Article

Not peer-reviewed version

BeamSNARKS: A proof algorithm is further veering right into Ethereum 3.0 with zkVM running on DePINs

Justin Drake *

Posted Date: 29 November 2024

doi: 10.20944/preprints202411.2216.v1

Keywords: BeamSNARKS; zkVM; SNARKification

 doi:10.20944/preprints202411.2216.v1

Article

BeamSNARKS: A Proof Algorithm Is Further Veering Right into Ethereum 3.0 with zkVM Running on DePINs

Justin Drake

Independent Researcher; justin.drake.eth@gmail.com

Abstract: The rapid evolution of Ethereum's infrastructure calls for innovative mechanisms to enhance scalability, security, and performance. This paper introduces BeamSNARKS, a cutting-edge framework designed to address critical challenges in zero-knowledge proof systems. BeamSNARKS encompasses two groundbreaking innovations: the Dynamic zkSNARKS Generation Optimization Mechanism and the Dynamic SNARKification Technology. The former revolutionizes computational efficiency by dynamically retrieving state data relevant to proof generation, minimizing bandwidth and storage requirements while maintaining validation accuracy. The latter introduces adaptive circuit design and hierarchical proof aggregation to optimize transaction throughput and reduce the computational and financial overhead of Layer 1 submissions. Together, these innovations establish BeamSNARKS as a pivotal advancement in scalable, efficient, and resource-optimized zero-knowledge proof systems. Through comprehensive analysis and targeted experiments, this paper evaluates the performance of BeamSNARKS's innovations, demonstrating their potential to transform Ethereum's decentralized ecosystem and lay the groundwork for future high-throughput applications.

Keywords: BeamSNARKS; zkVM; SNARKification

1. Introduction

The relentless pursuit of scalability, security, and decentralization has marked the evolution of Ethereum's blockchain infrastructure. At Devcon 2024, Justin Drake introduced the Beam Chain initiative, a bold reimagining of Ethereum's consensus layer that integrates Fast Finality and Zero-Knowledge Proofs (ZKPs) [1]. By leveraging cutting-edge technologies such as zk-SNARKs, zkEVMs, and novel cryptographic paradigms, Beam Chain offers a transformative approach to addressing the limitations of traditional blockchain architectures. This initiative represents a critical step toward achieving Ethereum's long-term vision: a decentralized ecosystem supporting global-scale applications without compromising performance or security.

Beam Chain's design is centered around optimizing the consensus process to achieve faster finality while maintaining high levels of security [2]. The initiative streamlines state validation and enhances network scalability by harnessing the efficiency of zero-knowledge proofs. These advancements and modular upgrades aim to eliminate computation and data availability bottlenecks, enabling Ethereum's infrastructure to support more performant and accessible decentralized applications (dApps).

To align with this vision, this paper introduces BeamSNARKS, a cutting-edge zero-knowledge proof framework tailored to optimize Ethereum's Layer 2 performance, implemented as a core component of Moonchain. This paper focuses exclusively on its underlying algorithms and innovations. The technology comprises two core components: the Dynamic BeamSNARKS Generation Optimization Mechanism and Dynamic SNARKification Technology. The former dynamically retrieves state data relevant to BeamSNARKS proof generation, optimizing bandwidth usage and reducing storage overhead without compromising validation accuracy. The latter employs adaptive circuit design and recursive proof aggregation to enhance transaction throughput and minimize the computational and financial costs associated with Layer 1 submissions.

 doi:10.20944/preprints202411.2216.v1

This paper systematically analyzes the design and performance of BeamSNARKS, focusing on its ability to address key challenges in computational efficiency, synchronization fidelity, and scalability. Through targeted experiments, we evaluate the proposed innovations under diverse workloads and transaction complexities, demonstrating their effectiveness in improving proof generation time, resource utilization, and overall system performance, which provides a robust framework for supporting scalable, efficient, and decentralized ecosystems by aligning with Ethereum's evolving architectural goals.

