ESCOLA POLITÉCNICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO
MESTRADO EM CIÊNCIA DA COMPUTAÇÃO

LEONARDO BARBOSA DA ROSA

STORDY: RECUPERAÇÃO E ARMAZENAMENTO EFICIENTE DE DADOS PARA BLOCKCHAINS DE BLOCOS EXPANSÍVEIS

Porto Alegre
2025

PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO GRANDE DO SUL
ESCOLA POLITÉCNICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

STORDY: RECUPERAÇÃO E ARMAZENAMENTO EFICIENTE DE DADOS PARA BLOCKCHAINS DE BLOCOS EXPANSÍVEIS

LEONARDO BARBOSA DA ROSA

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre em Ciência da Computação na Pontifícia Universidade Católica do Rio Grande do Sul.

Orientador: Prof. Dr. Avelino Francisco Zorzo
Co-Orientador: Prof. Dr. Roben Castagna Lunardi

Porto Alegre
2025

Ficha Catalográfica

R788s Rosa, Leonardo Barbosa da

Stordy : Recuperação e armazenamento eficiente de dados para blockchains de blocos expansíveis / Leonardo Barbosa da Rosa. – 2025.
68.
Dissertação (Mestrado) – Programa de Pós-Graduação em Ciência da Computação, PUCRS.

Orientador: Prof. Dr. Avelino Francisco Zorzo.
Coorientador: Prof. Dr. Roben Castagna Lunardi.

1. Blockchain. 2. Internet das coisas. 3. Armazenamento. 4. Recuperação de Dados. I. Zorzo, Avelino Francisco. II. Lunardi, Roben Castagna. III. , . IV. Título.

Elaborada pelo Sistema de Geração Automática de Ficha Catalográfica da PUCRS
com os dados fornecidos pelo(a) autor(a).
Bibliotecária responsável: Clarissa Jesinska Selbach CRB-10/2051

LEONARDO BARBOSA DA ROSA

STORDY: RECUPERAÇÃO E ARMAZENAMENTO EFICIENTE DE DADOS PARA BLOCKCHAINS DE BLOCOS EXPANSÍVEIS

Dissertação apresentada como requisito parcial para obtenção do grau de Mestre em Ciência da Computação do Programa de Pós-Graduação em Ciência da Computação, Escola Politécnica da Pontifícia Universidade Católica do Rio Grande do Sul.

Aprovado(a) em 17 de Março de 2025.

BANCA EXAMINADORA:

Prof. Dr. César Augusto Fonticielha De Rose (PPGCC/PUCRS)

Prof. Dr. Weverton Cordeiro (PPGC/UFRGS)

Prof. Dr. Roben Castagna Lunardi (IFRS- Co-Orientador)

Prof. Dr. Avelino Francisco Zorzo (PPGCC/PUCRS - Orientador)

“A simplicidade é o grau mais alto de sofisti-
cação”
(Leonardo da Vinci)

AGRADECIMENTOS

Esta dissertação foi apoiada pelo Ministério da Ciência, Tecnologia e Inovações, com recursos da Lei nº 8.248, de 23 de outubro de 1991, no âmbito do PPI-SOFTEX, coordenado pela Softex e publicado no Residência em TIC 02 - Aditivo, DOU 01245.012095/2020-56.

Gostaria de expressar meus sinceros agradecimentos à Creatus Desenvolvimento de Soluções, empresa na qual trabalho, pelo apoio contínuo ao longo desta jornada acadêmica. Agradeço especialmente ao meu sócio, João Severo, cuja confiança, orientação e parceria foram fundamentais para o meu crescimento pessoal e profissional. A Creatus não apenas permitiu que eu utilizasse sua propriedade intelectual como base para este trabalho de mestrado, mas também me proporcionou flexibilidade para conciliar minhas responsabilidades profissionais com os estudos, demonstrando um compromisso genuíno com o desenvolvimento de seus colaboradores.

Agradeço profundamente ao meu orientador, Prof. Dr. Avelino F. Zorzo, e ao meu co-orientador, Prof. Dr. Roben C. Lunardi, por todo o suporte, conhecimento e incentivo ao longo deste processo. Desde o primeiro momento no grupo CONSEG da PUCRS, ambos contribuíram não apenas com ensinamentos e materiais de apoio, mas também com uma motivação constante para que eu buscasse a excelência como pesquisador. Suas orientações foram essenciais para moldar esta pesquisa e alcançar os objetivos propostos.

Agradeço também à minha família, que sempre esteve ao meu lado, oferecendo amor, compreensão e encorajamento em todos os momentos. Em especial, dedico meus agradecimentos à minha namorada, Mariele Leiria, cujo apoio incondicional e palavras de incentivo foram fundamentais para superar os desafios enfrentados durante esta jornada. Sua presença foi uma fonte constante de força e motivação.

Por fim, agradeço a todos os colegas, amigos e professores que, direta ou indiretamente, contribuíram para a realização deste trabalho. Esta conquista não seria possível sem o apoio e a colaboração de cada um de vocês.

STORDY: RECUPERAÇÃO E ARMAZENAMENTO EFICIENTE DE DADOS PARA BLOCKCHAINS DE BLOCOS EXPANSÍVEIS

RESUMO

A tecnologia de *blockchain* emergiu como uma das inovações mais significativas das últimas duas décadas. Desde sua proposta em 2008, as aplicações de *blockchain* expandiram-se além das transações financeiras, abrangendo setores como agricultura, educação, saúde e Internet das Coisas (IoT). No entanto, essas aplicações enfrentam diversas limitações, especialmente em ambientes de IoT. Particularmente, os dispositivos IoT possuem restrições relacionadas ao consumo de memória volátil. Mesmo em *blockchains* como a SpeedyChain, desenvolvidas especificamente para esses ambientes, essas limitações não foram adequadamente abordadas. Para enfrentar esse problema, esta dissertação propõe uma estrutura de armazenamento e indexação que melhora significativamente a confiabilidade do sistema e reduz o uso de memória volátil. A estrutura proposta tem como objetivo aumentar a eficiência e a robustez no gerenciamento, armazenamento e acesso a dados, sendo projetada para lidar com os altos volumes de transações típicos de ambientes IoT. Avaliações experimentais demonstram melhorias substanciais na escalabilidade do sistema e no desempenho geral da *blockchain*. Esses resultados destacam o potencial da abordagem para aprimorar significativamente a funcionalidade e a eficiência de sistemas *blockchain* em aplicações comerciais de IoT. Ao otimizar a SpeedyChain, este estudo contribui para superar limitações existentes e estabelece o caminho para implementações mais escaláveis e confiáveis da SpeedyChain.

Palavras-Chave: Blockchains, Internet das Coisas, Armazenamento, Recuperação de Dados.

STORDY: EFFICIENT DATA RETRIEVAL AND STORAGE FOR APPENDABLE-BLOCK BLOCKCHAINS

ABSTRACT

Blockchain technology has emerged as one of the most significant innovations of the last two decades. Since its proposal in 2008, blockchain applications have expanded beyond financial transactions to sectors such as agriculture, education, healthcare, and the Internet of Things (IoT). However, these applications face several limitations, particularly in IoT environments. This paper addresses some of these challenges by presenting improvements to SpeedyChain, an appendable-block blockchain. To enhance SpeedyChain, this study proposes a storage and indexing structure, which significantly improves the system's reliability and reduces the volatile memory used. The proposed structure aims to enhance efficiency and robustness in managing, saving, and accessing data, being designed to handle the high transaction volumes typical in IoT environments. Experimental evaluations demonstrate substantial improvements in the system scalability and overall blockchain performance. These results highlight the potential of our approach to significantly enhance the functionality and efficiency of blockchain systems in commercial IoT applications. By optimizing SpeedyChain, this study contributes to overcoming existing limitations and sets the path for more scalable and reliable implementations of SpeedyChain.

Keywords: Blockchains, Internet of Things, Storage, Data retrieval.

LISTA DE FIGURAS

Figura 2.1 – Representação simplificada de uma *blockchain* (elaboração do autor, adaptado de Tschorsch e Scheuermann (2016)). 19
Figura 2.2 – Camadas da *blockchain* (Zorzo et al., 2018) 21
Figura 2.3 – Prova de Merkle (elaboração do autor, adaptado de Liu et al. (2021) 22
Figura 2.4 – Principais nodos *IoT* (Lunardi et al., 2019) 26
Figura 2.5 – Exemplo de estrutura de uma *appendable-block blockchain* (Lunardi et al., 2020) 27
Figura 2.6 – Estrutura da patente Silveira et al. (2024) 28
Figura 4.1 – Estrutura de um *gateway* utilizando *blockchain* com blocos extensíveis. 38
Figura 4.2 – Estrutura de serialização em arquivo. 40
Figura 4.3 – Estrutura de camadas do Stordy. 42
Figura 5.1 – Uso da memória volátil da Speedychain com e sem Stordy dependendo do número de transações já anexadas. 52
Figura 5.2 – Tempo para anexar uma nova transação considerando o número de transações já anexadas com Stordy. 52
Figura 5.3 – Esquemático dos dispositivos utilizados no teste. 53
Figura 5.4 – Consumo de memória *RAM* em dispositivo IoT. 54
Figura 5.5 – Consumo de memória não volátil em dispositivo *IoT*. 55
Figura 5.6 – Diagrama de sequência do processo de inserção de uma transação à blockchain. 56
Figura 5.7 – Tempo para adição de uma nova transação à cadeia. 57

LISTA DE ALGORITMOS

4.1 Adding a Block 43
4.2 Obtaining Block by PublicKey 43
4.3 Obtaining the Number of Transactions in a Block 44
4.4 Obtaining Last Block 45
4.5 Adding Transaction to Block 46
4.6 Obtaining Last Transaction of Block 47
4.7 Obtaining Transaction by Hash 48
4.8 Validating the existence of a transaction 48

LISTA DE SIGLAS

POW – *Proof of Work*

POS – *Proof of Stake*

IOT – *Internet of Things*

P2P – *Peer to Peer*

DPOS – *Delegated Proof of Stake*

PBFT – *Practical Byzantine Fault Tolerance*

DAG – *Directed Acyclic Graph*

RFC – *Request for Comments*

JSON – *JavaScript Object Notation*

XML – *Extensible Markup Language*

HP – Hewlett-Packard

RLP – *Recursive-length prefix*

IIOT – *Industrial Internet of Things*

AB-M TREE – *Adaptive Balanced Merkle Tree*

DRL – *Deep Reinforcement Learning*

A2C – *Advantage Actor-Critic*

PPO – *Proximal Policy Optimization*

MQTT – *Message Queuing Telemetry Transport*

UUID – *Universally unique identifier*

SO – Sistema Operacional

RAM – *Random access memory*

API – *Application Programming Interfaces*

IPFS – InterPlanetary File System

DFS – Distributed File System

SUMÁRIO

1 INTRODUÇÃO 13
1.1 OBJETIVOS 15
1.2 METODOLOGIA E PROPOSTA 15
1.2.1 METODOLOGIA 16
1.2.2 PROPOSTA 17
1.3 CONTRIBUIÇÕES DA PESQUISA 17
1.4 ORGANIZAÇÃO DO DOCUMENTO 18

2 REFERENCIAL TEÓRICO 19
2.1 *BLOCKCHAIN* 19
2.1.1 CAMADAS 20
2.1.2 ESTRUTURA DE DADOS 21
2.1.3 ESTRUTURA DE INDEXAÇÃO 22
2.2 SERIALIZAÇÃO 23
2.3 APPENDABLE-BLOCK BLOCKCHAIN 25

3 TRABALHOS RELACIONADOS 30
3.1 SE-CHAIN: *A SCALABLE STORAGE AND EFFICIENT RETRIEVAL MODEL FOR BLOCKCHAIN* 30
3.2 BW-TREE: A B-TREE FOR NEW HARDWARE PLATFORMS 31
3.3 *PERFORMANCE OPTIMIZATION FOR BLOCKCHAIN-ENABLED INDUSTRIAL INTERNET OF THINGS (IIOT) SYSTEMS: A DEEP REINFORCEMENT LEARNING APPROACH* 32
3.4 EDGECHAIN: *AN EDGE-IOT FRAMEWORK AND PROTOTYPE BASED ON BLOCKCHAIN AND SMART CONTRACTS* 33
3.5 *ADAPTIVE STORAGE OPTIMIZATION SCHEME FOR BLOCKCHAIN-IIOT APPLICATIONS USING DEEP REINFORCEMENT LEARNING* 34
3.6 COMPARAÇÃO ENTRE TRABALHOS RELACIONADOS 35

4 SOLUÇÃO: STORDY 37
4.1 SERIALIZAÇÃO 39
4.2 INDEXAÇÃO 40
4.3 SERVIÇOS 41
4.4 ADIÇÃO DE UM NOVO BLOCO 42

4.5 BUSCAR UM BLOCO POR SUA CHAVE PÚBLICA 43
4.6 OBTENÇÃO DA QUANTIDADE DE TRANSAÇÕES EM UM DETERMINADO BLOCO . 44
4.7 OBTENÇÃO DO BLOCO MAIS RECENTE DA CADEIA 45
4.8 ADIÇÃO DE TRANSAÇÕES A UM BLOCO 45
4.9 OBTENDO A ÚLTIMA TRANSAÇÃO ANEXADA A UM DETERMINADO BLOCO 47
4.10 OBTENDO UMA TRANSAÇÃO PELO SEU HASH 47
4.11 VALIDAÇÃO DA EXISTÊNCIA DE UMA TRANSAÇÃO EM UM BLOCO 47
4.12 CONSIDERAÇÕES FINAIS 48

5 AVALIAÇÃO EXPERIMENTAL 50
5.1 AVALIAÇÃO I: APLICABILIDADE DO MÓDULO DE ARMAZENAMENTO 50
5.1.1 RESULTADOS 50
5.2 AVALIAÇÃO II: COMPORTAMENTO DO STORDY EM DISPOSITIVOS IOT 51
5.2.1 RESULTADOS 51

6 DISCUSSÕES 58
6.1 RELEVÂNCIA DO CONSUMO DE MEMÓRIA VOLÁTIL 58
6.2 EFICIÊNCIA TEMPORAL NA INSERÇÃO DE TRANSAÇÕES 59
6.3 IMPACTO NO CONSUMO DE RECURSOS EM IOT 60
6.4 PERSPECTIVAS PARA ARMAZENAMENTO HÍBRIDO E OFF-CHAIN 61

7 CONCLUSÃO 63

1. INTRODUÇÃO

A tecnologia *blockchain* tem sido amplamente reconhecida como uma das inovações mais disruptivas do século XXI, revolucionando não apenas o setor financeiro, mas também a maneira como transações e interações digitais são realizadas de forma segura, transparente e descentralizada (Zhao et al., 2016; Pilkington, 2016). Desde a sua criação, a *blockchain* trouxe uma nova perspectiva sobre como garantir a confiança em ambientes digitais, eliminando a necessidade de intermediários e permitindo que os participantes em uma rede distribuída validem transações por meio de consenso (Crosby et al., 2016). A origem da *blockchain* remonta ao trabalho seminal de Satoshi Nakamoto, que, em 2008, introduziu o *Bitcoin*, uma moeda digital baseada em um modelo *peer-to-peer*, inaugurando o uso de tecnologias de registro distribuído para garantir a integridade e a transparência das transações (Nakamoto, 2008).

Embora a aplicação inicial da *blockchain* tenha sido predominantemente focada no setor financeiro, particularmente em criptomoedas, a evolução dessa tecnologia rapidamente demonstrou seu potencial em diversas outras áreas. Setores como *saúde*, onde a *blockchain* pode garantir a integridade e privacidade de registros médicos (Yli-Huumo et al., 2016), *educação*, com a emissão de certificados digitais imutáveis (Albeanu, 2017), *agricultura*, onde o rastreamento de produtos pode ser verificado de ponta a ponta (Bermeo-Almeida et al., 2018), e *Internet of Things (IoT)*, onde a *blockchain* proporciona segurança e integridade em ambientes altamente conectados e dinâmicos (Pan et al., 2018), têm se beneficiado dessa inovação. A capacidade da *blockchain* de fornecer confiança, sem a necessidade de uma autoridade central, torna-a uma ferramenta valiosa em cenários que requerem segurança e transparência. Como exemplo disso, a *blockchain* pode ser empregada para registrar de forma imutável os eventos do ciclo de vida dos dispositivos, permitindo a análise preditiva de falhas a partir dos *logs* armazenados. Essa abordagem possibilita intervenções preventivas e redução de perdas financeiras, ao fornecer informações confiáveis para a gestão e manutenção dos ativos (Silveira et al., 2024).

A segurança e a confiabilidade da *blockchain* são garantidas por meio de mecanismos criptográficos, que asseguram que as transações sejam registradas de maneira imutável e verificável por uma rede de nós distribuídos (Narayanan et al., 2016). O conceito de *mecanismo de consenso*, como o *Proof of Work (PoW)*, introduzido pelo *Bitcoin*, é fundamental para garantir que todas as partes envolvidas concordem sobre o estado atual da *blockchain* sem a necessidade de um intermediário confiável Bonneau et al. (2015). No entanto, o *PoW* é criticado por seu alto consumo de energia e sua limitada escalabilidade, o que levou ao surgimento de novas abordagens, como o *Proof of Stake (PoS)*, que promete reduzir os impactos ambientais e aumentar a eficiência da validação das transações (King e Nadal, 2012; Buterin, 2017).

Com o avanço contínuo dessas tecnologias, a *blockchain* consolidou seu papel como um componente central em diversas áreas. No contexto das aplicações de *blockchain* para IoT, surgem desafios específicos devido à natureza distribuída e ao grande volume de dados gerados por dispositivos conectados. A literatura apresenta diversas inovações em arquiteturas (Dorri et al., 2017), estruturas de dados (Lunardi et al., 2018) e algoritmos de consenso adaptados a ambientes de IoT (Zheng et al., 2017; Seliem e Elgazzar, 2019). Contudo, essas aplicações enfrentam limitações conhecidas, como a necessidade de maior escalabilidade e eficiência no armazenamento e recuperação de dados, além da segurança em dispositivos com recursos limitados (Dorri et al., 2016).

Para enfrentar esses desafios, Lunardi et al. (2018, 2019); Michelin et al. (2018) propuseram a *Speedychain*, uma *blockchain* com blocos extensíveis que permite a criação de transações após a adição de um bloco. Essa característica inovadora possibilita a inserção de transações em paralelo, otimizando tanto o desempenho quanto a latência da *blockchain* em ambientes IoT. Projetada para atender às demandas de sistemas que exigem alta taxa de transações, como *Smart Cities* e redes de sensores *IoT*, a *Speedychain* destaca-se por sua escalabilidade e eficiência, fatores cruciais nesses contextos (Lunardi et al., 2023). Embora os resultados iniciais tenham demonstrado melhorias significativas na latência e no desempenho geral (de Arruda et al., 2020), a proposta inicial da *Speedychain* foi concebida como uma prova de conceito, limitando-se a explorar a viabilidade de desassociar transações do conceito tradicional de bloco. Entretanto, aprimoramentos adicionais são indispensáveis para tornar a solução aplicável em larga escala em ambientes comerciais.

Entre os principais desafios, destaca-se a necessidade de otimizar o armazenamento e o acesso aos dados da *blockchain*. Nas versões anteriores da *Speedychain*, as informações eram armazenadas exclusivamente em memória volátil, resultando em perda de dados durante reinicializações inesperadas e limitando o processamento ao volume de memória disponível. Conforme apontado por Silveira et al. (2024), foram investigadas técnicas de armazenamento persistente utilizando estruturas de dados não voláteis, permitindo a preservação dos dados mesmo após interrupções. Essa abordagem, inicialmente explorada no contexto de uma patente da HP sobre o gerenciamento do ciclo de vida de dispositivos, trouxe maior confiabilidade e continuidade às operações da *blockchain*.

Apesar das melhorias, a abordagem inicial apresentou limitações. O armazenamento persistente exigia que o *backup* fosse explicitamente acionado, gerando dois problemas principais: (i) em situações de reinicializações inesperadas, havia risco de perda de dados devido à ausência de um mecanismo automatizado de *backup*; e (ii) as aplicações continuavam a depender fortemente de memória volátil para a operação, restringindo a eficiência em dispositivos com recursos limitados . Essas limitações destacam a necessidade de soluções mais robustas, capazes de automatizar o gerenciamento de armazenamento e otimizar o uso de recursos computacionais.

Adicionalmente, a dependência atual de memória volátil para o gerenciamento da cadeia faz com que o consumo de *RAM* aumente proporcionalmente ao crescimento da *blockchain*, criando um gargalo significativo em dispositivos IoT. Equipamentos típicos desse ambiente, frequentemente limitados em *RAM* e capacidade computacional, não conseguem suportar adequadamente as demandas de escalabilidade da *Speedychain*, dificultando sua integração eficiente e comprometendo a adoção plena dessa tecnologia em cenários reais. Superar essa limitação é essencial para viabilizar o uso da *blockchain* com blocos extensíveis em ambientes IoT, ampliando seu potencial de aplicação e impacto.

