Revista Eletrônica de Iniciação Científica em Computação, 2026, 24:1
ISSN: 3085-8461 • doi: 10.5753/reic.2026.7243

Artigo de Pesquisa/Research Paper

Análise de Desempenho entre Rollups Otimistas e Execução Nativa em Blockchains EVM

Performance Analysis between Optimistic Rollups and Native Execution in EVM Blockchains

Henrique Marlon Santos [Instituto de Tecnologia e Liderança (Inteli) | *henrique.santos@sou.inteli.edu.br*]
Emanuele Morais [Instituto de Tecnologia e Liderança (Inteli) |*emanuele.morais@sou.inteli.edu.br*]
Bryan Kano Ferreira [Instituto de Tecnologia e Liderança (Inteli) |*bryan.ferreira@prof.inteli.edu.br*]
Victor Takashi Hayashi [Universidade de São Paulo |*victor.hayashi@usp.br*]

Instituto de Tecnologia e Liderança (Inteli), São Paulo – SP – Brasil.

Resumo. Este artigo apresenta uma análise comparativa de desempenho entre rollups otimistas e execução nativa em Ethereum Virtual Machine (EVM). O estudo investiga as diferenças em termos de custo de gás, avaliando o impacto das soluções de Layer 2 na escalabilidade da blockchain Ethereum. Os resultados experimentais fornecem insights sobre os trade-offs entre execução on-chain tradicional e rollups otimistas, contribuindo para a compreensão das estratégias de escalabilidade em ambientes blockchain.

Abstract. This paper presents a comparative performance analysis between optimistic rollups and native execution on the Ethereum Virtual Machine (EVM). The study investigates differences in gas costs, evaluating the impact of Layer 2 solutions on Ethereum blockchain scalability. Experimental results provide insights into the trade-offs between traditional on-chain execution and optimistic rollups, contributing to the understanding of scalability strategies in blockchain environments.

Palavras-chave: Blockchain, Ethereum, EVM, Rollups Otimistas, Layer 2, Custo de Gás, Cartesi

Keywords: Blockchain, Ethereum, EVM, Optimistic Rollups, Layer 2, Gas Cost, Cartesi

Recebido/Received: 18 December 2025 • **Aceito/Accepted:** 10 April 2026 • **Publicado/Published:** 07 May 2026

1 Introdução

A tecnologia blockchain evoluiu de um mecanismo voltado à transferência de valor com regras de negócio definidas, introduzido pelo Bitcoin Nakamoto [2008], para uma infraestrutura programável, com muito mais espressividade para a implementação de casos de uso diversos, capaz de coordenar uma variedade maior de relações *peer-to-peer*. A blockchain do Ethereum, consolidou-se como um marco ao introduzir uma máquina virtual determinística (EVM) e contratos inteligentes, permitindo que aplicações descentralizadas (dApps) implementem regras de negócio arbitrárias e executáveis de forma transparente. Esse avanço ampliou significativamente o escopo de aplicações possíveis, a exemplo de finanças descentralizadas e mecanismos de governança, mas também tornou mais visível um gargalo estrutural: a camada base (Layer 1) precisa replicar e verificar a computação em todos os nós, o que impõe limites práticos de vazão e eleva o custo marginal de execução. Na EVM, a estratégia de *gas metering* implementada garante que cada operação da máquina virtual (opcode) tenha um custo de gas, esse custo é projetado para precificar o uso de recursos computacionais e de armazenamento e, assim, proteger a rede contra ataques, como aqueles relacionados a DoS (*Denial-of-Service*) Wood *et al*. [2014]. Contudo, conforme a demanda cresce na rede e a competição por inclusão em bloco se intensifica, o preço efetivo do gás se torna volátil e frequentemente elevado, penalizando especialmente aplicações com lógica mais pesada. Em particular, aplicações descentralizadas que dependem de ordenação, agregação, cálculos numéricos e operações iterativas tendem a se aproximar rapidamente de limites de gás por transação e a apresentar custos incompatíveis com uso em escala, de modo que o desafio de escalabilidade passa a ser também um problema de viabilidade econômica de classes inteiras de aplicações quando executadas nativamente na camada base.

Nesse contexto, soluções de Layer 2 ganharam relevância como estratégia para ampliar o poder computacional a disposição das aplicações da rede, sem abrir mão das garantias de segurança do Ethereum Buterin [2020]. Os rollups se destacam por deslocarem a execução para fora da cadeia principal (off-chain), mantendo na Layer 1 os elementos necessários para disponibilidade de dados e para a verificação de corretude da computação performada por meio de mecanismos denominados *proof systems*. Em linhas gerais, rollups buscam reduzir o custo da computação replicada por todos os nós, substituindo-a por um modelo no qual o Ethereum atua como "âncora de segurança" e camada de liquidação, enquanto a L2 realiza a maior parte do processamento e publica compromissos e dados essenciais. Os Rollups Otimistas, em particular, viabilizam isso por meio de um modelo no qual as transações são assumidas corretas por padrão, e a segurança é preservada via um mecanismo de contestação baseado em provas de fraude: resultados incorretos podem ser desafiados dentro de uma janela predefinida, levando à reversão de estados inválidos e à penalização de agentes maliciosos Canetti *et al*. [2011]. Embora esse desenho ofereça ganhos relevantes de escalabilidade, ele introduz trade-offs que afetam diretamente o custo do sistema, incluindo mudanças na noção prática no tempo de finalidade da aplicação, dependência de

disponibilidade de dados para reexecução e auditoria, e custos associados à publicação on-chain de dados. Por isso, comparar execução nativa e execução via rollups exige analisar não apenas "custo por transação", mas a recomposição do custo total entre computação e dados, bem como o impacto dessas escolhas sob cargas computacionalmente intensivas.

Tendo em vista a análise desses trade-offs entre execução nativa EVM em comparação com soluções de Rollups Otimistas, este estudo adota como caso de uso um Mercado de Capitais de Dívida (*Debt Capital Market*), no qual a execução envolve (i) ordenação e seleção de vetores de ofertas, (ii) cálculos de taxas de juros e alocações, (iii) regras de negócio que podem exigir iterações e agregações, operações potencialmente onerosas na execução nativa da EVM, tanto em custo direto de gás quanto em risco de atingir limites de execução por bloco. Mais especificamente, comparamos a execução direta na EVM com uma implementação em Rollups Otimistas utilizando a Cartesi Virtual Machine Teixeira *et al*. [2018], que permite executar lógica de negócio em um ambiente Linux determinístico e publicar na camada base (Layer 1) os artefatos mínimos necessários para integridade e disponibilidade da aplicação. A análise é conduzida sob uma ótica de desempenho centrada em custo: medimos e contrastamos o consumo de gás associado às operações do caso de uso e discutimos como a decomposição entre computação off-chain e publicação on-chain afeta o custo total e os trade-offs arquiteturais e de segurança. O estudo busca contribuir com três frentes principais: (i) uma medição empírica do custo de execução nativa na EVM para um caso de uso computacionalmente intensivo, identificando gargalos e componentes dominantes do consumo de gás; (ii) uma análise comparativa com uma implementação baseada em um Rollup Otimista, mais especificamente um que implementa uma microarquitetura RISC-V na sua camada de execução, evidenciando em quais condições a execução em Layer 2 reduz custos e como o perfil de custo se desloca entre computação e dados publicados on-chain; e (iii) uma discussão sobre implicações arquiteturais para desenvolvedores de aplicações descentralizadas que consideram migrar lógica de negócio para ambientes off-chain verificáveis.

2 Fundamentação Teórica

2.1 Ethereum Virtual Machine (EVM)

A *Ethereum Virtual Machine* (EVM) é o ambiente de execução determinístico responsável por processar transações e contratos inteligentes na blockchain Ethereum. Cada nó completo da rede incorpora uma instância da EVM e executa localmente as instruções de cada transação, garantindo que todos os participantes da rede cheguem a um consenso sobre o estado global do sistema Antonopoulos and Wood [2018]; Dannen [2017]. A EVM define, portanto, a função de transição de estado da Ethereum, sendo um componente central para a segurança e a verificabilidade do protocolo Wood *et al*. [2014]. Formalizada no *Ethereum Yellow Paper* Wood *et al*. [2014], a EVM é descrita como uma máquina de pilha quase Turing-completa. O termo "quase" decorre da introdução explícita de um mecanismo de limitação de recursos, o *gas*, que impõe um custo econômico à execução de instruções e assegura a terminação dos programas, prevenindo loops infinitos e ataques de negação de serviço Wood *et al*. [2014]; Zheng *et al*. [2020]. Esse modelo de execução controlada é fundamental para a operação segura de contratos inteligentes em um ambiente descentralizado e adversarial.

