PROOF OF SPACE

John Chmura

Introduction

- Proof of Space is a proposed solution to the massive amount of electricity used in Proof of Work blockchains.
- One popular Proof of Work coin is Bitcoin, and it is responsible for a bulk of this energy usage. Some major problems caused by this extreme power (in Bitcoin alone) include blackouts, high CO2 emissions, and over 10.8 billion dollars spent annually in the US alone.

- Source: https://www.cryptoninjas.net/crypto/study-bitcoin-mining-impact

Innovation

- Some blockchains have started to move away from the Proof of Work.
- Programmatic Proof of Work - Tried to make mining more accessible.
- Proof of Stake - Users stake their assets in order to show trust. Some flaws in balancing, as the more wealth you have the more power you hold in the blockchain. (Ethereum)
- Proof of Space and Time - Users plot their hashes in large files. Uses a time component to ensure that the plots were already stored. This prevents creating plots on the fly. (Chia)

Chia Coin

- An adopter of Proof of Space and Time.
- Goal is to be a "green" alternative to energy hungry Proof of Work coins.
- Users plot their hashes all at once and farm these hashes as new challenges appear.
- Uses a "Timelord" to run variable delay functions which standardize the time between blocks. These Timelords provide proof that a certain amount of time has passed between blocks by running an un-parallelizable function (VDF).
- This time component prevents grinding attacks, on-the-fly plotting, and a CPU component to winning a block. It effectively levels the board by slowing everyone down.

Proposed Solution

- Use "green hashing" and Proof of Space in order to lower the amount of electricity needed to power blockchain technology.
- Faster and more efficient hashing functions can greatly reduce the compute power for generating these plots. Focus on speed rather than password security.
- SHA-256 is an example of an old hashing algorithm that’s slow and inefficient. By switching from functions like SHA-256 to more modern and parallelizable functions such as BLAKE3, you increase the throughput astronomically decreasing the initial compute power required for farming in Proof of Space.
- Source: Abubaker et al., "Exploring Green Cryptographic Hashing Algorithms for Eco-Friendly Blockchains", SC’23

BLAKE3

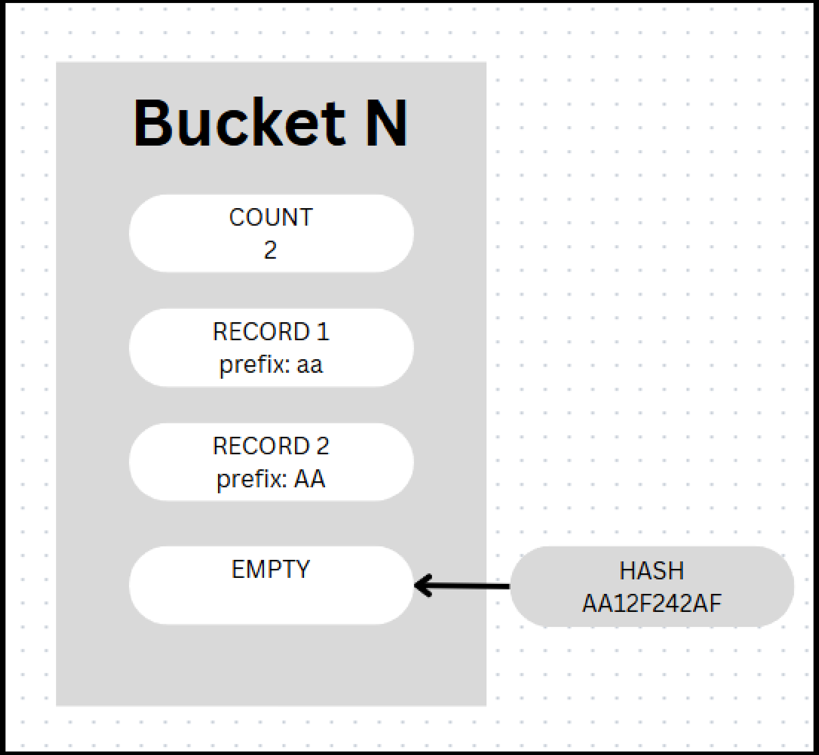
- A fast modern hashing algorithm. Benchmarked at 33.15 GB/s while SHA-256, the current popular blockchain hashing function, ran at 7.987 GB/s.
- Built with speed and parallelization in mind by utilizing a binary Merkle tree structure. The leaves can be ran independently and then merged in chunks rather than sequentially hashing a piece of data.