1.1 Objetivos

Diante do contexto apresentado, o presente trabalho tem como objetivo principal propor uma solução que aprimore o desempenho e a confiabilidade da *Speedychain*. Como objetivos específicos, definem-se: (i) propor uma estrutura de armazenamento em memória não volátil que elimine a dependência de *RAM* da aplicação; (ii) propor uma estrutura de indexação que viabilize o acesso rápido às informações, mesmo não estando elas em memória de aplicação; e (iii) definir a forma como essas informações serão consumidas, a fim de possibilitar o uso dessa *blockchain* em ambientes *IoT* com recursos limitados.

1.2 Metodologia e Proposta

Para a implementação da proposta deste trabalho, será utilizada a *SpeedyChain*, uma *blockchain* baseada no conceito de *appendable-blocks blockchain*, que possibilita a adição de novas transações a blocos já existentes (Lunardi et al., 2018). Essa característica favorece a flexibilidade e a eficiência do sistema de armazenamento de dados em ambientes de *IoT*. Estudos anteriores já demonstraram o potencial da *SpeedyChain* para o processamento paralelo de transações (Lunardi et al., 2019; Michelin et al., 2018; de Arruda et al., 2020), tornando-a adequada para cenários em que escalabilidade e desempenho são essenciais.

O foco deste trabalho consiste em aprimorar a *estrutura de armazenamento* da *SpeedyChain*, substituindo o uso de memória volátil por uma abordagem de *armazenamento persistente* em memória não volátil. Assim, busca-se garantir a integridade e a continuidade dos dados, mesmo em situações de falhas ou reinicializações inesperadas dos dispositivos. Além disso, pretende-se desenvolver uma *estrutura de indexação eficiente* que permita acesso rápido às informações armazenadas em disco, possibilitando a

recuperação de dados em sistemas com grandes volumes de transações sem comprometer o desempenho.

Para validar as soluções propostas, serão utilizados dados simulados obtidos em um ambiente operacional, empregando métricas de desempenho e cenários de alta demanda transacional. Os testes avaliarão a eficácia das novas estruturas de armazenamento e indexação, bem como sua influência no desempenho da *blockchain*. Assim, espera-se que as modificações implementadas aprimorem a eficiência e a escalabilidade da *SpeedyChain*, garantindo sua funcionalidade mesmo em ambientes *IoT* com recursos limitados.

1.2.1 Metodologia

Este trabalho foi realizado seguindo cinco passos fundamentais:

(i) Análise e definição do problema: inicialmente, será conduzida uma análise cuidadosa das limitações atuais da *SpeedyChain*, com foco no uso de memória volátil. Essa etapa visa identificar os principais gargalos que comprometem a escalabilidade e a viabilidade da solução em aplicações comerciais de larga escala.

(ii) Desenvolvimento da solução de armazenamento: propõe-se a criação de uma arquitetura de armazenamento baseada em memória não volátil, garantindo a persistência dos dados da *blockchain*. Com essa abordagem, espera-se que as informações permaneçam acessíveis mesmo após reinicializações ou falhas, eliminando a dependência de *RAM* e aumentando a robustez do sistema.

(iii) Desenvolvimento da estrutura de indexação: será projetada uma estrutura de indexação adaptada à nova arquitetura de armazenamento, permitindo acesso rápido e eficiente às informações armazenadas em disco. Para isso, técnicas como árvores B+ ou estruturas de índices otimizadas serão avaliadas, buscando garantir escalabilidade e reduzir a latência na recuperação de dados.

(iv) Testes com dados simulados: a solução será validada por meio de testes realizados em um ambiente controlado, utilizando dados gerados especificamente para simular cenários de alta demanda de transações. Esses testes terão como objetivo avaliar o desempenho da nova abordagem, considerando métricas como tempo de resposta, consumo de recursos e eficiência na recuperação de informações.

(v) Análise de desempenho e viabilidade: por fim, os resultados obtidos serão analisados e comparados com a versão original da *SpeedyChain*, destacando os ganhos em termos de desempenho, escalabilidade e confiabilidade. A viabilidade da solução

será discutida em contextos comerciais típicos de *IoT* e ambientes industriais, onde a alta demanda e a confiabilidade do sistema são fatores críticos.

1.2.2 Proposta

A presente proposta visa aprimorar a *SpeedyChain* de modo a torná-la mais robusta e eficiente em aplicações de *IoT*, contribuindo para o avanço do estado da arte em *blockchains* destinadas a ambientes com recursos limitados. Nesse sentido, serão desenvolvidas e avaliadas abordagens para garantir: (i) Armazenamento Persistente: explorar métodos para eliminar a dependência de memória volátil, assegurando a integridade e a continuidade dos dados diante de falhas ou reinicializações. (ii) Indexação Otimizada: conceber uma estrutura de indexação que possibilite acesso ágil, escalável e eficiente às informações armazenadas em memória não volátil, mesmo em cenários de alta demanda.

A intenção é que tais melhorias posicionem a *SpeedyChain* como uma opção viável para aplicações comerciais e industriais de grande escala, refletindo em maior escalabilidade, confiabilidade e desempenho. A proposta norteia as atividades de desenvolvimento, implementação e avaliação, delineando a trajetória a ser seguida no decorrer desta pesquisa.

1.3 Contribuições da Pesquisa

A partir da implementação e análise empírica da proposta, este trabalho apresenta as seguintes contribuições para o domínio de *blockchains* em cenários *IoT*:

(i) Armazenamento Não Volátil Implementado: concretização de um mecanismo de armazenamento persistente na *SpeedyChain*, demonstrando empiricamente sua capacidade de manter dados intactos mesmo após falhas, bem como sua adequação a ambientes de recursos limitados.

(ii) Estrutura de Indexação Otimizada Validada: desenvolvimento e validação de uma estratégia de indexação que, na prática, resultou em acesso rápido e eficiente às informações, reduzindo a latência e assegurando escalabilidade em cenários com alta demanda transacional.

(iii) Avaliação Empírica em Cenários *IoT*: condução de testes sistemáticos e análise de desempenho com dados simulados, permitindo a comprovação da eficácia da solução e fornecendo evidências robustas sobre a viabilidade da *SpeedyChain* em aplicações comerciais e industriais.

(iv) Avanço do Estado da Arte: ao demonstrar melhorias práticas e oferecer diretrizes claras para a integração de armazenamento não volátil e indexação eficiente, este trabalho contribui diretamente para o estado da arte, servindo de referência para pesquisas futuras e aplicações que demandem *blockchains* resilientes e escaláveis.

1.4 Organização do Documento

Este trabalho está estruturado em capítulos que, gradualmente, conduzem o panorama conceitual ao desenvolvimento e validação da solução proposta. O Capítulo 2 apresenta o referencial teórico, descrevendo os fundamentos da *blockchain* e suas camadas, bem como conceitos relacionados a estruturas de armazenamento, indexação e serialização de dados, aprofundando-se no funcionamento da *SpeedyChain*, analisando seu contexto, sua arquitetura atual e as limitações que motivam esta pesquisa.

No Capítulo 3, são discutidos trabalhos relacionados, contemplando diferentes abordagens no emprego de *blockchain* em cenários de recursos limitados, além de soluções de armazenamento não volátil e indexação eficientes. A partir desse conjunto de conhecimentos prévios, o Capítulo 4 introduz a solução proposta, descrevendo detalhadamente o seu projeto, a arquitetura elaborada e as adaptações necessárias à *SpeedyChain* para alcançar os objetivos do estudo. Logo após, o Capítulo 5 apresenta o delineamento experimental, os cenários de teste, as métricas de desempenho consideradas e a análise dos resultados obtidos, permitindo avaliar o impacto efetivo das melhorias propostas. Por fim, o Capítulo 7 encerra o trabalho ao sintetizar as contribuições alcançadas, refletir sobre as limitações encontradas e propor potenciais caminhos para pesquisas futuras, consolidando assim o conhecimento gerado ao longo desta dissertação.

2. REFERENCIAL TEÓRICO

Neste capítulo são apresentados os referenciais teóricos empregados nesta pesquisa. Na Seção 2.1 discute-se o conceito de *blockchain* e suas propriedades mais relevantes para este trabalho, como suas camadas, estruturas de dados e indexação. Enquanto na Seção 2.2 discute-se o conceito de serialização e suas aplicações. Por fim, na Seção 2.3 são apresentados os detalhes da *blockchain* de blocos extensíveis utilizada como objeto de estudo, destacando suas características e relevância para os experimentos realizados neste trabalho.

2.1 *Blockchain*

O conceito central por trás do uso da tecnologia *blockchain* é garantir um armazenamento altamente seguro de informações, seja para transações envolvendo criptomoedas ou para dados de diferentes sistemas. O objetivo principal é assegurar a imutabilidade desses dados de maneira descentralizada. Nesse modelo, as informações são armazenadas em blocos interligados, formando uma sequência linear, o que explica o nome "*blockchain*". Cada novo bloco possui uma ligação com o bloco anterior, feita por meio de um "ponteiro", que cria uma sequência encadeada (Figura 2.1). Essa conexão é realizada através do cálculo da função hash do cabeçalho do bloco que o antecede (Gupta et al., 2021). A estrutura de uma *blockchain* incorpora várias características, como o armazenamento distribuído, a comunicação *peer-to-peer* (P2P), mecanismos de consenso, criptografia, contratos inteligentes (smart contracts), entre outras tecnologias. Sua principal função é atuar como uma base para o desenvolvimento de aplicações descentralizadas (Tschorsch e Scheuermann, 2016).

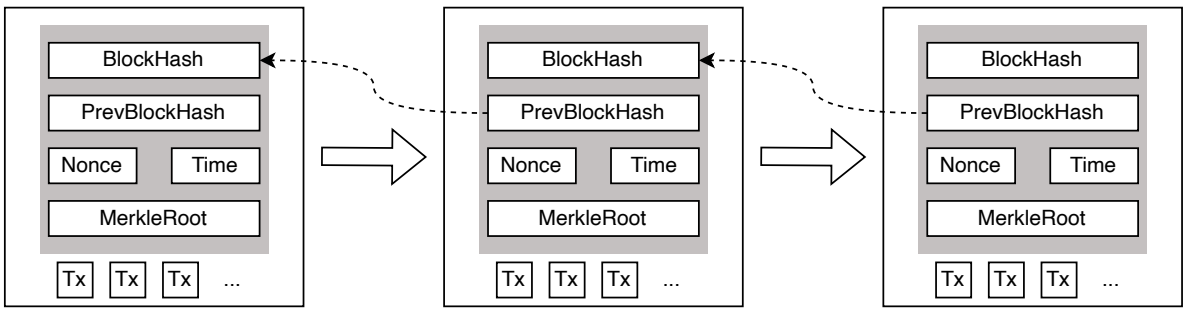

Figura 2.1 – Representação simplificada de uma *blockchain* (elaboração do autor, adaptado de Tschorsch e Scheuermann (2016)).

2.1.1 Camadas

Diversos estudos sobre *blockchain* frequentemente abordam a tecnologia com base em uma organização estruturada em camadas, o que facilita a compreensão de seus conceitos fundamentais e de suas funcionalidades (de Arruda et al., 2020; Zorzo et al., 2018; Rifat Hossain et al., 2024). A divisão em camadas é uma prática comum para descrever a *blockchain* de forma mais clara e acessível, oferecendo uma visão modular da sua complexidade. Neste trabalho, a *blockchain* é descrita com quatro camadas principais: **Comunicação**, **Consenso**, **Dados** e **Aplicação** (Figura 2.2). Originalmente concebida para aplicações no contexto da *Internet of Things* (IoT), essa organização pode ser adaptada para diferentes cenários e aplicações, demonstrando a flexibilidade da arquitetura *blockchains*.

A Camada de Comunicação é responsável pelos protocolos que regem a transmissão de informações, tanto no interior da rede *blockchains* quanto em suas interações externas. Isso inclui a forma como os nós da rede trocam informações e a definição das rotas de comunicação entre os blocos, além de determinar a arquitetura da rede, como centralizada ou descentralizada, que será usada para maximizar a eficiência na troca de dados. Esta camada garante que a troca de informações seja realizada de maneira eficaz, permitindo a interconectividade fluida dos componentes da rede.

A Camada de Consenso desempenha um papel fundamental ao implementar o mecanismo responsável por validar e autorizar a inclusão de novos blocos na cadeia. Vários mecanismos de consenso foram propostos ao longo do tempo, cada um com suas próprias características e aplicações. Uma análise detalhada de algoritmos de consenso em *blockchains*, como em Ferdous et al. (2020) destaca o *Proof of Work (PoW)*, amplamente utilizado na Bitcoin, e o *Proof of Stake (PoS)*, que se foca na redução do consumo de energia. Outras alternativas, como o *Delegated Proof of Stake (DPoS)* e mecanismos baseados em Tolerância a Falhas Bizantinas (como o PBFT), também são exploradas. Esses algoritmos são comparados em termos de segurança, escalabilidade, eficiência energética e adequação a diferentes contextos, sendo que a escolha do algoritmo ideal dependerá das necessidades específicas de cada aplicação *blockchain*.

A Camada de Dados é responsável pela definição da estrutura dos blocos e pela organização interna de como os dados são armazenados. Nessa camada, técnicas de criptografia, como o uso de funções de hash, são aplicadas para garantir a segurança e a integridade das informações armazenadas. Entre as principais formas de organização de dados utilizadas nessa camada estão as *Merkle Tree* (Nakamoto, 2008), listas encadeadas (Nakamoto, 2008), *Directed Acyclic Graph* (DAG) (Ji et al., 2018), e modelos de dados anexados diretamente dentro dos blocos (Lunardi et al., 2018).

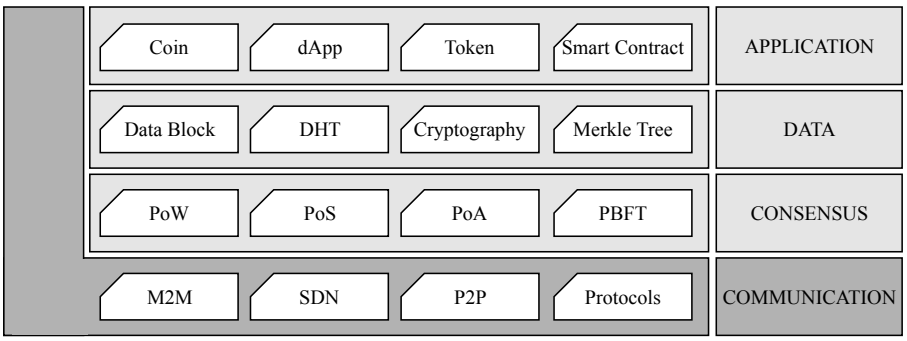

Figura 2.2 – Camadas da *blockchain* (Zorzo et al., 2018)

Por outro lado, a Camada de Aplicação é responsável pela gestão de diferentes funções e aplicações que operam sobre a *blockchain*, como as moedas digitais, *tokens*, ou qualquer outro tipo de dado. Um elemento essencial dessa camada, amplamente adotado nas *blockchains* modernas, são os *smart contracts*. Esses contratos são algoritmos que automatizam a execução de ações baseadas em condições predefinidas, eliminando a necessidade de uma autoridade central para o gerenciamento de processos dentro da rede (Zorzo et al., 2018).

2.1.2 Estrutura de Dados

Cada *blockchain*, em função da sua proposta de valor, pode empregar diferentes estruturas de dados, desempenhando um papel crucial na camada de dados. Essa camada organiza e gerencia informações essenciais, como transações, blocos e estados, garantindo integridade, acessibilidade e eficiência. Por exemplo, na Bitcoin, o design da camada de dados é otimizado para registrar transações financeiras de forma descentralizada e imutável. Já na Ethereum, essa funcionalidade é ampliada para incluir a execução de contratos inteligentes, o que exige maior complexidade na organização dos dados.

No Bitcoin (Nakamoto, 2008), os blocos contêm transações validadas pela rede, cuja integridade é garantida por uma *Merkle Tree*. A *Merkle Tree* organiza os *hashes* das transações de modo que sua raiz, chamada *Merkle Root*, seja armazenada no cabeçalho do bloco, funcionando como um resumo criptográfico de todas as transações, conforme Figura 2.3. Essa estrutura não armazena diretamente as transações, mas sim os *hashes* gerados a partir delas. Isso permite verificar a integridade de qualquer transação individual sem acessar todas as outras. Caso uma única transação seja alterada, a *Merkle Root* resultante será diferente, invalidando o bloco.

Embora a *Merkle Tree* seja usada para validar as transações, o encadeamento dos blocos é realizado de forma independente, com o *hash* criptográfico do bloco anterior sendo armazenado no cabeçalho de cada novo bloco. Esse mecanismo assegura a

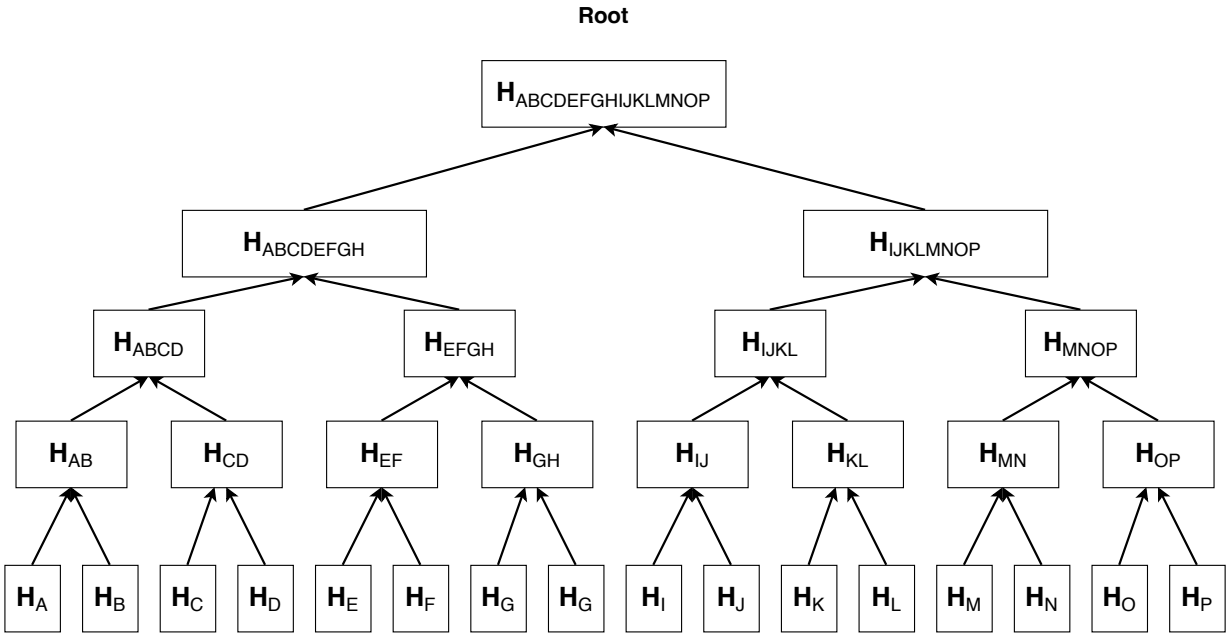

Figura 2.3 – Prova de Merkle (elaboração do autor, adaptado de Liu et al. (2021)

imutabilidade da cadeia, pois qualquer alteração em um bloco quebraria a sequência dos *hashes*, tornando evidente a inconsistência (Kuznetsov et al., 2024).

Por outro lado, a Ethereum adota estruturas de dados mais avançadas para suportar sua funcionalidade de contratos inteligentes e manter o estado global. Além de utilizar uma *Merkle Tree* para validar as transações, a Ethereum emprega uma *Merkle Patricia Tree* para gerenciar o Estado Global. Este estado contém informações sobre contas, saldos, códigos de contratos inteligentes e o armazenamento interno desses contratos (Kuznetsov et al., 2024).

A *Merkle Patricia Tree* combina as propriedades da *Merkle Tree* com as de uma *trie* (ou árvore de prefixo), otimizando a consulta, inserção e atualização dos dados armazenados. Essa estrutura é especialmente útil para a execução de contratos inteligentes, permitindo operações descentralizadas e complexas de forma eficiente e segura. A flexibilidade da Ethereum, ao lidar com esses requisitos computacionais avançados, destaca a diferença fundamental entre essas duas *blockchains*: enquanto o Bitcoin é projetado para transações financeiras descentralizadas, a Ethereum é voltada para ser uma plataforma para aplicações descentralizadas (*dApps*).

