Project Report

Plot generation study in Chia blockchain

CS-554 Data Intensive Computing
Spring 2023
Professor: Ioan Raicu

By
Prajakta Kumbhar (A20523710)
Shlok Mohan Chaudhari (A20525610)
Vishal Gawade (A20524009)

Table of Contents

Section 1: Introduction, Background, and motivation
1.1 Introduction to Cryptocurrencies and the Chia Network
1.2 Proof of Space and Time Consensus Algorithm
1.3 Importance of Chia Plot Generation
1.4 Challenges Faced in Plot Generation
1.5 Motivation for Studying Chia Plot Generation

Section 2: Proposed solution
2.1 Phases in Chia
2.2. Algorithm
- Phase 1: Forward Propagation
- Phase 2: Backpropagation
- Phase 3: Compression
- Phase 4: Checkpoints

2.3 Existing Software and Systems Used

Section 3:Evaluation
3.1 Evaluation Methodology
3.2 Explanation of Experiments, Variables, and Metrics Used
- Memory Consumption
- Average Disk I/O
- Frequency Distribution
- Device Power Report

Section 4: Related Work
4.1 Comparison with Existing Research
4.2 Differences in Our Work

Section 5: Conclusion
5.1 Key Findings
5.2 Future Work

Section 6:References

Appendix

Section 1: Introduction, Background, and motivation

1.1 Introduction to cryptocurrencies and the Chia network

Cryptocurrencies have emerged as a revolutionary digital asset class that offers secure, decentralized, and transparent transactions. Chia, a novel cryptocurrency, introduces an energy-efficient consensus algorithm known as Proof of Space and Time (PoST). This algorithm replaces the energy-intensive Proof of Work (PoW) used in traditional cryptocurrencies like Bitcoin. The Chia network is designed to promote a more eco-friendly and accessible approach to cryptocurrency, which has the potential to reshape the landscape of digital assets and encourage wider adoption. As the interest in and demand for cryptocurrencies continue to grow, the Chia network presents a sustainable alternative to energy-intensive cryptocurrencies, making it an important area of study.

1.2 Proof of Space and Time consensus algorithm

Proof of Space and Time is a consensus algorithm that relies on unused storage space instead of computational power. Users allocate storage space in the form of plots to participate in the Chia farming process. The more space a user allocates, the higher their chances of winning a block reward. This innovative approach to consensus not only reduces the environmental impact of cryptocurrency mining but also democratizes access to the rewards system, enabling users with varied resources to participate in the network. The PoST algorithm is a significant advancement in the field of cryptocurrencies, as it addresses the energy consumption and centralization concerns associated with traditional PoW-based systems.

1.3 Importance of Chia plot generation

Chia plot generation is a crucial aspect of the Chia network, as it determines the storage space required for farming. The process involves creating unique plots that are later used to prove the user's allocated space in the network, contributing to the overall security and stability of the Chia blockchain. Efficient plot generation is essential for optimizing storage resources and maximizing the potential returns from Chia farming, making it a critical component of the network's performance. A thorough understanding of Chia plot generation helps users make informed decisions about hardware requirements, resource allocation, and potential returns on investment in Chia farming.

1.4 Challenges faced in plot generation

Chia plot generation can be time-consuming and resource-intensive. It requires significant computational power, storage space, and memory, which can pose challenges to users with limited resources. Additionally, the process can generate a substantial amount of disk I/O, potentially impacting the longevity of storage devices. Identifying and addressing these challenges is crucial for ensuring the sustainability of the Chia network and fostering greater participation from users across diverse hardware configurations. As the Chia network continues to grow and evolve, understanding the intricacies of plot generation will become increasingly important in maintaining the network's efficiency and ecological sustainability.

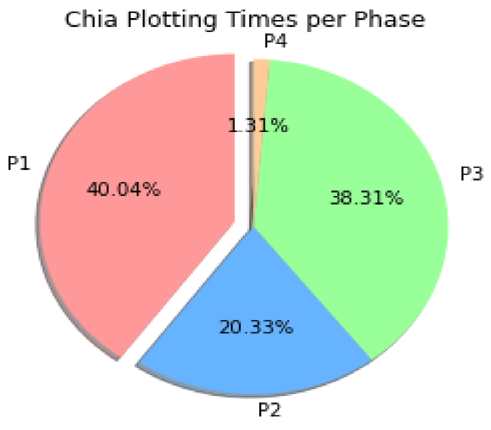
1.5 Motivation for studying Chia plot generation

This research aims to provide a comprehensive understanding of Chia plot generation, its technical aspects, and its impact on the overall Chia network. A better understanding of this process will help users and developers optimize the Chia network and improve its overall efficiency. By analyzing the resource consumption, performance bottlenecks, and opportunities for optimization, this study will contribute valuable insights into the Chia plot generation process and inform future developments in the Chia ecosystem. Furthermore, this research will facilitate a deeper understanding of the PoST consensus algorithm's practical implementation, helping to bridge the gap between theory and practice, and empowering users to make well-informed decisions when participating in the Chia network.