2. Core Innovations

2.1. Dynamic BeamSNARKS Generation Optimization Mechanism

Ethereum's Beam Sync upgrade marks a pivotal advancement in blockchain synchronization, introducing efficient on-demand state retrieval and enabling real-time validator functionality. Building upon these foundational enhancements, we strategically developed a Dynamic BeamSNARKS generation mechanism to align with and complement Ethereum's architectural evolution. This mechanism integrates on-demand state retrieval and incremental proof construction, optimizing resource utilization while maintaining high validation accuracy and synchronization fidelity. This innovation positions BeamSNARKS as a forward-compatible solution that synergizes with Ethereum's vision of scalable and efficient decentralized networks by addressing the demands of low-latency, high-throughput transaction processing.

To understand Beam Sync, it's essential to know two synchronization methods: Full Sync and Fast Sync [3], each with distinct advantages . The entire Sync processes every block from the genesis block, establishing an initial state of account balances and contract data. However, it's slow, especially on the mainnet, and the time required increases as the network grows. As illustrated in Figure 1, Fast Sync speeds up the process by downloading historical blocks and headers and selecting a recent block as the "launch block." It assumes miners follow valid EVM rules, allowing it to skip executing all previous blocks. Before executing the launch block, Fast Sync obtains the current state, including bytecodes and account information, by requesting a state snapshot from peers. After securing this information, it executes valid transactions and transitions into Full Sync to process all subsequent blocks.

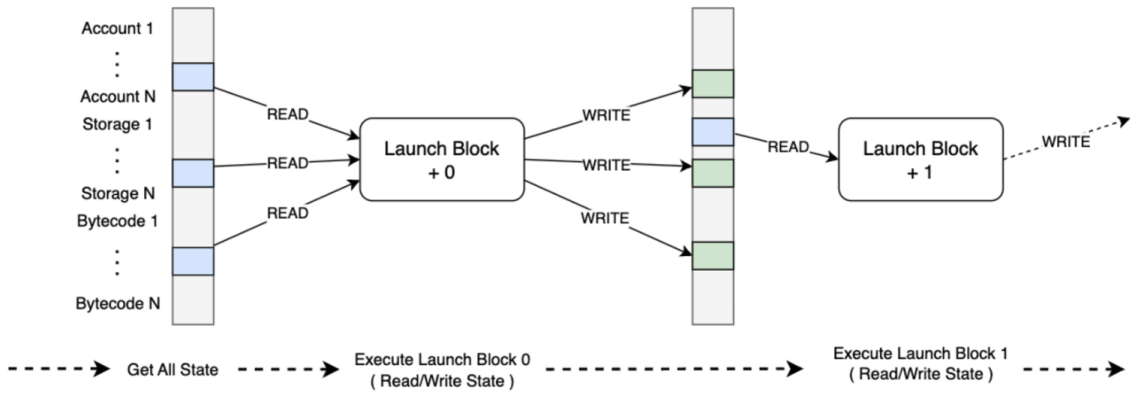

Figure 1. Illustration of the Fast Sync process, highlighting the retrieval of state snapshots from peers and the execution of the launch block to expedite blockchain synchronization.

Beam Sync is a direct evolution of the Fast Sync process, introducing a critical distinction to enhance synchronization efficiency. Unlike traditional methods, where validators must download and maintain the entire blockchain state, Beam Sync enables validators to execute the launch block immediately while requesting only the state data missing from the local database. This incremental

 doi:10.20944/preprints202411.2216.v1

synchronization process ensures that validators can start contributing to the network without waiting to download the complete state.

As shown in Figure 2, Beam Sync retrieves missing state elements on demand during block execution. Validators save the input and output states locally and proceed to the next block, repeating the process while requesting any additional missing data. Over time, more state data becomes available locally as more blocks are processed. Beam Sync also runs a background process to backfill states not accessed during block processing, ensuring that the database eventually contains all necessary state information. This approach minimizes bandwidth and storage demands, making participation feasible even for validators with limited hardware resources or connectivity.