2.1.3 Estrutura de Indexação

Sistemas *blockchain* dependem amplamente de estruturas de árvores especializadas, como *Merkle Trees* e *Patricia Trees*, para indexar, armazenar e validar dados de forma eficiente. *Merkle Trees* são fundamentais para garantir a integridade dos dados ao

aplicar resumos criptográficos recursivos, onde cada nó contém um resumo criptográfico derivado de seus nós filhos. Essa estrutura permite uma verificação rápida e confiável dos dados, já que qualquer alteração em um nó se propaga para cima, modificando o resumo criptográfico da raiz. Essas árvores são particularmente úteis em sistemas distribuídos, permitindo que as partes validem a consistência dos dados sem a necessidade de transferir todo o conjunto de dados.

Na Ethereum, a *Merkle Patricia Tree* combina funcionalidades de *Merkle Trees* e *Patricia Trees*, melhorando a eficiência de armazenamento e recuperação de dados. A estrutura usa três tipos de nós—ramo, extensão e folha—para representar as chaves de maneira compacta com caminhos e sufixos compartilhados. Os dados nessa árvore são resumidos criptograficamente e transformados em *arrays* de *bytes*, utilizando mecanismos como Codificação de Prefixo Hexadecimal e Prefixo de Comprimento Recursivo (*RLP*). Esses nós codificados são armazenados em bancos de dados de chave-valor, garantindo a imutabilidade dos dados e permitindo a localização eficiente de blocos e a validação de transações (Akcora et al., 2022).

Para resolver os desafios de escalabilidade, foram propostas estruturas de árvores adaptáveis, como a *Adaptive Merkle Tree* e a *Verkle Tree*. A *Adaptive Merkle Tree* prioriza dados mais frequentemente acessados, posicionando-os mais perto da raiz da árvore, o que reduz o tempo de verificação e o esforço computacional necessário para as buscas. Essa abordagem aplica os princípios de codificação de Shannon-Fano e Huffman para minimizar o comprimento médio dos caminhos, melhorando a eficiência em cenários de alta demanda. Já as *Verkle Trees* inovam ao substituir resumos criptográficos tradicionais por compromissos polinomiais, reduzindo significativamente o custo de armazenamento e a complexidade da validação (Kuznetsov et al., 2024).

Esses avanços nas estruturas de árvore desempenham um papel fundamental na escalabilidade e eficiência dos sistemas *blockchain*. Ao otimizar como os dados são indexados e validados, essas técnicas permitem o suporte a redes maiores e volumes mais altos de transações, mantendo a segurança e a confiabilidade. À medida que as aplicações de *blockchain* se expandem para além das criptomoedas, essas inovações são essenciais para garantir uma infraestrutura robusta e escalável em diversas áreas, como finanças, gestão da cadeia de suprimentos e IoT (Kuznetsov et al., 2024).

2.2 Serialização

Atualmente, existem diversos tipos de serialização disponíveis para trafegar dados pela rede, desempenhando um papel fundamental na camada de comunicação das *blockchains*. Essa camada é responsável por gerenciar a transmissão eficiente e confiável de informações entre os nós da rede, e a escolha do formato de serialização afeta dire-

tamente o desempenho e a eficiência dessa troca de dados. As opções de serialização variam em termos de eficiência, facilidade de leitura, desempenho e capacidade de carga útil. Entre as mais conhecidas estão o XML, o JSON e o Protobuf, desenvolvido pela Google, cada uma delas com características que as tornam adequadas para diferentes cenários no desenvolvimento de software (Bray et al., 1997; Currier, 2022).

O JSON é amplamente utilizado devido à sua simplicidade e integração nativa com linguagens como JavaScript. Ele organiza dados em pares de chave-valor, permitindo legibilidade e fácil manipulação. Apesar disso, possui limitações em dispositivos com recursos restritos, como os utilizados em IoT. Sua estrutura textual, que inclui redundância em metadados como chaves, aumenta o tamanho dos dados trafegados e pode limitar seu desempenho em cenários com restrições de largura de banda ou processamento.

O XML, por sua vez, é um formato mais antigo que organiza dados por meio de tags. Ele é robusto e extensível, mas extremamente verboso, como evidenciado nos experimentos realizados por Sumaray e Makki (2012), onde o XML apresentou os maiores tempos de serialização e tamanhos de dados entre os formatos testados. Sua verbosidade e falta de compressão o tornam inadequado para ambientes móveis ou limitados em recursos.

O Protobuf foi projetado para maximizar eficiência, utilizando uma abordagem binária e contratos explícitos definidos no código. Como apontado no estudo, ele oferece as menores taxas de uso de espaço e os tempos mais rápidos de serialização/deserialização entre os formatos testados. Isso ocorre porque os dados trafegados contêm apenas os valores, enquanto as definições de estrutura são mantidas em arquivos externos. Essa eficiência o torna ideal para aplicações que priorizam desempenho e economia de recursos, como sistemas móveis.

Os testes apresentados no estudo de Sumaray e Makki (2012) mostram que o XML deve ser evitado sempre que possível, pois é significativamente mais lento e gera arquivos maiores comparado ao JSON e aos formatos binários como Protobuf. Por outro lado, o JSON é mais eficiente que o XML e altamente compatível com diversas plataformas, sendo a escolha preferida para integração com *web services*. O Protobuf, por sua vez, é recomendado para novos projetos ou cenários que exigem alta performance e baixo consumo de recursos, especialmente em dispositivos móveis.

Além disso, um caso relevante para o trabalho é o de serialização que ocorre nos blocos do Bitcoin, armazenados nos arquivos .dat. Esse formato utiliza uma serialização binária customizada para economizar espaço e melhorar a eficiência no armazenamento e transmissão de informações da *blockchain*. Diferentemente de formatos textuais como JSON e XML, ele é altamente otimizado para leitura e escrita por máquinas, priorizando desempenho em detrimento da legibilidade humana.

Dessa forma, a escolha do formato de serialização depende do contexto e das necessidades específicas da aplicação. Formatos textuais como JSON e XML são mais

adequados para interoperabilidade e legibilidade humana, enquanto os formatos binários, como Protobuf, oferecem vantagens significativas em eficiência, sendo ideais para sistemas com restrições severas de recursos.

Para o contexto desta dissertação, foi escolhido o protocolo Protobuf como padrão de serialização devido à sua ampla disseminação e aos benefícios que oferece em termos de eficiência na serialização e desserialização. Protobuf apresenta vantagens significativas na economia de espaço para armazenamento, já que utiliza uma abordagem binária que reduz drasticamente a quantidade de caracteres de controle em comparação com outros protocolos, como JSON e XML. Além disso, esse protocolo também pode ser aplicado na serialização de informações destinadas a arquivos físicos, permitindo que os dados sejam armazenados de forma eficiente em memória não volátil. Essa característica é particularmente relevante para os objetivos desta pesquisa, que busca otimizar o gerenciamento e a persistência de dados em sistemas *blockchain*, especialmente em ambientes IoT com recursos limitados (Viotti e Kinderkhedia, 2022).

Portanto, a análise dos diferentes protocolos de serialização apresentada neste capítulo fornece subsídios fundamentais para justificar a escolha do Protobuf no contexto desta dissertação. Ao explorar as características de eficiência, compactação e desempenho proporcionadas pelo Protobuf, destaca-se sua adequação para aplicações *blockchain* voltadas para ambientes IoT com recursos restritos. Essa escolha não apenas possibilita a otimização do armazenamento e da transmissão de dados, mas também viabiliza a integração de técnicas de persistência em memória não volátil, alinhando-se diretamente aos objetivos deste trabalho. Assim, as reflexões deste capítulo são essenciais para embasar as decisões técnicas implementadas no módulo Stordy, proposto como uma solução para otimizar o gerenciamento e a recuperação de dados na *SpeedyChain*.

2.3 Appendable-block blockchain

As *blockchains* convencionais frequentemente enfrentam desafios relacionados à latência, desempenho, escalabilidade, além de limitações de *hardware* e consumo de energia (Dorri et al., 2016). Tais características tornam pouco viável a aplicação dessa tecnologia em ambientes de Internet das Coisas (IoT), que tipicamente enfrentam restrições severas de recursos. Para enfrentar essas limitações, Lunardi et al. (2018, 2019); Michelin et al. (2018) propuseram a SpeedyChain, uma *blockchain* com blocos extensíveis projetada para atender às demandas específicas desses cenários.

Os trabalhos conceituais sobre essa *blockchain* discutem sua arquitetura, que adota um modelo inovador ao separar os dados das transações do cabeçalho dos blocos. Essa abordagem permite uma inserção mais rápida e eficiente de dados nos blocos (de Arruda et al., 2020). Em outras palavras, a inclusão de novos blocos na cadeia é desa-

coplada da inclusão de novas transações, que podem ser adicionadas de forma paralela pelos nós, mesmo após a criação do bloco (Lunardi et al., 2019). Essa característica é particularmente vantajosa em cenários IoT, onde eficiência e escalabilidade são essenciais para lidar com grandes volumes de dados e dispositivos com recursos limitados.

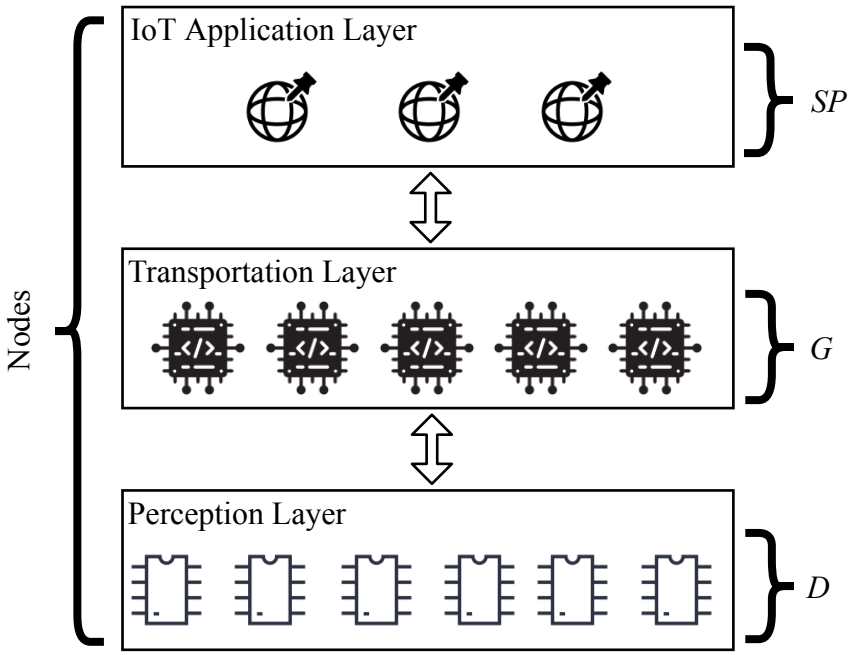

Figura 2.4 – Principais nodos *IoT* (Lunardi et al., 2019)

O modelo de *appendable-block* é baseado nas camadas apresentadas na Figura 2.2 e discutidas na Seção 2.1.1. Na estrutura da camada de comunicação, é adotada uma arquitetura dividida em três subcamadas, conforme ilustrado na Figura 2.4. Cada subcamada é responsável por uma etapa específica das operações de comunicação, sendo elas: a camada de Percepção, com *Devices (D)*; a camada de Transporte, com *Gateways (G)*; e a camada de Aplicação, com *Service Providers (SP)*.

Nessa arquitetura, os *devices* geram os dados e os enviam para os *gateways*, que têm a responsabilidade de inserir essas informações na *blockchain* e controlar o acesso aos dados armazenados. Os *service providers* podem acessar essas informações arquivadas na *blockchain* por meio dos *gateways*, permitindo a execução de operações sobre os dados.

Na camada de Dados, as transações são organizadas em uma estrutura que forma uma sequência encadeada. Cada transação é inserida em um bloco, onde o primeiro bloco é associado ao cabeçalho da cadeia principal por meio de um *hash*, enquanto os blocos subsequentes são conectados entre si através de ponteiros que apontam para as transações anteriores, como ilustrado na Figura 2.5. Essa abordagem resulta na criação de uma cadeia secundária de blocos conectada à principal (de Arruda et al., 2020).

A peculiaridade dessa estrutura dá origem ao termo “*appendable-block blockchain*”, uma vez que permite a inclusão de novas transações mesmo após a criação do

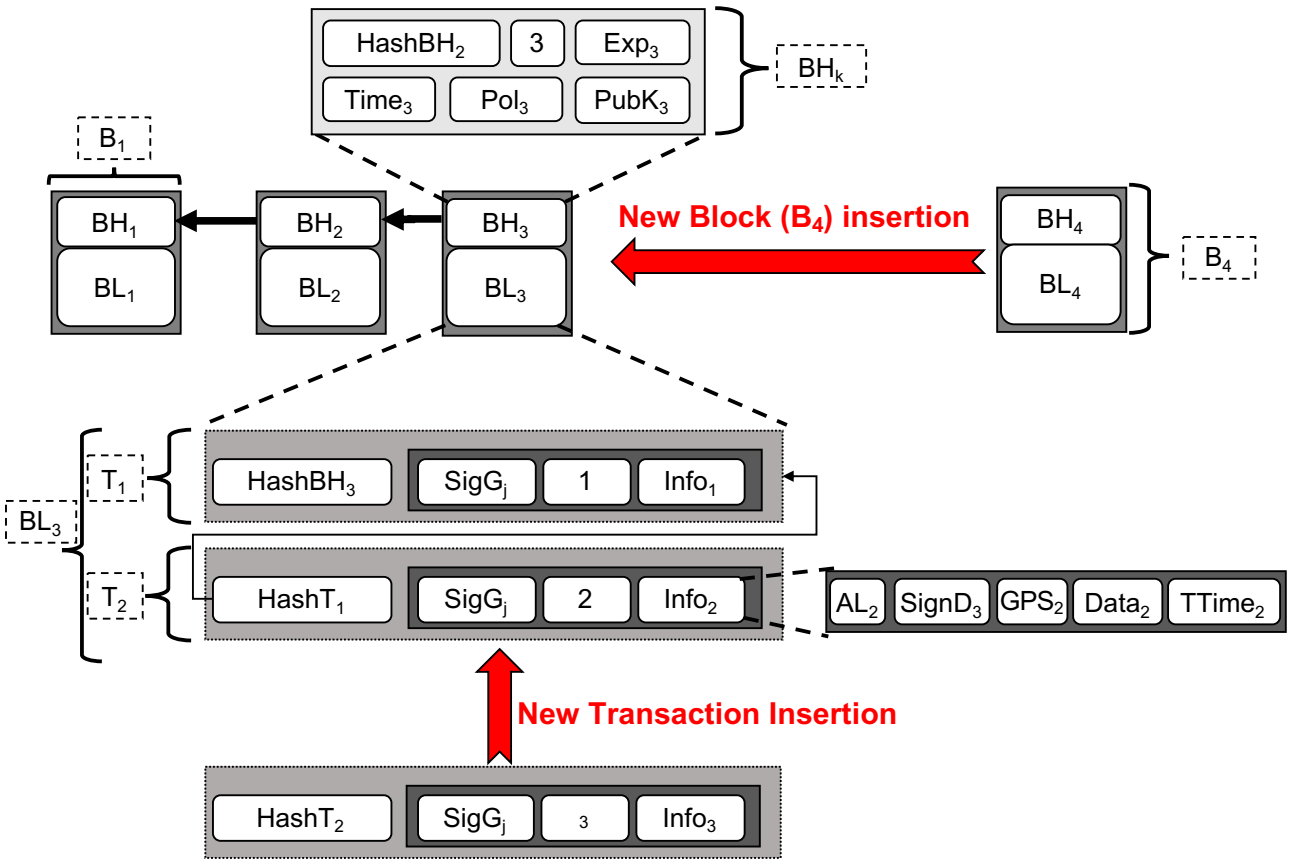

Figura 2.5 – Exemplo de estrutura de uma *appendable-block blockchain* (Lunardi et al., 2020)

bloco original, formando assim uma “cadeia de cadeias”. Por outro lado, a camada de Aplicação mantém as mesmas funções presentes em *blockchains* tradicionais, como a gestão de dados, a implementação de criptomoedas e a execução de *smart contracts*, sendo responsável por orquestrar as operações essenciais da *blockchain*.

Da mesma forma que ocorre com as *blockchains* convencionais, a camada de Consenso na *appendable-block blockchain* é responsável pela execução do algoritmo que insere novos blocos ou transações na cadeia. No entanto, os *gateways* apenas armazenam as transações na cadeia após a criação do bloco correspondente. Essa dinâmica, contudo, pode ser comprometida em cenários onde um dispositivo (*device*) tenta adicionar informações por meio de um *gateway* mal-intencionado.

Esse problema ocorre devido à arquitetura original da *appendable-block blockchain*, que pressupõe que os dispositivos conectam-se a um único *gateway* por vez. Nessa situação, um *gateway* mal-intencionado poderia comprometer o processo de inserção de informações de várias formas, como: (i) omissão de dados: o *gateway* pode deliberadamente não inserir as informações geradas pelo dispositivo na *blockchain*, efetivamente "eclipsando"o dispositivo. Isso significa que a produção de dados do dispositivo não será registrada, criando lacunas no registro; (ii) inserção de dados inválidos: um *gateway* comprometido pode introduzir transações inválidas, como execuções incorretas de contratos

inteligentes ou dados manipulados, comprometendo a integridade e a confiança na *blockchain*.

Para mitigar esses problemas, Lunardi et al. (2020) propuseram o algoritmo de consenso baseado em contexto (*context-based consensus*). Esse algoritmo exige que cada *device* se conecte a um número mínimo de *gateways*, reduzindo a probabilidade de influências de nós defeituosos, seguindo a lógica do consenso bizantino. No entanto, essa abordagem introduz novos desafios, como a duplicação de transações ou a inserção de transações inválidas. Para resolver essas questões, o armazenamento de novos dados é realizado segregando os *devices* em diferentes contextos.

O consenso é executado dentro de cada contexto e propagado para os *gateways* responsáveis por aquele contexto. Dessa forma, cada contexto realiza seu próprio consenso, determinado pelo *gateway* que o coordena. Além disso, o contexto utilizado em cada bloco é registrado em um campo específico do cabeçalho do bloco, garantindo rastreabilidade e integridade dos dados. Essa segregação contextual não apenas reduz os impactos de possíveis falhas, mas também permite maior escalabilidade e eficiência na propagação de transações e novos blocos na *blockchain*.

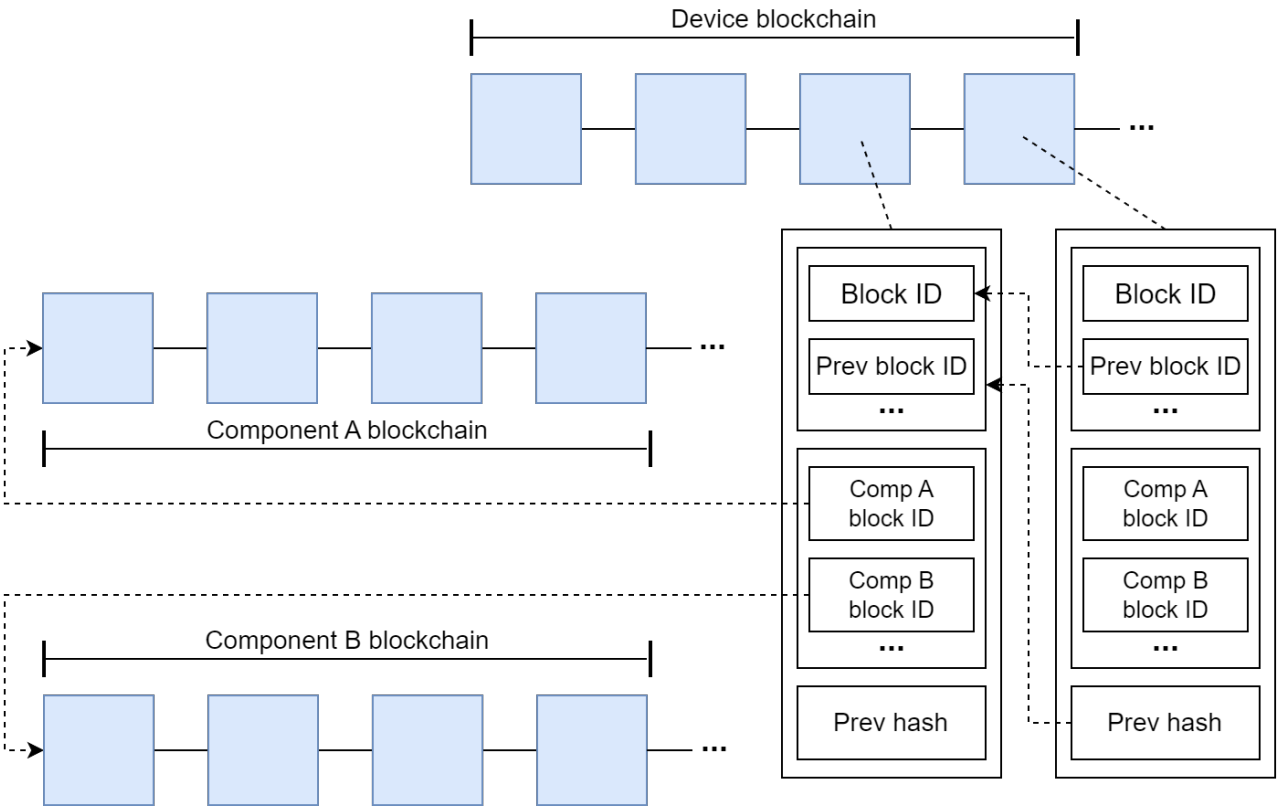

Figura 2.6 – Estrutura da patente Silveira et al. (2024)

A partir desses conceitos, Silveira et al. (2024) propuseram a aplicação dos princípios da SpeedyChain em um caso de uso real relacionado à validação de uma patente da HP. Essa patente aborda a gestão de eventos do ciclo de vida de dispositivos, como computadores, impressoras e notebooks, com foco na descentralização e segurança do

armazenamento das informações. No contexto dessa aplicação, a SpeedyChain foi adaptada para registrar dados de telemetria de dispositivos e de seus componentes em cadeias auxiliares conectadas ao bloco principal, permitindo a recuperação eficiente de informações específicas sem necessidade de percorrer toda a cadeia de blocos.

