VaultX Merge: Breaking Memory Barriers in Proof-of-Space Plot Generation

Arnav Sirigere
asirigere@hawk.illinoistech.edu
Illinois Institute of Technology
Chicago, IL, USA

Varvara Bondarenko*
vbondarenko@hawk.illinoistech.edu
Illinois Institute of Technology
Chicago, IL, USA

Ioan Raicu (advisor)
iraicu@illinoistech.edu
Illinois Institute of Technology
Chicago, IL, USA

Abstract

Proof-of-Work (PoW) has long powered blockchain systems like Bitcoin [6], but its high energy consumption has motivated the development of greener alternatives. Proof-of-Space (PoSp) [5] addresses this by relying on storage rather than computation. We contribute to MEMO, a high-throughput blockchain using VaultX [1], a lightweight PoSp algorithm capable of running on both small nodes, like Raspberry Pis, and high-performance systems. We introduce VaultX Merge, a novel out-of-memory method that generates large plots by first creating multiple small in-memory subplots and then merging them. This approach significantly reduces storage I/O, minimizes stored data size, and enables faster lookups with lower latency provided by large plots. We evaluate VaultX Merge across a range of machines and storage configurations, showing up to 50% faster plot generation compared to our previous Out-of-RAM implementation.

ACM Reference Format:
Arnav Sirigere, Varvara Bondarenko, and Ioan Raicu (advisor). 2018. VaultX Merge: Breaking Memory Barriers in Proof-of-Space Plot Generation. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation email (Conference acronym 'XX)*. ACM, New York, NY, USA, 3 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

Proof-of-Space (PoSp) offers a greener alternative to energy-intensive Proof-of-Work blockchains [4], but existing PoSp blockchains [3] still require substantial computation and memory, limiting scalability across devices. Large plots improve lookup performance by minimizing file accesses, but creating them efficiently is challenging due to memory limitations. Previous VaultX Out-of-RAM implementations generated large plots by streaming intermediate data to storage, repeatedly writing and reading partial results for both Table1 and Table2 structures [2], which caused redundant I/O and limited throughput. VaultX Merge addresses these challenges by generating smaller subplots fully in memory—retaining both Table1 and Table2 structures—and then merging them into large plots. This design reduces redundant I/O, overlaps CPU and storage operations, and enables efficient, scalable PoSp blockchain operation both on resource-constrained and high-performance systems.

*Developed the foundational VaultX work, upon which this project builds.

Conference acronym 'XX, Woodstock, NY

ACM ISBN 978-1-4503-XXXX-X/2018/06
https://doi.org/XXXXXXX.XXXXXXX

2 Lookup Performance

Larger plots consolidate stored data, reducing file system overhead and mitigating fragmented I/O that occurs when the same data is scattered across many smaller plots. As shown in Figure 1, lookup latency increases sublinearly with plot size, meaning that organizing the same dataset into fewer, larger plots results in a lower total lookup time.

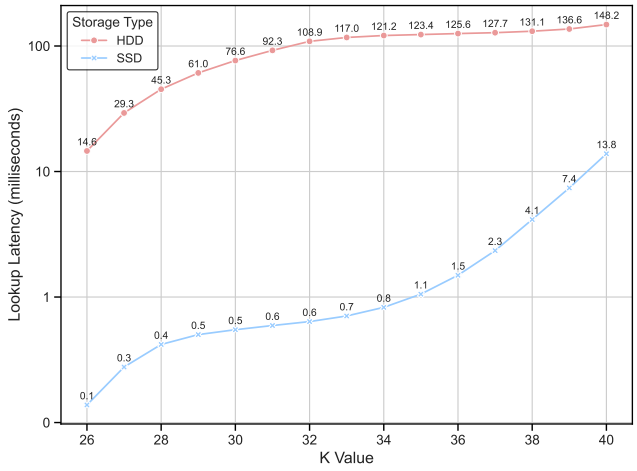

Figure 1: Lookup latency across plot sizes on EpycBox (see Table 2). Plot size doubles with each increase in K value.

3 VaultX Merge Design

The previous out-of-RAM pipeline generated large plot files by streaming intermediate data to storage in chunks, scattering bucket data and thus causing redundant I/O during both Table1 and Table2 [2] construction. These repeated reads and writes added significant overhead and slowed down performance. Our merge-based design restructures data writes to reduce redundant I/O and enable efficient plot generation within memory constraints, operating in two stages:

Stage 1: Subplot Generation. Using VaultX's in-memory approach, we generate smaller *subplots* that fit memory limits and write them to temporary storage. Each subplot contains a complete Table2 for a subset of the total records. By generating both Table1 and Table2 fully in memory per subplot, this approach avoids the multiple intermediate I/O operations required by the previous

Table 1: Storage I/O Comparison for $K = 34$ plot generation.

Metric	Out-of-RAM	Merge	Reduction
Plot Size	160 GB	128 GB	20% ↓
Reads	240 GB	128 GB	47% ↓
Writes	400 GB	256 GB	36% ↓
Total I/O	640 GB	384 GB	40% ↓

pipeline. Additionally, the storage writes for one subplot can overlap with CPU-intensive generation of the next, enabling concurrent processing and improving throughput.

Stage 2: Merge. After subplot generation, a final merge reorganizes data. Since each subplot holds partial entries for every bucket, we process them in batches—reading subplots, gathering bucket fragments, and consolidating them into contiguous buckets in the final output.