Section 2: Proposed solution

The process of plot generation in Chia is a complex and resource-intensive task that is essential for the platform's proof-of-space-and-time consensus algorithm. The plot generation process involves four stages, each with a specific function in constructing the evidence of space needed for validating blocks on the blockchain. The first stage involves creating seven tables of cryptographic hashes and saving them to a temporary directory. The second stage involves back-propagating through the hashes, while the third stage involves algorithmically compressing and sorting the hashes in the temporary directory. In the fourth and final stage, the compressed and sorted hashes are combined to construct the final plot file and sent to its designated location. We propose to study in detail all these phases in plot generation algorithm in order to gain a better grasp of how the Chia network functions and what makes it unique in the world of blockchain technology.

2.1: Phases in Chia:



1. Plotting time in chia

Phase 1 (Forward Propagation) is the most time-consuming phase, taking 40.04% of the total plotting time. This is expected since this phase involves computing all tables and final outputs (f_7).

Phase 2 (Backpropagation) takes 20.33% of the total plotting time. This phase focuses on optimizing the plot generation process by removing unnecessary data and adjusting positions in the next table.

Phase 3 (Compression) takes 38.31% of the total time, making it the second most time-consuming phase. It involves converting the data format and compressing it for more compact storage.

Phase 4 (Checkpoints) takes the least amount of time, at only 1.31% of the total plotting time. This phase is mainly concerned with compressing tables into checkpoint tables for efficient storage and retrieval.

Overall, Phases 1 and 3 are the most time-consuming, while Phases 2 and 4 require significantly less time to complete. This observation can be useful in guiding optimization efforts, as targeting the most time-consuming phases could lead to more significant improvements in overall plotting time.

2.2. Algorithm:

The chia plotting algo is an algorithm that takes a single input called plot_seed, which is a number between 1 and 2256, as well as a space parameter between 30 and 50, denoted as k, and a param object. The algorithm then produces a file called F in a deterministic manner.

On the other hand, Sort or Sort on Disk is a process that takes a table stored on a disk as input and performs a complete sort in ascending order, starting from a specified bit position.

Phase 1: Forward Propagation

During the forward propagation phase, the main goal is to calculate all tables where the matching conditions are satisfied and generate final outputs, denoted as f7. For each Tablei, we compute the corresponding f function on the x values (which are actually the collated values), and store the outputs in Tablei+1, along with their corresponding positions in Tablei, and the collated x values.

Next, the new table is sorted using Sort to arrange it in ascending order based on output fi. This makes it easy to check for matching conditions, as adjacent entries will have similar bucket_ids. Matches can be identified by reading groups of entries from disk using a sliding window approach. Here, C3 to C7 refer to the output of the collation function for each n.

In Phase 1 and 2, pointers are stored in the pos, offset format. This format represents the position and offset of an entry in the corresponding table. In Phase 1, entries are located close to their matches, making it efficient to store the offset along with the position.

However, this format will be replaced by the double pointer format, which stores two k-bit pointers for each entry. To reduce the size of each entry significantly closer to k bits, the double pointer format uses a combination of sorting and compression techniques.

```
# 1 Forward propagation phase
For x in 0...2^k - 1: Compute f1(x)
Write (f1(x), x) to table1
For table in 1..6:
Sort tablei by (fi(x), pos, offset). for table1, by f1(x) For entry in tablei:
if entries L and R match:
Compute fi+1(CL, CR) for table1, M=x
C = Collation_i(CL, CR)
Store (fi+1, pos, offset, C) in tablei+1
```

Phase 2: Backpropagation

Note that after Phase 1, the temporary file contains sufficient information to create proofs of space. The f7 value can be looked up, and the tables can be traced back to the x values in T able1. However, this approach is not space-efficient.

The main purpose of the backpropagation step is to remove unnecessary data that is not required for finding proofs. In this phase, all entries that do not form part of the matching condition in the next table are dropped, and the positions in the next table are adjusted accordingly. Additionally, the Collation outputs (C values) are also dropped since they are only useful for forward propagation. To make the compression phase more efficient, each entry is assigned a sort_key corresponding to its position in the table. This is more efficient than storing (ft,post−1,offset) to represent its position.