Na Figura 2.6, é ilustrada a estrutura utilizada para representar o ciclo de vida de dispositivos e seus componentes. Cada bloco da cadeia principal representa um dispositivo, enquanto suas cadeias auxiliares contêm as transações associadas aos componentes individuais. Isso viabiliza a manutenção do histórico completo de um componente, mesmo quando ele é transferido para outro dispositivo a um custo computacional reduzido e com maior rastreabilidade.

Além disso, foram introduzidas modificações na SpeedyChain para aprimorar o armazenamento e a recuperação de dados. Essas alterações incluem a implementação de um mecanismo de *dump*, que permite salvar o estado completo da *blockchain* em disco, garantindo persistência e continuidade em casos de reinicialização do sistema. Esse recurso é particularmente relevante em cenários corporativos, onde a preservação do estado da *blockchain* é essencial para evitar a perda de dados críticos e assegurar a consistência dos registros.

3. TRABALHOS RELACIONADOS

A evolução das tecnologias de *blockchain* trouxe uma série de desafios relacionados à escalabilidade, eficiência e gerenciamento de dados, especialmente em ambientes como a Internet das Coisas (IoT). Este capítulo apresenta uma revisão de trabalhos relacionados que exploram essas questões, destacando soluções inovadoras e suas conexões com os objetivos desta dissertação. O foco está em abordagens para armazenamento escalável, recuperação de dados eficientes e integração de sistemas *blockchain* em cenários com recursos limitados, fornecendo um panorama das principais contribuições acadêmicas nesse domínio.

3.1 SE-Chain: *A Scalable Storage and Efficient Retrieval Model for Blockchain*

Jia et al. (2021) propõem um modelo para lidar com os desafios relacionados à escalabilidade do armazenamento e à eficiência na recuperação de dados em sistemas *blockchain*. O trabalho justifica-se pelo crescimento constante do volume de dados em *blockchains*, que compromete a capacidade de armazenamento integral por todos os nós e dificulta a execução de consultas de forma eficiente em redes distribuídas. O modelo, denominado SE-Chain, organiza-se em três camadas principais: camada de dados, camada de processamento e camada de armazenamento.

A camada de dados utiliza uma estrutura chamada AB-M Tree (*Adaptive Balanced Merkle Tree*), que combina as características de árvores binárias balanceadas, para consultas rápidas, com as de *Merkle Tree*, para verificação de dados. Essa combinação visa reduzir o uso de espaço de armazenamento e melhorar o desempenho nas operações de consulta e verificação de dados. Em experimentos, a aplicação dessa camada resultou em uma redução de até 17% no tempo médio de consulta em comparação a modelos que utilizam exclusivamente *Merkle Tree*.

A camada de processamento incorpora um algoritmo para regulação da taxa de duplicação (*Duplicate Ratio Regulation Algorithm*), que ajusta dinamicamente o número de réplicas de cada bloco na rede. Este ajuste prioriza maior redundância para blocos mais recentes, devido à menor segurança associada, enquanto reduz o número de cópias para blocos mais antigos, nos quais o risco de perda de dados é menor. Essa abordagem busca equilibrar a segurança dos dados e a eficiência no uso dos recursos de armazenamento.

Por fim, a camada de armazenamento emprega um método de verificação de confiabilidade de nós (*Node Reliability Verification Method*), que avalia periodicamente a estabilidade dos nós na rede. Essa técnica permite priorizar nós mais confiáveis para o

armazenamento de dados, mitigando os riscos de perda de informações em caso de falhas ou comportamento inadequado por parte de nós instáveis.

Os resultados obtidos indicam que o modelo SE-Chain proporciona melhorias na escalabilidade do armazenamento e na eficiência de consultas. Além disso, o modelo mostrou-se capaz de reduzir o espaço médio de armazenamento ocupado por nó, particularmente em redes com maior número de participantes.

O modelo SE-Chain é relevante para esta dissertação, pois aborda desafios de escalabilidade e eficiência, objetivos que também orientam a proposta de melhorias na SpeedyChain. A estrutura AB-M Tree ressoa com a abordagem desta pesquisa em relação ao uso de indexação adaptativa para melhorar a recuperação de dados em *blockchains*. Além disso, os conceitos de redundância controlada e gerenciamento eficiente de nós fornecem diretrizes valiosas para o aprimoramento da robustez e escalabilidade em cenários de IoT.

3.2 BW-Tree: A B-tree for New Hardware Platforms

Levandoski et al. (2013) introduzem o Bw-Tree, uma variante da estrutura *B-tree* projetada para explorar o potencial de processadores multi-core e dispositivos de armazenamento em memória não volátil. Este trabalho aborda a limitação de desempenho das *B-trees* tradicionais em sistemas modernos, propondo melhorias na eficiência de processamento e na escalabilidade. O Bw-Tree emprega técnicas específicas para lidar com problemas relacionados à concorrência e ao acesso a armazenamento.

Uma das inovações apresentadas é a utilização de atualizações em *delta*, que evitam modificações no lugar (*update-in-place*). Este método melhora a eficiência do *cache* do processador ao reduzir a invalidação de linhas de *cache*. Além disso, as atualizações são gerenciadas por meio de uma tabela de mapeamento, que abstrai a localização física das páginas, permitindo operações atômicas e concorrentes utilizando instruções de comparação e troca (*compare-and-swap*, CAS).

Outra contribuição é a abordagem *latch-free*, que elimina a necessidade de bloqueios durante modificações na estrutura da árvore, como divisão e fusão de páginas. Essa técnica evita que os processos fiquem aguardando pela conclusão de operações em andamento, aumentando a eficiência em sistemas *multi-core*.

O Bw-Tree também inclui uma camada de armazenamento log-estruturado (*log-structured storage*, LSS), que organiza dados em blocos sequenciais para maximizar o desempenho de gravação em dispositivos de armazenamento *flash*. Essa camada reduz o impacto das características específicas de *flash*, como ciclos de apagamento antes da escrita, e melhora a eficiência do armazenamento ao consolidar deltas.

Os experimentos realizados demonstraram que o Bw-Tree supera significativamente sistemas tradicionais, como o BerkeleyDB em modo *B-tree*, tanto em cenários com operações intensivas de leitura quanto em cargas mistas de leitura e escrita. Além disso, a estrutura apresenta uma melhor eficiência de *cache* em comparação a listas de salto (*skip lists*), mostrando-se uma alternativa robusta para sistemas de gerenciamento de dados em hardware moderno.

O trabalho apresentado por Levandoski et al. (2013) é particularmente relevante para esta dissertação, pois a estrutura Bw-Tree foi utilizada como base para fornecer a estrutura de indexação do módulo de armazenamento Stordy. As inovações do Bw-Tree, como atualizações em *delta* e sua abordagem *latch-free*, foram fundamentais para garantir a eficiência e a escalabilidade da indexação na SpeedyChain, especialmente em cenários IoT com restrições de hardware. A capacidade do Bw-Tree de gerenciar operações concorrentes e de maximizar o desempenho em dispositivos modernos também ressoou com os objetivos desta pesquisa, permitindo que a solução proposta lidasse com grandes volumes de transações de forma eficiente e previsível. Assim, a integração dessa estrutura no Stordy não apenas reforça sua robustez, mas também demonstra a aplicabilidade do Bw-Tree em ambientes distribuídos de alto desempenho.

3.3 *Performance Optimization for Blockchain-Enabled Industrial Internet of Things (IIoT) Systems: A Deep Reinforcement Learning Approach*

Liu et al. (2019) propõem um *framework* baseado em aprendizado por reforço profundo (*deep reinforcement learning*, DRL) para otimização de desempenho em sistemas de *Industrial Internet of Things*(IIoT) habilitados por *blockchain*. Este trabalho aborda os desafios críticos relacionados à escalabilidade, descentralização, latência e segurança, conhecidos como o "trilema do *blockchain*"expandido para um contexto mais abrangente. O objetivo principal do *framework* é melhorar a escalabilidade do sistema enquanto mantém os outros fatores de desempenho equilibrados.

O *framework* proposto adota uma abordagem modular, permitindo ajustes dinâmicos em componentes fundamentais do sistema. Entre esses componentes estão o tamanho dos blocos, o intervalo entre blocos, a seleção dos produtores de blocos e a escolha do algoritmo de consenso. A implementação utiliza aprendizado por reforço profundo para tomar decisões adaptativas, ajustando o sistema conforme mudanças nas condições operacionais e características dinâmicas dos ambientes IIoT, como alta variação de tráfego e distribuição geográfica dos dispositivos.

Além disso, o *framework* apresenta uma metodologia quantitativa para avaliar o desempenho dos sistemas de *blockchain* no contexto do IIoT. Os quatro aspectos principais—escalabilidade, descentralização, latência e segurança—são analisados como parte

de um "*trade-off*"integrado, permitindo a otimização de uma métrica sem sacrificar as demais. Por exemplo, o aumento da escalabilidade é realizado ajustando dinamicamente o tamanho dos blocos e o intervalo entre eles, mas com restrições para garantir que a latência não ultrapasse limites aceitáveis e que a descentralização seja preservada.

Os resultados experimentais indicam que o *framework* é capaz de aumentar significativamente a escalabilidade de sistemas IIoT habilitados por *blockchain*, enquanto mantém um equilíbrio entre latência, descentralização e segurança. Os testes mostraram que o uso do DRL permite maior flexibilidade e eficiência na adaptação a mudanças em tempo real, como flutuações nas transações ou requisitos específicos de aplicações industriais.

Este trabalho é especialmente relevante para esta dissertação, pois destaca a importância de abordagens adaptativas e inteligentes para otimizar o desempenho de sistemas *blockchain* em ambientes de alta demanda, como o IIoT. Embora esta pesquisa não explore diretamente técnicas de aprendizado por reforço profundo, os desafios abordados por Liu et al. (2019), como escalabilidade, latência e descentralização, ressoam com os objetivos desta dissertação. Em particular, a abordagem modular e a otimização de parâmetros críticos, como o tamanho e o intervalo entre blocos, fornecem *insights* valiosos para o desenvolvimento de soluções escaláveis e eficientes, alinhadas às necessidades específicas de aplicações IoT. Além disso, este trabalho serve como inspiração para futuras investigações no contexto de armazenamento híbrido ou *off-chain*, explorando como a adaptação dinâmica de componentes do sistema pode ser utilizada para integrar de forma eficiente diferentes formas de armazenamento e melhorar a gestão de recursos em *blockchains* aplicadas a cenários industriais e IoT.

3.4 EdgeChain: *An Edge-IoT Framework and Prototype Based on Blockchain and Smart Contracts*

Pan et al. (2018) propõem o *framework* EdgeChain, que integra tecnologias de *blockchain* e contratos inteligentes para abordar desafios de escalabilidade, segurança e gerenciamento de recursos em sistemas de computação de borda (*edge computing*) para IoT. A principal contribuição do trabalho é a concepção de um sistema que vincula dispositivos IoT a recursos computacionais disponíveis em servidores de borda por meio de uma cadeia de blocos permissionada e um sistema interno de crédito baseado em moedas virtuais.

O EdgeChain é construído sobre um sistema de gerenciamento de recursos baseado em crédito, no qual cada dispositivo IoT é associado a uma conta na *blockchain*, podendo acessar recursos computacionais de borda com base em suas ações passadas, prioridade e tipo de aplicação. Contratos inteligentes são utilizados para automatizar e

regular essas interações, impondo políticas que garantem o comportamento adequado dos dispositivos IoT e previnem ações maliciosas de maneira eficiente. Todas as transações e atividades são registradas na *blockchain*, proporcionando um sistema robusto de auditoria e rastreabilidade.

Além disso, o *framework* adota um design orientado a recursos, permitindo alocar capacidade de computação, armazenamento e rede de maneira adaptativa às necessidades de diferentes dispositivos e aplicações. Uma característica notável é a compatibilidade retroativa com dispositivos legados, que são monitorados por meio de um proxy, sem a necessidade de integração direta com a *blockchain*.

Os resultados experimentais indicam que o EdgeChain é capaz de manter a segurança e a escalabilidade do sistema, com custos operacionais aceitáveis associados à integração do *blockchain* e contratos inteligentes. O *framework* também demonstrou ser flexível para atender a diversos cenários de aplicação, incluindo residências inteligentes, monitoramento em tempo real e serviços industriais. Este trabalho oferece uma perspectiva promissora sobre o uso de tecnologias de *blockchain* para gerenciar eficientemente dispositivos IoT em ambientes de computação de borda, destacando-se como uma solução prática para demandas de segurança, descentralização e escalabilidade.

O EdgeChain apresenta uma abordagem inovadora ao integrar *blockchain* e contratos inteligentes para gerenciar recursos computacionais em sistemas IoT, uma área de grande relevância para esta dissertação. Embora o foco do trabalho esteja em computação de borda, suas soluções para escalabilidade, segurança e descentralização oferecem *insights* valiosos sobre como tecnologias *blockchain* podem ser aplicadas para superar desafios em ambientes com recursos limitados. A ideia de alocar recursos de forma adaptativa e monitorar dispositivos por meio de um sistema de *blockchain* permissionada inspira novas estratégias de armazenamento distribuído, alinhando-se ao objetivo desta dissertação de explorar e melhorar soluções híbridas ou *off-chain*. Assim, o EdgeChain contribui como um referencial importante para expandir a eficiência e aplicabilidade de sistemas *blockchain* em cenários IoT.

3.5 *Adaptive Storage Optimization Scheme for Blockchain-IIoT Applications Using Deep Reinforcement Learning*

Akrasi-Mensah et al. (2022) apresentam um *framework* para otimização adaptativa de armazenamento em aplicações de *blockchain* voltadas para a IIoT. O trabalho busca enfrentar o problema do armazenamento escalável, comum em redes *blockchain*, ao propor um esquema que combina armazenamento em nuvem e local, utilizando aprendizado por reforço profundo (*Deep Reinforcement Learning*, DRL) para selecionar de forma otimizada os blocos a serem armazenados em cada local.

O principal objetivo do esquema proposto é minimizar os custos de armazenamento e latência, além de otimizar a ocupação do espaço local. Para isso, o problema de seleção de blocos foi modelado como um *Markov Decision Process* (MDP), onde as decisões do agente DRL consideram o estado atual do *ledger* e uma função de recompensa que combina métricas como frequência de consulta dos blocos, custo de armazenamento na nuvem e ocupação local. A abordagem permite a movimentação bidirecional de blocos, tanto para a nuvem quanto de volta ao armazenamento local, ajustando-se dinamicamente às mudanças nos padrões de consulta e na capacidade disponível.

Dois algoritmos de DRL, *Advantage Actor-Critic* (A2C) e *Proximal Policy Optimization* (PPO), foram implementados e avaliados em um ambiente simulado. Os resultados demonstraram que ambos os algoritmos alcançaram reduções significativas no armazenamento local, além de tempos de execução consideravelmente mais rápidos do que algoritmos evolutivos tradicionais, como NSGA-III e AT-MOPSO. Esse desempenho torna o esquema proposto uma alternativa eficiente e adaptativa para aplicações *blockchain-IIoT*.

O trabalho destaca também a relevância do DRL como ferramenta de otimização em cenários dinâmicos e de alta dimensionalidade, especialmente em sistemas *blockchain-IIoT*, onde restrições de armazenamento e latência impactam diretamente a escalabilidade e eficiência do sistema.

O esquema proposto por Akrasi-Mensah et al. (2022) fornece uma abordagem interessante para lidar com os desafios de armazenamento em sistemas *blockchain-IIoT*, ao combinar estratégias locais e em nuvem com aprendizado por reforço profundo. Embora o foco deste trabalho esteja na otimização dinâmica de blocos em diferentes camadas de armazenamento, suas contribuições inspiram reflexões diretamente aplicáveis à presente pesquisa. A proposta desta dissertação, centrada na melhoria da eficiência do armazenamento na SpeedyChain, pode se beneficiar dos *insights* apresentados sobre movimentação adaptativa de dados e decisões orientadas por padrões de acesso. Além disso, a ideia de integrar armazenamento híbrido, explorando tanto camadas locais quanto externas, ressoa com os desafios e oportunidades explorados nesta dissertação, oferecendo caminhos promissores para trabalhos futuros em *blockchain* aplicados ao IoT.

3.6 Comparação entre Trabalhos Relacionados

A análise comparativa dos trabalhos relacionados permite identificar as contribuições específicas de cada proposta, bem como situar a solução apresentada nesta dissertação no contexto do estado da arte. Nesta seção, são comparados aspectos fundamentais, tais como: (i) Escalabilidade: avalia-se como cada trabalho aborda o crescimento contínuo do volume de dados em *blockchains*, considerando sua capacidade de manter desempenho adequado à medida que a rede ou o número de transações aumenta; (ii) Efi-

ciência na Recuperação de Dados: analisam-se os métodos propostos para otimizar consultas e acesso às informações armazenadas, com ênfase na redução da latência e no uso eficiente de recursos computacionais; (iii) Aplicabilidade em Ambientes IoT: examina-se a adequação das soluções para cenários caracterizados por restrições severas de hardware, como dispositivos com limitações de memória volátil e capacidade computacional, típicos de aplicações de Internet das Coisas (IoT); (iv) Estratégias Híbridas de Armazenamento: investiga-se se e como as soluções integram abordagens combinadas, como o uso de armazenamento volátil e não volátil ou estratégias *on-chain* e *off-chain*, visando otimizar o gerenciamento de dados e a eficiência operacional.

Tabela 3.1 – Comparação entre trabalhos relacionados e a proposta desta dissertação.

Aspecto	**SE-Chain**	**Bw-Tree**	**EdgeChain**	**Adaptive Storage**	**Stordy**
Escalabilidade no Armazenamento	Sim	Não	Sim	Sim	Sim
Eficiência na Recuperação de Dados	Sim	Sim	Não	Sim	Sim
Aplicabilidade em Ambientes IoT	Sim	Não	Sim	Sim	Sim
Estratégias Híbridas de Armazenamento	Não	Não	Não	Sim	Não

A Tabela 3.1 apresenta uma comparação entre os trabalhos relacionados e o módulo proposto nesta dissertação, considerando os aspectos discutidos anteriormente. A partir dessa análise, é possível destacar algumas conclusões importantes. Por exemplo, no caso da BW-Tree (Seção 3.2), observa-se que ela não foi projetada para abordar diretamente desafios de escalabilidade no armazenamento de *blockchains*. Sua principal finalidade é otimizar o desempenho em sistemas modernos com *hardware* avançado, como processadores *multi-core* e dispositivos de memória não volátil. Embora a estrutura seja altamente eficiente para lidar com problemas específicos, como concorrência e acesso ao armazenamento, sua aplicação em *blockchains* não resolve diretamente as limitações de escalabilidade.

No contexto desta pesquisa, o Stordy utiliza a BW-Tree como base para indexação, mas a escalabilidade do sistema não depende exclusivamente dela. O armazenamento das informações pesadas é delegado ao sistema operacional, enquanto a BW-Tree desempenha o papel específico de indexar os dados da *blockchain* de forma eficiente. Essa separação de responsabilidades permite que o Stordy seja considerado escalável, pois o uso de recursos é otimizado e adaptado às restrições típicas de dispositivos IoT.