Arquiteturalmente, a EVM é baseada em uma máquina de pilha do tipo LIFO, com capacidade máxima de 1024 entradas, operando sobre unidades de 256 bits Antonopoulos and Wood [2018]. A adoção de 256 bits está alinhada às necessidades criptográficas do ecossistema Ethereum, facilitando operações como hashing, assinaturas digitais e aritmética modular utilizadas em primitivas de segurança e lógica de contratos Dannen [2017]. Além da pilha, a EVM dispõe de uma memória volátil linear, indexada por bytes e expandida sob demanda durante a execução, bem como de um armazenamento persistente (*storage*) estruturado como um mapa chave–valor associado a cada conta inteligente, que integra o estado global da blockchain Wood *et al*. [2014]; Zheng *et al*. [2020]. O código do contrato é armazenado separadamente e permanece imutável durante a execução, caracterizando uma arquitetura do tipo Harvard modificada, adotada por razões de segurança e previsibilidade Wood *et al*. [2014].

A execução na EVM é estritamente determinística em relação ao estado global, dada uma mesma entrada, composta pelo estado anterior e pela transação, todos os nós honestos que executam o mesmo conjunto de instruções produzem exatamente o mesmo estado final e os mesmos valores de retorno Wood *et al*. [2014]. Esse determinismo é essencial para o consenso distribuído, pois permite que a rede valide resultados de forma independente sem necessidade de comunicação adicional durante a execução Buterin [2014]. Cada instrução da EVM (*opcode*) resulta em modificações previsíveis na pilha, na memória ou no *storage*, consumindo uma quantidade específica de *gas* definida na especificação formal Wood *et al*. [2014]; Antonopoulos and Wood [2018]. Ao operar em um modelo de execução isolado e mediado por chamadas de mensagem, a EVM impõe fronteiras claras de interação entre contratos e com o ambiente externo, sustentando propriedades de verificabilidade e auditabilidade, ainda que à custa de limitações de desempenho e escalabilidade na execução on-chain Dannen [2017]; Zheng *et al*. [2020].

2.2 Custo de Gas e Limitações da EVM

O modelo de *gas* da Ethereum foi concebido como um mecanismo econômico para precificar o consumo de recursos computacionais e de armazenamento, com o objetivo de prevenir ataques de negação de serviço e incentivar a execução eficiente de contratos inteligentes em um ambiente descentralizado Wood *et al*. [2014]; Antonopoulos and Wood [2018]. Nesse modelo, cada instrução da Ethereum Virtual Machine (EVM) possui um custo predeterminado em unidades de *gas*, aproximadamente proporcional à carga computacional, ao impacto sobre o estado global ou ao custo de entrada/saída imposto aos nós validadores Wood *et al*. [2014]. Em particular, operações aritméticas simples e manipulações de pilha tendem a consumir poucas unidades de *gas*, enquanto instruções associadas a acesso e modificação de armazenamento persistente, bem como operações criptográficas, apresentam custos significativamente mais elevados Antonopoulos and Wood [2018]; Dannen [2017]. Antes da execução de uma transação, o emissor deve especificar tanto o limite máximo de

gas que está disposto a consumir quanto o preço por unidade de *gas* (*gas price*). Caso o *gas* fornecido seja esgotado antes da conclusão da execução, a transação é revertida, mas o *gas* já consumido não é reembolsado, assegurando que contratos maliciosos ou não terminantes não explorem recursos da rede sem custo econômico Wood *et al*. [2014].

Esse arranjo economicamente racional impõe restrições diretas à complexidade algorítmica dos contratos que podem ser executados de forma viável on-chain. Algoritmos computacionalmente intensivos, como ordenação de grandes volumes de dados ou cálculos matemáticos sofisticados, tornam-se rapidamente proibitivos do ponto de vista de custo, uma vez que demandariam milhões de instruções da EVM Antonopoulos and Wood [2018]. Além disso, a existência de um limite máximo de *gas* por bloco restringe a soma total de computação que pode ser incluída em cada bloco. Após a introdução da EIP-1559, esse limite passou a operar em torno de um alvo de aproximadamente 15 milhões de unidades de *gas*, com possibilidade de variação até cerca de 30 milhões em situações de pico, impondo um teto rígido ao throughput da camada base Buterin *et al*. [2019]; Wood *et al*. [2014]. Como consequência, o desempenho histórico da Ethereum na Camada 1 permaneceu limitado a uma ordem de grandeza de aproximadamente 12 a 15 transações por segundo, variando conforme o tipo de transação e a complexidade computacional envolvida, com picos teóricos próximos a 20 TPS em condições ideais Vukolić [2015]; Antonopoulos and Wood [2018]. Embora tal capacidade seja suficiente para determinados casos de uso, ela é várias ordens de magnitude inferior à exigida por aplicações de escala global. Em períodos de alta demanda, a competição por espaço em bloco intensifica-se, resultando em elevação significativa do preço do *gas* e, consequentemente, em custos de transação elevados, que inviabilizam microtransações e afetam a usabilidade de aplicações descentralizadas mais complexas Antonopoulos and Wood [2018]; Dannen [2017].

Em síntese, embora o modelo de *gas* seja fundamental para garantir segurança, vivacidade e previsibilidade econômica da rede, ele impõe limitações severas à escalabilidade da execução on-chain. O conhecido trilema da blockchain, conforme discutido por Buterin, estabelece que descentralização, segurança e escalabilidade não podem ser simultaneamente maximizadas Vukolić [2015]; Buterin [2018]. Nesse contexto, a Ethereum priorizou segurança e descentralização em sua camada base, aceitando menor *throughput* e maior latência, o que motivou a adoção progressiva de arquiteturas de escalabilidade em Camada 2 como estratégia predominante para expansão do ecossistema.

2.3 Rollups Otimistas

Os *rollups otimistas* constituem uma classe de soluções de escalabilidade em Camada 2 que deslocam a execução de transações para fora da blockchain principal, publicando periodicamente na Camada 1 apenas os dados necessários para verificação, como os dados brutos das transações ou compromissos criptográficos (por exemplo, raízes de Merkle) que representam o novo estado resultante Kalodner *et al*. [2018]; Buterin [2021]. O qualificativo "otimista"decorre da suposição de validade adotada por padrão, os estados publicados são aceitos como corretos pela Camada 1 sem verificação imediata, sendo a correção garantida ex post por meio de mecanismos de contestação. A segurança dos rollups otimistas baseia-se em provas de fraude (*fraud proofs*). Após a submissão de um novo estado, abre-se uma janela de disputa, tipicamente de alguns dias, durante a qual qualquer participante pode contestar a validade da transição de estado publicada Alvarez *et al*. [2024]. Caso uma inconsistência seja identificada, o protocolo permite que um validador honesto apresente uma prova demonstrando que a execução off-chain diverge da semântica correta da máquina virtual subjacente, acionando um mecanismo de arbitragem on-chain.

Cada rollup otimista é ancorado por um ou mais contratos inteligentes na Camada 1 que atuam como árbitros do sistema. Esses contratos exigem que os agentes responsáveis por publicar novos estados, comumente denominados *sequencers* ou *proposers*, depositem garantias econômicas (*bonds*). Em caso de fraude comprovada, o estado inválido é descartado e o depósito do agente desonesto é parcialmente ou totalmente confiscado, sendo redistribuído ao desafiante honesto como incentivo econômico Kalodner *et al*. [2018]. A maioria das implementações modernas emprega protocolos interativos de verificação para reduzir o custo de adjudicação on-chain. Tipicamente, tais protocolos assumem a forma de um jogo de bissecção do histórico de execução, no qual o desafiante e o propositor dividem iterativamente a sequência de transições até isolar uma única instrução ou passo divergente, que pode então ser verificado diretamente na Camada 1 Nehab and Teixeira [2022]. Esse mecanismo garante que o custo on-chain de verificação permaneça baixo, mesmo quando a computação off-chain envolve milhões de instruções.

Esse modelo de verificação pós-fato permite que computações arbitrariamente complexas sejam realizadas fora da blockchain principal, pagando-se on-chain apenas pelos dados publicados e, eventualmente, pelo custo de resolução de disputas, que idealmente ocorre de forma rara Buterin [2021]. Na prática, soluções como *Arbitrum* e *Optimism* demonstram reduções substanciais nas taxas por transação em comparação à execução nativa na Camada 1, ao mesmo tempo em que herdam as garantias de segurança do Ethereum Kalodner *et al*. [2018]; Buterin [2021].

