RISE: The High-Performance Rollup

Samuel Battenally *Hai Nguyen* *Thanh Nguyen*

RISE Labs

Abstract

We present RISE, an innovative Layer 2 (L2) platform designed to address the pressing performance limitations within the Ethereum rollup ecosystem. Despite notable advancements, current Ethereum L2 solutions are lagging in transaction throughput, significantly underperforming competitors like Solana. RISE leverages a parallel Ethereum Virtual Machine (EVM), a continuous execution pipeline, and a novel state access architecture built on Rust Programming Language-based Reth node infrastructure to enhance throughput and performance substantially. The core aim of RISE is to achieve a the target of The Surge [4], 100,000 transactions per second (TPS), with the potential for further scalability. This paper details the challenges of existing L2 technologies, the architectural innovations of RISE, and future directions for optimising blockchain scalability and efficiency. RISE promises to meet and exceed the most performant Layer 1 (L1) solution, establishing a new benchmark in blockchain technology.

1 Introduction

1.1 Technology Landscape

When Vitalik Buterin first set the Ethereum blockchain into motion in July 2015 [1], he was not only advancing the foundational technology introduced by Bitcoin [2] but also pioneering a profoundly innovative concept: smart contracts. Smart contracts have since unlocked applications and global coordination not dreamed of before their inception. It's clear from the adoption that the world is realising the potential of this incredible technology, though the adoption highlighted scaling challenges that ultimately led to the Ethereum Endgame proposed by Vitalik Buterin [3], which has evolved into the Ethereum Roadmap [4]. A core component of the roadmap is *The Surge*, which has a goal to reach 100,000 transactions per second and beyond on rollups. Arbitrum and Optimism have both played a significant role in the progress towards this goal by demonstrating EVM rollups as an undeniable solution to scaling, but their throughput is falling short [5][6]. The technology is primed for a second generation of rollups, hyper-focused on performance, to close this gap. Thanks to recent innovations, The Surge is becoming a reality, Vitalik argues we are reaching the exponential component of the capabilities S-curve [7]; RISE aims to unlock this rapid change and unlock the next wave of applications not yet dreamed of.

1.2 Introduction

Ethereum's rollup ecosystem has made commendable progress in enhancing blockchain scalability, securely supporting various applications and capturing substantial market value. However, even with these successes, the combined throughput of the Layer 2 (L2) ecosystem typically processes about 100 transactions per second (TPS)[8], which is far from meeting the performance of competitors like Solana, which consistently achieves over 1000 TPS, ten times that of all L2s combined [9]. Solana achieved this by prioritising scalability over decentralisation, with a lower validator count and high emissions to fund expensive infrastructure, its adoption indicates the market's support of a pragmatic approach to scaling. The success of Solana and the disparity in performance between Solana and L2s also sends a clear message that there is demand for high-performance, low-cost EVM L2s, but said demand is not being met by the technology available today.

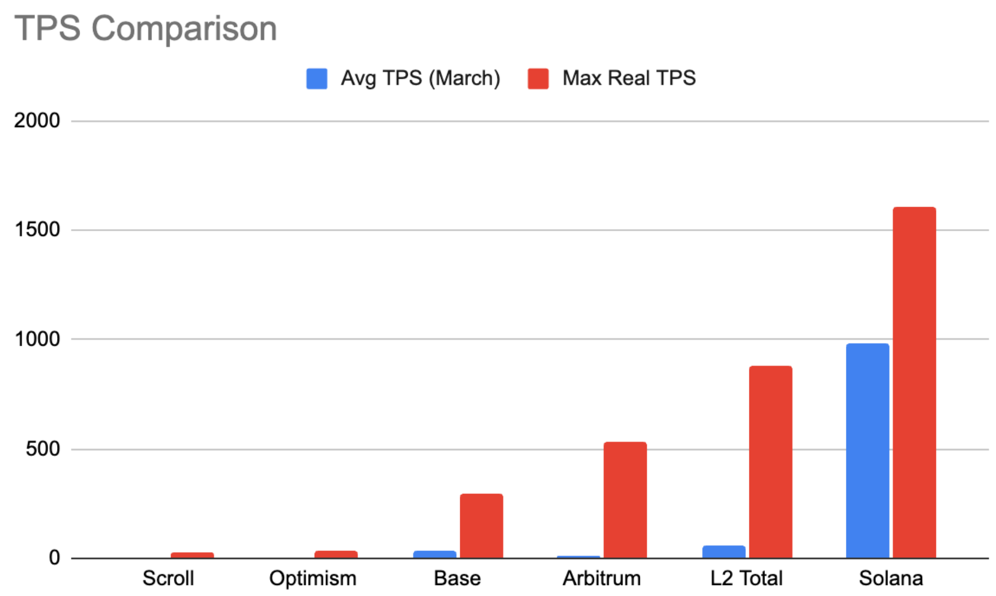
We introduce RISE, a next-generation optimistic Ethereum Virtual Machine (EVM) rollup stack. Designed from scratch, RISE focuses on maximising transaction throughput to achieve the ambitious target of 100,000 transactions per second (TPS) outlined in the Ethereum Roadmap. Historically, Data Availability (DA) has been a major bottleneck for rollups, but recent advances, such as EigenDA and EIP-4844, have significantly mitigated this issue. The challenge now shifts to execution performance. In tackling execution performance, RISE brings together the best technology current research offers along with novel innovations, including Parallel EVM Execution, Continuous Block Execution, and an innovative state access design. These are all integrated within the robust, rust-based Reth node infrastructure.