The implementation of this phase involves iterating through two tables simultaneously, a left and a right table, and checking which left entries are used in the right table. The entries that are not used are dropped, and a sort by position is performed before looking at the right table. This enables the reading of a window into both tables in a cache-local manner without having to read random locations on disk.

```
# 2 Backpropagation phase
For table in 6..1:
Iterate through tablei and tablei+1:
Drop unused entries in tablei
sort_key = table == 7 ? fi : tablei+1pos
Rewrite used entries in tablei as (sort_key, pos, offset) Rewrite entries in tablei+1 as (sort_key, pos, offset)
If i > 1:
Sort tablei by (pos, offset)
```

Phase 3: Compression

Phase 3 aims to convert from the (pos, offset) format, which necessitates sorting the tables based on their bucket_ids to retrieve the entries, to a double pointer format. This format enables storing entries in a more compressed way while sorting them by line_point. In the pos offset format, approximately k bits are required for the pos and 8 bits for the offset, resulting in a total of approximately k + 8 bits per entry. In contrast, the double pointer format requires an average of only 2 bits of overhead per entry.
In the compressed format, two pointers can be stored in approximately k + 2 bits using the techniques outlined in the Compressing Entry Data section. The objective of the double

pointer format is to store two k bit pointers to a prior table (2k bits of information) as compactly as possible. To accomplish this, each entry Et,i is represented as pointers to the previous table, which is sorted by line_point. As a result, positions are no longer sorted by matches/bucket_id, and these 2k bit line_points are used to encode the deltas between each one, which are roughly k bits each. This phase necessitates careful sorting and mapping from old positions to new positions since we have less memory than disk space.As we iterate from T able1 to T able7, the compression is performed, and each compressed table is written to the final file F.
The final output after phase 3 will be stored in file F, except for the checkpoint table and Table7.
The main bottlenecks in the reference implementation are searching for matches, and sorting on disk.

```
# 3 Compression phase
For table in 1..6:
Iterate through tablei and tablei+1:
Read sort_key, pos, offset from tablei+1
Read tablei entries in that pos and offset: eL, eR y, x = sort(eL.newPos, eR.newPos)
line_point = x*(x-1)//2 + y
Rewrite tablei+1 entry as (line_point, sort_key)
Sort tablei+1 by line_point
For entry e, i in enumerate(tablei+1):
newPos = i
Rewrite e as (sort_key, newPos)
Write compressed e.line_point deltas to table Pi
Sort tablei+1 by sort_key
```

Phase 4: Checkpoints

Phase 4 involved iterating through T.Table7, writing F.Table7, and compressing the f7s into the checkpoint tables.

```
# 4 Checkpoints phase
For entries in table7:
Compress f7 entries into C1,C2,C3 tables Write pos6 entries to table P7k
```

Note:

The reason Chia plotting performs complex steps like deltafying and encoding instead of just creating hashes and storing them on disk is because the goal of Chia plotting is to create proofs of space that are verifiable in a decentralized and trustless manner.

To achieve this, the plots must meet certain criteria, such as having a certain size, being unpredictable, and being resistant to attacks. Simply storing hashes on disk would not be sufficient to meet these requirements, as it would be too easy to generate plots that do not meet the criteria.

By using complex steps like deltafying and encoding, Chia plotting creates plots that are more difficult to predict and attack, while still being verifiable. These steps also allow for efficient storage and retrieval of plot data, which is critical for the Chia blockchain to function properly

2.3 Existing software and systems used:

We utilize the Chia blockchain, various plot generation tools, and monitoring tools to evaluate the performance of the plot generation process.

PowerTop: An open-source, command-line utility developed by Intel that provides real-time, detailed information on power consumption and energy efficiency in Linux systems.

htop: A real-time, interactive system-monitor for viewing the running processes, CPU usage, memory usage, and system load.

iotop: A tool that displays disk I/O usage by processes or threads on the system, which can be helpful in monitoring disk activity during Chia plotting.

nmon: A system performance monitoring tool that provides information on CPU, memory, disk I/O, network, and other system metrics in a single screen.

glances: A cross-platform system monitoring tool that offers a real-time, comprehensive overview of system performance, including CPU, memory, disk I/O, network, and more.

sysstat: A collection of performance monitoring tools, including sar, iostat, and mpstat, which can be used to gather and report system performance data.