4. SOLUÇÃO: STORDY

No Capítulo 3 discutimos diversos trabalhos relacionados que destacam a relevância do tema de armazenamento em ambientes IoT, conforme detalhado nas Seções 3.1, 3.3, 3.4 e 3.5. Apesar de apresentarem soluções robustas para o armazenamento em *blockchains* convencionais, esses trabalhos não abordam o potencial do paralelismo, uma característica essencial para atender às demandas de ambientes IoT, onde dispositivos geram grandes volumes de dados simultaneamente e exigem alta eficiência no processamento.

Na Seção 2.3 introduzimos a Speedychain, uma *blockchain* com blocos extensíveis projetada para explorar características específicas e vantajosas em contextos IoT. Essa abordagem permite a inserção de transações em blocos previamente anexados, o que possibilita a adição paralela de transações em diferentes blocos da cadeia. Apesar dessa inovação, os modelos de armazenamento analisados no Capítulo 3 mostram-se ineficientes para esse tipo de *blockchain*, pois não foram concebidos para aproveitar a natureza paralela e dinâmica de blocos extensíveis.

Embora a Speedychain represente um avanço significativo em relação às *blockchains* convencionais, ainda apresenta desafios importantes que limitam sua aplicação prática em dispositivos IoT. A dependência de memória volátil e a ausência de mecanismos eficientes para armazenamento persistente em tempo real tornam inviável seu uso em cenários que exigem maior eficiência de recursos. Dispositivos IoT, frequentemente equipados com hardware de baixo desempenho e restrições de memória, demandam soluções de armazenamento que sejam capazes de aliar persistência, resiliência e baixo consumo de recursos. Além disso, a possibilidade de falhas e reinicializações nesses dispositivos torna ainda mais crítica a necessidade de preservar o estado da *blockchain* de forma segura e eficiente.

Nesse contexto, propomos o Stordy (*Storage and Retrieval of Data for SpeedyChain*), um módulo de armazenamento desenvolvido para atender às necessidades específicas de *blockchains* com blocos extensíveis, como a Speedychain. O Stordy é uma solução modular e extensível, projetada para se integrar perfeitamente à arquitetura existente, otimizando tanto o armazenamento quanto a recuperação de dados. Ele aproveita o paralelismo intrínseco desse modelo de *blockchain* e introduz melhorias significativas em eficiência e desempenho. O módulo foi concebido para superar as limitações das soluções existentes, reduzindo a dependência de memória volátil por meio do uso de armazenamento persistente e garantindo a integridade e a acessibilidade dos dados mesmo em casos de falhas inesperadas.

Para atender às demandas de eficiência e escalabilidade da Speedychain, reformulamos a interface de retenção de dados da camada de armazenamento. A estrutura

previamente baseada em listas, que armazenava informações localmente, foi substituída por um modelo que utiliza a comunicação com um serviço externo dedicado. Este serviço foi projetado para otimizar as requisições de armazenamento e leitura de dados, garantindo maior eficiência e desempenho, conforme ilustrado na Figura 4.1.

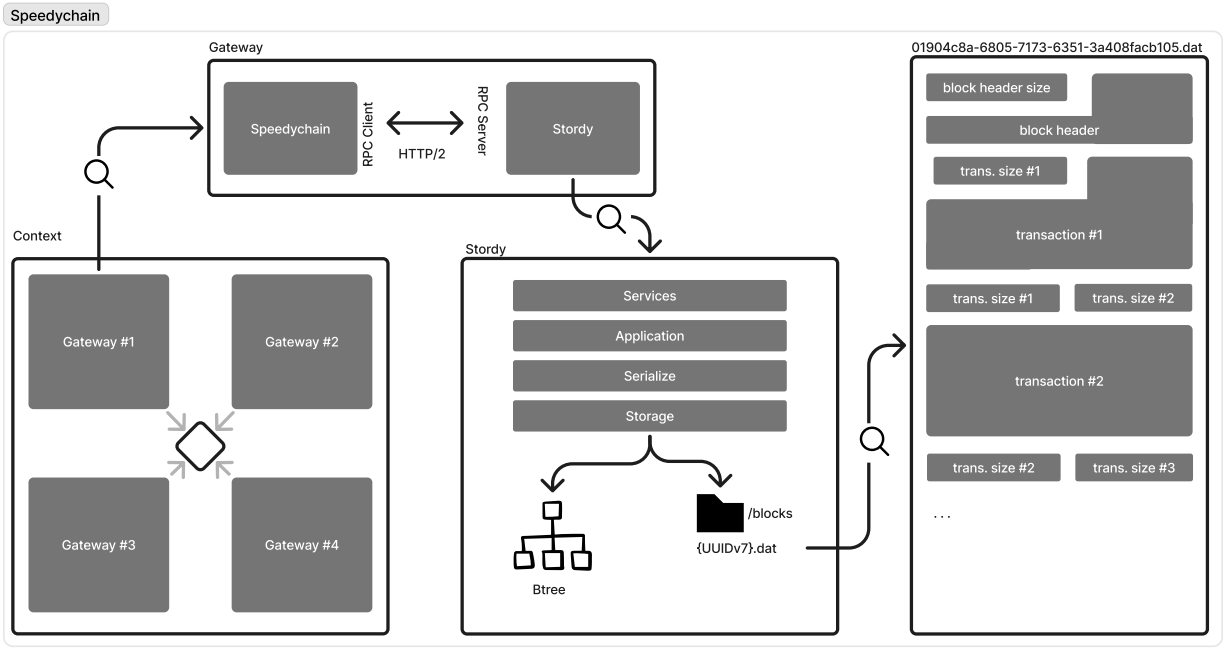

Figura 4.1 – Estrutura de um *gateway* utilizando *blockchain* com blocos extensíveis.

A comunicação entre o *gateway* e o serviço de armazenamento foi implementada utilizando um modelo de *Remote Procedure Call* (RPC), sob o protocolo HTTP/2, empregando o gRPC da Google. Essa escolha se justifica pelas vantagens que o gRPC proporciona em cenários de alto tráfego de dados. Com suporte a multiplexação de conexões e compressão de cabeçalhos via HTTP/2, o gRPC reduz a latência, aumenta o *throughput* e otimiza o uso de recursos, características essenciais para o desempenho de sistemas como a Speedychain (Ritu et al., 2024).

O modelo de armazenamento desenvolvido é estruturado em quatro camadas principais: *serviço*, *aplicação*, *serialização* e *armazenamento*, cada uma desempenhando um papel essencial para assegurar a modularidade, eficiência e escalabilidade do sistema. A Camada de Serviço estabelece a interface de comunicação entre a *blockchain* e o módulo de armazenamento, intermediando as operações de leitura e escrita de forma consistente e eficiente. Já a Camada de Aplicação é responsável por implementar as regras de negócio relacionadas ao armazenamento, incluindo políticas de consistência, validação de dados e organização lógica das transações, garantindo que tudo funcione de acordo com os critérios estabelecidos, promovendo a integridade e a confiabilidade das operações.

A Camada de Serialização utiliza o *Protobuf* para compactar e organizar os dados, possibilitando uma transmissão mais eficiente e reduzindo o espaço necessário para

armazenamento (Popić et al., 2016). Esse processo melhora significativamente o desempenho do sistema ao lidar com grandes volumes de informações. Por fim, a Camada de Armazenamento gerencia a indexação e o armazenamento físico dos dados em arquivos externos, assegurando a persistência, resiliência e controle de acesso. Essa arquitetura em camadas promove um design modular e flexível, permitindo que o sistema se adapte facilmente a novos requisitos, ao mesmo tempo em que garante escalabilidade para lidar com volumes crescentes de dados, essenciais em aplicações como *blockchain* e IoT.

Nas próximas seções exploramos em detalhe como o módulo proposto gerencia os processos de serialização, indexação e os serviços disponibilizados para o *client*. Neste contexto, o termo *client* refere-se à aplicação principal da *blockchain*, que interage diretamente com o módulo para realizar operações essenciais.

4.1 Serialização

A serialização das informações dos blocos na Speedychain foi projetada para otimizar o armazenamento e garantir a integridade dos dados. Para cada bloco anexado à cadeia, um novo arquivo com extensão *dat* é criado no diretório *blocks*. Nesse arquivo, são armazenados apenas os *fields* do cabeçalho do bloco, que incluem informações essenciais, como: (i) *index*, (ii) *previous_hash*, (iii) *timestamp*, (iv) *hash*, (v) *nonce*, (vi) *public_key*, (vii) *block_context*, (viii) *device*, (ix) *previous_expired_block_hash*, e (x) *previous_block_signature* .

Esses dados são serializados utilizando o *Protocol Buffers* (Protobuf), uma tecnologia desenvolvida pela Google que oferece uma maneira altamente eficiente de compactar e estruturar informações. O Protobuf foi escolhido devido à sua capacidade de gerar arquivos binários significativamente menores em comparação com formatos textuais, como JSON e XML, reduzindo assim o consumo de espaço no armazenamento e na transmissão de dados. Além disso, sua abordagem baseada em esquemas facilita a compatibilidade entre diferentes sistemas e linguagens de programação, garantindo interoperabilidade mesmo em ambientes heterogêneos. Essa eficiência é particularmente relevante para sistemas distribuídos como a SpeedyChain, onde a redução no tamanho dos dados e o aumento na performance algorítmica são essenciais para lidar com os desafios impostos por dispositivos com recursos limitados, como em aplicações IoT.

Adicionalmente, considerando que novas transações são adicionadas ao bloco, as informações dessas transações são concatenadas ao conteúdo existente nos arquivos. Contudo, apenas anexar dados ao arquivo sem uma delimitação adequada pode dificultar a identificação das seções correspondentes ao cabeçalho do bloco ou às transações. Por exemplo, seria impossível distinguir onde começa e termina o cabeçalho ou separar transações individuais.

Para resolver esse desafio, adotamos um mecanismo de serialização que emprega 2 *bytes* de controle no início e 2 *bytes* no final de cada elemento serializado, seja um cabeçalho de bloco ou uma transação. Essa abordagem, inspirada na estrutura *Recursive-length prefix (RLP)* da Ethereum (Omahs, et al., 2025), permite navegação bidirecional no arquivo: enquanto os *bytes* iniciais atuam como delimitadores direcionais (indicando tipo e comprimento do dado, conforme o padrão RLP), os *bytes* finais funcionam como âncoras para navegação reversa.

A dupla de *bytes* no final de cada elemento serializado serve como um ponto de referência fixo, permitindo que o ponteiro de leitura retroceda eficientemente a partir do fim do arquivo. Essa estratégia é essencial para identificar rapidamente a última transação anexada sem exigir uma varredura sequencial completa – ao alcançar o final do arquivo, os *bytes* finais fornecem um *offset* preciso para recuar até o início da transação mais recente. Assim, combinamos a eficiência de serialização do RLP com a capacidade de leitura reversa, otimizando operações críticas como recuperação da última transação. A Figura 4.2 ilustra essa estrutura, destacando como os pares de *bytes* habilitam a varredura bidirecional.

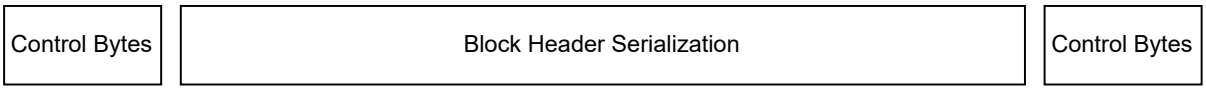

Figura 4.2 – Estrutura de serialização em arquivo.

O uso de 2 *bytes* para representar o tamanho de cada elemento é mais do que suficiente para os casos considerados, uma vez que esse tamanho pode alcançar até 2^{16} (65.536 bits). Tanto o cabeçalho de um bloco quanto as transações têm tamanhos significativamente menores que esse limite, tornando essa abordagem eficiente e apropriada. Além de garantir a compatibilidade e integridade das informações, essa solução preserva a simplicidade e a eficiência do design do sistema.

4.2 Indexação

Para a criação dos arquivos com a estrutura mencionada anteriormente, utiliza-se uma nomenclatura baseada no padrão UUID versão 8 (UUIDv8). Esse padrão, conforme especificado no RFC 9562 (Davis et al., 2024), permite a geração de identificadores únicos baseados em tempo, garantindo que os identificadores sejam ordenáveis cronologicamente. Essa abordagem é essencial para a obtenção do último bloco anexado à cadeia, um requisito funcional da Speedychain.

No entanto, ao adotar esse padrão de identificadores, o bloco não pode ser diretamente identificado por sua chave pública. Na Speedychain, a chave pública desempenha um papel central, pois é utilizada para endereçar novas transações aos respectivos blocos. Essa chave pública é gerada no dispositivo que realiza a conexão com o *gateway* (Lunardi et al., 2019). Para superar essa limitação, é necessária a implementação de uma estrutura de mapeamento que vincule a chave pública de cada bloco ao seu identificador UUID e, consequentemente, ao nome do arquivo onde as informações do bloco estão armazenadas.

Além disso, surge um problema relacionado à localização das transações no *buffer* do arquivo. Sem uma forma eficiente de determinar a posição dos *bytes* de controle de uma transação, seria necessário iterar sobre o arquivo para identificar e acessar uma transação específica. Esse procedimento resultaria em um custo $O(n)$, o que é incompatível com os objetivos de desempenho de uma *blockchain*, especialmente em dispositivos de recursos limitados. Para evitar esse custo e possibilitar o acesso direto a uma transação específica, a estrutura de indexação é concebida com base na Bw-Tree, conforme detalhado na Seção 3.2. Essa estrutura utiliza uma abordagem *latch-free* que elimina bloqueios durante as operações de escrita e leitura, garantindo alta eficiência e concorrência em sistemas multi-core.

A Bw-Tree organiza os dados de forma a evitar alterações no local (*update-in-place*), gerenciando atualizações por meio de uma tabela de mapeamento e operações atômicas como *compare-and-swap* (CAS). Ao adotar essa estrutura, o sistema pode manter metadados que relacionam as transações aos blocos aos quais pertencem e ao deslocamento no *buffer* do arquivo. Isso permite localizar e acessar diretamente o trecho relevante do arquivo com custo constante, reduzindo significativamente o consumo de memória volátil e melhorando o desempenho geral da Speedychain.

4.3 Serviços

Os serviços sugeridos possuem os objetivos de suprir as necessidades da Speedychain quanto ao gerenciamento e anexo de informações, isso para que seja possível apenas adicionar uma abstração à *blockchain* que realiza a chamada ao Stordy, que realiza as demais tarefas. Considerando a estrutura da *blockchain*, os seguintes serviços são sugeridos: (i) adição de um bloco à cadeia (Seção 4.4); (ii) buscar um bloco por sua chave pública (Seção 4.5); (iii) obter a quantidade de transações existentes em um bloco a partir da sua chave pública (Seção 4.6); (iv) obter o último bloco anexado à cadeia (Seção 4.7); (v) adicionar uma transação a um bloco (Seção 4.8); (vi) obter a última transação anexada a um determinado bloco (Seção 4.9); (vii) obter uma transação por meio de seu

hash (Seção 4.10); (viii) verificar se uma determinada transação existe em um bloco. A estrutura do módulo proposto pode ser visualizada conforme Figura 4.3.

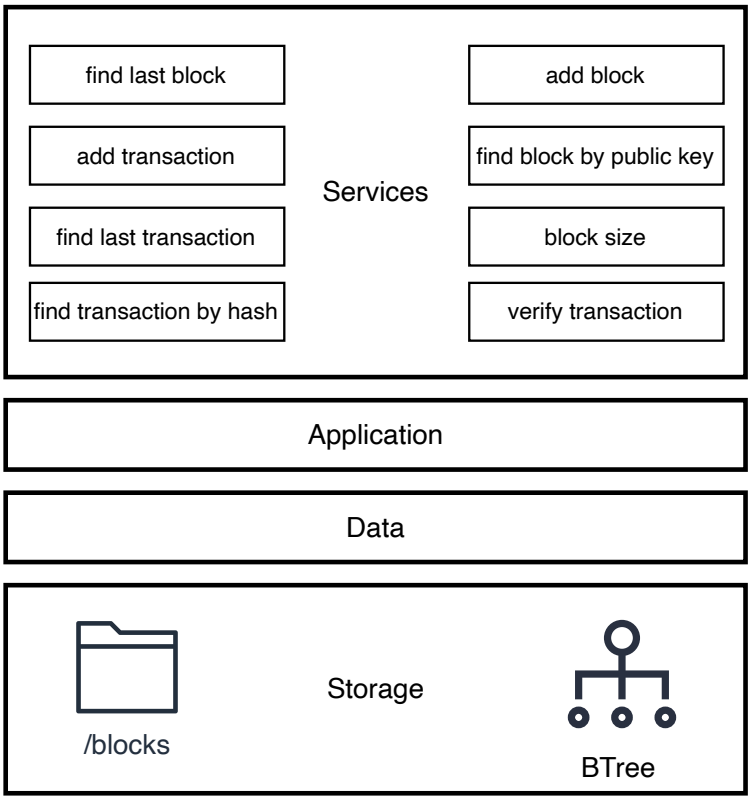

Figura 4.3 – Estrutura de camadas do Stordy.

4.4 Adição de um novo bloco

A principal característica distintiva da *blockchain*, composta por blocos extensíveis, reside em sua capacidade de anexar transações a blocos que fazem parte da cadeia. Consequentemente, tornou-se imprescindível a formulação de um modelo de armazenamento que possibilite essas anexações de dados com um custo linear consistente. Para permitir a adição de novos blocos à cadeia, foi desenvolvido um sistema que utiliza a serialização do cabeçalho do bloco, juntamente com um par de *bytes* de controle projetados para especificar o tamanho dos dados serializados (Alg. 4.1, linhas 1 a 4). Posteriormente, um novo arquivo é criado no disco para armazenar esses dados, vinculando assim um novo bloco à cadeia existente (Alg. 4.1, linhas 7 a 9).

Assim como o protocolo MQTT (Soni e Makwana, 2017), que frequentemente incorpora campos para especificar o tamanho da carga útil, o Stordy utiliza *bytes* de controle

para otimizar a eficiência na transferência de dados, especialmente em dispositivos IoT com recursos limitados. Esses *bytes* de controle facilitam o reconhecimento eficiente da carga útil, independentemente de a carga representar a serialização de um cabeçalho de bloco ou transações anexadas. Esse mecanismo garante um processamento de informações simplificado e preciso.

Algorithm 4.1 Adding a Block

Require: Block header by *blk*
Ensure: Status *s*
1: *publicKey* ← *blk*.*publicKey*
2: *blkHeaderBuffer* ← ConvertBlockToBytes(*blk*)
3: *blkSize* ← Size(*blkHeaderBuffer*)
4: *blkSizeArrayU*16 ← ConvertU64ToArrayU16(*headerBlkSize*)
5: *blkSizeBytes* ← Array.slice(*headerBlkSizeArrayU*16, 2)
6: *bytes* ← Concatenate(*blkSizeBytes*, *blkHeaderBuffer*)
7: *uuid* ← UUIDv7(Now)
8: Bwtree.write(*publicKey*, *uuid*)
9: File.write(*uuid*, *bytes*)
10: **return** True

4.5 Buscar um bloco por sua chave pública

Devido à ausência de um mapeamento direto entre chaves públicas e blocos, é necessário utilizar armazenamento não volátil para localizar, de forma eficiente, o arquivo que contém as informações de um bloco específico. Para isso, emprega-se a estrutura de indexação da Bw-tree.

Algorithm 4.2 Obtaining Block by PublicKey

Require: Block identified by *publicKey*
Ensure: Block *blk*
1: *path* ← Bwtree(*publicKey*)
2: **if** *path* = *null* **then**
3: **return** *null*
4: **end if**
5: *f* ← File.open(path)
6: *headerBlkSizeBuffer* ← f.Seek(0, 2)
7: *headerBlkSize* ← ConvertU16ArrayToU32(*headerBlkSizeBuffer*)
8: *headerBytes* ← *f*.*Seek*(2, headerBlkSize + 2)
9: *blk* ← ConvertBytesToBlock(headerBytes)
10: **return** *blk*

Com a utilização dessa estrutura, é possível referenciar de maneira eficiente o UUIDv8 associado à chave pública de um bloco específico na cadeia, conforme descrito

na Seção 4.2. Após identificar a localização do arquivo de um bloco, apenas a porção relevante do arquivo é acessada, minimizando a sobrecarga na memória volátil da aplicação (Alg. 4.2, linhas 5 a 9).

4.6 Obtenção da Quantidade de Transações em um Determinado Bloco

Na proposta apresentada, a quantidade de transações dentro de um bloco não é essencial para o funcionamento do livro-razão. Isso é possível devido ao uso de um identificador numérico incremental para a última transação na blockchain com blocos extensíveis. Consequentemente, obter a última transação é suficiente para determinar o tamanho de um bloco. Contudo, esta abordagem foi concebida para cenários específicos onde esses dados são explicitamente requisitados por um dispositivo.