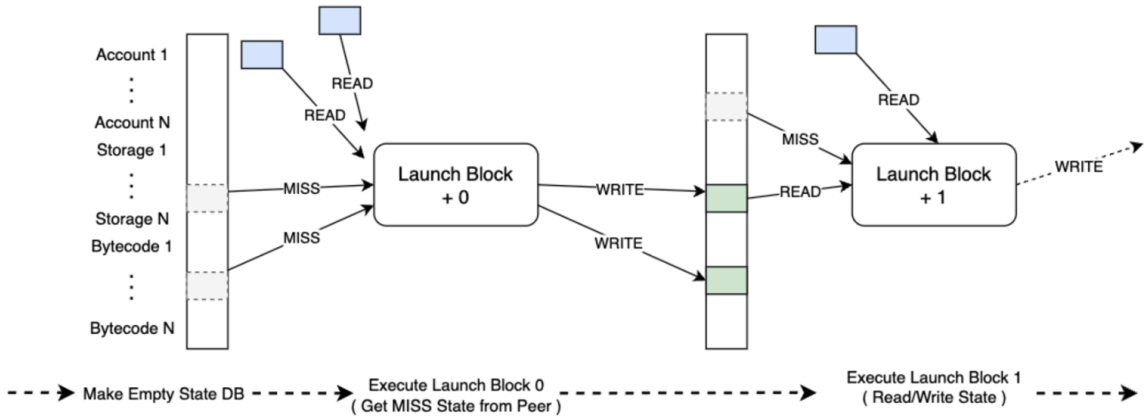

Figure 2. Visualization of Beam Sync, showcasing incremental state retrieval and progressive database backfilling.

Building upon the technical innovations introduced by Ethereum's Beam Sync, we proposed a targeted optimization framework: Dynamic zkSNARKS Generation Optimization. This framework addresses the computational and synchronization challenges inherent in zk-Rollup systems, ensuring seamless compatibility with Ethereum's evolving architecture. Leveraging the on-demand data retrieval mechanism of Beam Sync, this optimization dynamically constructs zkSNARKS proof inputs by synchronizing only the state data directly relevant to the verification process, thus eliminating the need to preload the entire Layer 2 (L2) state. This targeted approach minimizes bandwidth usage and significantly reduces storage requirements, making zk-Rollup systems more accessible to a wider range of participants. Additionally, the proof generation process is incrementally partitioned to improve computational efficiency. Transaction streams are divided into smaller, logically grouped subsets, each independently generating sub-proofs. These sub-proofs are later aggregated into a final BeamSNARK, preserving succinctness while distributing the computational workload across multiple cycles. This incremental architecture reduces peak processing demands and aligns seamlessly with the real-time synchronization capabilities enabled by Beam Sync.

2.1.1. On-Demand State Synchronization

Traditional zk-Rollup systems preload the entire Layer 2 (L2) state S , incurring high storage and bandwidth costs. BeamSNARK eliminates this inefficiency by dynamically retrieving only the relevant subset $S_r \subset S$. The cost of state retrieval is modeled as Equation (1). By dynamically requesting state elements during proof generation, the mechanism significantly reduces $|S_r|$, making the system more efficient. Validators only retrieve data relevant to the transactions being processed, minimizing bandwidth usage and enabling broader participation with limited resources.

$$\mathcal{C}\text{retrieve} = \sum_{i=1}^{k} g(S_r, t_i), \tag{1}$$

 doi:10.20944/preprints202411.2216.v1

where $g(S_r, t_i)$ represents the bandwidth cost of retrieving S_r at time t_i , and k denotes the total number of retrieval steps.

2.1.2. Incremental Proof Generation

Instead of processing an entire batch of transactions $\mathcal{B}$ at once, BeamSNARKS partitions $\mathcal{B}$ into subsets $\mathcal{B}_j$, generating proofs incrementally. The total computational cost for a batch $\mathcal{B}$ with n transactions is:

$$\mathcal{C}\text{compute} = \sum_{j=1}^{m} \sum_{T \in \mathcal{B}_j} (\alpha + \beta \cdot d(T)), \tag{2}$$

where α is the fixed setup cost, β is the per-step computation cost, and $d(T)$ is the computational depth of transaction T.

As expressed in Equation (2), incremental proof generation distributes computational workloads across multiple cycles, significantly reducing peak processing demands. This approach aligns seamlessly with Beam Sync's real-time block processing, ensuring low latency and scalability.