Section 3: Evaluation

3.1 Evaluation methodology

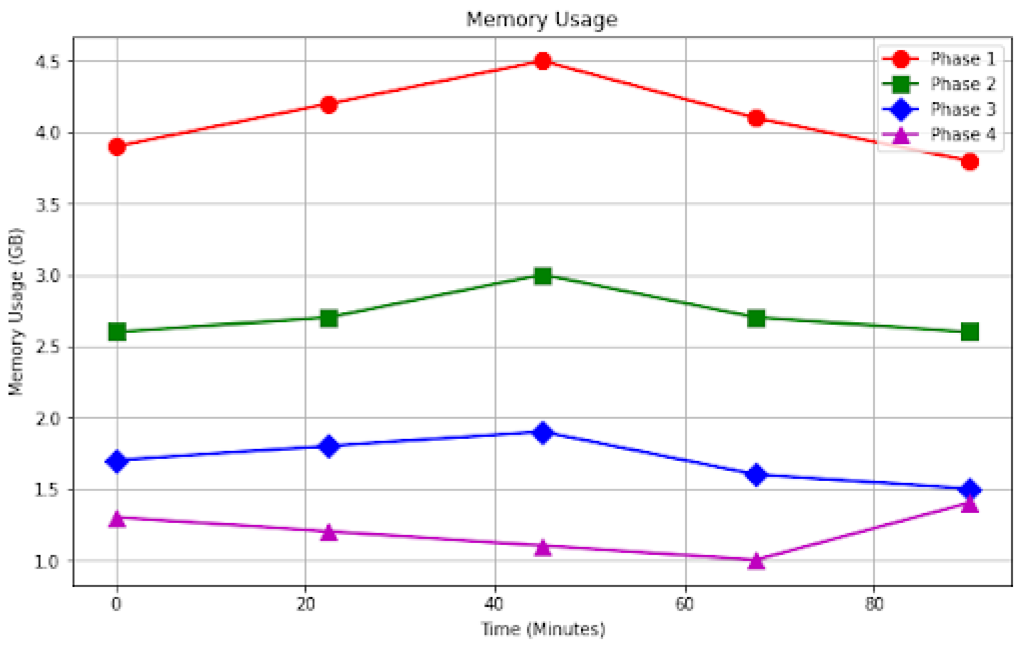
In this study of Chia plot generation, we implemented a robust evaluation methodology to accurately assess various performance metrics. Our primary focus was on memory consumption, average disk I/O, frequency distribution, and device power reports.

To ensure a thorough analysis, we began by designing controlled experiments for each performance metric under investigation. For memory consumption, we monitored and recorded the usage of system memory throughout the plot generation process using specialized tools. Similarly, we captured average disk I/O by tracking read and write operations on the storage devices during the various phases of plot generation. To examine frequency distribution, we assessed the distribution of resource utilization patterns over time, thereby gaining insights into potential bottlenecks or areas of high demand. Lastly, we generated device power reports to evaluate the energy efficiency of the system and the impact of the Chia plotting process on overall power consumption. By employing a systematic and comprehensive approach, our evaluation methodology provided valuable insights into the performance of Chia plot generation and the potential avenues for optimization.

3.2 Explanation of experiments, variables, and metrics used

Our experiments involve testing various plot generation tools under different conditions, such as varying storage space, memory, and processing power. We measure metrics such as plot generation time, CPU usage, memory usage, and storage space efficiency.

Memory Consumption:



2.1. Memory Usage

Phase 1, memory usage is relatively high, fluctuating between 3.9 GB and 4.5 GB. This phase, which involves creating a series of temporary files, demands a significant amount of memory resources to efficiently process the data. The peak memory usage in this phase is observed at 4.5 GB.

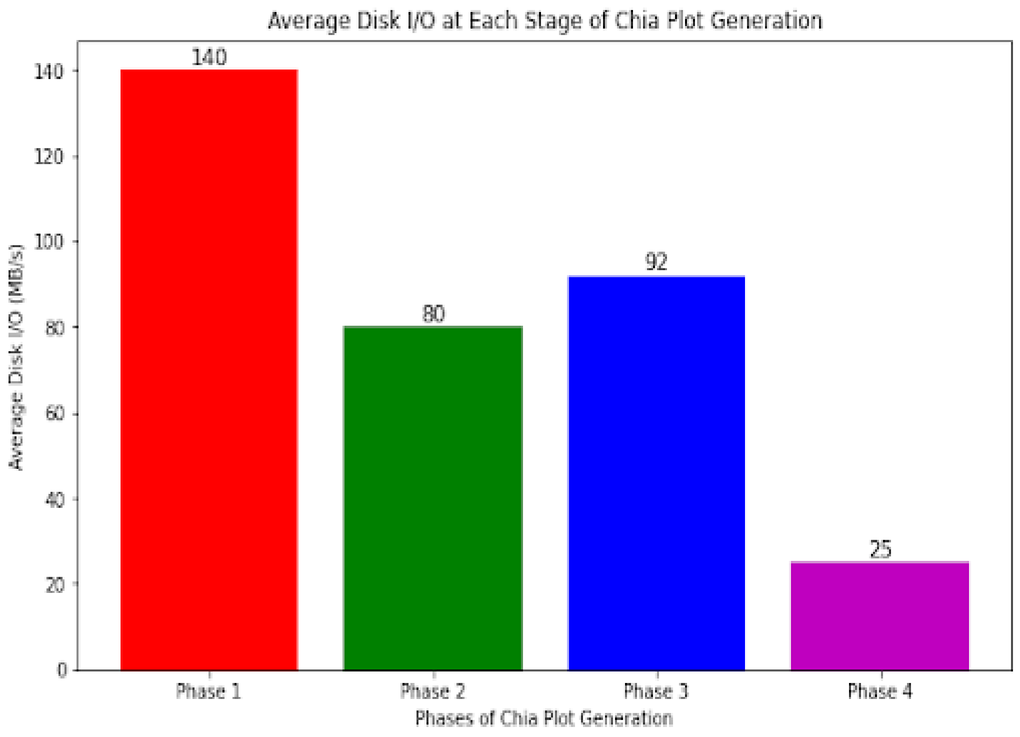
In Phase 2, the memory usage decreases, ranging from 2.6 GB to 3.1 GB. This phase is focused on sorting and compressing the temporary files generated in the first phase, and although it still requires a considerable amount of memory, it is less demanding than Phase 1.