The RISE stack is for a rollup-centric future; to enable flexible deployments, both Sequencing and DA are agnostic in design, meaning RISE app chain deployments can choose a custom Sequencing and DA stack. The core of the RISE stack is high-performance execution. Not only will the RISE mainnet support a throughput to match the most performant Layer 1's (L1), but so will any number of RISE app chains.

This paper introduces the RISE execution technology stack and its performance innovation. However, it does not cover RISE interoperability via shared sequencing; this will be covered in future work by the RISE Labs team.

1.3 The state of rollups

The Ethereum approach to scaling has ostensibly been a success. L2s have held over $40 billion in value combined and have averaged over 100 transactions per second (TPS) since February 2024 [2]. With over 40 rollups now live, we are witnessing the early days of Ethereum horizontal scaling. This growth hasn't been frictionless; the lack of a shared state causes fragmentation of applications, users, and liquidity. Given the growth of horizontal scaling in L2s, it's fair to conclude vertical scaling has experienced limitations and anti-network effects slowing adoption and activity in individual rollups.



Performance comparison, Layer 2s and Solana.

This is even clearer when we compare L2 adoption and performance to the most performant competitor, Solana. Solana averaged close to 1000 TPS [2] during March 2024, ten times that of all Ethereum L2s combined. Arbitrum and Base reached noteworthy TPS highs of 532 TPS and 293 TPS. However, both networks experienced outages during their peak congestion, whereas Solana steadily supports 2000+ TPS alone. Solana has made significant gains on Ethereum despite the EVM tailwind, further strengthening the conclusion that there is an unmet demand for performance L2s.

1.4 Virtual Machine Selection

The EVM tailwind is powerful; EVM sports more live products, more investment, more active talent building and more users than any other Smart Contract VM [10]. Building the technology to support 100,000 TPS is meaningless without adoption from both developers and users. There are design decisions in the EVM that make scaling extremely difficult; however, the EVM tailwind is far too difficult to contend with; due to this RISE focuses solely on an EVM-compatible L2 stack.

1.5 Zero Knowledge vs Optimistic

zkEVM L2's have made significant progress recently, adoption is at all-time highs and proofing speeds are trending north. The goal of RISE is to usher in The Surge [6] and reach the 100,000 TPS target. This will be possible in the future with a zkEVM, though it will require hardware solutions such as zkASIC to achieve it, which will take to become available [x]. With an optimistic rollup, however, we're confident high throughput can be achieved without the need for custom hardware acceleration. Due to this, the RISE team will focus on an optimistic stack initially and will transition to zkEVM once suitable hardware becomes available.

1.6 Structure of the paper

In this paper, we begin in section 2 by identifying the performance challenges faced when building a high-performance rollup stack; in section 3, we provide a detailed breakdown of the RISE technology stack, and we finish in section 4 with future work and closing remarks.

2 Performance Challenges

The primary performance challenge faced by L2s to date has been data availability (DA). However, thanks to innovations in external DAs such as Celestia and eigenDA, and enshrined solutions like EIP-4844 [8], this is no longer the case. We're now seeing many other bottlenecks emerge, such as Networking, Execution, Block Size limitations, Fraud Proof Restrictions, Merkelisation and Storage IO. Seemingly overnight, L2 performance has become an exciting and challenging design space with various design directions.

