Chia Network and Accelerated Plotter

Ting-Wei Su
Texas A&M University
College Station, United States
willytwsu@tamu.edu

ABSTRACT

Chia network, a new decentralized network utilizing the proof of space and verified delay function is a recent spotlight in blockchian which came live around May of 2021. The utilization of disk to contribute proof of space is not only innovative but also ecologically friendly. Compared with the most well-known cryptocurrency, bitcoin, the energy consume in chia network is extremely less than the power required during bitcoin mining.

1 INTRODUCTION

Recently, Blockchian has become a new popular terms and area people talk about. Not only does it has tons of innovative application in technology but also the impact it impose is also huge which will overturn the world that we live in. Decentralized is the key component that catch everyone's attention. Ranging from financial technology, semiconductor industry, entertainment industry and art collection, blockchain is providing a whole new architecture of protecting and contributing the system integrity and security at the same time. The key concept that blockchain was based on decentralized consensus is like a voting system where every node participating in the network could have equal contribution to the decision of the network.
Bitcoin, as the most famous cryptocurrency, has been making a huge impact to the blockchain society and the whole world. Different kind of consensus theory has been proposed ever since the launch of bitcoin. As the first truly decentralized cryptocurrency, Bitcoin launched at 2008, invented by the mistyrious anonymous founder Satoshi Nakamoto, is not only the forerunner but the motivation to any application in the blockchain expertise.
Etherrum, intially started with a more robust scripting language a Turing-complete programming language, was a improvement to bitcoin core algorithm. Not only does it slight tune the 21 million limitation of Bitcoin but the extension on decentralized contract system has triggered many different application accross financial technology and art exhibition. With the recent migration of the original consensus protocol, proof of work, to a more stable protocol, proof of stake, Ethereum is constantly become a better network in all aspect than its predecesor, Bitcoin. Proof of space is noval consensus protocol that was proposed recently since 2017. Chia network is one of network that utilize the proof of space consensus theory

2 CHIA NETWORK

Chia was founded in May 2021 by Bram Cohen, who was known for the peer-to-peer file-sharing internet protocol, BitTorrent. The intention of building the chia decentralized network is to improve the energy consumption spend on Bitcoin mining by utilizing a new consensus, proof of space, as the fundamental of the network. Chia is a decentralized consensus network that uses the combination of Proof of Space and Proof of time (VDF) as the fundamental consensus algorithm.
When a person owns a storage disk with 1 GB, the person can request to join the network by implementing the network protocol to claim and tell the network that he or she has a storage disk with 1 GB. How do we let the network know that you did actually hold a 1 GB storage space? In order to prove you have 1 GB space, we could simply give you a 1GB file and tell you to store the file on your disk. When we need a specific segment of information in the file, we request it from you, you should be able to pull out the exact same information.
In order to store the user information in the File that the network gave to the farmer and create the proof of space of earning the rewards, three stages are required in the participation of Chia network: In the following chapters, we will discuss more in detail. Chia network is composed of three roles, timelord, farmer, and verifier. Whenever timelord releases the lottery, a sub-plot, to the network every ten minutes, farmers who have a completed plot file will construct a proof of space from the plot file they store and send the proof of space to the verifier to verify. When the verifier verifies that the exact proof of space corresponds to a sub-slot, the rewards will go to the selected farmer. The reward for successfully finding the valid proof of space is 2 XCH. Why the block generation rate is 10 mins per block
Three steps have to be completed before winning the rewards.
Plotting Plotting will store a huge amount of hash data in a plot file in order for the farmer to quickly fetch the proof of space in the next step, farming. Not only the hash data but the checkpoint data, which will assist in faster proof of space retrieval, will be stored in the plot file. The plotting works from table 1 through table 7.
Farming Farming is the process of consrtucting prove to the network that ones actually owns a specific amount of disk space. To begin with, the network will release a challenge to every farmer who has participating in the game. Upon receiving the challenge, farmer goes to the last table, table 7, to find the matching data along with the corresponding checkpoint, stored in plot file, subsequently traverse the search by the matching data in the previous table, table 6, and following this routine to the first table, constructing a proof of space and send it out for verifying. In reverse than plotting, farming retrive data from the last table.
Verifying The verifying stage require **Plot filter** The plot filter is a threshold to decide whether the challenge has any possibility of getting rewards. It is used to prevent the farmer on finding proof of space that one's doesn't have. Farmers receive these signage points and compute a hash for each plot, at each signage point. If the hash starts with nine zeros, the plot passes the filter for that signage point, and can proceed. This disqualifies around 511/512 of all plot files in the network, for that signage point. The formula to compute

the filter hash is

$$\text{plot filter bits} = sha256(\text{plot id} + \text{sub-slot challenge} + \text{cc signage point})$$