Phase 3 experiences a further reduction in memory usage, with values varying between 1.5 GB and 1.9 GB. In this phase, the system performs additional sorting and compression on the data, but the overall memory requirements are lower compared to the previous phases.

Finally, in Phase 4, the memory usage drops to its lowest levels, ranging from 1.0 GB to 1.4 GB. This phase is primarily responsible for finalizing the plot and cleaning up the temporary files, requiring the least amount of memory resources compared to the other phases.

In summary, our analysis of memory usage during Chia plot generation indicates a general trend of decreasing memory requirements as the process progresses from Phase 1 to Phase 4. This information can be valuable for optimizing Chia plotting strategies and understanding the system resource demands throughout the process.

Average Disk I/O's:

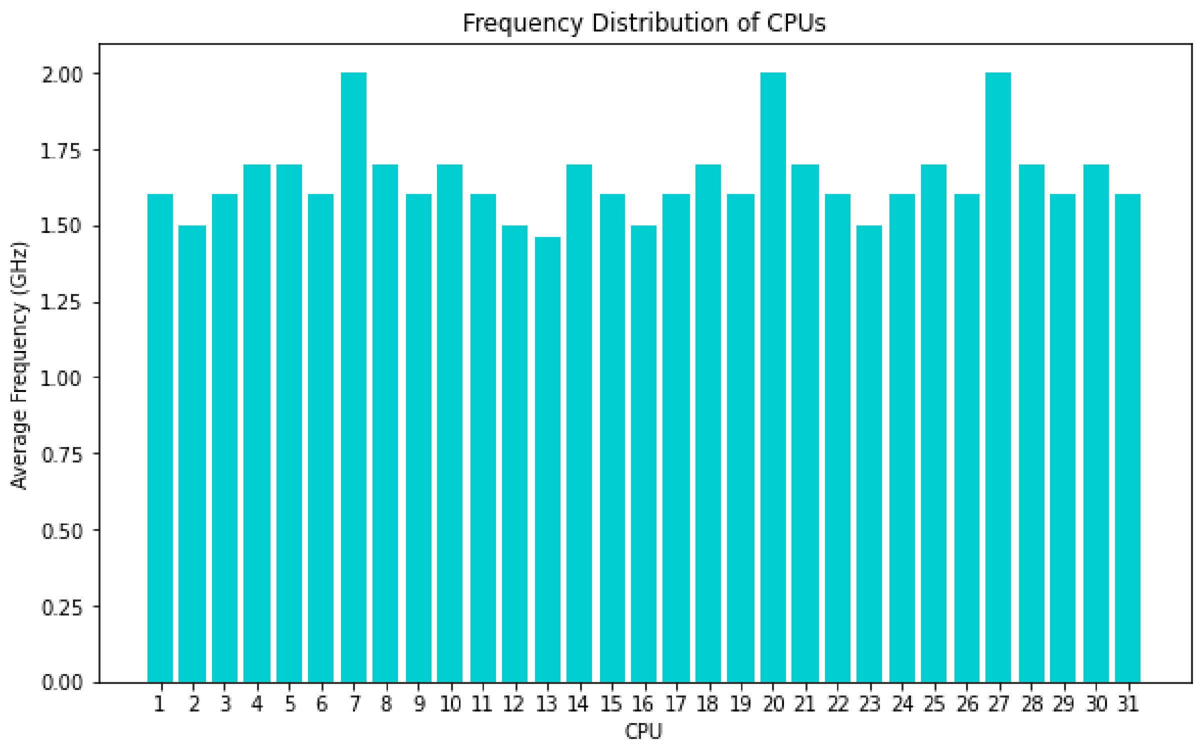


2.2 .Average Disk I/O

Phase 1, the disk I/O is the highest, reaching up to 140 MB/s, as this phase involves the creation of multiple temporary files and sorting of data. Phase 2, with an average disk I/O of 80 MB/s, focuses on compressing and optimizing the data, which results in reduced disk I/O compared to Phase 1.