2.1 EVM Execution

Most implementations of EVM execution are single-threaded. Due to this, today, the upper limit for performance is dictated by the clock rate of a single CPU. Adopting a parallel execution model is also non-trivial, given the transactions in a block are inherently sequential.

2.2 Ethereum State Access

Ethereum requires authenticated data structures encoded as modified Merkle-Patricia Tries (MPTs) to store and authenticate its key-value data.

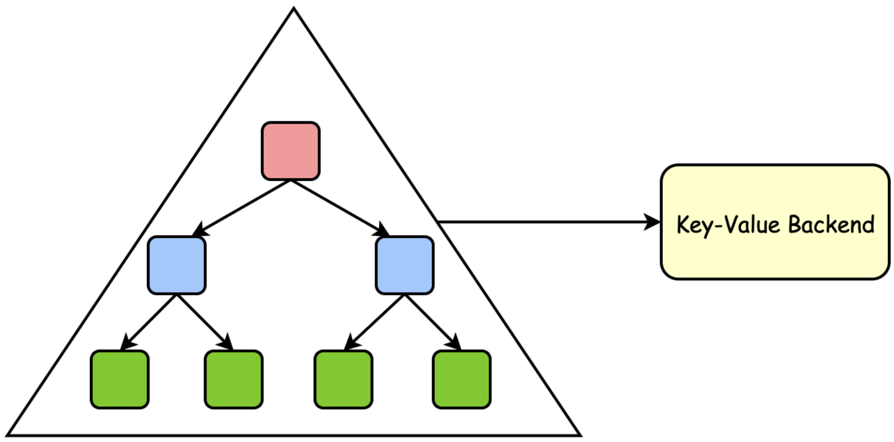


Figure: Key-value data in Ethereum is encoded in an MPT which is eventually stored a key-value backend database. The backend database itself is usually implemented as a B-Tree or LSM-Tree data structure.

Unfortunately, storage access in Ethereum is slow due to several reasons.

- Key-value data isn't stored in the database; it is encoded in an MPT that is stored in the database. Therefore, a read/write of a key in the application layer would be amplified to multiple reads/writes to the database backend. This read/write amplification results in low storage speed, translating into low execution throughput.
- A write operation would require multiple hash re-computations of all inner nodes along the MPT path.

- Lastly but most importantly, the storage I/O is synchronous, that is, execution threads must wait for costly read/write operations to complete before they continue processing the next transaction.

The impact of this is significant. State bloat leads to an anti-network effect. As the state grows, so does the average trie depth, slowing storage access even further. Additionally, the MPT root must be re-computed every block, resulting in a significant portion of the execution time going towards Merkleisation.

2.3 Storage IO

State storage isn't explicit in EVM design; node clients have the freedom to innovate on storage solutions. LevelDB, PebbledDB and MDBX are all Key-Value stores used in Ethereum and Layer 2 clients, though, they are not natively Authenticated Key-Value Stores (AKVS). A native AKVS solution would have its advantages, though no suitable FOSS solutions are available today.

2.4 Block Size

An obvious limitation of Layer 2s is the block gas limit. This configuration is easily increased if the system can support higher throughput. However, the increased block limit may impact the fraud-proof system. If challengers are to submit an entire block for fraud-proofs, they risk running into Ethereum block size limitations.

2.5 Layer 2 Block Productions

All L2 clients began as Ethereum clients and often hold onto L1 concepts. The block production pipeline for an L2 is inherently different to the L1. This offers opportunities to optimise and parallelise the block production pipeline in the L2 sequencer and execution client.

2.6 Interoperability and Decentralisation

Interoperability through shared sequencing will be a significant unlock for Layer 2's and decentralisation of sequencers is also an important step. Each introduces restrictions to the execution and sequencer that must be considered when building a future-proof system.