2.4 Cartesi Virtual Machine (CVM)

A *Cartesi Virtual Machine* (CVM) representa uma extensão do paradigma de rollups otimistas ao prover um ambiente de execução off-chain significativamente mais expressivo do que a *Ethereum Virtual Machine*, sem comprometer o determinismo necessário para verificação on-chain Teixeira *et al*. [2018]. Desenvolvida no contexto do projeto Cartesi, a CVM é baseada na arquitetura RISC-V de 64 bits e implementada como um emulador determinístico capaz de executar um sistema operacional Linux completo, possibilitando a reutilização de ferramentas, bibliotecas e linguagens amplamente consolidadas no ecossistema de software tradicional. Diferentemente da EVM, que opera com um conjunto restrito de opcodes, unidades de 256 bits e severas restrições impostas pelo modelo de *gas*, a CVM permite que desenvolvedores utilizem linguagens de programação convencionais, como C/C++, Rust, Python e Go, bem como bibliotecas padrão do ambiente Linux, para implementar a lógica de aplicações descentralizadas Teixeira *et al*. [2018]. Essa abordagem supera limitações de expressividade inerentes à Camada 1, tornando

viável a execução off-chain de computações que seriam economicamente ou tecnicamente impraticáveis em contratos inteligentes tradicionais.

O principal desafio ao ampliar a expressividade da execução off-chain consiste em preservar segurança e verificabilidade. A CVM aborda esse desafio por meio de garantias estritas de determinismo, dado um mesmo estado inicial, incluindo memória, registradores e imagem do sistema de arquivos, e as mesmas entradas, a execução produz sempre o mesmo resultado Teixeira *et al*. [2018]. Essa propriedade é essencial para a validação por provas de fraude, uma vez que permite que qualquer parte interessada reproduza a computação e verifique sua correção de forma independente. Para assegurar determinismo bit-a-bit em um ambiente Linux, o projeto a CVM emprega técnicas como o uso de um kernel customizado, a eliminação ou controle rigoroso de fontes de entropia e a restrição de interações não determinísticas com o ambiente externo Teixeira *et al*. [2018]. Em cenários de disputa, o protocolo de *Permissionless Refereed Tournaments*, proposto por Nehab et al., possibilita a refutação eficiente de computações off-chain potencialmente fraudulentas, mesmo quando estas envolvem milhões de instruções Nehab and Teixeira [2022]. Nesse protocolo, a verificação é estruturada como um torneio eliminatório, reduzindo progressivamente o espaço de divergência até um passo mínimo que pode ser verificado on-chain com custo controlado.

A Figura 1 ilustra como a CVM consegue combinar expressividade off-chain com verificabilidade on-chain: em vez de tentar "levar Linux para a L1", o protocolo reduz o que a blockchain precisa conhecer a um único *commitment* do estado, tipicamente a raiz de uma *merkle tree* (m), que autentica todo o espaço de endereçamento do estado da máquina (memória e dispositivos virtualizados). Assim, a execução off-chain pode operar sobre um estado rico e particionado (p.ex., RAM e diferentes regiões de flash associadas a sistemas de arquivos), enquanto a L1 retém apenas (m) como referência pública. Quando é necessário justificar um resultado, não se publica o estado inteiro: prova-se apenas a porção relevante por meio de um caminho da *merkle tree* e os seus respectivos de *siblings*, permitindo verificar que uma palavra (v) efetivamente pertence ao endereço (a) no estado representado por (m). De modo análogo, uma atualização local, substituir (v) por (v') induz um novo *commitment* (m') sem reexecutar ou revalidar todo o estado, capturando a intuição de que o estado pode ser "lido/escrito" de forma verificável via primitivas como (slice/splice) (e (read/write)). Essa estrutura é

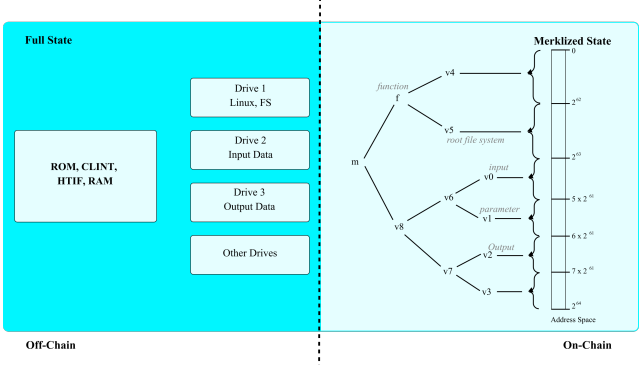

Figura 1. Mecanismo de integração e sincronização do estado da máquina entre os ambientes on-chain e off-chain

exatamente o que viabiliza a resolução eficiente de disputas: caso duas partes discordem do resultado de uma computação, elas conseguem restringir interativamente o desacordo até um único passo (ou uma região mínima) cuja transição pode ser checada em L1, tornando o custo on-chain proporcional apenas ao tamanho da prova e ao passo discrepante, e não ao total de instruções executadas off-chain Teixeira *et al*. [2018]; Nehab and Teixeira [2022].

Na prática, a CVM viabiliza que aplicações descentralizadas executem algoritmos computacionalmente intensivos, como ordenação de grandes conjuntos de dados, processamento de imagens ou tarefas de aprendizado de máquina, inteiramente off-chain, submetendo à blockchain apenas os dados essenciais de entrada e saída, bem como as provas criptográficas necessárias Teixeira *et al*. [2018]. O white-paper *The Core of Cartesi* demonstra que implementações equivalentes em C++ executadas na CVM podem superar em várias ordens de magnitude a eficiência de versões escritas em Solidity e executadas na EVM, justamente por não estarem sujeitas às restrições do modelo de *gas* e à simplicidade intencional da máquina virtual da Camada 1. Importante salientar que esse ganho expressivo de desempenho não compromete a segurança do sistema. Caso um agente publique um resultado inválido oriundo da execução na CVM, qualquer nó honesto pode reproduzir a computação e acionar um mecanismo de prova de fraude, identificando a divergência exata. Graças ao uso de protocolos de torneio, a verificação on-chain permanece eficiente, pois apenas o passo discrepante final precisa ser avaliado por um contrato inteligente na Ethereum, em vez de toda a computação off-chain Nehab and Teixeira [2022].

3 Revisão da Literatura

A tecnologia de registros distribuídos com consenso descentralizado foi inaugurada pelo Bitcoin Nakamoto [2008], posteriormente estendida pela plataforma Ethereum para suportar contratos inteligentes Turing-completos Buterin [2014], formalizados na especificação da *Ethereum Virtual Machine* (EVM) por Wood Wood *et al*. [2014]. Dadas as restrições de desempenho e custo inerentes à execução replicada na camada base Antonopoulos and Wood [2018], a comunidade passou a investir em arquiteturas de segunda camada (*layer 2*) para contornar essas limitações.

Esse desafio insere-se no chamado *trilema* de escalabilidade de blockchains, que postula uma tensão entre descentralização, segurança e escalabilidade Vukolić [2015]; Buterin [2018]. A solução direta de aumentar a capacidade computacional dos nós, por exemplo, exigindo hardware mais poderoso, comprometeria a descentralização da rede Vukolić [2015]; Buterin [2018]. Assim, a comunidade passou a investigar soluções de escalabilidade que movem a maior parte do processamento para fora da camada base, sem sacrificar a segurança fornecida pelo consenso global. Entre essas soluções, destacam-se os *rollups*, hoje considerados peça central na estratégia de escalabilidade do ecossistema Ethereum Buterin [2020]. Um rollup consiste essencialmente em uma instância de computação off-chain cuja integridade é garantida pela cadeia principal. A Ethereum mantém um *roadmap* centrado em rollups Buterin [2020], e o Vitalik propôs um guia abrangente detalhando esse conceito Buterin [2021]. Em linhas gerais,

transações dos usuários são registradas on-chain de forma compactada, por exemplo, como dados de chamada ou *calldata*, enquanto sua execução completa ocorre off-chain em uma camada de execução separada Buterin [2021]. Periódica ou continuamente, o estado resultante do rollup é consolidado na L1 por meio de checkpoints ou *state roots*. A grande questão passa a ser: como a camada base pode ter certeza da validade de um estado proposto pelo rollup sem reexecutar todas as transações? As respostas atuais bifurcam-se em duas categorias principais Buterin [2021]: (i) *rollups* de prova de validade (ZK rollups), que fornecem provas criptográficas sucintas de correção a cada atualização de estado, e (ii) *rollups otimistas*, que assumem temporariamente a validade das transações e recorrem a provas de fraude (*fraud proofs*) apenas se uma inconsistência for alegada.