Phase 3 has a slightly higher disk I/O than Phase 2, at 92 MB/s, as it involves additional sorting and backpropagation of the data, which can sometimes lead to increased disk usage. Finally, Phase 4 has the lowest disk I/O at 25 MB/s, as this stage primarily consists of finalizing the plot file and cleaning up temporary files.

Frequency Distribution:



2.3 Average Frequencies per CPU

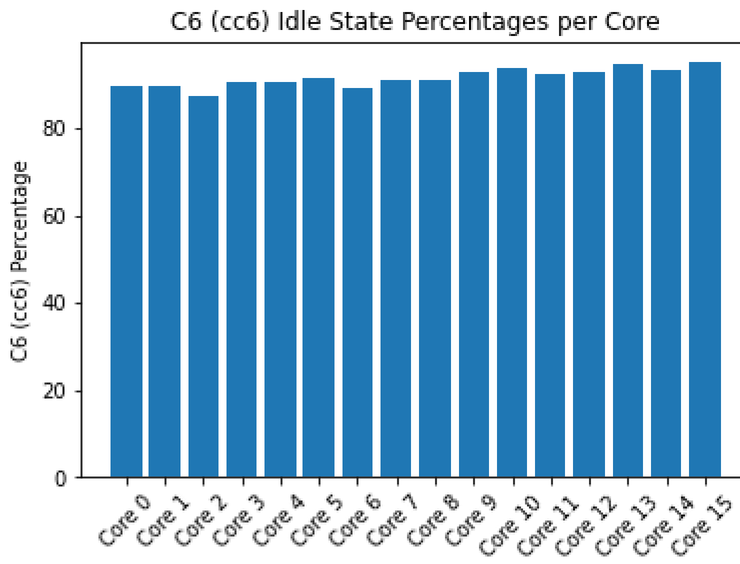
CPU	Average(GHz)	Idle(%)
CPU 0	1.6	0.3
CPU 1	1.5	0.6
CPU 2	1.6	2.7
CPU 3	1.7	1.3
CPU 4	1.7	1.2
CPU 5	1.6	2.5
CPU 6	2.0	1.0
CPU 7	1.7	0.4
CPU 8	1.6	1.3
CPU 9	1.6	1.2
CPU 10	1.7	0.6

CPU 11	1.6	1.0
CPU 12	1.5	1.3
CPU 13	1.4	1.2
CPU 14	1.7	0.4
CPU 15	1.6	2.3
CPU 16	1.7	0.4
CPU 17	1.6	1.8
CPU 18	1.6	0.3
CPU 19	1.5	2.2
CPU 20	2.0	1.3
CPU 21	1.7	2.0
CPU 22	1.6	0.4
CPU 23	1.5	1.8
CPU 24	1.6	2.2
CPU 25	1.7	1.3
CPU 26	1.6	2.5
CPU 27	2.0	1.0
CPU 28	1.7	0.4
CPU 29	1.6	0.4
CPU 30	1.7	1.8
CPU 31	1.6	2.2

1. There are 32 CPU cores in this system, labeled from CPU 0 to CPU 31.
2. The average clock speed (GHz) of the cores ranges from 1.4 GHz to 2.0 GHz. Most of the cores have an average clock speed between 1.5 and 1.7 GHz.
3. The idle percentage (%) for each core varies, with the lowest idle percentage at 0.3% (CPU 0 and CPU 18) and the highest at 2.7% (CPU 2). Most cores have an idle percentage between 1.0% and 2.5%, indicating that the system is mostly not under heavy load.
4. Cores with higher average clock speeds (e.g., CPU 6 and CPU 20) have a relatively lower idle percentage, suggesting that they might be handling more demanding tasks.