Sub-slot A segement of challenge that is generated by the timelord and will be issue to the farmer. The format of the sub-slot is as below

```
class ChallengeChainSubSlot(Streamable):
  challenge_chain_end_of_slot_vdf: VDFInfo
  infused_challenge_chain_sub_slot_hash: [bytes32]
  subepoch_summary_hash: [bytes32]
  new_sub_slot_iters: [uint64]
  new_difficulty: [uint64]
```

Singage Point The sinage point is a temporarily stage for sub-slot generating. Within 64 iteration of the sinage point, composing a complete sub-slot, each sinage point will be release by the timelord to the farmer every 9.375 seconds. The farmer, upon receiving the sinage point will compute the plot filter to see if it pass the threshold.

3 CHIA PLOTTING PROCESS

The plotting process could be analogous to the work done, such as fertilizing and irrigating then cropping the seed in the farm, while farming could be analogous to the process of harvesting the product. It begins with storing many hash-like number in a file that stores on the farmer disk. Seven tables is initiate in the plot file with a specific plot constant k. Two parameters will be required when plotting initialize. A plot seed, which is received from the network, a plot constant k, which is assigned by the farmer.
Plot constants k The plot constants k represent the number of entries each table, which is determine by the farmer. The minumum disk space of the plot with $k = 32$ is 515.4 (GB) for temperarily storage and 104.7 (GB) for final storage.

$$\text{Number of entries in each table} = 2^k, 32 \leqq k \leqq 50$$

$$\text{Temporarily plot file size} \approx 3.75 \times k \times 2^k (byte)$$

$$\text{Final plot file size } \approx 0.762 \times k \times 2^k (btye)$$

Plot seed The plot seed is a 32 byte constant received from the network to initialize each plot file, so that no two plot file will be the same.

$$\text{plot\_seed} \in [2^{256}]$$

That is said, all the farmer need within the plotting process is the plot seed receiving online, everything else will be able to construct offline. The design of chia plotting process is intentionally design so that whomever gets a powerful machine, such as a ASIC or GPU won't create absolute advantage over those who don't, because the main motivation is letting the farmer purely use The plotting process will be distinguish into the following four parts.
1. Forward propogation The forward propogation is to insert data in each entries of the table, from table 1 to table 7. The dependency of the table is table i + 1 will relie on the talbe i (i = 0, 1, ... 6). Because this dependency, farmer who wish to accelerate the plotting process will not have the ability to predict what the next table content without waiting for the completion of the previous table. This will create a bottle-neck for those who wish to create a faster plotting method using a high performance machine. Other than the first stage, the rest are extra optimization feature created by Chia network. With a fully propagation plot file, the farmer are allow to construct the proof of space. But considering the efficiency, Step 2 to 3 are recommended to operating, in order to reduce unused data and make good use of the disk space
2. Backward propogation Backward propogation is to reduce the size of each table by getting rid of those entry that's not in use. In each table, a left entry will contain data from the previous table, right entry contain data it calculate. The entry is useful to the next table, only if the left entry is equal to the right. Therefore, in this stage, it is dropping unused data where the left is not equal to the right.
3. Compression The compression stage is to compacting the content in each table and create a format for the efficiently retrival with bucket id. The
4. Checkpoint Checkpoint will create a faster index for future proof retrieval.

Chacha8 and Blake3 hash-like function The fundamental of chia plot generation is based on two hash functions, chacha8 and blake3, where chacha8 is used in the gerneration of the first table and blake3 is used in generation of the rest 6 tables. The hash function will fetch in a **Table 1** The generation of the table 1 required iteration through the entire entries with value up to $2^k - 1$, which is more than 4 billion entries with k32 plot.

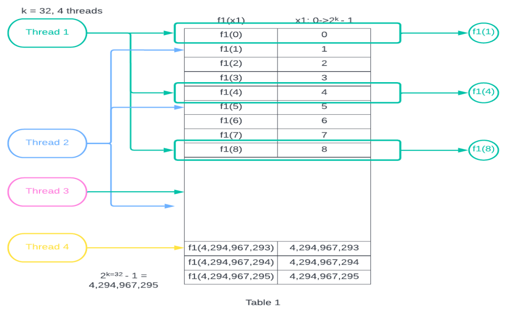

Fig. 2 The generation of table 1 required at lease 4 billion iteration in a k32 plot file.