Diferentes projetos de rollups otimistas implementaram variações desse sistema. A Arbitrum e a Optimism, por exemplo, optaram por manter compatibilidade total com a EVM, facilitando a migração de contratos existentes Kalodner *et al*. [2018]. Nessas plataformas, os contratos inteligentes são executados em máquinas virtuais equivalentes à EVM rodando off-chain, o que melhora o *throughput* mas não elimina todas as limitações de desempenho inerentes à EVM. Ainda que essas soluções elevem significativamente a capacidade de transações por segundo e reduzam as taxas *em relação à* camada 1, aplicações com lógica altamente intensiva (algoritmos de complexidade $O(n^2)$, computação científica, aprendizado de máquina, etc.) continuam impraticáveis se implementadas de forma ingênua na EVM, mesmo em um rollup. Em parte, isso ocorre porque o pior caso ainda exige uma verificação on-chain: se um cálculo extenso for disputado, o protocolo deve conseguir executá-lo (ainda que fragmentado) dentro dos limites de gas da L1. Assim, os projetos atuais de rollup geralmente impõem limites implícitos ao tamanho ou complexidade de cada lote de computação, para garantir que a verificação de fraude permaneça viável em última instância. Contudo, esses limites restringem o ganho de desempenho para casos de uso que demandam computação massiva, evidenciando uma lacuna para pesquisas adicionais em torno de execuções intensivas.

Uma abordagem promissora para contornar as limitações da EVM é prover ambientes de execução off-chain mais poderosos, integrados a um mecanismo de prova que os conecte à blockchain. Nesse contexto, destaca-se a *Cartesi Virtual Machine* (CVM), uma máquina virtual baseada em arquitetura RISC-V que roda um sistema operacional Linux completo de forma determinística Teixeira *et al*. [2018]. A proposta da CVM é permitir que dApps executem cálculos complexos com as mesmas ferramentas de desenvolvimento disponíveis na computação tradicional, por exemplo, bibliotecas Python ou C++, ao invés de ficarem restritos à linguagem Solidity e ao modelo de execução da EVM. A Cartesi encapsula essa execução em uma camada de rollup otimista: o estado inicial da máquina (por exemplo, o sistema de arquivos e entradas) é fixado, e após processar as transações o estado final produz um hash criptográfico de verificação Teixeira *et al*. [2018]. Apenas esse hash e eventuais saídas essenciais são registrados on-chain, representando de forma compacta o resultado da computação. Se nenhum participante contestar o resultado, a transição de estado off-chain é aceita tal como em outros rollups. Em caso de disputa, o Cartesi utiliza um protocolo de provas de fraude interativo especializado que permite reproduzir a execução da máquina Linux passo a passo dentro da blockchain, de modo semelhante ao jogo de verificação da EVM, porém suportando um conjunto de instruções muito mais amplo (as instruções RISC-V) Teixeira *et al*. [2018]. Para viabilizar isso, a CVM foi projetada visando determinismo, possibilitando dividir a execução de milhares ou milhões de instruções em segmentos menores verificáveis on-chain Nehab and Teixeira [2022]. Na prática, essa solução expande o leque de aplicações descentralizadas possíveis, mantendo garantias de correção semelhantes às de um contrato on-chain tradicional.

4 Metodologia

4.1 Abordagem Experimental

Para conduzir a análise comparativa de desempenho, foram desenvolvidas duas versões do sistema com requisitos funcionais equivalentes: uma implementação de execução nativa na camada base (EVM) e uma implementação com execução off-chain via rollup otimista. A metodologia parte do princípio discutido nas seções anteriores de que a principal diferença econômica entre as abordagens não é apenas o "custo por transação", mas a recomposição do custo total entre computação replicada na L1 e custos associados à publicação e liquidação on-chain. Assim, selecionou-se uma operação específica presente em ambas as implementações e caracterizada por complexidade significativa, por envolver múltiplas etapas de processamento, incluindo validações, agregações, cálculos de distribuição e emissão de múltiplas saídas observáveis. Essa escolha busca maximizar a sensibilidade do experimento aos trade-offs de execução apresentados previamente, uma vez que operações com maior intensidade computacional tendem a evidenciar mais claramente o impacto do modelo de *gas* na EVM e, por contraste, o deslocamento da computação para fora da cadeia em um rollup otimista. Em ambas as implementações, a operação selecionada corresponde à função `closeCampaign`. Essa operação representa o encerramento lógico do ciclo de vida do caso de uso analisado, isto é, o momento em que a lógica de negócio consolida o resultado das ordens e define as transferências necessárias, tornando-a adequada para comparar custos em um cenário onde a computação e a movimentação de valor se encontram diretamente acopladas.

4.2 Escolha da Implementação de Rollup Otimista

A plataforma selecionada para representar a execução via rollup otimista foi a Cartesi, por oferecer um ambiente de execução off-chain mais expressivo do que a EVM, ao mesmo tempo em que preserva o determinismo necessário para verificação por provas de fraude. Essa escolha está alinhada com a fundamentação teórica apresentada: em rollups otimistas, a L1 não reexecuta a computação por padrão, mas precisa ser capaz de arbitrar disputas e verificar a correção de forma reprodutível. A CVM provê esse requisito ao executar uma microarquitetura RISC-V de 64 bits em ambiente Linux determinístico, o que permite implementar a lógica do sistema com ferramentas e linguagens convencionais, sem perder a

propriedade de reexecução determinística em cenários de contestação. Consequentemente, a comparação proposta isola, de um lado, a execução totalmente on-chain sujeita aos limites de *gas* e ao custo marginal da computação replicada na L1 e, de outro, um desenho no qual a computação é deslocada para off-chain e a L1 é utilizada como camada de disponibilidade e liquidação.

4.3 Equivalência Funcional entre Implementações

Para garantir uma comparação justa, as duas versões do sistema foram mantidas funcionalmente equivalentes no nível de requisitos e efeitos observáveis. Em particular, a operação analisada foi definida de modo que, dadas entradas semanticamente equivalentes, ambas as implementações produzam o mesmo conjunto de consequências lógicas: validações aplicadas às ordens, regras de seleção e consolidação, cálculo de valores finais a serem distribuídos e efeitos de transferência associados ao encerramento. Na versão EVM, esses efeitos ocorrem atomicamente dentro da transação `closeCampaign`, de modo que a lógica de encerramento e os efeitos de liquidação sejam resolvidos no mesmo contexto transacional na L1. Na versão baseada em rollup, os mesmos efeitos lógicos são computados off-chain no `closeCampaign` e materializados na L1 por meio do mecanismo de *voucher*, um dos *outputs* gerados pela CVM que pode ser executado, mediante verificação, na camada base, preservando o fato de que, ao término do fluxo completo, os fundos estejam efetivamente disponíveis ao usuário e liquidados na camada base.

4.4 Métricas de Avaliação

A métrica principal adotada foi o custo total em *gas* necessário para que o ciclo de vida da operação seja concluído e para que os fundos estejam disponíveis ao usuário na Layer 1, pois essa definição captura diretamente o trade-off central analisado neste trabalho: execução replicada e atômica na L1 versus execução deslocada para off-chain com liquidação posterior na L1. No caso da execução nativa na EVM, o custo total é o gás consumido pela transação `closeCampaign`, que executa a lógica e realiza as transferências no mesmo contexto transacional. No caso da execução via Cartesi, o custo total foi definido como a soma do gás consumido pelas interações necessárias para completar o fluxo até a disponibilidade final na L1, abrangendo o envio do input de finalização (*input close*), o envio do input de solicitação de saque (*input withdraw*) e a execução do *voucher* de transferência na L1. Essa definição evita comparações parciais que poderiam subestimar o custo da abordagem em rollup ao considerar apenas o input inicial e ignorar o custo inevitável de liquidação e transferência on-chain.

4.5 Cenários de Teste

A avaliação empírica foi conduzida por meio do envio de entradas equivalentes para ambas as implementações, solicitando a execução da rotina de encerramento de campanha. Para observar a sensibilidade do custo às dimensões que tipicamente amplificam a carga computacional, como cardinalidade de participantes e número de ordens, foram definidos três cenários de teste com complexidade crescente, mantendo-se constante a natureza da operação e variando-se apenas o tamanho do conjunto processado. O Cenário Base consiste em uma campanha com um investidor e uma ordem de investimento; o Cenário Médio consiste em uma campanha com cinco investidores e cinco ordens; e o Cenário Complexo consiste em uma campanha com dez investidores e dez ordens. Essa gradação busca capturar como o custo se comporta à medida que aumentam as etapas de validação, agregação e distribuição, o que está diretamente relacionado às limitações econômicas e operacionais discutidas para a execução nativa na EVM, bem como ao deslocamento de custo entre computação off-chain e publicação/execução on-chain na abordagem de rollup otimista.