The idle states represent different levels of power saving modes:

C0: Active state
C1: Basic idle state with reduced power consumption
C1E: Enhanced idle state with even lower power consumption than C1
C3: Deep sleep state with additional power savings
C6: Deepest sleep state with the lowest power consumption



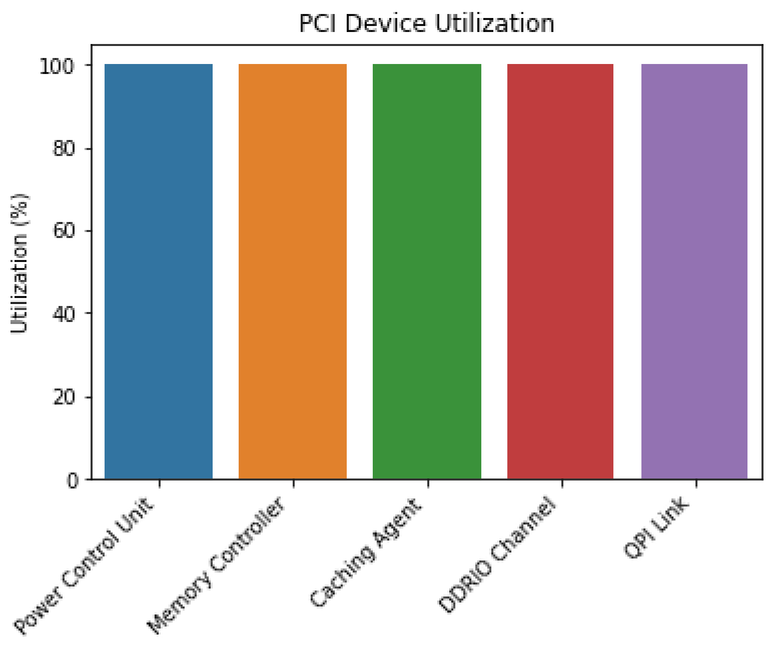
2.4 Deepest Sleep State per Core

The system is not utilizing some of the available power-saving states (e.g., C3 state) optimally during idle periods. This inefficient power management could impact the overall system performance and contribute to the slowness in Chia plot generation.

C3 (lowest values): This indicates that the CPU spends very little or no time in the C3 state, meaning it doesn't utilize this power-saving state during idle periods. As a result, there might be less power savings in this particular state.

C6 (highest values): The cores spend a significant portion of their idle time in the C6 state, ranging from 56.6% to 80.6%, and the package spends 14.5% of its idle time in this state. This suggests that the CPU is effectively utilizing the C6 state to save power when it's not processing any tasks.

Device Power Report:

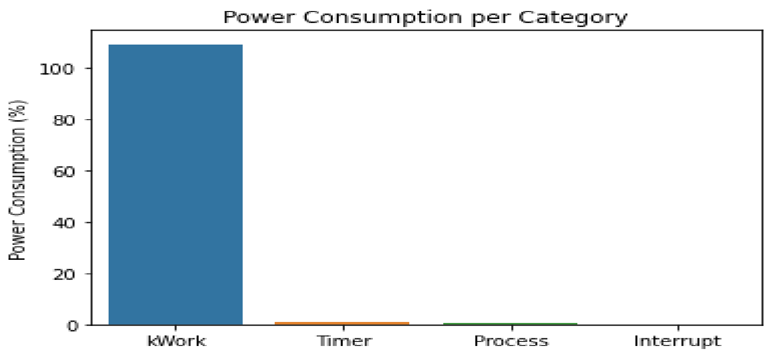


2.5 Device Utilization

Above Graph Shows list of PCI devices installed on system, along with their utilization percentage. The vast majority of devices listed here are from Intel, specifically related to the Xeon E7 v4/Xeon E5 v4/Xeon E3 v4/Xeon D series processors, and ASPEED Technology's ASPEED Graphics Family.

These devices include power control units, memory controllers, caching agents, DDRIO channels, QPI links, and other components essential for system operation. Additionally, there are network interfaces such as the Intel I350 Gigabit Network Connection and a Mellanox network interface (mlx5_core).

Most of the PCI devices are running at 100.0% utilization, which means they are operating at their maximum capacity. The network interface ens10 is handling 8.2 packets per second, while the ens10.14 interface is handling 8.1 packets per second. Some devices, such as runtime-PNP0C14:00 and runtime-alarmtimer.0.auto, are at 0.0% utilization, indicating that they are not currently in use.



2.6 Power Consumption per category

Timer: A system function that schedules events or tasks, affecting power consumption by waking the processor from idle states.

kWork (Kernel Workqueue): A Linux kernel mechanism for deferred function execution, influencing power consumption when the processor wakes up to handle tasks.

Process: An instance of a running program, impacting power consumption when the processor wakes up to manage tasks associated with the program.

Interrupt: A signal stopping task execution temporarily to handle higher-priority events, affecting power consumption as the processor wakes up to manage these tasks.