Dependency between table (i) and table (i + 1) The dependency of table i and table i + 1 lie in that table i + 1 will required the matching entries in the left and right to calculate the offset of the tale i.

4 RELATED WORK

Approach by madMax One of the contributors of the open-source project, XPM GPU Miner, [1], has successfully provided a highly parallelize architecture to support more efficient parallel computing to accelerate chia plotting by creating multiple threads on a multi-core processor. The result of his is 17 seconds for the table 1 using a 256 GB RAM dual Xeon® E5-2650v2@2.60GHz R720 processor with 16 threads. The method he uses is to create a thread

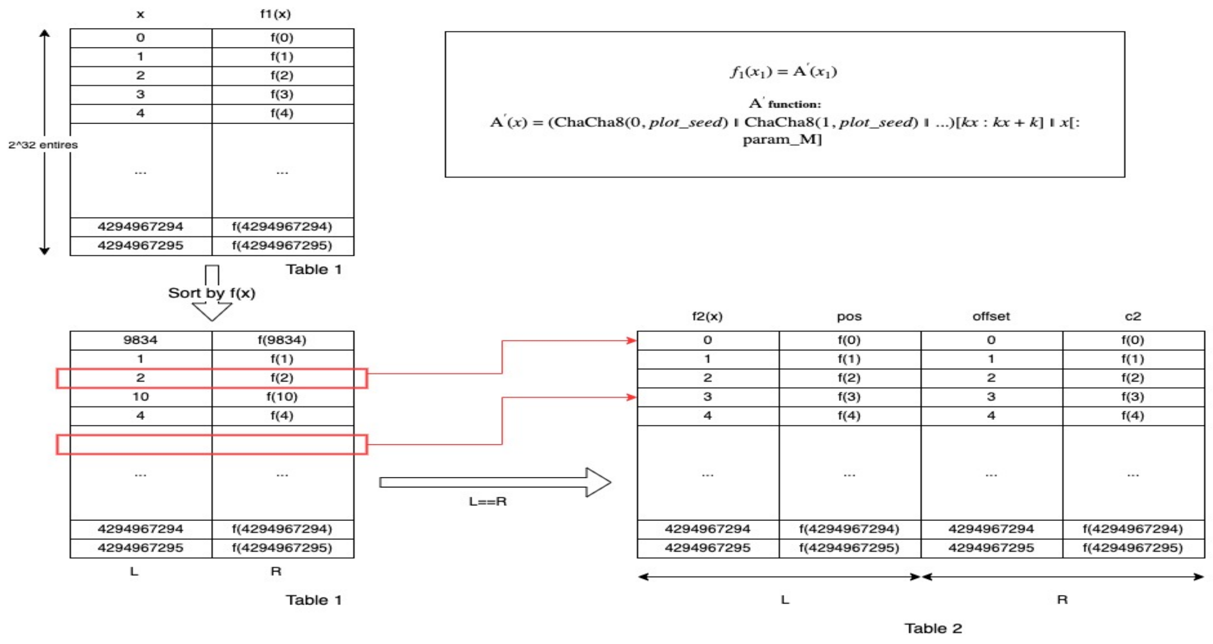
x
f1(x)
2^32 entires
Table 1
Sort by f(x)
L
R
Table 1
L==R
f2(x)
pos
offset
c2
L
R
Table 2
$f_1(x_1) = \mathrm{A}'(x_1)$
A′ function:
$\mathrm{A}'(x) = (\mathrm{ChaCha8}(0, plot\_seed) \parallel \mathrm{ChaCha8}(1, plot\_seed) \parallel \ldots)[kx : kx + k] \parallel x[: \mathrm{param\_M}]$

Fig. 3 The dependcy between table 1 and table 2.

pool manager in managing the status of each thread. The thread pool manager would not benefit from the table 1 generation since all the entry calculation is done in parallel. However, the thread pool manager could benefit from the matching function in the rest of the table generation, from table 2 to table 7. When generating table 2, some thread will be doing the matching calculation, some threads will drop those that is not in use. Therefore, introducing the locking mutex in the first stage of table 2 to table 7 is necessary. In addition, madMax solution also provides a larger RAM buffer usage. Take the first stage of k32 table 1 plot file generation, the disk operation wouldn't be operating until the all entry calculation is done. About 68 (GB) of memory space will be occupied if not flushing the data to disk during table 1 plotting.

$$\begin{aligned}\text{Temperarily RAM usage} &= 2^k|_{k=32}(entry/table) \times 16(bytes/entry)\\ &= 6.872 \times 10^{10}(Byte)\end{aligned}$$

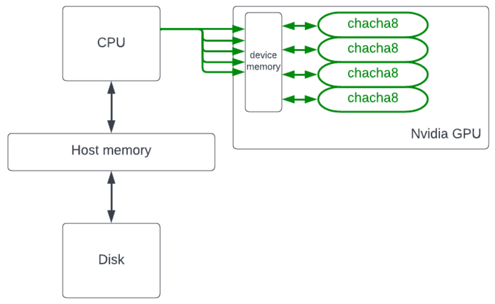

Fig. 4 The architecture of GPU accelerated computing Chia plot.

