Improving the Performance of Proof-of-Space in Blockchain Systems

Varvara Bondarenko[1], Renato Diaz[2], Lan Nguyen[1] (*advisor*), Ioan Raicu[1] (*advisor*)

[1]Illinois Institute of Technology, Chicago, IL, [2]University of Central Florida, Orlando, FL

ABSTRACT

Blockchain technologies enable the success of digital currencies by providing security, decentralization, and trustless operation. Chia's Proof-of-Space (PoSpace) is a consensus algorithm that uses storage (instead of computation) for validation in the network while maintaining decentralization. Our work takes a clean-slate approach to implementing an efficient PoSpace system that is lightweight and operates on small nodes (e.g., Raspberry Pis with 4 cores and 2GB RAM) to large systems (HPC servers with 192 cores, 768GB RAM, and multiple NVMe devices). Our C and Rust implementations achieve significantly higher performance than Chia (XCH) in both plot generation and lookup efficiency across all systems.

BLOCKCHAIN MEETS HPC

- PoSpace transforms the compute-intensive problem of PoW into a data-intensive one, improving the energy efficiency of blockchain systems.
- It's important to efficiently use large HPC resources with many cores and multiple storage devices.
- Simplifying and streamlining I/O processes extends the lifespan of non-volatile storage [1].

VAULT DESIGN

- The hash-initialization phase [Fig. 1, Fig. 2] stores a sorted file with records for a participant node on disk, and uses the BLAKE3 hashing function for faster hash generation. [2, 5].
- The program completes two I/O writes and one I/O read.

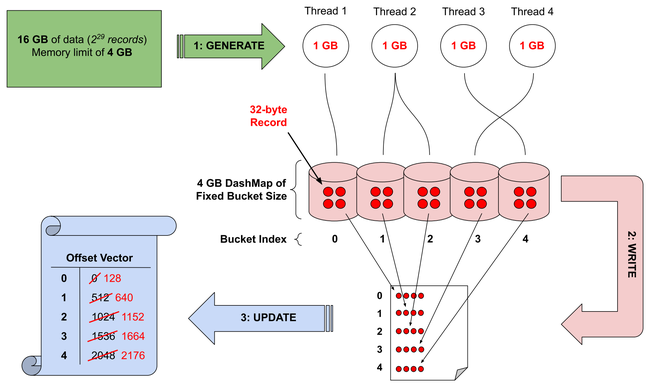

Fig. 1: Hash-initialization: generating hashes stage

- The search-verification phase [Fig. 3] checks if a participant node possesses a winning hash in its storage.

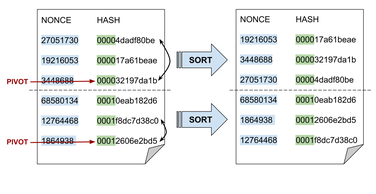

Fig. 2: Hash-initialization: sorting hashes

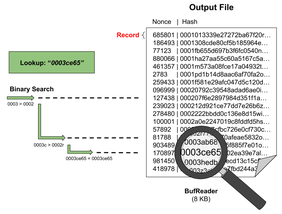

Fig. 3: Search-verification: lookup

HASH-INITIALIZATION PERFORMANCE

To evaluate the performance of *Vault-76*, we generated files with 2^k 32-byte records and collected throughput data from machines with varying compute power [3], as well as on different drives.

MACHINES	CPU	CORES	RAM	STORAGE	ISA
8Socket	Intel Xeon Platinum 8160 @2.10GHz	192	768GB	SATA, NVMe SSD	x86_64
EpycBox	AMD EPYC 7501 @2.00GHz	64	320GB	SATA, NVMe SSD, SATA HDD	x86_64
Raspberry Pi 4	Broadcom BCM2711, ARM Cortex-A72 @1.8GHz	4	2GB	SATA SSD, SATA HDD	aarch64

Tabel 1: Tech specifications of Mystic nodes used for testing [4]

In the experiments shown in Fig. 4, we set the memory limit to half the size of the file being generated to ensure the data could fit into memory, with a maximum limit of 16GB. The same memory limit was applied to programs with the same k-value to ensure consistency among all experiments.