5 Desenvolvimento

Para viabilizar a análise comparativa de desempenho, foram desenvolvidas duas implementações *compliant* de um mercado de capital de dívida descentralizado, ambas atendendo aos mesmos requisitos funcionais, porém ancoradas em modelos de execução distintos: (i) execução nativa na camada base via EVM; e (ii) execução off-chain via rollup otimista com o Cartesi Rollups. A comparação busca evidenciar, no nível de engenharia do sistema, como o custo e as limitações operacionais se redistribuem entre computação replicada (EVM) e liquidação/publicação on-chain (rollup), mantendo o mesmo caso de uso e os mesmos efeitos observáveis ao final do fluxo. Nesse contexto, embora ambas as versões encerrem campanhas e consolidem resultados sob o mesmo conjunto de regras, a implementação EVM adota simplificações orientadas a economia de *gas* que podem reduzir a optimalidade do leilão. Portanto, as rotinas de fechamento não são algoritmicamente equivalentes no nível de passos internos:

$$\text{closeCampaign}_{\text{EVM}}(C, O) \not\equiv_{\text{alg}} \text{CloseCampaign}_{\text{CVM}}(C, O) \tag{1}$$

Entretanto, elas permanecem funcionalmente equivalentes em termos de efeitos observáveis ao final do fluxo, isto é, a campanha é encerrada, as ordens são consolidadas e os fundos tornam-se disponíveis na L1 conforme as regras estabelecidas pelo protocolo:

$$\text{closeCampaign}_{\text{EVM}}(C, O) \equiv_{\text{func}} \text{CloseCampaign}_{\text{CVM}}(C, O) \tag{2}$$

As duas implementações foram projetadas para cobrir o mesmo conjunto de requisitos funcionais: (RF01) criação de campanhas com validação de colateral mínimo; (RF02) submissão de ordens de investimento com taxas competitivas e demais validações; (RF03) encerramento de campanha e consolidação do resultado, com determinação de valores captados e taxas aplicáveis; (RF04) repagamento do empréstimo com distribuição proporcional aos investidores; e (RF05) execução do colateral em cenário de inadimplência. As diferenças entre as versões concentram-se principalmente na forma como o estado é persistido (on-chain na EVM vs. banco SQLite off-chain na CVM) e como os efeitos monetários são materializados (transferências atômicas imediatas na EVM vs. vouchers para posterior liquidação no rollup).

5.1 Algoritmo de Leilão Reverso

O DCM implementado em ambas versões adota um mecanismo de leilão reverso para selecionar as melhores ofertas

de investimento, privilegiando taxas de juros menores e, em caso de empate, a prioridade temporal (primeiro a chegar). A formulação matemática do problema é comum às duas versões; a diferença central reside na viabilidade de implementar o algoritmo de forma completa na EVM versus off-chain na CVM.

5.1.1 Formalização Matemática

Seja uma campanha C definida pelos parâmetros D (montante de dívida a ser captado), $r_{\max}$ (taxa máxima aceita pelo criador), t_{close} (*timestamp* de encerramento) e $O = \{o_1, o_2, \ldots, o_n\}$ (conjunto de ordens de investimento). Cada ordem o_i é caracterizada por a_i (valor ofertado), r_i (taxa de juros ofertada) e t_i (instante de criação). Consideram-se válidas as ordens que obedecem à restrição $r_i \leq r_{\max}$ para todo i e cuja soma agregada dos valores ofertados é suficiente para cobrir a demanda D:

$$r_i \leq r_{\max} \quad \forall i \in \{1, 2, \ldots, n\} \tag{3}$$

$$\sum_{i=1}^{n} a_i \geq D \tag{4}$$

5.1.2 Algoritmo de Finalização

Ordenação. As ordens elegíveis são ordenadas por taxa crescente e, em caso de empate, por *timestamp* crescente:

$$o_i \prec o_j \iff (r_i < r_j) \vee (r_i = r_j \wedge t_i < t_j) \tag{5}$$

Seleção. Seja $O' = \{o'_1, o'_2, \ldots, o'_m\}$ o conjunto de ordens ordenado. Define-se o conjunto de ordens aceitas A como o menor prefixo de O' cuja soma atinge D:

$$A = \{o'_1, o'_2, \ldots, o'_k\} \quad \text{onde} \quad k = \min \left\{ j : \sum_{i=1}^{j} a'_i \geq D \right\} \tag{6}$$

Taxa efetiva. A taxa efetiva do empréstimo (r_{eff}) é calculada como a média ponderada das taxas das ordens em A:

$$r_{\text{eff}} = \frac{\sum_{i=1}^{k} a'_i \cdot r'_i}{\sum_{i=1}^{k} a'_i} \tag{7}$$

Rejeição e devolução. As ordens em O que não pertencem a A (ou seja, $O \setminus A$) são marcadas como rejeitadas e seus valores ofertados devem ser devolvidos aos respectivos investidores.

5.2 Implementação Nativa em EVM

A implementação nativa foi desenvolvida em Solidity utilizando o framework Foundry. O contrato principal `DCM.sol` concentra a lógica de negócio on-chain, com aproximadamente 535 linhas de código, incluindo validações de entradas, gerenciamento de armazenamento e emissão de eventos para observabilidade. Por se tratar de uma execução integralmente replicada na camada base, o estado do sistema é mantido diretamente em `storage` na L1 e todos os efeitos relevantes do fluxo (p. ex., atualizações de estado e transferências de ERC20) são materializados no mesmo contexto transacional, fazendo com que o custo econômico seja diretamente refletido em *gas*.

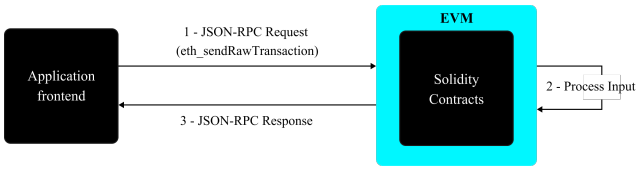

Figura 2. Diagrama de componentes e fluxo de execução da implementação nativa em Solidity.

A Figura 2 sintetiza esse modelo de execução em um fluxo curto e direto entre o *Application frontend* e o ambiente EVM. No passo (1), o frontend submete uma transação assinada via JSON-RPC, tipicamente por meio do método `eth_sendRawTransaction`, delegando à rede a inclusão da transação em um bloco. No passo (2), a própria EVM processa o *input* dessa transação ao executar o código dos contratos Solidity, o que envolve leitura e escrita em `storage`, verificação de condições de negócio, emissão de eventos e chamadas externas necessárias (por exemplo, `transfer` de ERC20). Esse processamento é determinístico e replicado por todos os nós validadores, de modo que o custo marginal de cada etapa de computação e de acesso a armazenamento é internalizado no modelo de *gas*. Por fim, no passo (3), a aplicação recebe a resposta via JSON-RPC, que representa a confirmação de que a execução foi incluída e seus efeitos já foram materializados on-chain. Dessa forma, considerando as limitações econômicas e operacionais da EVM, a rotina de encerramento da campanha (`closeCampaign`) foi implementada de forma a privilegiar uma estratégia de menor custo em detrimento da optimalidade algorítmica. O armazenamento de dados do leilão é mantido em estruturas eficientes (p. ex., *mappings* e listas de participantes), evitando a materialização de um conjunto completo de ordens e, assim, dispensando uma ordenação explícita por (r_i, t_i) no momento do encerramento. Dessa forma, a seleção das ordens tende a seguir a ordem de inserção dos participantes, reduzindo comparações e escritas em `storage`. A taxa efetiva r_{eff} do empréstimo pode ser simplificada por um parâmetro pré-definido ou por um cálculo reduzido, já que a computação integral da média ponderada com seleção ótima implicaria maior custo e risco de exceder limites de *gas* em cenários de alta cardinalidade. Em consequência, o custo assintótico da rotina de fechamento é tipicamente $O(n)$ em relação à quantidade de participantes/ordens processadas, com consumo de *gas* dominado por operações de escrita em `storage` e transferências de tokens ERC20 realizadas no mesmo contexto transacional.