Following the definition of the cost of state retrieval and computational cost, the total zkSNARKS generation cost integrates the retrieval, computation, and proof finalization phases can be modeled as follow:

$$\mathcal{C}\text{proof} = \mathcal{C}\text{retrieve} + \mathcal{C}\text{compute} + \mathcal{C}\text{final}, \tag{3}$$

where $\mathcal{C}_{\text{final}}$ accounts for overheads such as proof aggregation and Layer 1 submission.

Equation (3) demonstrates how BeamSNARKS reduces the overall zkSNARKS generation cost. By dynamically synchronizing state retrieval and incrementally constructing proofs, the mechanism lowers computational and bandwidth requirements, enabling validators with standard hardware to participate effectively. By coupling dynamic state synchronization with incremental proof generation, this approach achieves a high degree of resource efficiency while maintaining the cryptographic integrity and performance standards expected in zk-Rollup systems. The proposed optimization significantly lowers the hardware and bandwidth barriers for zk-Rollup nodes, making it feasible for a broader range of participants, including those with limited computational resources, to engage in the network actively. This enhancement boosts scalability and democratizes participation, fostering a more decentralized ecosystem.

2.2. Dynamic SNARKification

SNARKification represents a pivotal innovation in zero-knowledge proof systems, addressing scalability and throughput challenges in Layer 2 rollup mechanisms [4]. By compressing transaction batches into a single succinct proof, this technology significantly reduces the computational and storage overhead required for Layer 1 submission [5]. Unlike traditional rollup designs, which often rely on fixed circuits, SNARKification employs adaptive circuit generation and recursive proof aggregation, enabling efficient validation of transactions with varying complexity. These advancements lower gas costs and ensure compatibility with Ethereum's consensus mechanism, making high-throughput decentralized applications more feasible. As a cornerstone of BeamSNARKS, this technique exemplifies the potential of zero-knowledge technologies to reshape the efficiency and accessibility of blockchain systems.

Central to BeamSNARKS's innovations is its dynamic SNARKification technology, which breaks away from traditional fixed-circuit designs by introducing adaptive circuit generation and hierarchical proof optimization. This advancement allows blockchains to tailor its zero-knowledge proof schemes to accommodate varying transaction complexities efficiently. Dynamic SNARKification avoids over-design by tailoring circuits to the complexity of transactions. For a batch $\mathcal{B}$ partitioned into m clusters $\{\mathcal{B}_1, \ldots, \mathcal{B}_m\}$, the circuit complexity is given by:

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 29 November 2024 doi:10.20944/preprints202411.2216.v1

$$\mathcal{C}\text{dynamic} = \sum_{j=1}^{m} \left(|\mathcal{B}_j| \cdot \mathcal{C}_j \right), \tag{4}$$

where $|\mathcal{B}_j|$ represents the number of transactions in cluster $\mathcal{B}_j$, and $\mathcal{C}_j$ is the circuit complexity of $\mathcal{B}_j$.

Additionally, to minimize Layer 1 submission costs, we aggregates individual proofs $\{P_1, P_2, \ldots, P_n\}$ into a single succinct proof P_{agg} :

$$P_{\text{agg}} = \mathcal{A}\left(P_{i i=1}^{n}\right), \tag{5}$$

where $\mathcal{A}$ is the recursive aggregation function. The computational cost of aggregation scales logarithmically with the batch size n :

$$\mathcal{C}_{\text{agg}} = \gamma \cdot \log n, \tag{6}$$

with γ as the aggregation coefficient. As shown in Equations (5) and (6), recursive aggregation drastically reduces the size of proofs submitted to Ethereum's Layer 1. This efficiency lowers gas costs and ensures scalability for high-throughput rollup systems.

By clustering transactions with similar computational characteristics, BeamSNARKS reduces unused resources and optimizes circuit utilization. As shown in Figure 3, during the circuit construction phase, the system dynamically adjusts circuit pathways based on transaction types and batch sizes. For simple transactions, such as single payments or IoT device data validations, lightweight circuits minimize proof generation time and memory usage. Conversely, for complex batch transactions, recursive proof aggregation consolidates multiple sub-proofs into a single BeamSNARK, drastically reducing the data volume submitted to Layer 1 and lowering verification costs. This dynamic mechanism avoids over-design in proof generation, improving circuit utilization, reducing gas consumption, and optimizing overall Layer 2 performance.

