Chia Network and Proof of Space Acceleration by GPU

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

ABSTRACT

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.

Chia network, a new network utilizing the proof of space to verify the farmer for farming power, is a recent spotlight in blockchian forum which came live at June of 2021. The utilization of disk to contribute proof of power is not only innovative but also ecologically friendly. Compared with the most well-known cryptocurrency, bitcoin, the energy consume in chia network is 100 times less than the power required during bitcoin mining.

The motivation of my approach is to accelerate the process of chia farming, which is analogous to bitcoin mining, via a accerlation unit, a GPU computing node on super-computer.

1 INTRODUCTION

Developed by the founder of bittorrent, Cohn Bram. Seek to improve the energy wasted in the Bitcoin mining.

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 with

what is proof of space? 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 useful information in the File that the network gave it to the farmer and create the proof of space of earn the rewards, three stages are required in the participating of Chia network: In the following chapters, we will discuss more in detail.

Plotting -> Farming -> Verifying Consensus algorithm: VDF Timelord creating the challenge Farmer extracts the proof of space

2 CHIA NETWORK

Firstly, let's start with what is Chia 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. Two kinds of people are included in the network. The farmer, persons who provide the storage space, and the verifier, persons who request the information stored by farmer.

3 CHIA PLOTTING PROCESS

The plotting process is when a farmer preperation work that the farmer has to completed before contributing to the network. The plotting process 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.

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.

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

$$\text{Temporarily plot file size} \approx 3.75 \times k \times 2^k (byte) \text{Final plot file size } \approx 0.762 \times k \times$$

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.

2. Backward propogation Backward propogation is to reduce the size of each table by getting rid of those entry that's not in use.

3. Compression The compression stage is to sort the content in each table.

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. **Table 1** The generation of the table 1 required iteration through the entire entries with value up to $2^k - 1$.

Adaptive DLP. Except for scheduling optimization, [1] proposes a method applying different DLP approaches in specific situations. Although this paper mainly focuses on the kernel utilization, we can treat binary translation processing elements as enormous kernels inside. Data partitioning can be done according to the properties of basic blocks. If the basic blocks are comprised of small items, machine tends to adopt kernel partitioning to fit with the instructions. In contrary, basic blocks with big items are appropriate for inter-kernel strategy.

Kernel Improvement. [2] introduces a solution efficiently applying kernel computation. It duplicates the kernels with potential following computations (multiplication). Machine operates the sequential computation in parallel that decreases the time spending on recursive computations. The additional computation cost is acceptable in exchange of performance reinforcement. [1] also includes improvements about kernels. It demonstrates how to operate kernel partitioning. Small pieces of kernels help accelerate data fetching and computations due to shorter distance between PE and targets.

SIMD Exploitation. With the support of SIMD engine, such as ARM NEON and Intel AVX, the main processor could utilize vectorized code regions to realize DLP. Along with the dynamic Assembler we could extend the DLP to a dynamically detection at the run-time execution. The approach could be achieve by vectorizing the instruction set architecture (ISA) or convert the short instruction into longer instruction.

Previously-mentioned methods are seemingly solutions to deal with assembly sequences. Machine should pay extra cycles to complete their preliminary works. This cause them not capable of universally adopted. In spite that there are approaches available to address the loss in translation, they do not have a in-depth research throughout the whole computer architecture. Current papers simulate the data stream without taking penalty and exceptions into account. They have deficiency though, we can still figure out more and more possibilities following their directions. Notable advances will be made if the algorithm of DLP is optimized and avoid exceptional hardware efforts.

4 APPROACH WITH PARALLEL COMPUTING

Nvidia GPU In [3], researcher also proposes a dynamic SIMD Assembler (DSA) generating the SIMD instruction to trigger the ARM NEON engine by detecting the vectorized code regions. The need of the DSA to detect the vectorized loop run-time is to compensated the weakness of the ARM auto-vectorization compiler count loop detection at compile time. Such condition as dynamic range loop, conditional loop and loop with a function call are three condition that the compiler couldn't possibly has optimized; thus, it will required the assistance from the run-time engine to parallelize the execution. However, the two memory space is reside in the GPU model, the host memory and device memory. In order to invoke
Related Work by Max One of the contributor of the open source project, XPM GPU Miner, [4], has successfully provide highly parallelize archtecture to support more efficient parallel computing to accelerate chia plotting by creating multiple threads on a multi-core processor. The methods he use is create a thread pool manager on the plotting thread in order to synchronize the status of each thread. However, the API he provide is not thread-safe api. Therefore, with the motivation of the ability to parallelize the plotting into each
Software architecture The main Inherited from the source code repositories, I add a 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. The generation

5 CONCLUSIONS

After evaluating the enhancements and implementation difficulties, we propose our view of possible solutions on this topic. Programming GPU plotter has the concurrency issue that we have taken serious handle.

Our survey shows that DLP assists the system performance either in aspect of energy or speedup perspective. Considering computer structure limitations in overall could we find the best architecture to our targeted design. We aim to fulfill a DLP optimization with more practical constraints, e.g. memory dependencies, cold start overhead, data stream in reality with the proposed solutions. By doing so, unexpected conditions have a chance to be addressed. Probably, novel methods will be realized while we address them. [**chia_github_source**] [5]

6 FUTURE WORK

7 REFERENCE

[1] Lili Song et al. "C-Brain: A Deep Learning Accelerator That Tames the Diversity of CNNs through Adaptive Data-Level Parallelization". In: *Proceedings of the 53rd Annual Design Automation Conference*. DAC '16. Austin, Texas: Association for Computing Machinery, 2016. ISBN: 9781450342360. DOI: 10.1145/2897937.2897995. URL: https://doi.org/10.1145/2897937.2897995.

[2] Franjo Plavec, Zvonko Vranesic, and Stephen Brown. "Exploiting Task- and Data-Level Parallelism in Streaming Applications Implemented in FPGAs". In: *ACM Trans. Reconfigurable Technol. Syst.* 6.4 (Dec. 2013). ISSN: 1936-7406. DOI: 10.1145/2535932. URL: https://doi.org/10.1145/2535932.

[3] Ding-Yong Hong et al. "Exploiting Longer SIMD Lanes in Dynamic Binary Translation". In: *2016 IEEE 22nd International Conference on Parallel and Distributed Systems (ICPADS)*. 2016, pp. 853–860. DOI: 10.1109/ICPADS.2016.0115.

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

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