3.1 Serial Merge Approach

This approach executes the three phases in strict sequence. Each phase completes fully before the next begins, resulting in idle CPU or storage time, as resources wait for the other stage to finish. While straightforward to implement, this approach does not maximize resource utilization on high-end servers capable of handling the three phases concurrently, potentially leading to longer total runtimes.

3.2 Pipelined Merge Approach

The pipelined merge approach employs task-based parallelism with OpenMP to run reading, reorganization, and writing concurrently. Most setups typically use temporary storage for plot generation and separate storage for the final merged plot. This separation allows the pipeline to fully overlap reading and writing operations without contention. Data streams through the three stages as a pipeline, with different threads working on different phases simultaneously. This concurrency hides I/O latency and keeps both CPU and storage devices busy, maximizing resource utilization, improving throughput, and reducing overall merge time.

4 Data Compression

The merge approach improves storage efficiency by allowing the use of smaller nonces. Unlike out-of-RAM plotting, which requires a large nonce to ensure global uniqueness across the entire plot, the merge approach employs repeated smaller nonces that remain unique across subplots by using a unique plot ID as a domain separator in the hashing function.

5 Performance Evaluation

To evaluate the performance improvements of VaultX Merge, we measured generation times for a K-34 plot across both resource-constrained machines and high-performance computing systems, using a variety of storage devices. For each configuration, we generated subplots with varying K values and reported the best observed time. The specifications of the machines used are listed in Table 2. Figure 2 presents the plot generation times across these storage configurations, demonstrating the improvements achieved by our

Table 2: Tech specifications of machines used for experiments

Machine	CPU	Cores	RAM	Storage	ISA
EpycBox	AMD EPYC 7501 (2.00 GHz)	64	470 GB	SATA SSD, SATA HDD	x86_64
Orange Pi 5	ARM Cortex-A76/A55 (2.3/1.8 GHz)	8	31 GB	NVMe SSD, SATA HDD	aarch64
Raspberry Pi 5	ARM Cortex-A72 (1.5 GHz)	4	8 GB	NVMe SSD, SATA HDD	aarch64

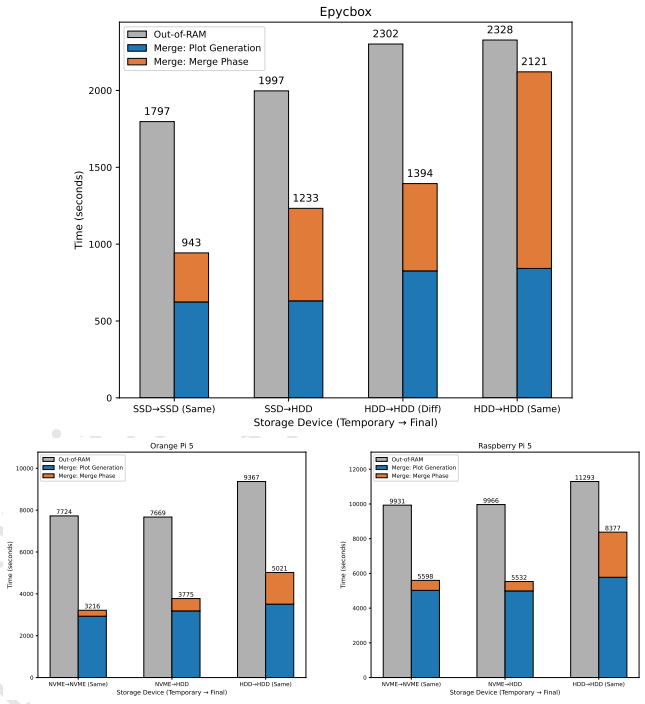

Figure 2: Performance evaluation plots illustrating plot generation times across storage configurations. ("Same" = single disk; "Diff" = two separate disks with identical specs.)

approach. We observe that the improvements in the HDD→HDD (same) case are minimal. While our approach reduces the overall disk I/O load compared to the baseline, the data is distributed across multiple files, leading to repeated I/O calls. On spinning disks, this introduces additional file system overhead, which limits the performance gains. However, when using two separate spinning hard drives, we see significant improvement since both disks can be utilized concurrently. We also expect the approach to scale with higher core counts, as hash generation can directly benefit from additional parallelism.

Acknowledgments

This work was supported in part by the National Science Foundation under award OAC-2150500.

References

[1] Varvara Bondarenko, Renato Diaz, Lan Nguyen, and Ioan Raicu. 2024. Improving the Performance of Proof-of-Space in Blockchain Systems. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage, and Analysis (SC '24)*. ACM, New York, NY, USA.

[2] Varvara Bondarenko and Ioan Raicu. [n. d.]. Securing Proof-of-Space Against Hellman Attacks.

[3] Bram Cohen and Krzysztof Pietrzak. 2019. The Chia Network Blockchain. White Paper, Chia.net. Version 9.
[4] Alex de Vries-Gao. 2018. Bitcoin's Growing Energy Problem. *Joule* 2 (05 2018), 801–805. doi:10.1016/j.joule.2018.04.016
[5] Stefan Dziembowski, Sebastian Faust, Vladimir Kolmogorov, and Krzysztof Pietrzak. 2015. Proofs of Space. In *Annual Cryptology Conference*. Springer, 585–605.
[6] Satoshi Nakamoto. 2008. Bitcoin: A Peer-to-Peer Electronic Cash System. https://bitcoin.org/bitcoin.pdf.