O mecanismo proposto baseia-se na realocação do índice em um arquivo. Dessa forma, torna-se possível ler os *bytes* de controle de um bloco sem a necessidade de carregar o bloco completo para a memória. Isso é realizado movendo o índice pelos *bytes* de controle e avançando até os caracteres de controle da próxima transação. Quando o índice atinge o final do arquivo, é possível determinar quantas transações existem em um bloco (Alg. 4.3, linhas 8 a 11). Apesar da sobrecarga computacional associada a essa proposta, ela evita a necessidade de carregar o arquivo completo para a memória volátil.

Algorithm 4.3 Obtaining the Number of Transactions in a Block

Require: Block identified by *publicKey*
Ensure: Number of transactions *n*

```
 1: path ← Bwtree(publicKey)
 2: f ← File.open(path)
 3: n ← 0
 4: pointer ← 0
 5: bytes ← f.seek(pointer, 2)
 6: pointer ← ConvertToU32(bytes)
 7: size ← file.size
 8: while pointer < size do
 9:     n ← n + 1
10:     bytes ← f.seek(pointer, 2)
11:     pointer ← ConvertToU32(bytes)
12: end while
13: File.close(f)
14: return n
```

4.7 Obtenção do Bloco Mais Recente da Cadeia

Devido ao modelo de consenso utilizado pelas blockchains com blocos extensíveis, torna-se necessário um método eficiente para recuperar o último bloco anexado à cadeia, uma etapa crucial para a adição de novos blocos subsequentes. Ademais, os métodos anteriores não possuem um sistema definitivo de ordenação baseado em *hash* ou chave pública, o que não está alinhado com o ordenamento cronológico dos blocos anexados à cadeia.

Propomos o uso de *Universally unique identifier* (UUIDs) para otimizar a recuperação das informações do bloco mais recente. Embora essa abordagem resolva o problema da identificação dos blocos por meio dos nomes dos arquivos, ela não aborda diretamente o requisito de ordenação. Para isso, nossa estratégia baseia-se na recomendação mais recente para esses identificadores, com foco nos requisitos técnicos delineados. A decisão de adotar o UUIDv8 foi motivada pelas necessidades e desafios identificados (Alg. 4.4, linhas 1 a 3). Após isso, os dados extraídos do bloco são recuperados e convertidos para bytes (linhas 5 a 8). Com isso, cada novo bloco anexado é denotado por um UUIDv8, que se diferencia das versões anteriores por incorporar um *timestamp*, permitindo assim a ordenação dos identificadores.

Algorithm 4.4 Obtaining Last Block

Ensure: Last Block *blk*

1: *files* ← ReadDir(”/blocks”)
2: *last* ← List.Sort(files, “desc”)
3: *f* ← File.open(last)
4: *headerBlkSizeBuffer* ← *f*.*Seek*(0, 2)
5: *headerBlkSize* ← ConvertU16ArrayToU32(*headerBlkSizeBuffer*)
6: *headerBytes* ← *f*.*Seek*(2, headerBlkSize + 2)
7: *blk* ← ConvertBytesToBlock(headerBytes)
8: **return** *blk*

4.8 Adição de Transações a um Bloco

Para incluir transações em um bloco, é necessário, primeiramente, verificar a autenticidade do bloco, validando a chave pública recebida, conforme descrito na Seção 4.5. Esse processo de verificação, abordado nas linhas 1 a 4 do Algoritmo 4.5, assegura que a chave pública corresponde a um bloco existente. Após a validação, os detalhes da transação são serializados, e 2 *bytes* de controle são adicionados tanto no início quanto no final da serialização resultante (Alg. 4.5, linhas 5 a 8 e linhas 9 a 12). Esses *bytes*

de controle, conforme detalhado na Seção 4.4, especificam o tamanho das informações anexadas e são essenciais para garantir a integridade dos dados.

A informação serializada, encapsulada com os *bytes* de controle, é então adicionada ao final do arquivo do bloco, conforme detalhado nas linhas 9 a 12 do Algoritmo 4.5. Esse mecanismo de controle duplo simplifica o processo de iteração pelo arquivo. A Figura 4.1 demonstra a serialização de um bloco junto com suas transações.

Algorithm 4.5 Adding Transaction to Block

Require: Block identified by *publicKey*
Require: Transaction *tr*
Ensure: Status *s*
1: *path* ← Bwtree(*publicKey*)
2: **if** *path* = *null* **then**
3: **return** *False*
4: **end if**
5: *serialize* ← SerializeTransaction(*tr*)
6: *sizeTr* ← *serialize.size*
7: *sizeToU*16*Array* ← ConvertU32ToArrayU16(*sizeTr*)
8: *bytes* ← Concatenate(*sizeToU*16*Array*, *serialize*, *sizeToU*16*Array*)
9: *f* ← File.open(*path*)
10: *f.append*(*bytes*)
11: *f.close*
12: **return** *True*

Após o processo de serialização, a adição de informações ao bloco é realizada por meio do método *append*, que é nativamente fornecido pelo sistema de arquivos do sistema operacional (Alg. 4.5, linha 10). Essa técnica permite o registro eficiente dos dados no final do arquivo, eliminando a necessidade de modificar o conteúdo existente. Assim, aumenta-se a eficiência do processo de incorporação de novos dados a um arquivo pré-existente, mantendo um custo linear. As vantagens associadas à utilização dessa abordagem nativa incluem: (i) permitir que transações sejam adicionadas a um bloco sem a necessidade de reescrever o bloco, economizando tempo e recursos (Andrew e Herbert, 2015); (ii) manter a integridade dos dados, uma vez que não há necessidade de sobrescrever informações anteriores (Silberschatz et al., 2018); (iii) possibilitar a adição de transações em paralelo, visto que novas transações são adicionadas ao final de um bloco, evitando conflitos de escrita (Andrew e Herbert, 2015).

Para facilitar a localização de uma transação dentro de um bloco, uma instância separada da Bw-tree é empregada. Ela mantém o *hash* da transação como chave e contém detalhes sobre o bloco específico ao qual a transação está relacionada, juntamente com sua posição sequencial no arquivo serializado do bloco. Esse método permite a recuperação de uma transação específica sem a necessidade de carregar todos os dados do bloco para a memória volátil da aplicação.

4.9 Obtendo a Última Transação Anexada a um Determinado Bloco

Para obter a última transação anexada a um bloco, utilizam-se os *bytes* de controle adicionados ao final da transação, conforme descrito nas Seções 4.4 e 4.8 . Os dois últimos *bytes* do arquivo do bloco são utilizados para recuperar o tamanho da última transação. Posteriormente, o tamanho da transação é decrementado por meio da técnica demonstrada na Seção 4.10 (Alg. 4.6).

Algorithm 4.6 Obtaining Last Transaction of Block

Require: Block identified by *publicKey*
Ensure: Last transaction *t*

```
1:  path ← Bwtree(publicKey)
2:  f ← File.open(path)
3:  size ← f.size
4:  pointer ← size − 2
5:  bytes ← f.seek(pointer, 2)
6:  sizeOfLastTransaction ← ConvertToU32(bytes)
7:  pointer ← size − sizeOfLastTransaction − 2
8:  bytes ← f.seek(pointer, sizeOfLastTransaction)
9:  t ← BytesToTransaction(bytes)
10: return t
```

4.10 Obtendo uma Transação pelo seu Hash

Uma vez que a transação tenha sido indexada com base no seu *hash* e contenha o valor do bloco em que reside, juntamente com o seu deslocamento, utiliza-se o método *seek*. A proposta desse método visa minimizar o tempo necessário para acessar os dados, permitindo o acesso direto a qualquer bloco dentro do sistema de arquivos. Isso é possível por meio da realocação do ponteiro de leitura para a posição desejada (Ritchie e Thompson, 1974). Consequentemente, essa funcionalidade permite a recuperação parcial de informações na memória, movendo o ponteiro para os bytes iniciais da transação durante o processo de serialização do arquivo. Assim, somente os dois primeiros bytes são recuperados para determinar o tamanho da transação, seguidos pela recuperação subsequente dos bytes restantes (Alg. 4.7).

4.11 Validação da Existência de uma Transação em um Bloco

Após a estrutura de indexação delineada nas Seções 4.4 e 4.5, torna-se evidente a simplicidade de verificar a presença de uma transação específica em um bloco. Utili-

Algorithm 4.7 Obtaining Transaction by Hash

Require: Transaction identified by *hash*
Ensure: Transaction *t*

```
1: pathAndOffset ← Bwtree(hash)
2: path ← pathAndOffset.split(0)
3: offset ← pathAndOffset.split(1)
4: f ← File.open(path)
5: pointer ← offset
6: bytes ← f.seek(pointer, 2)
7: sizeOfLastTransaction ← ConvertToU32(bytes)
8: pointer ← pointer + 2
9: bytes ← f.seek(pointer, sizeOfLastTransaction)
10: t ← BytesToTransaction(bytes)
11: return t
```

zando o *hash* da transação, é possível identificar o bloco em que ela reside. Consequentemente, é necessário confirmar a presença de uma saída válida nesta busca na estrutura de indexação. A ausência de uma saída válida indicaria que a transação não foi localizada no respectivo bloco (Alg. 4.8).

Algorithm 4.8 Validating the existence of a transaction

Require: Transaction identified by *hash*
Ensure: Status *s*

```
1: path ← Bwtree(hash)
2: if path = null then
3:     return False
4: end if
5: return True
```

4.12 Considerações Finais

Neste capítulo, apresentamos o Stordy, um módulo de armazenamento persistente desenvolvido para suprir as limitações da Speedychain em ambientes IoT. A solução proposta combina três elementos-chave: (i) um mecanismo de serialização bidirecional com *bytes* de controle, inspirado no padrão RLP da Ethereum; (ii) estratégias de indexação baseadas em UUIDv8 cronologicamente ordenáveis e estrutura Bw-Tree para acesso eficiente; e (iii) uma arquitetura em camadas que desacopla serialização, persistência e lógica de negócio. Essa abordagem permite não apenas a recuperação eficiente da última transação anexada via navegação reversa, mas também garante resiliência contra falhas através do armazenamento persistente em arquivos estruturados.

Os algoritmos e serviços descritos demonstram como o Stordy atende aos requisitos específicos de *blockchain* extensíveis, permitindo operações críticas como adição

paralela de transações (Algoritmo 4.5) e recuperação direta de dados via *hash* (Algoritmo 4.7) com complexidade constante. A integração com protocolos como gRPC e técnicas como *memory-mapped files* posiciona a solução como uma alternativa viável para dispositivos com recursos limitados. No Capítulo 5 avaliaremos experimentalmente o módulo proposto, analisando métricas de desempenho como latência na inserção de transações e consumo de memória.

5. AVALIAÇÃO EXPERIMENTAL

O presente capítulo tem como objetivo avaliar os benefícios da incorporação do módulo de armazenamento na *appendable-block blockchain*, explorando seu impacto no desempenho, escalabilidade e uso de recursos em diferentes cenários. Para essa finalidade, as avaliações foram organizadas em duas categorias distintas: (i) Avaliação I: Aplicabilidade do módulo de armazenamento, que busca analisar o consumo de recursos e o desempenho geral da *blockchain* em cenários de alta demanda; e (ii) Avaliação II: Comportamento do Stordy em dispositivos IoT, que avalia a viabilidade da solução em contextos com restrições de hardware.

Em ambas as avaliações os mesmos critérios foram utilizados, porém em ambientes distintos. As métricas avaliadas foram: (i) consumo de memória volátil (RAM), em função do número de transações anexadas; e, (ii) tempo médio para adicionar novas transações à cadeia, em milissegundos. Por fim, para assegurar a robustez estatística dos resultados, cada avaliação foi executada 10 vezes em condições controladas (mesmo ambiente, *hardware* e versões de *software*).

5.1 Avaliação I: Aplicabilidade do módulo de armazenamento

Nas versões anteriores da *blockchain*, a aplicação enfrentava limitações impostas pela capacidade de memória RAM disponível na máquina onde era executada, restringindo tanto sua escalabilidade quanto sua viabilidade prática em ambientes com recursos limitados. O módulo de armazenamento proposto busca superar essas limitações, introduzindo uma abordagem que visa melhorar a eficiência no consumo de memória e reduzir a dependência de recursos voláteis, especialmente em cenários de alto volume de transações. Para esta avaliação, foi utilizado um sistema equipado com processador M1 PRO e 16 GB de memória RAM.

5.1.1 Resultados

Para o primeiro teste, foi implementado um método de inicialização e popularização da *blockchain*, permitindo emular um cenário inicial com um conjunto de blocos e transações previamente anexados. Essa funcionalidade foi incorporada diretamente ao módulo de armazenamento, automatizando a replicação de transações por meio da modificação de parâmetros como índice, *hash* criptográfico, assinatura digital e *hash* da

transação anterior. Durante o experimento, a *blockchain* foi conectada ao módulo Stordy, permitindo a realização de *dumps* do estado atual da cadeia para backup.

Dois cenários foram configurados: (i) execução da *blockchain* sem o módulo Stordy; e (ii) execução integrada com o Stordy. Em ambos os casos, monitorou-se o consumo de memória volátil à medida que novas transações eram adicionadas. No cenário sem o módulo Stordy, observou-se um aumento progressivo no consumo de memória, evidenciando a dependência direta da *blockchain* de recursos voláteis. Por outro lado, a utilização do módulo Stordy demonstrou uma redução significativa no consumo de memória, especialmente quando o número de transações ultrapassou 32.000, como apresentado na Figura 5.1.

No segundo experimento, avaliou-se a variação no tempo necessário para adicionar novas transações à *blockchain*. Para isolar o impacto do módulo de armazenamento, simulou-se a adição de transações diretamente a blocos preexistentes, sem realizar outros processamentos típicos da *blockchain*. A API do módulo Stordy foi utilizada para simular o comportamento de um produtor de transações, variando o número de transações anexadas ao bloco de 1 até mais de 16 milhões.

Os resultados, ilustrados na Figura 5.2, mostram que o tempo médio para adicionar uma nova transação permaneceu estável, com um valor médio de 1,09 ms e um desvio padrão de 0,35 ms. Pequenas variações observadas são atribuídas à preempção do sistema operacional, mas não comprometeram a consistência do processo. Esse desempenho estável indica que o custo computacional para adicionar novas transações à *blockchain* é previsível, mesmo em cenários de alta demanda.

5.2 Avaliação II: Comportamento do Stordy em dispositivos IoT

Para avaliar o desempenho do módulo de armazenamento em dispositivos com restrições de hardware, como aqueles encontrados em aplicações de *Internet of Things* (IoT), conduzimos dois experimentos principais. O objetivo foi investigar o impacto do módulo de armazenamento no consumo de recursos e no tempo necessário para inserir novas transações na *blockchain*, utilizando um ambiente experimental composto por *Raspberry Pi 3*, cada um equipado com 1 GB de *RAM* e um cartão *SD* de 32 GB Classe C2 de 2M, conforme é possível visualizar na Figura 5.3.

5.2.1 Resultados

No primeiro experimento, avaliamos o consumo de memória *RAM* em função do número de transações anexadas à *blockchain*. A versão original da *blockchain*, conforme

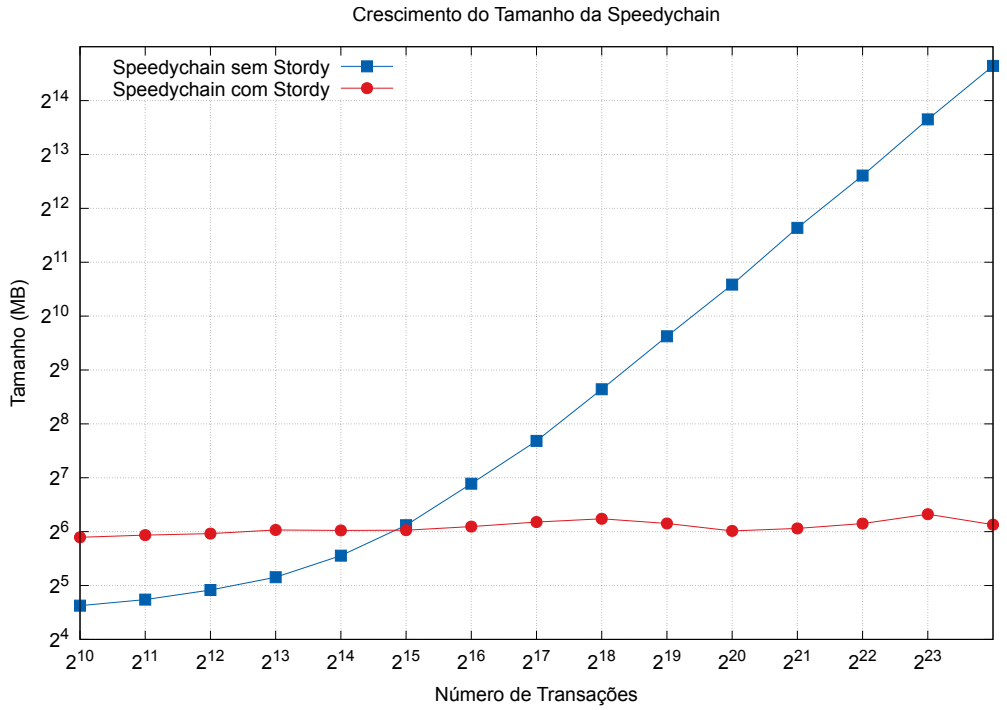

Figura 5.1 – Uso da memória volátil da Speedychain com e sem Stordy dependendo do número de transações já anexadas.

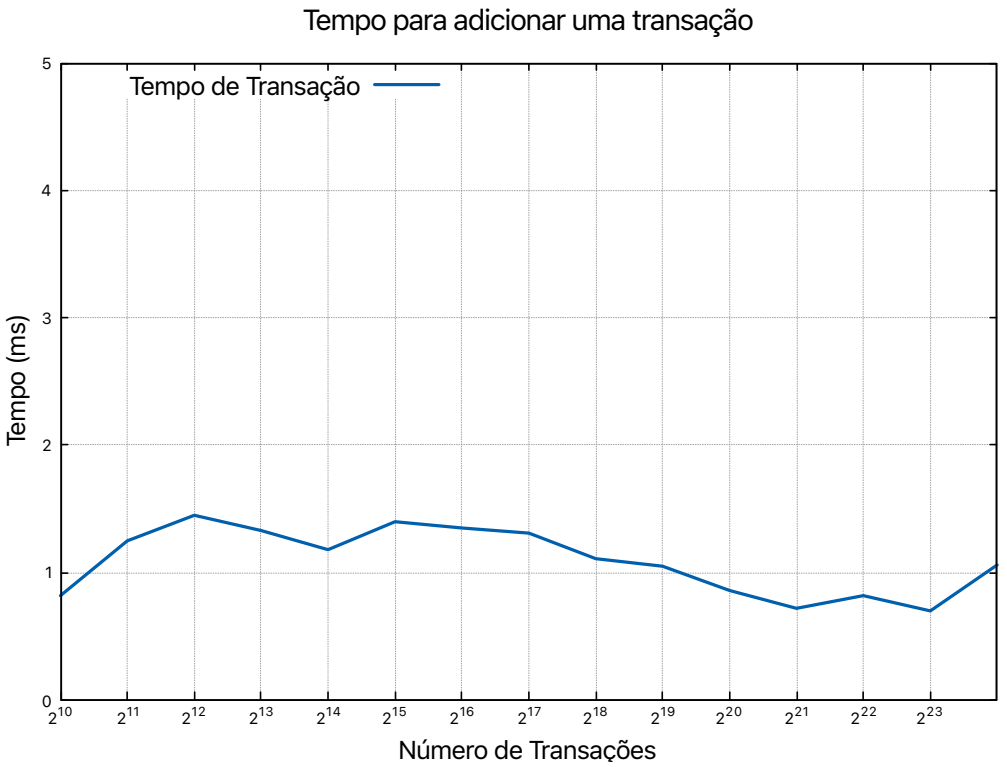

Figura 5.2 – Tempo para anexar uma nova transação considerando o número de transações já anexadas com Stordy.