5.3 Implementação com Cartesi Rollups

A Figura 3 ilustra o fluxo de execução do dApp utilizando Cartesi Rollups, composto por quatro etapas principais numeradas (1) a (4) que conectam os componentes *Application frontend*, *Cartesi Rollups Contracts*, *Cartesi Machine* e *Application backend*. No passo (1), o *Application frontend* (interface do usuário) envia um *input* para a camada base (L1) por meio dos *Cartesi Rollups Contracts*, os quais fazem o registro desse *input* na blockchain, tornando-o disponível para processamento off-chain. Em seguida, no passo (2), o nó do Cartesi Rollups (componente fora da cadeia) detecta o novo *input* publicado e o encaminha à instância da *Cartesi Machine* correspondente, dando início à execução da lógica do dApp fora da L1. No passo (3), dentro da *Cartesi Machine*, o *Application backend*, que corresponde ao código da aplicação

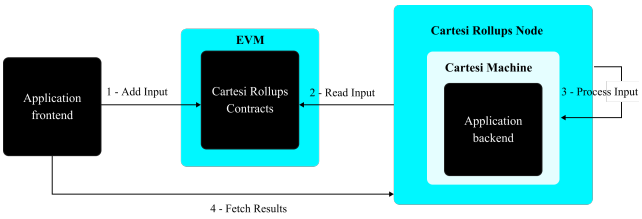

Figura 3. Arquitetura em camadas da aplicação Cartesi.

implementado em Go, processa o *input* recebido. Essa lógica de backend é executada na *Cartesi Virtual Machine* (CVM), um ambiente Linux isolado e determinístico proporcionado pela Cartesi, permitindo o uso de ferramentas e bibliotecas tradicionais. Durante a computação, o backend pode manter e atualizar o estado da aplicação de forma persistente; esses dados são armazenados em disco local (no contexto da CVM) utilizando SQLite no driver da aplicação. Toda a computação intensiva ocorre nesse ambiente off-chain, explorando o poder computacional da CVM para realizar cálculos complexos com desempenho muito superior ao da EVM, mas sem comprometer a verificabilidade, pois a execução permanece determinística e auditável. No passo (4), ao término do processamento, o *Application backend* produz saídas (*outputs*) verificáveis. Em particular, são gerados *vouchers*, que representam transações a serem executadas na camada base (L1), e possivelmente *notices* contendo informações de resultado para acompanhamento. Os *Cartesi Rollups Contracts* na L1 são então responsáveis por agendar e executar os *vouchers* emitidos, efetivando na blockchain as operações decorrentes da lógica off-chain. Dessa forma, a camada L1 cuida apenas de publicar os *inputs* e de realizar a execução final dos *outputs* (vouchers), enquanto toda a computação pesada permanece segregada na CVM off-chain. É importante destacar que a execução na CVM é totalmente determinística e reexecutável por qualquer parte interessada, característica que viabiliza o modelo otimista de rollups: caso um resultado incorreto seja reivindicado, o mecanismo de prova de fraude poderá ser acionado, reexecutando a computação de forma verificável e identificando discrepâncias, de modo que apenas o estado válido seja aceito na cadeia principal.

A rotina de encerramento da campanha, implementada no *handler* `CloseCampaign`, ativado por um *input* do tipo `campaign/close`, permite executar integralmente o algoritmo de leilão conforme formalizado. As ordens relativas à campanha são recuperadas do banco SQLite da aplicação, por exemplo, via `SELECT * FROM orders WHERE campaign_id = ?`, e então ordenadas por (r_i, t_i) utilizando `sort.Slice`, resultando em complexidade típica $O(n \log n)$. Em seguida, aplica-se a seleção ótima das ofertas com base no critério de parada assim que a soma acumulada atinge D, e a taxa efetiva r_{eff} é calculada como média ponderada das ordens aceitas utilizando aritmética inteira de alta precisão, evitando erros de ponto flutuante. Diferentemente do caso EVM, os efeitos monetários não são refletidos imediatamente durante a computação off-chain: em vez disso, a aplicação gera *vouchers* correspondentes às transferências de tokens necessárias, os quais serão materializados na L1 posteriormente, após a janela de disputa (challenge period) definida pelo algoritmo de *Fraud-Proof*, tipicamente 7 dias na mainnet do Ethereum. Esse arranjo preserva a equivalência funcional ao término do fluxo, ou seja, o resultado final observado na L1 é o mesmo, mas desloca o peso computacional para fora da cadeia, mantendo na L1 apenas o custo associado aos *inputs* e à execução dos *vouchers* resultantes.

5.4 Ambiente de Testes

A validação da arquitetura proposta neste trabalho demandou a estruturação de um ambiente de testes com o objetivo principal de garantir a equivalência semântica entre as duas abordagens, a implementação de referência em Solidity e a implementação otimizada com Cartesi, e realizar medições precisas de desempenho, consumo de recursos e consistência dos efeitos observáveis da operação escolhida para a análise, a operação `closeCampaign`. No caso da implementação nativa EVM, os testes foram realizados em uma rede Ethereum local baseada na ferramenta Anvil, com um limite de *gas* por bloco configurado em 30.000.000, *gas price* fixado em 1 gwei e intervalo de blocos de aproximadamente 2 segundos. Essa configuração permitiu executar os cenários de teste com controle sobre os custos de *gas* envolvidos, garantindo também reprodutibilidade dos experimentos. Já para a versão utilizando o Cartesi Rollups, foi utilizado um ambiente de desenvolvimento baseado em contêineres Docker, contemplando a instância da Cartesi Virtual Machine (CVM) e os demais componentes do Cartesi Rollups gerenciados pela Cartesi CLI. A simulação de fluxos de liquidação foi viabilizada por meio de contratos de teste, permitindo executar ciclos completos de operações do dApp.

Visando reprodutibilidade dos experimentos, as versões exatas das ferramentas utilizadas foram: `Foundry` com `forge-std 1.10.0` e `OpenZeppelin Contracts 5.4.0`, com a opção `via_ir` habilitada no compilador Solidity; `Go 1.24.4`, com as bibliotecas `rollmelette` (commit `20250617235715`) e `go-ethereum 1.15.11`; `@cartesi/cli 2.0.0-alpha.20`, com `machine-emulator 0.19.0`, `xgenext2fs 1.5.6` e imagem de kernel Linux `6.5.13-ctsi-1-v0.20.0`; e `SQLite` via `gorm.io/driver/sqlite 1.5.6`. O determinismo da CVM e as versões congeladas em `foundry.toml`, `go.mod` e no *workflow* de CI do repositório garantem medições reproduzíveis bit-a-bit entre execuções.

5.4.1 Metodologia de Medição

A medição de consumo de *gas* foi realizada por instrumentação direta, tomando-se a diferença entre `gasleft()` imediatamente antes e depois da operação de interesse. Na versão EVM, a medição é aplicada a uma única chamada de `closeCampaign` nos cenários de 1, 5 e 10 investidores; na versão Cartesi, é aplicada separadamente a cada uma das três transações L1 que compõem o ciclo, *input* de fechamento, *input* de *withdraw* e execução do *voucher*, cuja soma constitui os 140.385 gás reportados. O desempenho da execução off-chain é caracterizado pela mesma heurística de diferença, aplicada aos contadores internos `Mcycle` e `Icycleinstret` da CVM antes e após o *handler* de fechamento, dos quais se deriva a razão *Cycles Per Instruction* (CPI) como indicador de intensidade computacional do *workload* RISC-V. O determinismo bit-a-bit da CVM garante que essas medidas sejam invariantes entre execuções.

5.4.2 Testes de Correspondência Funcional

A implementação dos testes em Solidity foi desenvolvida utilizando o framework Foundry, que oferece uma suíte de testes capaz de realizar a medição de custos de *gas* em chamadas para funções específicas. Na versão utilizando o Cartesi Rollups, por sua vez, foi utilizado o framework de testes Vitest para realizar as asserções em conjunto com um *binding* da CVM em Node.js `@tuler/node-cartesi-machine 0.8.0`, que possibilita a invocação programática de sequências de *inputs*, observação de *outputs* (*vouchers* e *notices*) e inspeção dos registradores internos da CVM. Para assegurar a equivalência funcional entre as duas implementações, foi desenvolvida uma bateria de testes automatizados em ambas as bases de código. Na implementação EVM, foram criados 11 testes com Foundry, abrangendo cenários como criação de campanha, submissão de ordens, encerramento de leilão, execução de colateral e repagamento. A cobertura de código atingiu 73,37%. Do outro lado, na versão baseada em Cartesi, foi desenvolvida uma suíte com mais de 20 testes automatizados. Essa abordagem permitiu a execução de sequências equivalentes de *inputs*, simulando interações realistas de usuários e comparando os resultados semânticos observados. A metodologia adotada consistiu em definir cenários idênticos de entrada para ambas as implementações, executar os testes nos dois ambientes; e contrastar os estados finais, os eventos observáveis, bem como os cálculos financeiros, por exemplo, taxa efetiva e distribuição de retorno. Esse processo de validação cruzada foi fundamental para garantir que ambas as versões produzissem efeitos funcionalmente equivalentes em todas as fases do ciclo de vida do sistema.

