

A COMPARATIVE STUDY OF BLOCKCHAIN ALGORITHMS
FOR NON-FUNGIBLE TOKEN

WOO JAN YIN

Bachelor of Computer Science
(Computer System & Networking)
UNIVERSITI MALAYSIA PAHANG

UNIVERSITI MALAYSIA PAHANG

DECLARATION OF THESIS AND COPYRIGHT

Author's Full Name : WOO JAN YIN

Date of Birth

Title : A COMPARATIVE STUDY OF BLOCKCHAIN ALGORITHMS
FOR NON-FUNGIBLE TOKEN

Academic Session : 2021/2022

I declare that this thesis is classified as:

- CONFIDENTIAL (Contains confidential information under the Official Secret Act 1997) *
- RESTRICTED (Contains restricted information as specified by the organization where research was done) *
- OPEN ACCESS I agree that my thesis to be published as online open access (Full Text)

I acknowledge that Universiti Malaysia Pahang reserves the following rights:

1. The Thesis is the Property of Universiti Malaysia Pahang
2. The Library of Universiti Malaysia Pahang has the right to make copies of the thesis for the purpose of research only.
3. The Library has the right to make copies of the thesis for academic exchange.

Certified by:



(Student's Signature)

(Supervisor's Signature)

New IC/Passport
Number Date: 17
February 2023

Dr Zahian Ismail
Name of Supervisor
Date: 17 February 2023

NOTE: * If the thesis is CONFIDENTIAL or RESTRICTED, please attach a thesis declaration letter.

THESIS DECLARATION LETTER (OPTIONAL)

Librarian,
Perpustakaan Universiti Malaysia Pahang,
Universiti Malaysia Pahang,
Lebuhraya Tun Razak,
26300, Gambang, Kuantan.

Dear Sir,

CLASSIFICATION OF THESIS AS RESTRICTED

Please be informed that the following thesis is classified as RESTRICTED for a period of three

Author's Name
Thesis Title

Reasons (i)

(ii)

(iii)

(3) years from the date of this letter. The reasons for this classification are as listed below.

Thank you.

Yours faithfully,

(Supervisor's Signature)

Date:

Stamp:

Note: This letter should be written by the supervisor, addressed to the Librarian, *Perpustakaan Universiti Malaysia Pahang* with its copy attached to the thesis.



SUPERVISOR'S DECLARATION

I hereby declare that I have checked this thesis and in my opinion, this thesis is adequate in terms of scope and quality for the award of the degree of Bachelor of Computer Science in Computer System & Networking

Zahian Ismail

(Supervisor's Signature)

Full Name : Dr Zahian Ismail

Position : Senior Lecturer

Date : 17 February 2023

(Co-supervisor's Signature)

Full Name :

Position :

Date :



STUDENT'S DECLARATION

I hereby declare that the work in this thesis is based on my original work except for quotations and citations which have been duly acknowledged. I also declare that it has not been previously or concurrently submitted for any other degree at Universiti Malaysia Pahang or any other institutions.

A handwritten signature in black ink, appearing to be "Woo Jan Yin", is written above a horizontal line.

(Student's Signature)

Full Name : WOO JAN YIN

ID Number : CA19079

Date : 17 February 2023

RESEARCH ON A COMPARATIVE STUDY OF BLOCKCHAIN
ALGORITHMS FOR NON-FUNGIBLE TOKEN

WOO JAN YIN

Thesis submitted in fulfillment of the requirements
for the award of the degree of
Bachelor of Computer Science in Computer System & Networking

Faculty of Computing
UNIVERSITI MALAYSIA PAHANG

JANUARY 2023

ACKNOWLEDGEMENT

I would like to convey my heartfelt thanks to Dr Zahian Ismail, the supervisor of the Final Year Project at Universiti Malaysia Pahang's Faculty of Computing, for enabling me to work under her supervision and providing helpful guidance during the Final Year Project. Her zeal, vision, sincerity, and drive have all made a lasting effect on me. She had given me instructions on how to complete the job and communicate the outcomes as clearly as possible. It was a wonderful pleasure and joy to work and learn under her direction. I am appreciative of everything she has done for me. I'd want to thank her for her friendship, understanding, and excellent sense of humors as well. I want to thank her for her patience and compassion during our project work and thesis preparation conversations.

For their love, commitment, care, and efforts in teaching and educating me for the future, I owe my parents a debt of appreciation. I am appreciative for their patience, compassion, commitment, and unwavering support in assisting me in completing my study endeavors.

I'd want to convey my heartfelt gratitude to Yew Wei Zhiang and Lim Aun Xian, two of my closest friends, for their unwavering support. Throughout this semester, I've appreciated their encouragement and suggestions when I've been having trouble with my research. Their efforts and commitment are valuable, and it pays out in the end.