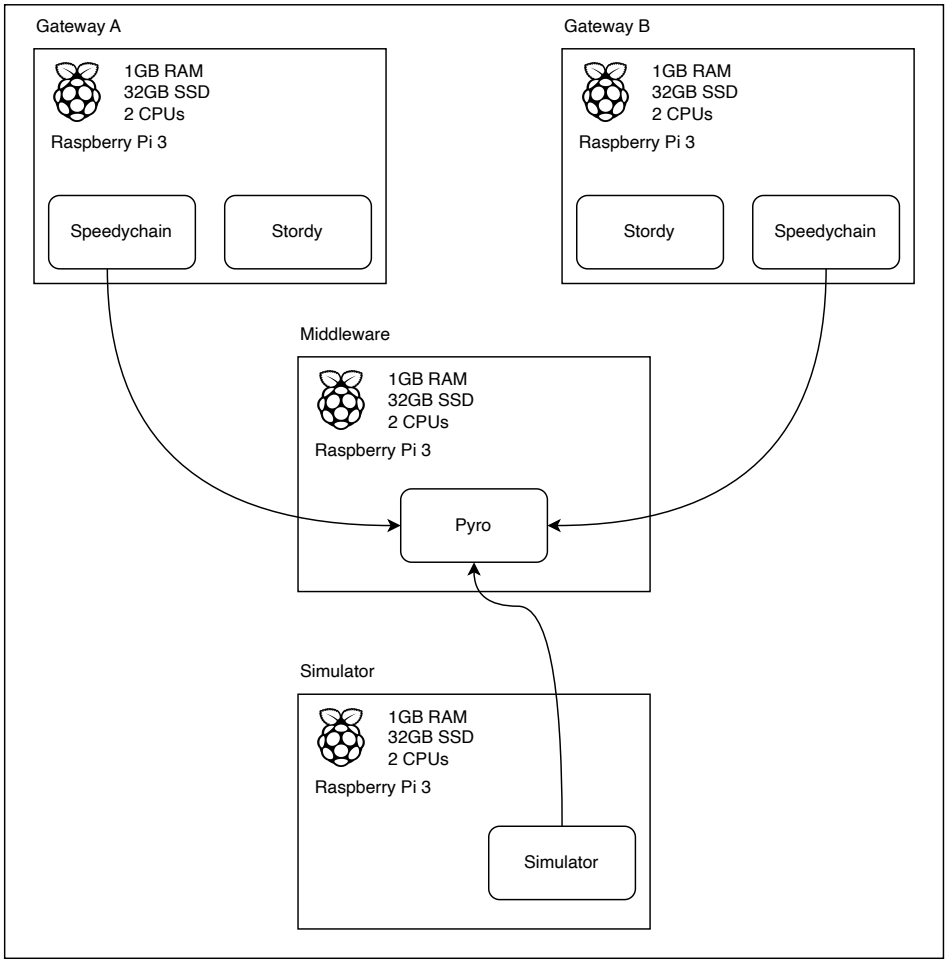

Figura 5.3 – Esquemático dos dispositivos utilizados no teste.

descrito por Silveira et al. (2024), demonstrou um consumo de memória proporcional ao número de transações, atingindo cerca de 728 MB após anexar 300.032 transações, o que resultou no travamento do sistema, como ilustrado na Figura 5.4. Por outro lado, a versão proposta neste trabalho apresentou um consumo inicial de aproximadamente 37 MB de *RAM*, permanecendo estável independentemente do número de transações adicionadas.

Adicionalmente, avaliamos o consumo de memória não volátil (*disk*) utilizando a versão da *blockchain* com o uso do Stordy. A Figura 5.5 apresenta o consumo de armazenamento no cartão *SD*, que cresceu proporcionalmente ao número de transações anexadas à cadeia. Para 262.144 transações, o consumo foi de aproximadamente 120 MB. Esse comportamento constante pode ser explicado pelo tamanho médio de 470 bytes por transação, que é diretamente influenciado pelas informações simuladas utilizadas nos testes.

No segundo experimento, investigamos o tempo necessário para adicionar novas transações à cadeia, incluindo tanto a operação local no módulo de armazenamento quanto a disseminação da transação entre os *gateways*. A Figura 5.6 apresenta o di-

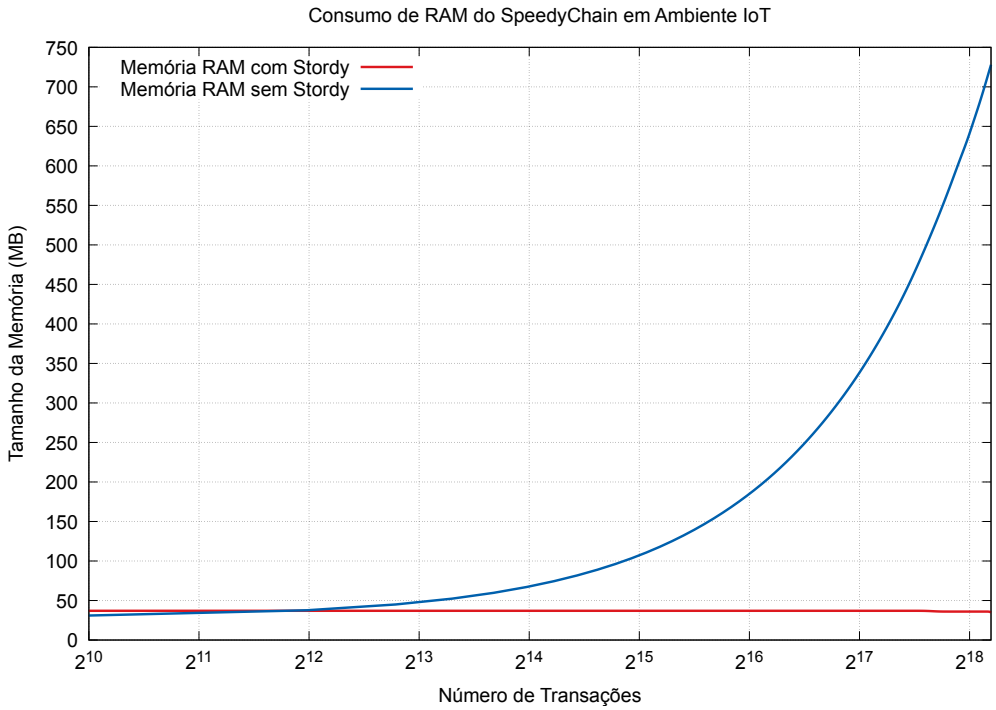

Figura 5.4 – Consumo de memória *RAM* em dispositivo IoT.

agrama de sequência do processo de inserção de uma transação, ilustrando as etapas envolvidas e as trocas de mensagens entre os dispositivos.

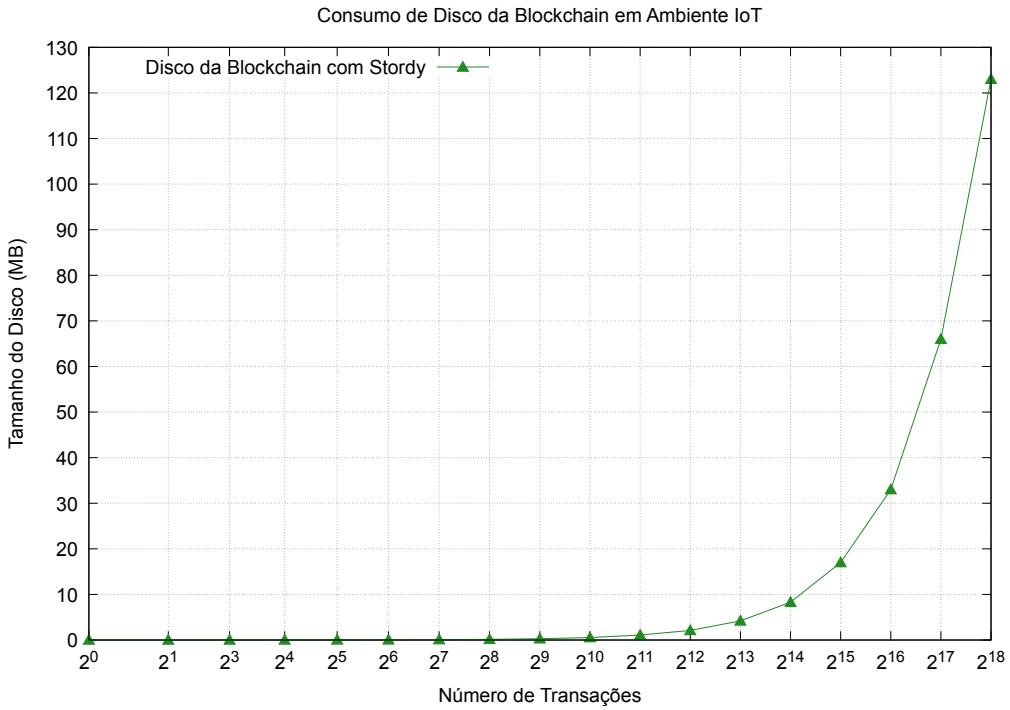

Figura 5.5 – Consumo de memória não volátil em dispositivo *IoT*.

Os resultados, apresentados na Figura 5.7, indicam que a versão sem o módulo de armazenamento apresentou um aumento significativo no tempo necessário para adicionar novas transações à medida que o volume de transações no bloco aumentava. Em contrapartida, a versão com o Stordy demonstrou tempos de adição consistentes, mesmo em cenários de alta demanda, com variações mínimas atribuídas ao *Sistema Operacional* (SO). Nesta versão, o tempo médio para adicionar uma transação foi de 14,29 ms, com um desvio padrão de 0,765 ms.

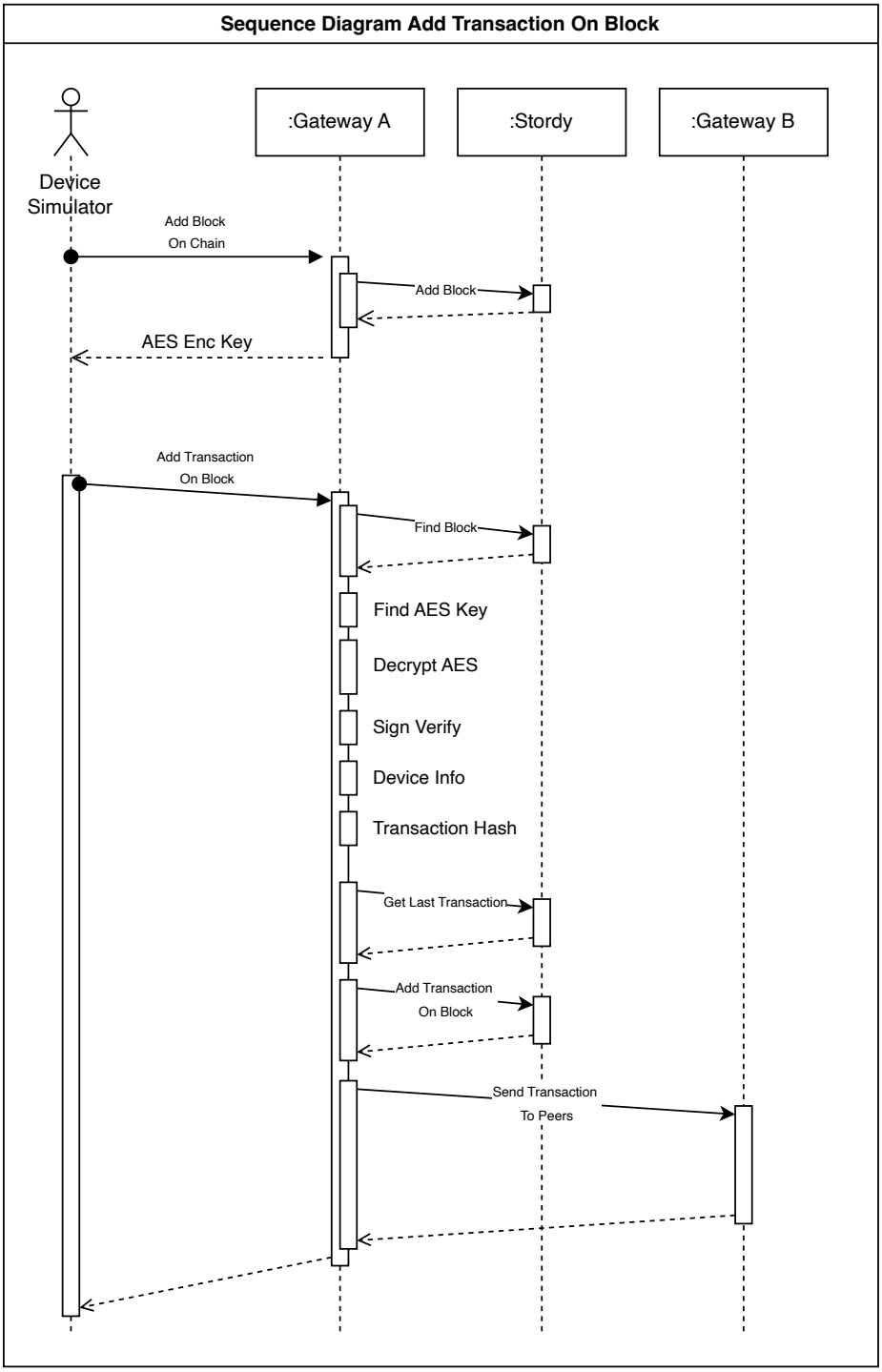

Figura 5.6 – Diagrama de sequência do processo de inserção de uma transação à block-chain.

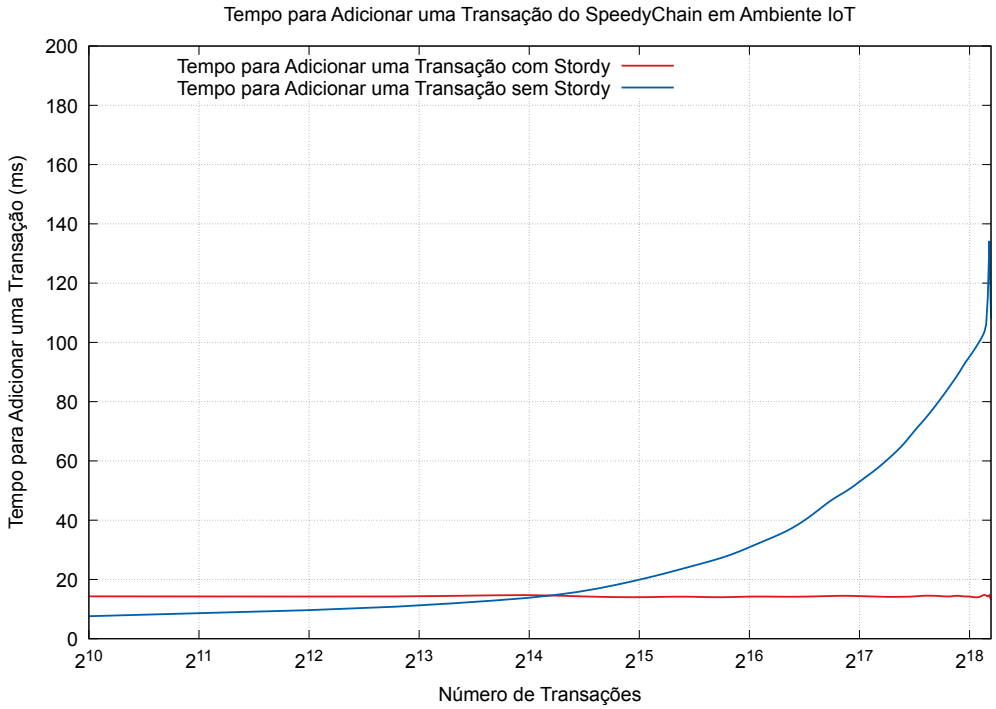

Figura 5.7 – Tempo para adição de uma nova transação à cadeia.

6. DISCUSSÕES

O objetivo deste capítulo é realizar uma análise crítica dos resultados apresentados na Seção 5, avaliando sua relevância frente aos desafios enfrentados e às contribuições propostas por este trabalho. Essa análise busca não apenas validar a eficácia da solução no contexto de aplicações IoT, mas também identificar como os resultados obtidos podem ser interpretados em termos de impacto prático e acadêmico. Além disso, são discutidas as implicações práticas da solução, destacando seus benefícios em cenários reais, bem como suas limitações e os principais gargalos observados durante os experimentos. Por fim, são exploradas oportunidades para melhorias futuras, considerando tanto a evolução tecnológica quanto possíveis novos domínios de aplicação que podem ampliar o alcance da solução proposta.

6.1 Relevância do Consumo de Memória Volátil

A redução no consumo de memória volátil é um dos aspectos mais críticos abordados neste trabalho, especialmente no contexto de aplicações IoT, onde dispositivos operam com restrições severas de *hardware*. Os experimentos realizados e apresentados nas Seções 5.1 e 5.2 evidenciaram que a utilização do Stordy na Speedychain permite manter o consumo de *RAM* constante, independentemente do número de transações adicionadas à cadeia. Essa característica é fundamental para garantir a operação contínua de sistemas baseados em dispositivos de baixo custo, como *Raspberry Pi 3*, que possuem apenas 1 GB de memória disponível, como foi possível observar nos experimentos da Seção 5.2.

Conforme ilustrado na Figura 5.1, a versão da *blockchain* sem o módulo de armazenamento apresentou um aumento drástico no consumo de *RAM* à medida que novas transações eram inseridas, culminando em gargalos operacionais e, eventualmente, no travamento do sistema quando a memória foi esgotada. Por outro lado, a versão com o Stordy manteve um consumo estável de memória volátil, mesmo após a inserção de mais de 8 milhões de transações. Essa estabilidade permite que dispositivos IoT, geralmente limitados a configurações de *hardware* simplificadas, possam operar de forma confiável e eficiente.

A relevância prática dessa melhoria é evidente quando consideramos cenários reais, como redes de sensores distribuídos, monitoramento ambiental ou dispositivos de saúde conectados. Nesses contextos, a escalabilidade é frequentemente comprometida pela sobrecarga de *RAM* causada pelo crescimento da *blockchain*. A solução proposta neste trabalho mitiga esse problema, viabilizando a utilização de *blockchains* em ambientes que, historicamente, eram considerados inviáveis devido às limitações de recur-

sos. Além disso, a análise comparativa entre os dois cenários demonstrou que o uso de memória volátil na versão com o Stordy não apenas se mantém constante, mas também é significativamente menor do que a versão anterior ao lidar com grandes volumes de transações. Esse comportamento reduz a necessidade de atualizações frequentes de *hardware*, prolongando a vida útil dos dispositivos e contribuindo para a sustentabilidade tecnológica.

Entretanto, apesar dos resultados positivos, algumas limitações devem ser consideradas. Por exemplo, o módulo de armazenamento depende de um sistema de arquivos eficiente para garantir que as operações sejam realizadas dentro do tempo esperado, especialmente em dispositivos IoT que utilizam cartões *SD*, conhecidos por sua limitada taxa de leitura/escrita. Trabalhos futuros poderiam explorar alternativas de armazenamento, como memórias *flash* integradas ou soluções híbridas que combinem armazenamento *on-chain* e *off-chain*, ampliando ainda mais a eficiência e a aplicabilidade da solução.

6.2 Eficiência Temporal na Inserção de Transações

A eficiência temporal na inserção de transações é um aspecto crucial para o desempenho de sistemas *blockchain* em ambientes IoT, onde a capacidade de processar grandes volumes de transações de maneira consistente e eficiente é essencial. Os experimentos apresentados na Seção 5.2 evidenciaram que o módulo de armazenamento Stordy mantém estabilidade no tempo necessário para a inserção de novas transações, independentemente do número de transações já armazenadas em um bloco. Conforme ilustrado na Figura 5.2, o tempo médio para a inserção de uma transação no módulo de armazenamento permaneceu consistentemente abaixo de 2 milissegundos em todas as condições testadas, com um desvio padrão de apenas 0,35 milissegundos. Essa constância reforça a capacidade do Stordy de lidar com operações em larga escala sem impacto significativo na eficiência temporal.

Durante a análise comparativa com a versão da *blockchain* sem o módulo de armazenamento, duas hipóteses foram formuladas: a primeira sugeria que a versão sem o Stordy seria mais eficiente, uma vez que todas as informações estavam armazenadas diretamente na *RAM*, permitindo acessos teoricamente mais rápidos; a segunda hipótese, no entanto, considerava que o aumento no volume de informações armazenadas na *RAM* poderia levar a tempos de acesso progressivamente mais lentos, devido à sobrecarga no gerenciamento dessas informações, tornando o uso do Stordy vantajoso para a inserção de transações em cenários de grande escala.

Os resultados apresentados na Seção 5.2, ilustrados pela Figura 5.7, corroboram a segunda hipótese, isso pois à medida que o número de transações em um bloco aumenta, o tempo necessário para a inserção de novas transações na versão sem o Stordy

cresce progressivamente, enquanto a versão com o módulo de armazenamento mantém uma estabilidade temporal significativa. Essa consistência reflete a eficácia do Stordy na execução de operações internas, como a busca por um bloco, a obtenção da última transação anexada e a atualização do bloco com a nova transação, mesmo em cenários de alta densidade de informações. A habilidade do Stordy em gerenciar essas operações de forma eficiente demonstra sua adequação para lidar com cenários de escalabilidade exigente.