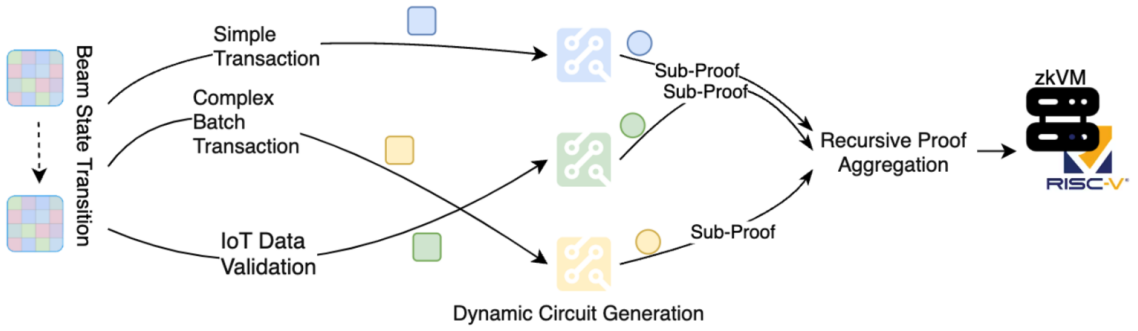

Figure 3. Adaptive Circuit Design demonstrated through transaction clustering based on complexity

At the hardware level, our implementation of RISC-V-based zkVMs further enhances the efficiency of dynamic SNARKification. Leveraging the modularity of the RISC-V architecture, these zkVMs are optimized explicitly for cryptographic operations such as elliptic curve multiplications and Merkle tree updates [6]. This results in a 30%-50% improvement in proof generation throughput on standard hardware configurations. Edge nodes equipped with lightweight zkVMs execute simple zero-knowledge validations, while core nodes, utilizing high-performance zkVMs, perform recursive aggregation to generate compressed proofs for Layer 1 submission. This hierarchical validation architecture reduces the hardware requirements for validator participation and ensures low latency and high throughput across the network.

By seamlessly integrating dynamic SNARKification with zkEVM capabilities, this framework delivers a scalable and efficient solution tailored to the specific demands of decentralized physical infrastructure networks (Depin). The technology excels in processing high-frequency microtransactions,

 doi:10.20944/preprints202411.2216.v1

securely validating IoT device data, and enabling efficient resource allocation. Compared to traditional zk-Rollup solutions, this approach significantly lowers resource consumption, accelerates proof generation and verification, and demonstrates superior gas efficiency and scalability. These innovations provide robust foundational support for Depin applications, aligning with Ethereum's vision for a high-performance blockchain ecosystem.

3. Experiments

As the first Layer 2 solution tailored for Beam Chain, BeamSNARKS optimizes key aspects such as on-chain state synchronization efficiency, zero-knowledge proof generation performance, and transaction throughput capabilities by introducing the Dynamic BeamSNARKS Generation Optimization Mechanism and Adaptive SNARKification Technology. This chapter evaluates BeamSNARKS's performance through three pivotal experiments, highlighting its theoretical advancements and demonstrating its practical potential.

3.1. Performance Evaluation of Chain SNARKification

This experiment evaluates the optimization effects of dynamic SNARKification technology in large-scale transaction processing, focusing on its performance in proof generation efficiency, verification time, and resource utilization. The experiment comprises two test groups: one employing a traditional fixed-circuit design for Snarkification and the other utilizing the dynamic SNARKification technology. Transaction data was generated in batches using Python scripts, with batch sizes of 1000, 10000, and 50000 transactions. The transaction types included simple ERC-20 token transfers and IoT device data validation scenarios requiring multi-step logical verification.

Dynamic SNARKification enhances performance by dynamically adjusting circuit paths on demand and aggregating proofs recursively. In this experiment, transaction streams were partitioned into logically related subsets, with each subset generating individual sub-proofs independently. These sub-proofs were aggregated recursively into a final BeamSNARK, which was submitted to Ethereum Layer 1 for verification. The experimental environment was constructed using the Circom framework, and proof generation and recursive aggregation tasks were executed with SnarkJS. Key performance metrics were monitored throughout the experiment, including the time required to generate sub-proofs for each subset under the dynamic SNARKification approach, the total time for aggregating all sub-proofs into a final BeamSNARK, and the time for verifying the final BeamSNARK on Ethereum Layer 1. Additionally, CPU usage, memory consumption, and storage IO performance were tracked to comprehensively assess the performance of the dynamic SNARKification technology under varying transaction scales and complexities.