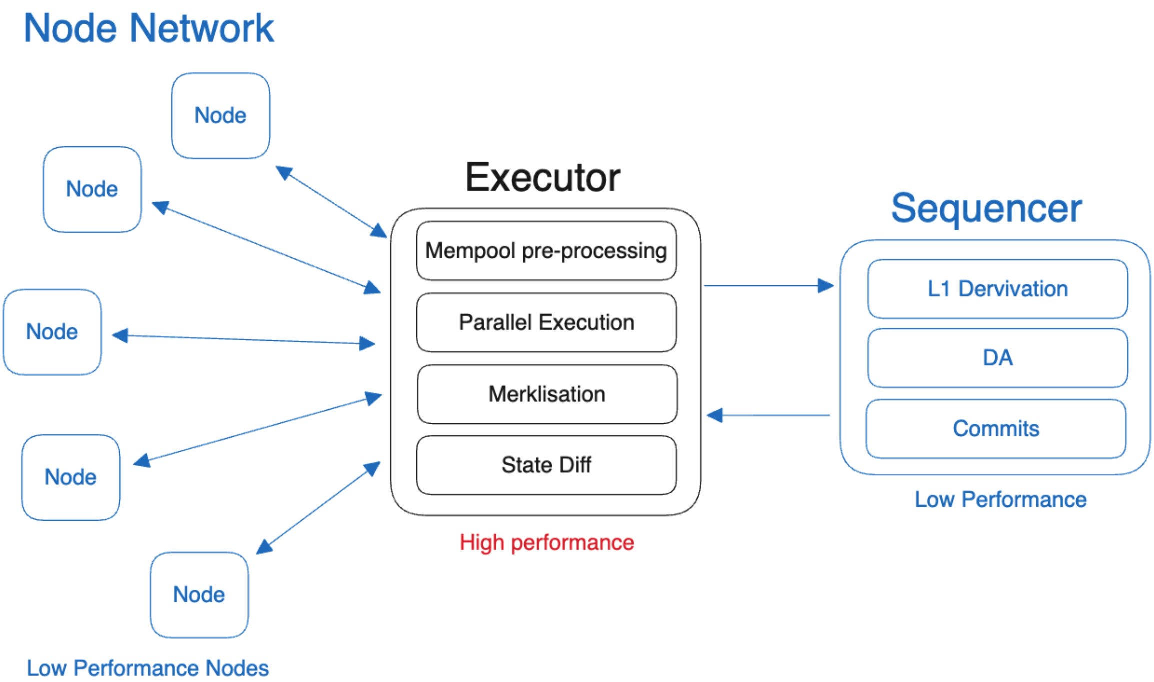
3 RISE System Design

3.1 RISE Network Architecture

The RISE system rethinks the status quo of L2 network architecture. Node clients in L2s today execute every transaction that passes through the network. However, the execution step does not improve the security model for the network; nodes rely on the fraud-proof system for trustlessness. Learning from Solana, RISE takes advantage of diversity in the Compute systems; RISE node clients do not execute unnecessary transactions, allowing a high-performance and expensive sequencer, or “executor” (see section 3.1.1), while maintaining node affordability.

3.1.1 The “Executor”

There’s a common misconception in the industry today that Layer 2’s generate their revenue through sequencing. It is true that there is value to be gained in sequencing, though it must be extracted at the user's expense through MEV. No major L2 today participates in value extraction from their users for obvious adoption reasons. The revenue generated by rollups today is predominantly from execution; L2s offer execution as a product. Here, we introduce the term “Executor”, the party responsible for executing transactions in blocks, maintaining the network state, generating the state root, and providing state diffs to the sequencer and broader network. The specifications of the executor are tailored to suit the demand of the particular RISE network deployment, be it an app chain or RISE mainnet.



Simplified RISE Compute Architecture. Expensive Executor, affordable Nodes.

3.1.2 High-performance Executor

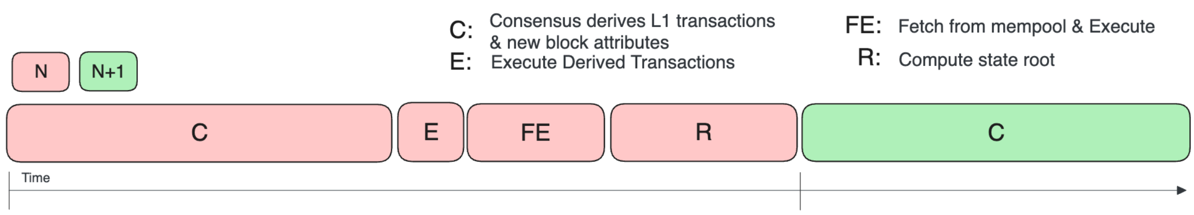
By unlocking a high-performance executor we are not only able to scale performance with a higher clock rate and more cores, recall the impact of Storage IO on performance, we're able to tackle that overhead by caching the most relevant state in RAM.