Essa constatação tem implicações práticas significativas, especialmente para aplicações IoT, onde a escalabilidade e a previsibilidade são fatores críticos. Dispositivos de baixo custo frequentemente operam em limites restritos de memória e processamento, e a estabilidade proporcionada pelo Stordy garante que o desempenho não seja comprometido mesmo em situações de alta demanda. Esses resultados consolidam o módulo de armazenamento como uma alternativa viável e eficiente para superar os desafios impostos por sistemas *blockchain* em ambientes de recursos limitados.

6.3 Impacto no Consumo de Recursos em IoT

A análise dos resultados experimentais evidenciou que a solução proposta é altamente eficaz em lidar com os desafios típicos de dispositivos IoT, como restrições severas de *hardware* e a necessidade de operar com volumes crescentes de dados em tempo real. A incorporação do módulo de armazenamento Stordy mostrou-se fundamental para superar essas limitações, oferecendo uma abordagem que alia eficiência, previsibilidade e escalabilidade.

Um dos aspectos mais significativos é como a solução impacta diretamente a viabilidade de aplicações IoT em cenários reais. Redes de sensores ambientais, por exemplo, demandam dispositivos que operem de forma autônoma por períodos prolongados, frequentemente sem supervisão constante. A estabilidade temporal e a otimização no uso de recursos promovidas pelo Stordy permitem que esses dispositivos mantenham sua funcionalidade sem a necessidade de intervenções frequentes, aumentando sua vida útil operacional. Esse mesmo benefício pode ser observado em contextos industriais ou de monitoramento de saúde, onde a confiabilidade e a previsibilidade do sistema são essenciais para evitar falhas críticas.

A compatibilidade da solução com dispositivos de baixo custo, como *Raspberry Pi 3*, também representa um avanço significativo no uso de *blockchains* em sistemas distribuídos. A capacidade de operar em plataformas acessíveis democratiza a tecnologia, possibilitando sua adoção por *startups* e pequenas empresas que antes eram limitadas pelo alto custo associado a soluções mais robustas. Esse fator não apenas amplia o al-

cance da tecnologia, mas também promove sua aplicação em novas áreas que antes eram consideradas inviáveis.

Outro ponto relevante é o comportamento escalável do Stordy no gerenciamento de armazenamento não volátil. Conforme demonstrado na Figura 5.5, o consumo de memória não volátil permaneceu diretamente proporcional ao volume de transações inseridas, garantindo que o sistema possa operar de maneira sustentável mesmo em cenários de alta densidade de dados. Essa característica fortalece ainda mais a aplicação da solução em redes IoT que exigem escalabilidade e desempenho consistente.

Em síntese, os resultados obtidos consolidam a solução proposta como uma alternativa viável e eficiente para o uso de *blockchains* em dispositivos IoT, ampliando seu potencial de aplicação em cenários distribuídos e de recursos limitados. A combinação de acessibilidade, eficiência e escalabilidade oferece uma base sólida para a exploração de novas oportunidades no uso de *blockchains* em sistemas conectados.

6.4 Perspectivas para Armazenamento Híbrido e Off-Chain

Embora o módulo Stordy tenha demonstrado avanços significativos na redução do consumo de memória volátil e na eficiência temporal para sistemas *blockchain*, uma oportunidade promissora reside na adoção de arquiteturas híbridas que combinem armazenamento *on-chain* e *off-chain*. Essa abordagem poderia mitigar ainda mais as limitações de escalabilidade e custos computacionais em cenários com recursos restritos, como ambientes IoT.

A literatura destaca diversas implementações de armazenamento híbrido, como o uso de soluções baseadas em *Merkle Tree* e *Patricia Tree* para otimizar o armazenamento *on-chain*, enquanto dados de menor prioridade ou menos críticos são armazenados *off-chain* em *Distributed File System* (DFS), como *InterPlanetary File System* (IPFS) ou soluções baseadas em nuvem, conforme discutido na Seção 3. Esse modelo reduz a carga de dados diretamente armazenados na *blockchain*, aliviando os gargalos de desempenho sem comprometer a integridade das informações. No contexto do Stordy, a aplicação de estratégias híbridas poderia incluir:

(i) Armazenamento de Metadados On-Chain: apenas os metadados ou resumos (*hashes*) das transações seriam armazenados diretamente na cadeia de blocos, garantindo a rastreabilidade e a integridade. Os dados completos das transações poderiam ser mantidos em camadas *off-chain*, reduzindo significativamente o uso de espaço e recursos computacionais.

(ii) Integração com Sistemas Distribuídos de Arquivos: utilizar ferramentas como IPFS para armazenar transações menos acessadas, enquanto os dados de alta prioridade

ou frequentemente consultados permaneceriam acessíveis diretamente na *blockchain*. Essa abordagem permitiria otimizar a recuperação de dados sem sacrificar a escalabilidade.

(iii) Mecanismos de Indexação Avançados: a combinação de técnicas como a *Adaptive Balanced Merkle Tree (AB-M Tree)* para indexação e recuperação eficiente poderia complementar a estrutura híbrida, proporcionando maior agilidade nas operações de leitura e escrita Jia et al. (2021).

(iv) Flexibilidade e Sustentabilidade: o modelo híbrido, ao reduzir o armazenamento em dispositivos IoT, promoveria maior durabilidade desses dispositivos e menores custos de manutenção, uma vez que a dependência de *hardware* específico seria minimizada.

Além das vantagens técnicas, essa abordagem também abre caminho para uma melhor adequação a casos de uso emergentes, como em Cidades Inteligentes, monitoramento ambiental e logística, onde o volume de dados continua a crescer exponencialmente. A implementação de armazenamento híbrido permitiria escalar as soluções de forma sustentável, sem comprometer os princípios de descentralização e segurança inerentes às *blockchains*.

7. CONCLUSÃO

O conceito de *blockchain* expansível, exemplificado pela SpeedyChain, representa um avanço significativo no paradigma de armazenamento e recuperação de dados em redes descentralizadas, especialmente aquelas com recursos limitados. Esta pesquisa explorou essa abordagem promissora, propondo otimizações substanciais no consumo de memória volátil e na persistência de dados, elementos cruciais para a implementação eficaz dessa tecnologia em contextos como ambientes IoT e a gestão de dispositivos com restrições de recursos.

O módulo de armazenamento desenvolvido, denominado Stordy, desempenhou um papel fundamental na mitigação de gargalos relacionados ao consumo de memória. Isso se mostrou particularmente evidente em cenários de alta demanda transacional, nos quais o uso de recursos foi significativamente otimizado. Tal aprimoramento permitiu que a *blockchain* operasse de forma ininterrupta e confiável, mesmo em dispositivos com capacidade limitada, conforme evidenciado nas avaliações apresentadas no Capítulo 5. Adicionalmente, a adoção de uma estrutura de serialização eficiente, baseada no Protobuf, e a incorporação de mecanismos de controle inspirados no *Recursive-Length Prefix* da Ethereum, asseguraram que os dados fossem armazenados e transmitidos de maneira compacta, preservando a integridade e o desempenho do sistema.

Os resultados experimentais revelaram que a implementação do módulo de armazenamento não apenas possibilitou a redução do consumo de RAM, mas também manteve um tempo de inserção de transações estável, mesmo diante do aumento do volume de dados. Esse comportamento é essencial para aplicações de IoT, nas quais a previsibilidade do tempo de resposta é um requisito crítico. Com um consumo médio de 1,09 ms por transação e a escalabilidade demonstrada na persistência dos dados em memória não volátil, a solução proposta se destacou pelo equilíbrio entre desempenho e economia de recursos.

Nos testes realizados em ambientes IoT, utilizando Raspberry Pi 3, a robustez da solução foi confirmada em condições reais de operação. Enquanto a versão original da *blockchain* apresentou gargalos significativos ao atingir um alto volume de transações, a versão com o Stordy manteve-se estável, evidenciando o impacto positivo das otimizações propostas. O consumo de memória não volátil também apresentou um crescimento linear proporcional ao volume de transações, corroborando os dados de teste utilizados. Isso indica que a solução pode ser escalada sem comprometer o desempenho em cenários mais amplos.

Outro ponto relevante foi a introdução de um serviço de indexação de dados, que possibilita a recuperação rápida de informações cruciais na cadeia. Essa abordagem é especialmente relevante considerando o crescimento exponencial do volume de dados

em aplicações de *blockchain* e IoT. A capacidade de acessar dados específicos sem a necessidade de percorrer toda a cadeia representa um avanço significativo em termos de eficiência. No entanto, como qualquer solução, o modelo apresentado nesta pesquisa também enfrenta desafios. Questões como escalabilidade em larga escala e a integração com outras tecnologias, como sistemas *off-chain* ou infraestruturas de armazenamento distribuído, como o IPFS, demandam investigações mais aprofundadas.

Sugere-se que pesquisas futuras explorem mecanismos de armazenamento híbrido, nos quais seja possível armazenar em disco apenas as informações essenciais para acesso rápido, enquanto as informações menos utilizadas sejam distribuídas entre os participantes, similar ao SE-Chain, conforme detalhado na Seção 3.1. Adicionalmente, a implementação de soluções de armazenamento externo à cadeia, como a proposta no trabalho intitulado "*Adaptive Storage Optimization Scheme for Blockchain-IIoT Applications Using Deep Reinforcement Learning*"(Seção 3.5), pode gerar impactos positivos em ambientes IoT, nos quais os dispositivos geralmente possuem capacidade de disco limitada.

Durante o desenvolvimento deste trabalho, um artigo científico foi produzido e submetido à conferência *World Conference on Information Systems and Technologies* (*WorldCIST*). O artigo, intitulado "*Efficient Data Retrieving Mechanism for Appendable-block Blockchain*", foi aceito para publicação, embora ainda não tenha sido publicado oficialmente[1].

[1] Abril/2025

REFERÊNCIAS BIBLIOGRÁFICAS

Akcora, C. G., Gel, Y. R., and Kantarcioglu, M. (2022). Blockchain networks: Data structures of bitcoin, monero, zcash, ethereum, ripple, and iota. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 12, pp. e1436.

Akrasi-Mensah, N. K., Agbemenu, A. S., Nunoo-Mensah, H., Tchao, E. T., Ahmed, A.-R., Keelson, E., Sikora, A., Welte, D., and Kponyo, J. J. (2022). Adaptive storage optimization scheme for blockchain-iiot applications using deep reinforcement learning. *IEEE Access*, vol. 11, pp. 1372–1385.

Albeanu, G. (2017). Blockchain technology and education. In: *The 12th International Conference on Virtual Learning (ICVL)*, pp. 271–275.

Andrew, S. T. and Herbert, B. (2015). *Modern operating systems*. Pearson Education.

Bermeo-Almeida, O., Cardenas-Rodriguez, M., Samaniego-Cobo, T., Ferruzola-Gomez, E., Cabezas-Cabezas, R., and Bazan-Vera, W. (2018). Blockchain in agriculture: A systematic literature review. In: *Technologies and Innovation: 4th International Conference (CITI), Guayaquil, Ecuador, November 6-9, Proceedings 4*, pp. 44–56. Springer.

Bonneau, J., Miller, A., Clark, J., Narayanan, A., Kroll, J. A., and Felten, E. W. (2015). Sok: Research perspectives and challenges for bitcoin and cryptocurrencies. In: *IEEE Symposium on Security and Privacy*, pp. 104–121.

Bray, T., Paoli, J., Sperberg-McQueen, C. M., Maler, E., and Yergeau, F. (1997). Extensible markup language (xml). *World Wide Web Journal*, vol. 2, pp. 27–66.

Buterin, V. (2017). Ethereum: Platform review and use cases.

Crosby, M., Pattanayak, P., Verma, S., and Kalyanaraman, V. (2016). Blockchain technology: Beyond bitcoin. *Applied Innovation*, vol. 2, pp. 6–10.

Currier, C. (2022). Protocol buffers. In: *Mobile Forensics–The File Format Handbook: Common File Formats and File Systems Used in Mobile Devices*, pp. 223–260. Springer.

Davis, K., Peabody, B., and Leach, P. (2024). RFC 9562: Universally Unique IDentifiers (UUIDs). Relatório Técnico, Internet Engineering Task Force (IETF), USA.

de Arruda, E. H., Lunardi, R. C., Nunes, H. C., Zorzo, A. F., and Michelin, R. A. (2020). Appendable-block blockchain evaluation over geographically-distributed iot networks. In: *IEEE International Black Sea Conference on Communications and Networking (BlackSeaCom)*, pp. 1–6. IEEE.

Dorri, A., Kanhere, S. S., and Jurdak, R. (2016). Blockchain in internet of things: Challenges and solutions.

Dorri, A., Kanhere, S. S., and Jurdak, R. (2017). Towards an optimized blockchain for iot. In: *2nd International Conference on Internet-of-Things Design and Implementation (IOTDI)*, pp. 173–178.

Ferdous, M. S., Chowdhury, M. J. M., Hoque, M. A., and Colman, A. (2020). Blockchain consensus algorithms: A survey.

Gupta, S. P., Gupta, K., and Chandavarkar, B. (2021). The role of cryptography in cryptocurrency. In: *2nd International Conference on Secure Cyber Computing and Communications (ICSCCC)*, pp. 273–278. IEEE.

Ji, Q., Bouri, E., Gupta, R., and Roubaud, D. (2018). Network causality structures among bitcoin and other financial assets: A directed acyclic graph approach. *The Quarterly Review of Economics and Finance*, vol. 70, pp. 203–213.

Jia, D.-Y., Xin, J.-C., Wang, Z.-Q., Lei, H., and Wang, G.-R. (2021). Se-chain: A scalable storage and efficient retrieval model for blockchain. *Journal of Computer Science and Technology*, vol. 36, pp. 693–706.

King, S. and Nadal, S. (2012). Ppcoin: Peer-to-peer crypto-currency with proof-of-stake.

Kuznetsov, O., Kanonik, D., Rusnak, A., Yezhov, A., Domin, O., and Kuznetsova, K. (2024). Adaptive merkle trees for enhanced blockchain scalability. *Internet of Things*, vol. 27, pp. 101315.

Levandoski, J. J., Lomet, D. B., and Sengupta, S. (2013). The bw-tree: A b-tree for new hardware platforms. In: *IEEE 29th International Conference on Data Engineering (ICDE)*, pp. 302–313. IEEE.

Liu, H., Luo, X., Liu, H., and Xia, X. (2021). Merkle tree: A fundamental component of blockchains. In: *International Conference on Electronic Information Engineering and Computer Science (EIECS)*, pp. 556–561.

Liu, M., Yu, F. R., Teng, Y., Leung, V. C., and Song, M. (2019). Performance optimization for blockchain-enabled industrial internet of things (iiot) systems: A deep reinforcement learning approach. *IEEE Transactions on Industrial Informatics*, vol. 15, pp. 3559–3570.

Lunardi, R. C., Alharby, M., Nunes, H. C., Zorzo, A. F., Dong, C., and van Moorsel, A. (2020). Context-based consensus for appendable-block blockchains. In: *IEEE international conference on blockchain (Blockchain)*, pp. 401–408. IEEE.

Lunardi, R. C., Michelin, R. A., Alharby, M., Dedeoglu, V., Nunes, H. C., Arruda, E., Zorzo, A. F., and van Moorsel, A. (2023). When blockchain meets smart cities: Opportunities, security and future research. In: *Blockchains: A Handbook on Fundamentals, Platforms and Applications*, pp. 423–463. Springer.

Lunardi, R. C., Michelin, R. A., Neu, C. V., Nunes, H. C., Zorzo, A. F., and Kanhere, S. S. (2019). Impact of consensus on appendable-block blockchain for iot. In: *16th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*, pp. 228–237.

Lunardi, R. C., Michelin, R. A., Neu, C. V., and Zorzo, A. F. (2018). Distributed access control on iot ledger-based architecture. In: *IEEE/IFIP Network Operations and Management Symposium (NOMS)*, pp. 1–7. IEEE.

Michelin, R. A., Dorri, A., Steger, M., Lunardi, R. C., Kanhere, S. S., Jurdak, R., and Zorzo, A. F. (2018). Speedychain: A framework for decoupling data from blockchain for smart cities. In: *15th EAI International Conference on Mobile and Ubiquitous Systems: Computing, networking and services*, pp. 145–154.

Nakamoto, S. (2008). Bitcoin: A peer-to-peer electronic cash system.

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

Omahs, et al. (2025). Recursive-length prefix (RLP) serialization. https://ethereum.org/en/developers/docs/data-structures-and-encoding/rlp/.

Pan, J., Wang, J., Hester, A., Alqerm, I., Liu, Y., and Zhao, Y. (2018). Edgechain: An edge-iot framework and prototype based on blockchain and smart contracts. *IEEE Internet of Things Journal*, vol. 6, pp. 4719–4732.

Pilkington, M. (2016). Blockchain technology: principles and applications. In: Olleros, F. X. and Zhegu, M., editores, *Research Handbook on Digital Transformations*, pp. 225–253. Edward Elgar Publishing.

Popić, S., Pezer, D., Mrazovac, B., and Teslić, N. (2016). Performance evaluation of using protocol buffers in the internet of things communication. In: *International Conference on Smart Systems and Technologies (SST)*, pp. 261–265.

Rifat Hossain, M., Nirob, F. A., Islam, A., Rakin, T. M., and Al-Amin, M. (2024). A comprehensive analysis of blockchain technology and consensus protocols across multilayered framework. *IEEE Access*, vol. 12, pp. 63087–63129.

Ritchie, D. M. and Thompson, K. (1974). The unix time-sharing system. *Communications of the ACM*, vol. 17, pp. 365–375.

Ritu, Arora, S., Bhardwaj, A., Kukkar, A., and Kaur, S. (2024). A comparative analysis of communication efficiency: Rest vs. grpc in microservice- based ecosystems. In: *International Conference on Emerging Innovations and Advanced Computing (INNOCOMP)*, pp. 621–626.

Seliem, M. and Elgazzar, K. (2019). Biomt: Blockchain for the internet of medical things. In: *IEEE International Black Sea Conference on Communications and Networking (BlackSeaCom)*, pp. 1–4. IEEE.

Silberschatz, A., Galvin, P., and Gagne, G. (2018). *Operating System Concepts*. Wiley.

Silveira, R. W., Lunardi, R. C., and Zorzo, A. F. (2024). Data Management in Appendable-Block Blockchains: A Case Study for IT Life-Cycle Management. In: Zaslavsky, A., Ning, Z., Kalogeraki, V., Georgakopoulos, D., and Chrysanthis, P. K., editores, *Mobile and Ubiquitous Systems: Computing, Networking and Services*, pp. 476–488, Cham. Springer Nature Switzerland.

Soni, D. and Makwana, A. (2017). A survey on MQTT: a protocol of internet of things (iot). In: *International conference on telecommunication, power analysis and computing techniques (ICTPACT)*, vol. 20, pp. 173–177.

Sumaray, A. and Makki, S. K. (2012). A comparison of data serialization formats for optimal efficiency on a mobile platform. In: *6th International Conference on Ubiquitous Information Management and Communication (ICUIMC)*, pp. 1–6.

Tschorsch, F. and Scheuermann, B. (2016). Bitcoin and beyond: A technical survey on decentralized digital currencies. *IEEE Communications Surveys & Tutorials*, vol. 18, pp. 2084–2123.

Viotti, J. C. and Kinderkhedia, M. (2022). A Survey of JSON-compatible Binary Serialization Specifications.

Yli-Huumo, J., Ko, D., Choi, S., Park, S., and Smolander, K. (2016). Where is current research on blockchain technology?—a systematic review. *PLOS ONE*, vol. 11, pp. e0163477.

Zhao, J. L., Fan, S., and Yan, J. (2016). Overview of business innovations and research opportunities in blockchain and introduction to the special issue. *Financial Innovation*, vol. 2, pp. 28.

Zheng, Z., Xie, S., Dai, H., Chen, X., and Wang, H. (2017). An overview of blockchain technology: Architecture, consensus, and future trends. In: *IEEE international congress on big data (BigData congress)*, pp. 557–564. IEEE.

Zorzo, A. F., Nunes, H. C., Lunardi, R. C., Michelin, R. A., and Kanhere, S. S. (2018). Dependable iot using blockchain-based technology. In: *Eighth Latin-American Symposium on Dependable Computing (LADC)*, pp. 1–9. IEEE.

Pontifícia Universidade Católica do Rio Grande do Sul
Pró-Reitoria de Pesquisa e Pós-Graduação
Av. Ipiranga, 6681 – Prédio 1 – Térreo
Porto Alegre – RS – Brasil
Fone: (51) 3320-3513
E-mail: propesq@pucrs.br
Site: www.pucrs.br