5 APPROACH WITH PARALLEL COMPUTING

My intention of accelerating the Chia plotting starts from a post [2], stating that it is impossible to use GPU to assist the plotting process. The reason is that using GPU to the plot will require a huge amount of temporary memory usage. If the device memory is not large enough, it will result in disk I/O operation penalties. Take k32 plot, for example, it will require more than 36 GB of memory space for storage in order to calculate table 1. The maximum device storage on an Nvidia GPU is less than 30 MB per node. Therefore, using GPU to assist the plotting, we still have to store the temporary plot file somewhere before writing back to disk space. **Software architecture** Adding the threading queue in front of GPU kernel function. Memory copying from host to device side is costly, and the approach is to copy data together to the kernel device at once to reduce the cost of issuing memory copy seperatly. The queue will collect data derived and iterated from the table 1 function and issue to the kernel when the queue is collecting enough data for GPU to calculate at the same working cycle.

Regarding the synchronize issue among all different thread, no mutex is requird for the generation of the 1st table. Since each entry is independent to each other, so the writing to disk would be affected by the lock mechanism.

6 RESULT - WITHOUT GPU

The implementation of mine is based on the calculation of the table 1. The generation of table 1 will require the chacha8 hash function which could be done in parallel. The result running on the TAMU Grace super computer cluster is approximately 900 seconds for k32 plot file and 1833 seconds for k33 plot file. The temporarily storage in k32 plot file is around 36 GB. While k33 plot file is around 76 GB.

Table 1: Frequency of Special Characters

Plot constant k	Thread number	Time (sec)	Average (sec)
32	4	911.52	908.23
	8	908.96	
	16	906.99	
	32	905.46	
33	4	1840.32	1833.53
	8	1840.91	
	16	1825.87	
	32	1827.00	

7 CONCLUSIONS

After evaluating the enhancements and implementation difficulties, I proposed my view of possible solutions to this topic. Using super-computer clusters with accelerated units or with a larger memory space will effectively improve the plotting speed in different aspects, however, the hardware cost is also one aspect everyone dedicating to using this solution has to take into serious consideration.

Conclusively, the result shows positive outcomes with respect to using the official source code on a simple desktop machine. Although future optimization is required to perfect the solution, I could conclude that the survey has left with some positive consequences. [3] [4] [5]

8 FUTURE WORK

The future work will be focusing on improving the threading algorithm and the memory cache issue, with specifically what to keep in the temporarily storage and what not to keep in the temporarily storage.

RAM usage Memory usage has been very cautious, my intention is not to overuse the memory space at the first attempt, therefore, my approach will still be limited by the disk I/O latency whenever writing 256 buckets of entry data from memory to disk during plotting frequently. But as discussed in the madMax solution, caching more data in memory before writing to disk will eliminate a huge amount of time. Therefore, the first optimization I will do regarding the current development is to cache more entries in memory and to write more buckets out to disk at the same event.

Thread Pool A threading pool manager will be added in order to handle the completion of each thread. The existence of the thread pool manager is to assist the rest of the tables when some of the threads might be doing one thing differently from the other. In addition to the entry calculation in table 2, the matching function search through table 1 will also be one of the tasks that could be done in parallel along with the entry calculation.

9 REFERENCE

[1] Max - madMAx43v3r. *Chia Plotter*. Github. URL: https://github.com/madMAx43v3r/chia-plotter.git.

[2] Christian Balcom. *Why not GPU*. Github. URL: https://gist.github.com/computer-whisperer/42b8507b2fb506faf0b1c9ee1189cd58.

[3] Gene Hoffman Bram Cohen. *Chia Network*. Github. 2021. URL: https://github.com/Chia-Network.

[4] Gene Hoffman Bram Cohen. *Chia Network*. 2021. URL: https://docs.chia.net/.

[5] Dan Boneh, Benedikt Bünz, and Ben Fisch. “A Survey of Two Verifiable Delay Functions”. In: *IACR Cryptol. ePrint Arch.* 2018 (2018), p. 712.