3.2 Continuous Block Pipeline

L2 block production consists of the following sequential steps [13]:

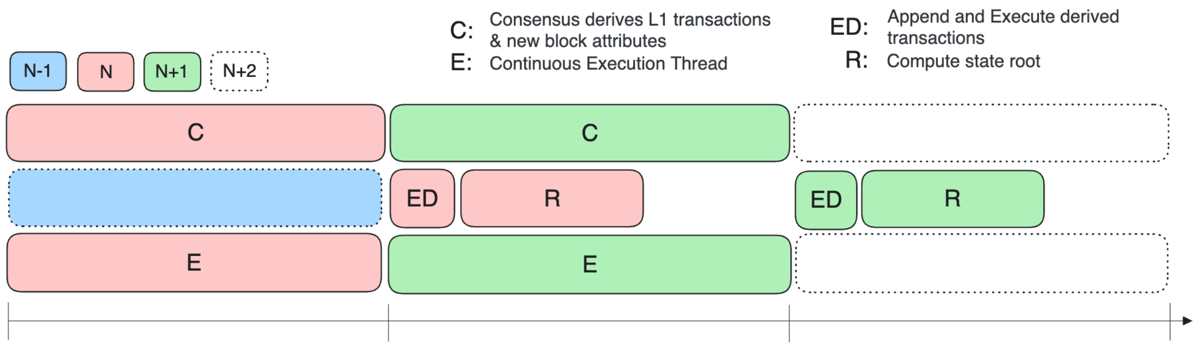
1. Consensus deriving L1 transactions and new block attributes
2. Execute Derived Transactions
3. Fetch transaction from mempool and execute
4. Prepare the header and compute the state root

Typically, these steps are sequential; the figure below illustrates a typical block pipeline for an L2 with a one-second block time and large state. The block gas limit will be limited by how much gas can be processed in the time allocated to execution; however, in this system, execution is allowed only a minority of the block-building pipeline. In practice, C consumes 40-80% of the block time, and as the state grows, R consumes up to 60% of the remaining execution time. In a worst-case scenario, E and FE will only have 8% of the block time to execute, which is efficient and block gas limits need to consider the worst-case scenario closely to avoid block time drift and reorgs.



Sequential Block Pipeline

RISE introduces the Continuous Block Pipeline (CBP), a parallelised block pipeline with concurrent stages and a Continuous Execution (CE) thread. The CE thread monitors the Mempool for transactions and executes in multiple block segments, no longer waiting for consensus to request a new block. Recall the worst-case scenario of 8% execution; in comparison, CBP executes transactions close to 100% of the available time. The state root computation also occurs concurrently with execution.



Continuous Block Pipeline (CBP)

3.3 Parallel EVM Execution

Here, we introduce another parallelisation layer, now, at the raw execution level. The EVM demands sequential execution; if this requirement is not met, non-deterministic outcomes will break consensus. However, in RISE, most transactions are executed in parallel. RISE utilises Block-STM, which Aptos first introduced [11]. The core technique of Block-STM is to take a block and optimistically execute every transaction in parallel, one outputs being transaction dependencies. Transactions with no dependencies are concluded and transactions with common dependencies are executed sequentially. There is further complexity, though this is the core technique of Block-STM.

3.3.1 Mempool Preprocessing

Unlike previous rollups that ordered transactions by first-come-first-served or gas auctions, RISE innovates a new Mempool structure that balances latency and throughput. The goal is to pre-order transactions to minimise shared states and maximise parallel execution. This has a relatively similar effect as the local fee market on Solana, where congested contracts & states are more expensive regarding gas & latency. Since the number of threads to execute transactions is much smaller than our intended TPS, we can still arrange dedicated threads to execute high-traffic contract interactions sequentially and others in parallel in other threads.

3.4 Layered MPTs

Recall Ethereum State Access significantly impacts throughput; our initial solution is to store the entire MPT in memory to reduce read/write overhead. Unfortunately, the state representation can grow extremely large. RISE will process hundreds of thousands of transactions per second; therefore, the state is likely to grow very fast. As the state grows, storing the entire MPT in memory is not feasible, though storing a small part of it would be possible. For example, we can store frequently-accessed addresses (e.g, Uniswap, USDT, bridges, etc,.) in memory for faster reads/writes.

We make a number of optimisations to enable high-speed storage access, such as separating read/write operations and caching MPTs. Instead of using a single MPT for the state, we add smaller intermediary in-memory MPTs on top of the existing on-disk MPT to reduce read and write amplification from high-latency disk storage. Our design, inspired by [11], consists of three different MPTs acting as caches for data access.

