.

Como extensão deste trabalho, são propostos os seguintes direcionamentos:

- **Avaliação de segurança e privacidade:** Estender os testes para considerar cenários com transações maliciosas, análise de resistência à censura, e controle de

revogação em massa.
- **Cenários multi-organizacionais:** Implementar casos de uso com múltiplas entidades emissoras e verificadoras distribuídas, incluindo interoperabilidade entre diferentes redes permissionadas.
- **Análise de custo:** Estimar o custo computacional e energético das operações em cada arquitetura, considerando cenários de produção com larga escala.
- **Simulação de falhas:** Avaliar o comportamento das redes diante de falhas de nós validadores, particionamento de rede ou sobrecarga de transações.

Esses caminhos complementares poderão ampliar a compreensão sobre as arquiteturas analisadas e apoiar a tomada de decisão no desenvolvimento de soluções de identidade digital descentralizada mais robustas, interoperáveis e eficientes.

7. Agradecimentos

Os autores agradecem o apoio dado a este trabalho, pelo MCTI-Ministério da Ciência, Tecnologia e Inovação, com recursos financeiros do FUNTTEL e administrados pela FINEP, no âmbito especificamente dos projetos AERF - Ações Estratégicas para Redes Futuras, Contrato 01.22.0471.00, Referência 1508/22 e TECSEG - Desenvolvimento de tecnologias e metodologia de avaliação e investigação de segurança para redes e aplicações de governo digital, Contrato 01.21.0163.01, Referência 1196/21.

Referências

Afraz, N. and Ruffini, M. (2020). 5g network slice brokering: A distributed blockchain-based market. In *2020 European Conference on Networks and Communications (EuCNC)*, pages 23–27.

Allen, C. (2016). The path to self-sovereign identity. *Life with Alacrity Blog*.

Bastos, M., Veloso, A., Sousa, J., Evaristo, B., Abreu, D., and Abelém, A. (2024). Minindy: Automating the deployment and management of hyperledger indy networks. In *11th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO)*.

Besu, H. (2023a). Consensus protocols overview. `https://besu.hyperledger.org/how-to/configure/consensus/`.

Besu, H. (2023b). Hyperledger besu documentation. `https://besu.hyperledger.org`.

Besu, H. (2023c). Privacy and permissioning features. `https://besu.hyperledger.org/HowTo/Use-Privacy/Privacy-Overview/`.

Choi, W. and Won-Ki Hong, J. (2021). Performance evaluation of ethereum private and testnet networks using hyperledger caliper. In *22nd Asia-Pacific Network Operations and Management Symposium (APNOMS)*.

Community, H. (2023). Challenges and limitations in hyperledger indy. `https://wiki.hyperledger.org/display/indy`.

Community, H. (2024a). Indy-besu: An experimental vdr implementation for self-sovereign identity. `https://github.com/hyperledger/indy-besu`.

Community, H. (2024b). Indy besu experimental project. https://github.com/hyperledger/indy-besu.

Fan, C., Lin, C., Khazaei, H., and Musilek, P. (2022). Performance analysis of hyperledger besu in private blockchain. In *2022 IEEE International Conference on Decentralized Applications and Infrastructures (DAPPS)*, pages 64–73.

Foundation, S. (2020). Sovrin network. https://sovrin.org/.

Hang, L. and Kim, D.-H. (2019). Sla-based sharing economy service with smart contract for resource integrity in the internet of things. *Applied Sciences*, 9(17).

Hyperledger Foundation (2019). Hyperledger indy. https://www.hyperledger.org/use/hyperledger-indy.

Indy, H. (2022). Hyperledger indy documentation. https://hyperledger-indy.readthedocs.io.

Indy, H. (2023). Indy vdr specification. https://hyperledger.github.io/indy-did-method/.

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

Kshirsagar, A. and Pachghare, V. (2022). Performance evaluation of proof of scope consensus mechanisms on hyperledger. In *IEEE International Conference on Blockchain and Distributed Systems Security (ICBDS)*.

Melo, C., Gonçalves, G., Silva, A. S., and Soares, A. (2024). Performance modeling and evaluation of hyperledger fabric: An analysis based on transaction flow and endorsement policies. In *IEEE Symposium on Computers and Communications (ISCC)*.

Mostarda, L., Pinna, A., Sestili, D., and Tonelli, R. (2023). Performance analysis of a besu permissioned blockchain. In *Advanced Information Networking and Applications*, pages 279–291.

Thorstensson, J. (2018). Erc-1056: Ethereum did registry. https://eips.ethereum.org/EIPS/eip-1056. Ethereum Improvement Proposal.

Veloso, A., Sousa, J., Evaristo, B., Abreu, D., Saraiva, F., and Abelém, A. (2024). Minindy: Um framework para automatizar a implantação e o gerenciamento de redes blockchain hyperledger indy. In *Anais do VII Workshop em Blockchain: Teoria, Tecnologias e Aplicações*, pages 55–68, Porto Alegre, RS, Brasil. SBC.

W3C (2022a). Decentralized identifiers (dids) v1.0. https://www.w3.org/TR/did-core/. W3C Recommendation.

W3C (2022b). Verifiable credentials data model 1.1. https://www.w3.org/TR/vc-data-model/. W3C Recommendation.