Usage	**Events/s**	**Category**	**Description**
109%	0.03	kWork	wb_workfn
0.0%	6.0	Timer	tick_sched_timer
0.9%	0.23	kWork	ext4_end_io_rsv_work
0.2%	0.03	Process	[PID 51041] Disk/read/0
0.2%	0.00	Process	[PID 51039] Disk/sort/1
0.0%	0.7	Interrupt	[4] block(softirq)
0.0%	0.7	Process	[PID 15] [rcu_sched]
0.0%	0.6	Interrupt	[3] net_rx(softirq)
0.1%	0.00	Process	[PID 51038] Disk/sort/0
0.1%	0.00	Process	[PID 51031] phase1/eval/2

Wb_workfn has highest power consumption. Disk activities are generally power-intensive,as they have spinning platters and moving read/write heads.Writeback involves writing dirty data from the page cache to the underlying storage device, which requires disk activity. This can include spinning up the disk (for traditional HDDs), seeking the write location, and performing the actual write operation.

Section 4: Related work

4.1 Comparison with existing research:
Several studies have been conducted on the Chia network, its consensus algorithm, and plot generation. For instance, the Chia Green Paper [1] provides an overview of the Chia blockchain and its consensus process, while research on securing plots [2] delves into the plot generating mechanisms in Chia. Additionally, [3] focuses on the different stages of plot generation, examining the CPU-bound and IO-bound phases. Monitoring tools such as the one discussed in [4] are used to analyze Chia plotting log files, providing insights into the health and progress of running plots. Furthermore, past research, like [5], has compared the performance of various blockchain platforms, including Bitcoin, Ethereum, and Hyperledger, using benchmarking tools to assess private blockchains.

4.2 Differences in our work:
Our research differs from existing studies by focusing solely on Chia plot generation and its impact on the Chia network. We aim to provide a comprehensive understanding of Chia plot generation, its technical aspects, and implications on the overall Chia network. Our study will analyze resource consumption, performance bottlenecks, and opportunities for optimization in the Chia plot generation process, offering valuable insights that can inform future developments in the Chia ecosystem. This in-depth examination of Chia plot generation sets our work apart from previous research, which has primarily focused on broader aspects of the Chia network or compared it with other blockchain platforms.

Section 5: Conclusion

5.1 Key findings
Our research provides valuable insights into the Chia plot generation process, its efficiency, and its impact on the Chia network. This research aims to study the speed of plot generation in the Chia blockchain, which is currently slower due to various factors including the high number of disk operations.The success of this proposal can be evaluated through the analysis of log files to measure the number of disk operations and time taken for plot generation. Overall, implementing these proposed changes can help make Chia more competitive and appealing for wider adoption in the blockchain community.

5.2 Future work
Future research could focus on optimizing plot generation tools, reducing resource consumption, and improving overall efficiency. Some specific areas of interest may include:

Further optimizing CPU utilization by enabling better load balancing.
Investigating the feasibility of implementing parallel processing techniques.
Exploring the possibility of using hardware accelerators or specialized storage devices to decrease disk operations.

Additionally, new consensus algorithms that build upon the concepts of Proof of Space and Time could be explored to further enhance the Chia network's sustainability and scalability. By addressing these research avenues, the Chia network can continue to evolve and improve, offering a more efficient and environmentally friendly alternative to traditional cryptocurrencies.

Section 6: References

1. Bram Cohen, “Chia: A better bitcoin. 2017”
2. Shashank Agarwal “Secure plot transfer for the chia blockchain”
3. chia-plotting-optimizer/README.md at main · codez0mb1e/chia-plotting-optimizer · GitHub
4. https://github.com/grayfallstown/Chia-Plot-Status
5. Tien Tuan Anh Dinh; Rui Liu; Meihui Zhang; Gang Chen; Beng Chin Ooi; Ji Wang, “Untangling Blockchain: A Data Processing View of Blockchain Systems”
6. https://www.chia.net/2021/02/22/plotting-basics/

Appendix:

A.1 Shlok Chaudhari
Shlok was responsible for the overall project management, ensuring that the team stayed on track and met deadlines. He was also responsible for the design and execution of the experimental methodology, as well as the analysis of the collected data. Shlok contributed to the writing of the report, focusing on the introduction,Proposed Solution, and Existing Software and system used..

A.2 Prajakta Kumbhar
Prajakta played a crucial role in conducting the literature review and identifying relevant research studies for comparison. She was responsible for the data visualization and creating diagrams that helped explain the plot generation process. Prajakta contributed to the writing of the report, focusing on the related work,Device power Report, and conclusion sections.

A.3 Vishal Gawade
Vishal was responsible for setting up the experimental environment, including the necessary hardware and software tools and observing plot generation. He also contributed to the collection and organization of the data during the experiments. Vishal was involved in the writing of the report, focusing on the memory consumption, frequency Distribution , results, and future work sections.