- **Snapshot MPT (SMPT).** SMPT is the same as the regular MPT used in Ethereum client, except that SMPT only holds the blockchain state at a snapshot block height instead of the most recent blockchain state. SMPT is stored on disk; therefore, access to SMPT is costly.

- **Intermediate MPT (IMPT).** IMPT is built on top of SMPT and is stored in memory.

- **Delta MPT (DMPT).** DMPT is built on top of IMPT and is also stored in memory. DMPT holds the most recent blockchain data and is the place where new updates take place.

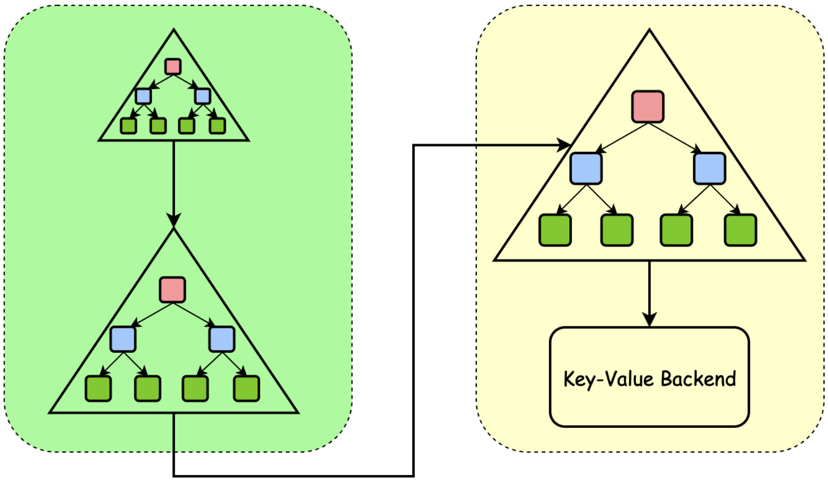


Figure. *There are three MPTs, two of which are kept in memory (green) while the largest one is stored on disk (yellow). A write is updated in DMPT and periodically flushed to disk. A read first touches DMPT for data, if not existing, it queries IMPT and reaches SMPT if the first two MPTs do not have the queried data.*

Write operations are updated on the DMPT which resides in memory. The DMPT will most likely keep the data of most frequently used accounts such as USDT or Uniswap, therefore, updates on these accounts will be reflected very quickly. For a read access, the request first touches the DMPT. If the data is not found, it continues to search the IMPT and finally, the SMPT is searched if the data is not found on the two previous MPTs.

To keep DMPT and IMPT efficient enough to fit in memory, at a predefined interval, updates in these tries are flushed to disk by merge operations. A merge operation consists of the following steps.

- Changes in IMPT are flushed to SMPT to create a new checkpoint.

- DMPT becomes the new IMPT.

- The new IMPT is emptied.

Note that the choice of the interval affects the system performance. If the interval is too short, DMPT and IMPT will be too small to hold enough cached data and merge operations will happen very frequently. If the interval is too long, DMPT and IMPT will be too large to fit in memory.

3.4 Additional Improvements

Some additional network features that contribute to the RISE stacks include alternate networking and JIT compilation.

3.4.1 Networking

The executor will be connected to thousands of nodes, each of these connections has overhead. The RISE network employs the QUIC protocol, an alternative to traditional TCP that offers faster connection setups and reduced latency, among other features.

3.4.2 JIT Compilation

Just-in-time allows for native VM execution rather than execution via an interpreter. JIT will have an incremental impact on the EVM execution stage in the RISE stack.

4 Closing

4.1 Future Work

4.1.1 RISE DB

Layered MPTs operate as cache layers for state access to help reduce the I/O amplifications for frequently accessed addresses. However, to further unlock high-performance EVM, removing additional data structures like B-trees or LSM-trees is unavoidable. RISE DB is an ongoing research project in this direction. Ideally, we use the native trie structure of the data as the index on-disk that never needs to be compacted, combined with a few low-level disk access optimisations, to reduce I/O overheads significantly.