Table 1. Performance Evaluation of Batch Processing with FCD and DS

Batch Size	Technique	Generation Time (ms) ± SD	Verification Time (ms) ± SD	CPU Usage (%) ± SD	Memory Usage (GB) ± SD
1,000 Transactions	FCD	502.3 ± 18.4	51.7 ± 3.2	30.5 ± 1.8	4.1 ± 0.3
	DS	298.7 ± 14.6	29.4 ± 2.1	19.7 ± 1.3	3.2 ± 0.2
10,000 Transactions	FCD	5,027.6 ± 95.3	203.4 ± 10.7	61.2 ± 3.5	16.4 ± 0.8
	DS	2,489.2 ± 81.7	101.8 ± 7.2	40.6 ± 2.4	12.3 ± 0.6
50,000 Transactions	FCD	30,127.5 ± 480.3	1,014.8 ± 50.1	90.7 ± 4.6	64.5 ± 1.7
	DS	14,856.9 ± 340.2	608.3 ± 25.4	70.3 ± 3.2	48.2 ± 1.1

The experimental results, presented in Table 2, indicate that the dynamic SNARKification technology consistently outperforms the traditional fixed-circuit design across all evaluated scenarios, with notable improvements in proof generation time, verification time, and resource utilization. This approach minimizes unnecessary computational complexity and enhances resource efficiency by dynamically adjusting circuit configurations. Furthermore, the recursive proof aggregation method

 doi:10.20944/preprints202411.2216.v1

effectively distributes computational workloads, demonstrating superior scalability in processing large-scale transaction batches.

3.2. Performance Analysis of zkVM Proof Generation

This experiment investigates the performance of the Dynamic BeamSNARKS Generation Optimization Mechanism, focusing on proof generation efficiency, throughput, and resource utilization under varying transaction complexities and workload conditions. Three transaction scenarios were selected for analysis: standard ERC-20 token transfers (simple transactions), ERC-712 multi-signature transactions (moderately complex transactions), and ERC-721-based NFT minting (complex transactions). The workload was configured for two levels of throughput, 100 TPS and 500 TPS, with transaction streams generated using Locust and processed by the zkVM module. The zkVM module leverages dynamic optimization to enhance performance through on-demand state retrieval and incremental proof generation. Transaction streams were divided into smaller logical subsets, with each subset independently generating sub-proofs. These sub-proofs were aggregated into a final BeamSNARK, maintaining succinctness while distributing computational workloads. The proof generation environment was built using the Halo2 framework, with recursive circuit designs employed for proof generation. The experimental infrastructure included dual Intel Xeon CPUs, 256 GB of memory, and NVMe SSD storage.

Throughout the experiment, several key metrics were monitored to assess performance. Proof generation time was recorded as the time taken to generate the final BeamSNARK from the transaction input. At the same time, throughput was measured as the number of transactions processed per second. Additionally, CPU usage, memory consumption, and storage I/O performance were tracked to evaluate resource efficiency across varying workload intensities.

Table 2. Performance Analysis of zkVM Proof Generation

Transaction Type	Load (TPS)	Proof Generation Time (ms) ± SD	Throughput (TPS) ± SD	CPU Usage (%) ± SD	Memory Usage (GB) ± SD
Simple	100	52.3 ± 3.1	998.4 ± 12.7	21.2 ± 1.5	5.2 ± 0.3
	500	72.8 ± 4.6	502.6 ± 9.8	41.8 ± 2.1	10.3 ± 0.6
Moderate Complexity	100	153.7 ± 5.8	498.2 ± 10.1	31.7 ± 1.8	8.5 ± 0.4
	500	201.4 ± 7.2	305.4 ± 11.3	52.9 ± 2.5	12.1 ± 0.7
Complex	100	497.5 ± 15.4	102.3 ± 6.9	62.4 ± 3.3	12.4 ± 0.8
	500	812.6 ± 18.7	82.1 ± 5.4	82.7 ± 4.1	18.6 ± 1.2