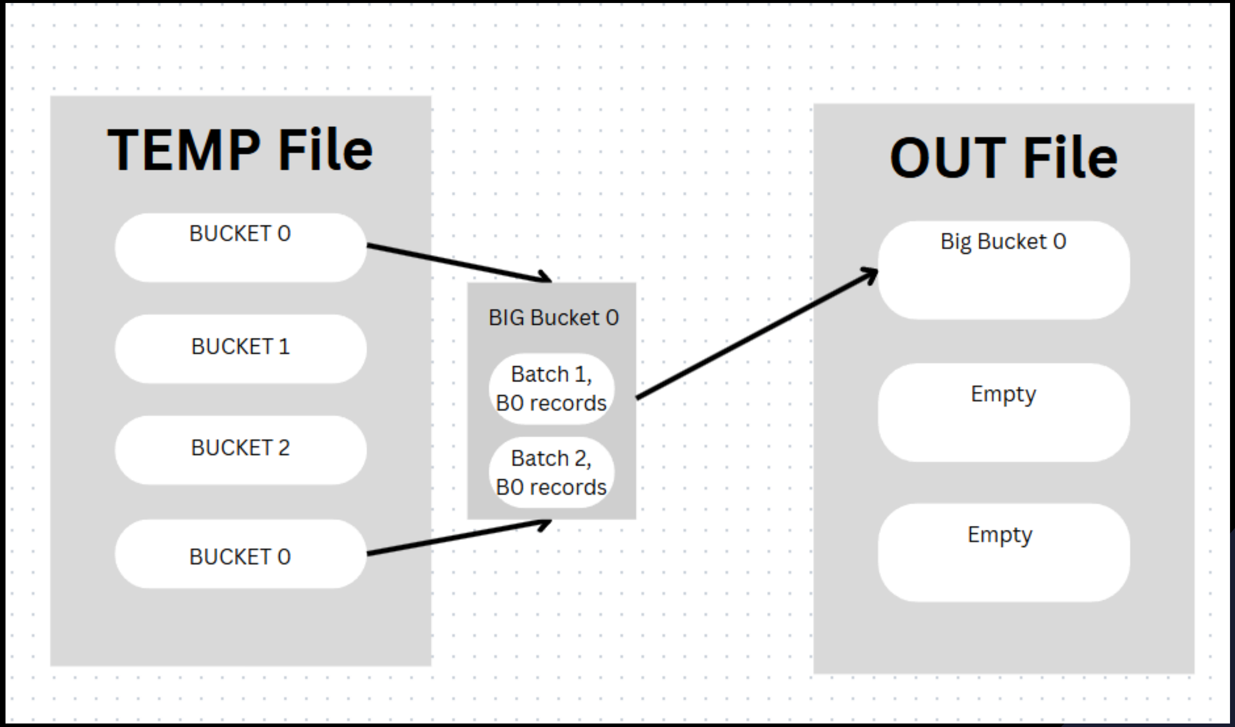
- Source: Abubaker et al., "Exploring Green Cryptographic Hashing Algorithms for Eco-Friendly Blockchains", SC'23
- Source: https://github.com/BLAKE3-team/BLAKE3/ (The BLAKE3 REPO and paper)

Implementation

- My implementation of a Proof of Space table consists of using BLAKE3 as the backbone to quick hash generation. By multi-threading the already light-weight BLAKE3, throughput is increased heavily.
- On top of generating the initial plot - lookup speed is also needed.
- Therefore, buckets are needed in order to minimize time spent inserting, moving, and searching for these hashes.

The Idea


Bucket N
COUNT
2
RECORD 1
prefix: aa
RECORD 2
prefix: AA
EMPTY
HASH
AA12F242AF


TEMP File
BUCKET 0
BUCKET 1
BUCKET 2
BUCKET 0
BIG Bucket 0
Batch 1,
B0 records
Batch 2,
B0 records
OUT File
Big Bucket 0
Empty
Empty