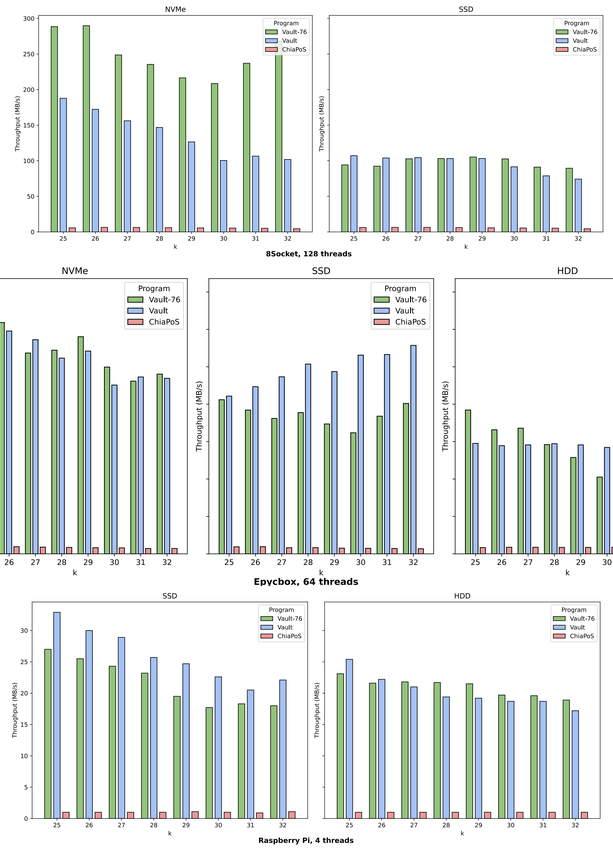

Fig. 4: Throughput comparison of *Vault-76, Vault (C), & Chia's default plotter* for k=25-32 on various machines

The experiments shown in Fig. 5 compare *Vault-76* and *Vault* with various Chia plotters.

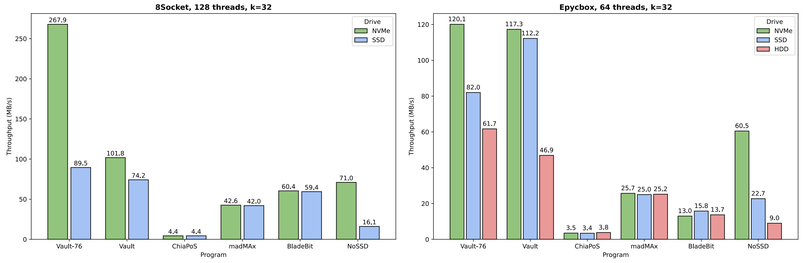

Fig. 5: Throughput comparison on two machines to various Chia plotters

SEARCH-VERIFICATION PERFORMANCE

The lookup latency was tested on the Epycbox machine [Table 1] after generating a file with k=32 records [Table 2].

DRIVE	MIN (ms)	AVG (ms)	MAX (ms)
NVMe SSD	1.25	2.31	3.71
SATA SSD	2.10	31.86	155.08
SATA HDD	33.83	195.57	393.89

DRIVE	MIN (ms)	AVG (ms)	MAX (ms)
NVMe SSD	8.00	17.14	118.00
SATA SSD	10.00	87.28	346.00
SATA HDD	224.00	310.66	411.00

Tabel 2: Search-Verification phase latency: *Vault-76 (left)*; *ChiaPoS (right)*;

CONTRIBUTIONS

- Improved PoSpace performance (in both write throughput and search time) by up to an order-of-magnitude across various hardware.
- Reduced I/O requirements and improved non-volatile memory longevity.

CONCLUSIONS AND FUTURE WORK

- *Vault-76* demonstrates significantly higher throughput and lower latency compared to *Chia's* plotters and effectively leverages multiple threads, with increased thread counts consistently leading to higher throughput.
- When writing to HDD, the bottleneck is disk speed, minimizing differences between Rust and C. However, on NVMe, Rust outperforms C due to native BLAKE3 support and superior parallel hash generation.
- The search-verification phase is highly optimized, achieving faster speeds than lookup in Chia.

Future Work:

- Implementing memory caching and compression, as shown in Fig. 6, to further enhance the performance of the Rust implementation.

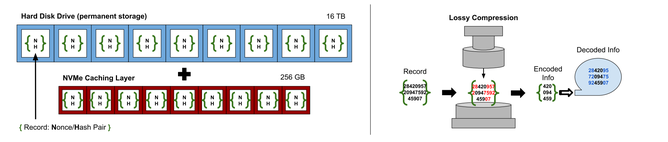

Fig. 6: Memory caching & compression

REFERENCES

[1] URL: https://docs.chia.net/ssd-endurance/.

[2] Aahad Abubaker *et al.* „Exploring Green Cryptographic Hashing Algorithms for Eco-Friendly Blockchains" (2023).

[3] AWS. *What is Compute?* Toim. Amazon. [Online; 2024]. 2024. URL: https://aws.amazon.com/what-is/compute/.

[4] Al Orhean *et al.* „Mystic: Programmable systems research testbed to explore a stack-wide adaptive system fabric". Teoses: *8th Greater Chicago Area Systems Research Workshop (GCASR)*. 2019.

[5] Jack O'Connor *et al. one function, fast everywhere.*

ACKNOWLEDGEMENTS

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

CONTACT: Varvara Bondarenko (vbondarenko@hawk.iit.edu), Renato Diaz (renato.diaz@ucf.edu)