The results are presented in Table 2 .For simple transactions, the Dynamic BeamSNARKS mechanism achieved a proof generation time of 52.3 ms at 100 TPS and 72.8 ms at 500 TPS, with throughput reaching up to 1000 TPS under lower workload conditions. Moderately complex transactions exhibited longer proof generation times of 153.7 ms at 100 TPS and 201.4 ms at 500 TPS, with throughput decreasing to 500 TPS and 300 TPS, respectively. The proof generation time for complex transactions increased to 497.5 ms at 100 TPS and 812.6 ms at 500 TPS, with throughput reduced to 100 TPS and 80 TPS. Resource utilization scaled with transaction complexity and workload intensity, with CPU usage peaking at 80% and memory consumption reaching 18 GB for high-complexity transactions at 500 TPS.

The findings demonstrate that the Dynamic zk-SNARK mechanism effectively optimizes proof generation and resource utilization across different transaction complexities and workloads. The incremental proof generation and recursive aggregation strategies enable the zkVM to efficiently handle diverse transaction types while maintaining scalability under high-throughput conditions. These results highlight the potential of BeamSNARKS to support a broad range of applications requiring low-latency, high-performance Layer 2 solutions.

4. Conclusions

Moonchain represents a transformative advancement in Layer 2 solutions for Ethereum's Beam Chain framework. By introducing the Dynamic BeamSNARKS Generation Optimization Mechanism and Dynamic SNARKification Technology, BeamSNARKS effectively addresses key challenges in

 doi:10.20944/preprints202411.2216.v1

scalability, resource efficiency, and real-time synchronization. The optimization mechanism leverages on-demand state retrieval and incremental proof generation, significantly lowering hardware and bandwidth requirements while maintaining cryptographic integrity. Complementing this, dynamic SNARKification employs adaptive circuit design and recursive proof aggregation to streamline transaction processing and enhance throughput, achieving superior scalability and reduced costs for Layer 1 submissions.

Experimental evaluations have demonstrated BeamSNARKS's ability to outperform traditional fixed-circuit approaches, delivering substantial improvements in proof generation time, resource utilization, and overall performance across diverse transaction complexities and workloads. These results highlight the platform's potential to support high-performance decentralized applications, fostering inclusivity and democratizing blockchain participation by reducing barriers to entry.

As Ethereum progresses toward a modular and resilient architecture, Moonchain provides a complementary solution that aligns with the technical objectives of the Beam Chain initiative. By addressing critical scalability, resource efficiency, and real-time synchronization limitations, BeamSNARKS contributes to the foundational infrastructure necessary for enabling scalable and efficient decentralized ecosystems. Its methodological approach ensures compatibility with Ethereum's evolving framework while maintaining performance and security standards.

References

1. Ethereum Beam Chain: FAQ to Unpack the Big DevCon Announcement — polygon.technology. https://polygon.technology/blog/ethereum-beam-chain-faq-to-unpack-the-big-devcon-announcement. [Accessed 27-11-2024].
2. The 1.x Files: The State of Stateless Ethereum | Ethereum Foundation Blog — blog.ethereum.org. https://blog.ethereum.org/2019/12/30/eth1x-files-state-of-stateless-ethereum. [Accessed 27-11-2024].
3. Matevž Pustišek, Anton Umek, and Andrej Kos. Approaching the communication constraints of ethereum-based decentralized applications. *Sensors*, 19(11):2647, 2019.
4. Duc Anh Luong and Jong Hwan Park. Privacy-preserving blockchain-based healthcare system for iot devices using zk-snark. *IEEE Access*, 10:55739–55752, 2022.
5. Xinglin Shang, Liang Tan, Keping Yu, Jing Zhang, Kuljeet Kaur, and Mohammad Mehedi Hassan. Newton-interpolation-based zk-snark for artificial internet of things. *Ad Hoc Networks*, 123:102656, 2021.
6. Enfang Cui, Tianzheng Li, and Qian Wei. Risc-v instruction set architecture extensions: A survey. *IEEE Access*, 11:24696–24711, 2023.