6 Análise dos Resultados

Esta seção apresenta os dados coletados, com foco quantitativo no custo de gás e no desempenho das implementações testadas. Primeiramente, comparamos o consumo de gás total necessário para concluir o ciclo completo de encerramento de campanha nas versões nativa (EVM) e em rollup otimista (Cartesi). Em seguida, abordamos a viabilidade econômica e os trade-offs inerentes à adoção do rollup, além de apresentar métricas de execução observadas na CVM que ilustram o esforço computacional off-chain.

Os testes mediram o consumo de gás para o ciclo completo de finalização de campanha. Os dados obtidos estão detalhados na Tabela 1. Nela, observa-se o custo total de gás para execução nativa na EVM comparado ao custo total na abordagem Cartesi, que soma todas as interações na Layer 1 (inputs de fechamento, saque e execução do voucher de transferência).

O fator de 8,6x no cenário complexo compara implementações *funcionalmente equivalentes, mas algoritmicamente distintas*, conforme discutido na Seção de Desenvolvimento. A implementação EVM executa uma versão simplificada do leilão reverso ($O(n)$ linear, com taxa efetiva reduzida e sem seleção ótima), enquanto a implementação Cartesi executa o algoritmo completo ($O(n \log n)$ com ordenação, seleção ótima e cálculo preciso da média ponderada das ordens aceitas). Portanto, o ganho medido não expressa apenas redução de custo de uma mesma computação: ele expressa um deslocamento arquitetural, no qual a Camada 2 viabiliza lógica de negócio mais rica e expressiva pagando *menos* na Camada 1. Em outras palavras, o que a EVM entrega a 1,2 milhão de gás é estritamente inferior, em termos de qualidade algorítmica, ao que a Cartesi entrega a 140 mil gás.

Tabela 1. Custo total de gás para finalização de campanha: EVM nativa vs. Cartesi (rollup otimista).

Cenário	Investidores	EVM Nativo	Cartesi (Total L1)*	Fator de Eficiência
Base	1	354.535	140.385	**2,5x** mais eficiente
Médio	5	731.207	140.385	**5,2x** mais eficiente
Complexo	10	1.204.547	140.385	**8,6x** mais eficiente
Tendência		*Crescimento Linear*	*Constante*	*Escalabilidade crescente*

*Soma do gás de: `input close` (60.811) + `withdraw` (32.415) + `voucher transfer` (47.159).

Cabe ressaltar que a constância de 140.385 gás refere-se exclusivamente ao ciclo de fechamento (*input close* + *withdraw* + *voucher*), e não ao ciclo completo desde o recebimento das ordens. O envio de cada ordem individual, em ambas as plataformas, consome *calldata* proporcional ao número de investidores: na EVM via chamadas externas de submissão de ordens e na Cartesi via inputs no *InputBox*. Como o mesmo número de ordens é enviado nos dois ambientes em cada cenário, a simetria estrutural do custo marginal de *calldata* por ordem preserva a validade da comparação. Portanto, o que se mede aqui é o custo da *finalização* do leilão, onde reside a diferença algorítmica relevante entre as duas arquiteturas.

6.1 Viabilidade Econômica e Escalabilidade

A análise ilustrada na Figura 4 demonstra que, no cenário base (1 investidor), a abordagem otimista já provou ser 2,5 vezes mais barata do que a execução totalmente on-chain. Essa vantagem se amplia nos casos de uso mais intensivos: no cenário complexo (10 investidores), a EVM gastou cerca de 1,2 milhão de gás, enquanto o Cartesi Rollup permaneceu em apenas 140 mil, resultando em um ganho de custo de 8,6 vezes. Isso implica que, ao escalar para cenários ainda maiores, as diferenças tornam-se ainda mais dramáticas. Por exemplo, projetando o crescimento linear observado, 100 investidores resultariam em um gasto superior a 10 milhões de gás na execução nativatornando a operação economicamente inviável. Em contraste, o modelo baseado em Rollups, mantém-se praticamente constante em torno de 140 mil de gás na L1, viabilizando aplicações de alta complexidade computacional que seriam proibitivas no modelo on-chain tradicional.

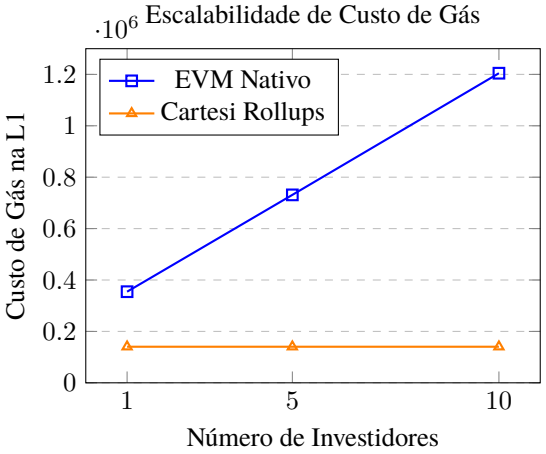

Figura 4. Comparação de escalabilidade de custos: a linha laranja (Cartesi Rollups) mantém-se estável à medida que cresce a complexidade, enquanto a linha azul (EVM nativa) cresce linearmente com o número de investidores.

Tabela 2. Custo estimado em USD (preços médios de 2025).

Cenário	Investidores	EVM Nativo (USD)	Cartesi (USD)	Economia (USD)
Base	1	\$2,93	\$1,16	\$1,77
Médio	5	\$6,04	\$1,16	\$4,88
Complexo	10	\$9,97	\$1,16	\$8,81
Projetado	100	\$77,75	\$1,16	\$76,59

Para contextualizar os valores em termos econômicos, a Tabela 2 converte o consumo de gás observado em custos aproximados em dólares, utilizando como referência o preço médio do gás (2,7 gwei) e do Ether (US$ 3.064,34) durante o ano de 2025 Etherscan [2025]; CoinGecko [2025].

Mesmo em condições de preço de gás historicamente baixas como as observadas em 2025, após a introdução do *proto-danksharding* via *upgrade* Dencun, o custo de operações complexas na EVM nativa cresce significativamente com o número de investidores. Em momentos de alta congestão da rede (picos históricos de 50–200 gwei), o diferencial se amplifica em até duas ordens de grandeza.

6.2 Trade-offs Arquiteturais

A expressiva redução de custo observada, até 8,6× no cenário complexo, vem acompanhada de trade-offs inevitáveis em termos de latência e modelo de segurança. Enquanto a execução EVM finaliza atômica e imediatamente no mesmo bloco em que a transação é minerada, a arquitetura de rollup otimista impõe uma janela de disputa (challenge period) antes da confirmação final. No ambiente de produção, essa janela costuma ser de vários 7 dias para permitir que qualquer nó honesto apresente provas de fraude contra transições incorretas. Consequentemente, aplicações que exigem liquidez imediata podem ser impactadas pela latência adicional. Contudo, para casos de uso como mercados de dívida ou procedimentos financeiros mais complexos, onde a urgência de liquidez é menor, essa troca (custo por tempo) pode ser vantajosa. Ademais, o modelo otimista desloca parte da responsabilidade de validação para fora da cadeia: presume-se que os resultados propostos sejam corretos, cabendo às partes interessadas o dever de fiscalizar ativamente. Embora existam mecanismos econômicos *bonds* que penalizem proposições fraudulentas, permanece a dependência de pelo menos um validador honesto e ativo para submeter provas quando necessário. Essas considerações de segurança são aspectos críticos a serem gerenciados por desenvolvedores e operadores de rollups, ainda que o protocolo garanta mecanismos eficientes para conter o custo de verificação on-chain em caso de disputa.

6.3 Desempenho de Execução na Cartesi Machine

Para contextualizar o esforço computacional off-chain, coletaram-se métricas de desempenho da Cartesi Machine durante a execução da operação `closeCampaign`. Um exemplo representativo é mostrado na Tabela 3, que exibe contagens de ciclos de máquina e instruções executadas (`mcycle` e `instret` do RISC-V) para o Cenário Médio de testes. Esses números indicam que a rotina de encerramento de campanha utilizou dezenas de milhões de ciclos de CPU e milhares de instruções RISC-V para completar o cálculo off-chain. Esse esforço de processamento é absorvido totalmente pela infraestrutura do Cartesi Rollups *off-chain*, não sendo refletido nos custos de gás da Layer 1. Em outras palavras, operações

Tabela 3. Métricas de desempenho da CVM para a operação de `closeCampaign`.

Métrica	Valor
Machine Cycles (Δ mcycle)	29,757,835
Instructions Retired (Δ instret)	14,416

computacionalmente intensivas, como ordenações, agregações e cálculos de alocação empregados neste caso, podem ser executadas eficientemente na CVM, utilizando hardware comum, enquanto apenas o resultado sintetizado (via compromissos e vouchers) é cobrado em gás on-chain. A capacidade de delegar tal carga de trabalho off-chain libera a camada base para atuar principalmente como settlement layer, preservando as garantias de segurança sem sobrecarregar a EVM com o processamento completo.