4.1.2 Interoperability

RISE is building for a rollup-centric future, where *Rollup* is synonymous with *Server* and applications have sovereignty over their blockspace. The market values block space far above the applications that bring demand to said block space. We foresee a future where applications deploy sovereign rollups, owning their block space and the value they attract to it. However, this isn’t feasible with the technology today; rollup infrastructure does not support the performance or interoperability many applications need. The lack of atomic interoperability (Universal Synchronous Composability) between rollups inhibits applications from leveraging adjacent applications to benefit their own. RISE aims to tackle both of these challenges, performance directly and interoperability indirectly, by working with other innovators in the space. This whitepaper focuses solely on performance; more work is to come from the RISE Labs research team on tackling atomic interoperability for rollups.

4.2 Conclusion

The development of RISE represents a significant stride forward in the evolution of blockchain scalability and performance. This whitepaper has explored the inherent challenges faced by existing Layer 2 technologies within the Ethereum ecosystem and introduced RISE as a transformative solution designed to surpass these limitations. By implementing a parallel EVM, refined execution processes, and innovative database designs, RISE targets an ambitious throughput of 100,000 TPS, thereby setting a new standard in blockchain technology.

The ongoing development of RISE aims not only to meet current demands but also to anticipate and innovate ahead of future challenges in the blockchain space. We invite the community to join us in this exciting journey as we work towards building a more scalable, efficient, and inclusive blockchain ecosystem.

Acknowledgements

We extend our gratitude to those who have paved the way for advancements in Ethereum scalability and rollup technology. Particularly, we acknowledge the foundational work and significant contributions of the teams at Arbitrum and Optimism. Their pioneering efforts in optimistic rollups proved rollups as a viable scaling option for Ethereum.

We are also thankful to the Reth team for building the execution client on which RISE is being built. The modularity and performance Reth offers is crucial to RISE, the team are forward-thinking leaders in the space. We hope to pay it forward by contributing to Reth where possible.

References

[1] Buterin, V. (2014). Ethereum: A Next-Generation Smart Contract and Decentralized Application Platform. Retrieved from https://ethereum.org/content/whitepaper/whitepaper-pdf/Ethereum_Whitepaper_-_Buterin_2014.pdf

[2] Nakamoto, S. (2008). Bitcoin: A Peer-to-Peer Electronic Cash System. Retrieved from https://bitcoin.org/bitcoin.pdf

[3] Buterin, V. (2021, December 6). Endgame. Retrieved from https://vitalik.eth.limo/general/2021/12/06/endgame.html

[4] Understanding the Ethereum Development Roadmap, Retrieved April 10, 2024, from https://www.blocknative.com/blog/ethereum-roadmap-guide

[5] Kalodner, H., Goldfeder, S., Chen, X., Weinberg, S. M., & Felten, E. W. (2018). Arbitrum: Scalable, Private Smart Contracts. Princeton University. Retrieved from https://www.usenix.org/system/files/conference/usenixsecurity18/sec18-kalodner.pdf

[6] TPS, Max TPS, TTF, Block Time & Governance Model. (2024). Retrieved April 14, 2024, from https://chainspect.app/dashboard?range=30d

[7] Buterin, V. [@VitalikButerin]. (2024, March 29). What does this mean more broadly? I argue that now that the merge and EIP-4844 are done, we are decidedly on the right side of the S-curve. Further L1 changes will be quite meaningful and significant, but relatively milder. [X Post]. Retrieved from https://x.com/VitalikButerin/status/1773356271300706507

[8] Value Locked. (2024). Retrieved April 10, 2024, from https://l2beat.com/scaling/summary

[9] Buterin, V., & Beiko, T. (2024). EIP-4844: Shard Blob Transactions. Ethereum Improvement Proposals. Retrieved April 2, 2024, from https://eips.ethereum.org/EIPS/eip-4844

[10] Electric Capital. (2023). 2023 Crypto Developer Report.

[11] Choi, J., Beillahi, S., Singh, S., Michalopoulos, P., Li, P., Veneris, A., & Long, F. (2022). LMPT: A Novel Authenticated Data Structure to Eliminate Storage Bottlenecks for High Performance Blockchains. IEEE.

[12] Ethereum Optimism. (2024). Optimism: Optimistic Ethereum. Retrieved from https://github.com/ethereum-optimism/optimism

[13] Gelashvili, R., Spiegelman, A., Xiang, Z., Danezis, G., Li, Z., Malkhi, D., Xia, Y., & Zhou, R. (2022). Block-STM: Scaling Blockchain Execution by Turning Ordering Curse to a Performance Blessing. arXiv preprint arXiv:2203.06871. https://doi.org/10.48550/arXiv.2203.06871