Implementation Continued

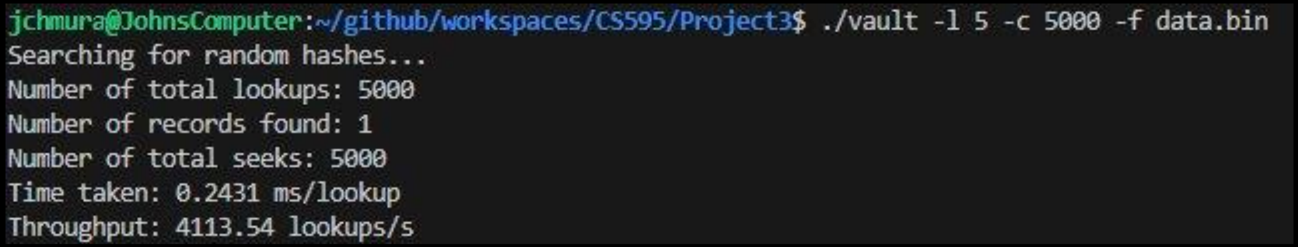
- I used C to implement my proposed solution to a Proof of Space system. It was about 1200 lines of code (including spaces).
- I used the OpenMP library to multi-thread the hash generation and merging/sorting of the buckets.
- Generate all the records (hash and nonce pairs)
- Each time one is generated find the bucket it belongs to by jumping to the bucket at the index of the hash prefix
- Dump all the buckets into a temp file
- Repeat until temp file is filled with all the records
- Go back in and merge the buckets of the same index together and sort
- Dump these "big" buckets into the actual output file

Performance

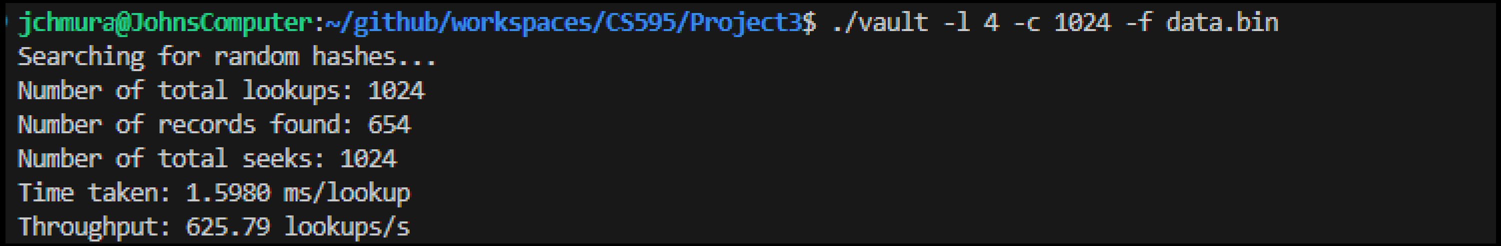
THREADS	MEMORY (MB)	FILE SIZE (MB)	K	TIME (S)	MB/S	MH/S
16	1024	1024	26	3.87	277.46	17.34
16	1024	2048	27	7.22	297.52	18.59
16	1024	4096	28	13.82	310.84	19.43
16	1024	8192	29	30.47	272.6	17.04
16	1024	16384	30	59.12	290.58	18.16
16	1024	32768	31	122.42	280.67	17.54
16	4096	65536	32	271.01	253.56	15.85

Searching

- Lookups are done by indexing to the prefix of the hash and then performing a binary search on the grouped together buckets.



```
jchmura@JohnsComputer:~/github/workspaces/CS595/Project3$ ./vault -l 5 -c 5000 -f data.bin
Searching for random hashes...
Number of total lookups: 5000
Number of records found: 1
Number of total seeks: 5000
Time taken: 0.2431 ms/lookup
Throughput: 4113.54 lookups/s
```



```
jchmura@JohnsComputer:~/github/workspaces/CS595/Project3$ ./vault -l 4 -c 1024 -f data.bin
Searching for random hashes...
Number of total lookups: 1024
Number of records found: 654
Number of total seeks: 1024
Time taken: 1.5980 ms/lookup
Throughput: 625.79 lookups/s
```

Conclusion

- This project was a success because I was able to decrease the time taken to generate the hashes and sort them using multi-threading.
- Also being able to fully generate the 64GB file was a challenge at first, but after trying different ways of threading and moving things around I was able to make it dump the final records in sorted order.
- In the future I could implement a second table that stores the hashes that were close enough together to create valid hashes, as talked about in lecture.

Sources

- https://www.cryptoninjas.net/crypto/study-bitcoin-mining-impact
- Abubaker et al., "Exploring Green Cryptographic Hashing Algorithms for Eco-Friendly Blockchains", SC'23
- https://github.com/BLAKE3-team/BLAKE3/ (The BLAKE3 REPO and paper)