7 Conclusão

Os resultados obtidos neste estudo evidenciam de forma clara os benefícios e os compromissos envolvidos na adoção de rollups otimistas em comparação com a execução nativa na EVM. Observou-se que a abordagem off-chain não elimina a computação intensiva, mas a desloca para um ambiente verificável fora da camada principal, reduzindo o custo de gás na L1: no cenário mais complexo, o consumo total foi cerca de 8,6 vezes menor do que na versão nativa em EVM. Essa economia expressiva confirma, de forma empírica, a hipótese de que o deslocamento da computação para a Layer 2 viabiliza casos de uso que seriam economicamente impraticáveis na camada base, ao mesmo tempo em que preserva as garantias de segurança ancoradas na blockchain do Ethereum. Além do ganho direto em custo, a execução via Cartesi permitiu implementar integralmente algoritmos mais sofisticados, como a ordenação e seleção ótima do leilão reverso, sem as simplificações extremas exigidas pela EVM. Dessa forma, a lógica de negócio pôde ser expressa de maneira mais fiel aos requisitos do domínio, evidenciando que soluções de Layer 2 não apenas ampliam a eficiência econômica, mas também expandem a expressividade e o escopo das aplicações descentralizadas. Esses achados corroboram a visão de uma arquitetura *rollup-centric*, na qual a Layer 1 atua prioritariamente como camada de liquidação e segurança, enquanto a computação intensiva é delegada a ambientes off-chain verificáveis, como é o caso da CVM.

Entretanto, os ganhos observados não são isentos de trade-offs arquiteturais relevantes. Diferentemente da execução nativa na EVM, que oferece finalização atômica e imediata no mesmo bloco, os rollups otimistas introduzem uma latência adicional até a confirmação definitiva das operações, decorrente da janela de contestação necessária para a apresentação de provas de fraude. Em implementações atuais, esse período pode se estender por vários dias, o que implica que a liquidez efetiva na Layer 1 só se materializa após o encerramento do *challenge period*. Para o caso de uso analisado neste trabalho, um mercado de capitais de dívida, essa latência é compatível com a dinâmica do domínio, no qual o encerramento de campanhas e a liquidação de recursos não exigem instantaneidade absoluta. Ainda assim, aplicações sensíveis ao tempo de liquidação devem considerar cuidadosamente esse aspecto ao optar por uma arquitetura baseada em rollups

otimistas. Por outro lado, avanços recentes em protocolos de resolução de disputas, como o BoLD, apontam para caminhos promissores na redução do tempo e do custo de arbitragem on-chain, tornando o modelo otimista progressivamente mais competitivo também sob a ótica de latência.

Sob a perspectiva prática, os resultados deste trabalho oferecem implicações diretas para desenvolvedores de aplicações descentralizadas. A migração de lógicas complexas para ambientes off-chain verificáveis, como a CVM, mostra-se particularmente vantajosa para aplicações que envolvem ordenações, agregações e cálculos numéricos intensivos, os quais rapidamente atingem limites econômicos na EVM. Ao permitir o uso de linguagens e bibliotecas tradicionais em um ambiente Linux determinístico, a abordagem baseada em CVM reduz significativamente a complexidade de desenvolvimento e amplia o conjunto de ferramentas disponíveis, sem abrir mão da auditabilidade e da segurança providas pela Layer 1. Em contrapartida, essa escolha demanda maior atenção ao desenho do fluxo de liquidação, ao gerenciamento da latência e à necessidade de monitoramento ativo da rede para detecção de possíveis comportamentos fraudulentos durante a janela de disputa.

Por fim, este estudo abre espaço para diversas extensões e investigações futuras. Uma direção natural consiste em ampliar a comparação para outros modelos de rollup, incluindo rollups baseados em provas de validade (ZK rollups), que eliminam o período de contestação ao custo de maior complexidade criptográfica. Outra linha promissora envolve a análise de arquiteturas híbridas, que combinem execução off-chain expressiva com mecanismos de liquidação acelerada, por meio de provedores de liquidez, mitigando o impacto da latência em aplicações sensíveis ao tempo. Ademais, estudos adicionais poderiam explorar diferentes perfis de carga e domínios de aplicação, avaliando até que ponto os ganhos observados neste trabalho se mantêm em cenários ainda mais complexos. Em conjunto, os resultados apresentados reforçam a viabilidade técnica e econômica dos rollups otimistas como estratégia de escalabilidade e indicam que a separação clara entre computação e liquidação tende a ser um elemento central no futuro das aplicações blockchain em larga escala.

Contribuições dos autores

Henrique Marlon Santos, Emanuele Morais, Victor Takashi Hayashi e Bryan Kano Ferreira contribuíram para a concepção do estudo, implementação, experimentos e redação do manuscrito. Todos os autores leram e aprovaram o manuscrito final.

Conflitos de interesse

Os autores declaram que não têm nenhum conflito de interesses.

Disponibilidade de dados e materiais

Os conjuntos de dados (e/ou softwares) gerados e/ou analisados durante o estudo atual serão feitos mediante solicitação.

Referências

Alvarez, M. *et al*. (2024). BoLD: Fast and cheap dispute resolution. In *6th Conference on Advances in Financial Technologies (AFT)*. Schloss Dagstuhl. DOI: 10.4230/LIPIcs.AFT.2024.2.

Antonopoulos, A. M. and Wood, G. (2018). *Mastering Ethereum: Building Smart Contracts and DApps*. O'Reilly Media.

Buterin, V. (2014). A next-generation smart contract and decentralized application platform. White Paper. Disponível em: `https://ethereum.org/whitepaper/`.

Buterin, V. (2018). Layer 1 should be innovative in the short term but less in the long term. Ethereum Foundation Research. Disponível em: `https://vitalik.eth.limo/general/2018/08/26/layer_1.html`.

Buterin, V. (2020). A rollup-centric ethereum roadmap. Fellowship of Ethereum Magicians forum post. Disponível em: `https://ethereum-magicians.org/t/a-rollup-centric-ethereum-roadmap/4698`.

Buterin, V. (2021). An incomplete guide to rollups. Ethereum Foundation. Disponível em: `https://vitalik.eth.limo/general/2021/01/05/rollup.html`.

Buterin, V. *et al*. (2019). EIP-1559: Fee Market Change for ETH 1.0 Chain. Disponível em: `https://eips.ethereum.org/EIPS/eip-1559`.

Canetti, R., Riva, B., and Rothblum, G. N. (2011). Practical delegation of computation using multiple servers. In *Proceedings of the 18th ACM conference on Computer and communications security*, pages 445–454. DOI: 10.1145/2046707.2046759.

CoinGecko (2025). Ethereum (eth) historical price data. Disponível em: `https://www.coingecko.com/en/coins/ethereum/historical_data`.

Dannen, C. (2017). *Introducing Ethereum and Solidity*. Apress. DOI: 10.1007/978-1-4842-2535-6.

Etherscan (2025). Ethereum Average Gas Price Chart. Disponível em: `https://etherscan.io/chart/gasprice`.

Kalodner, H., Goldfeder, S., Chen, X., Weinberg, S. M., and Felten, E. W. (2018). Arbitrum: Scalable, private smart contracts. In *27th USENIX Security Symposium (USENIX Security 18)*, pages 1353–1370, Baltimore, MD. USENIX Association. Disponível em: `https://www.usenix.org/conference/usenixsecurity18/presentation/kalodner`.

Nakamoto, S. (2008). Bitcoin: A peer-to-peer electronic cash system. White Paper. Disponível em: `https://doi.org/10.2139/ssrn.3440802`.

Nehab, D. and Teixeira, A. (2022). Permissionless refereed tournaments. DOI: 10.48550/arXiv.2212.12439.

Teixeira, A., Nehab, D., dos Santos, M., *et al*. (2018). The core of cartesi. Technical report, Cartesi Foundation. White Paper. Disponível em: `https://cartesi.io/cartesi_whitepaper.pdf`.

Vukolić, M. (2015). The quest for scalable blockchain fabric: Proof-of-work vs. BFT replication. In *Open Problems in Network Security*. Springer. DOI: 10.1007/978-3-319-39028-4_9.

Wood, G. *et al*. (2014). Ethereum: A secure decentralised generalised transaction ledger. Ethereum Project Yellow Paper. Disponível em: `https://ethereum.github.io/yellowpaper/paper.pdf`.

Zheng, Z., Xie, S., Dai, H.-N., Chen, X., and Wang, H. (2020). An overview of blockchain technology: Architecture, consensus, and future trends. *IEEE International Congress on Big Data*. DOI: 10.1109/BigDataCongress.2017.85.