Finally, I'd want to thank everyone who has assisted me in completing the research work, whether actively or passively.

ABSTRAK

Dari segi prestasi, skalabilitas, dan kependaman, NFT adalah pelaksanaan penting bagi manusia tidak kira dalam cryptocurrency atau perlindungan pemilikan aset digital. Walau bagaimanapun, terdapat beberapa kebimbangan yang perlu dipertimbangkan oleh masyarakat sebelum menggunakan universal teknologi NFT. Isu terpenting yang wajib diatasi adalah prestasi algoritma blockchain dalam NFT dan kesan skalabiliti dalam setiap algoritma blockchain. Oleh kerana sistem blockchain sangat bergantung pada protokol, masalah berprestasi rendah mungkin dialami. Selain itu, sistem blockchain mungkin tidak dapat memproses jika blok dihasilkan terlalu cepat. Oleh itu, penyelidikan ini akan menjadi kejayaan NFT yang dapat menilai prestasi algoritma blockchain dan menentukan algoritma yang paling sesuai yang mungkin berlaku untuk NFT. Algoritma yang dicadangkan akan dinilai dari segi prestasi, skalabiliti, dan kependaman. Dalam penyelidikan ini, akan ada beberapa algoritma terpilih, yang merupakan Bukti Kerja dan Bukti-Berjaya dari sumber terbuka dan diubah suai menjadi kriteria ujian yang sesuai untuk menilai data yang sah dan mencukupi untuk visualisasi data. Semua data akan dilakukan dalam paparan carta untuk menjadikan semua perbandingan lebih jelas. Dari hasilnya, salah satu algoritma yang paling sesuai akan dipilih iaitu Proof-of-Stakes dan alasannya dibenarkan seperti prestasi yang lebih baik, skalabilitas yang baik, dan kurang latensi. Oleh itu, penyelidikan ini akan dilakukan pada pengujian algoritma dan menentukan algoritma terbaik yang perlu diterapkan dalam NFT.

ABSTRACT

In terms of performance, scalability, and latency, NFTs are a crucial implementation for humans no matter in cryptocurrency or digital asset ownership protection. However, there are some concerns for the community to consider before using NFTs technology universalness. The most important issue that is compulsory to overcome is the performance of blockchain algorithms in NFTs and the effect of scalability in each blockchain algorithm. Since a blockchain system is highly dependent on the protocols, a low-performance problem might be experienced. Other than that, a blockchain system might not be able to process if blocks are produced too fast. Therefore, this research will be a breakthrough of NFTs which could evaluate the performance of blockchain algorithms and determine the most suitable algorithms that may apply to NFTs. The proposed algorithms will be evaluated in terms of performance, scalability, and latency. In this research, there will be some selected algorithms, which is Proof-of-Work and Proof-of-Stakes from open resource and modified to the suitable testing criteria to evaluate valid and sufficient data for data visualization. All data will be done in chart view to make all comparative more obvious. From the result, one of the most suitable algorithms will be selected which is Proof-of-Stakes and the reasons are justified as in better performance, good scalability, and less latency. Therefore, this research will be on a testing for algorithms and determine the best algorithms that need to apply in NFTs.

TABLE OF CONTENT

DECLARATION	
TITLE PAGE	
ACKNOWLEDGEMENTS	II
ABSTRAK	III
ABSTRACT	IV
TABLE OF CONTENT	V - VII
LIST OF FIGURES	VIII
LIST OF TABLES	IX
CHAPTER 1 INTRODUCTION	
1.1 Overview	1
1.2 Background	2 - 3
1.3 Problem Statement	4
1.4 Objectives	5
1.5 Scope	5
1.6 Hypothesis	6
1.7 Research Contribution	6
1.8 Organization of Thesis	7
CHAPTER 2 LITERATURE REVIEW	
2.1 Blockchain Algorithms	8
2.1.1 Proof of Work	8
2.1.2 Proof of Stake	8
2.1.3 Proof of Space	9
2.2 Related Work	10
2.2.1 Bitcoin	10 - 12
2.2.2 Ethereum	13
2.2.3 Chia	14
2.3 Related Resources	15
2.4 Summary	16
CHAPTER 3 METHODOLOGY	
3.1 Introduction	17
3.2 Project Management Framework	18

3.2.1 Data Collection & Token Randomization	18
3.2.2 Code Modification	18
3.2.3 Experimental Algorithms	18
3.2.4 Result Evaluation	18
3.3 Project Requirements	19
3.3.1 Input	19
3.3.2 Output	19
3.3.3 Process Description	19
3.3.4 Constraint and Limitation	19
3.3.5 Software Requirement	20
3.3.6 Hardware Requirement	20
3.4 Proposed Design	21 - 22
3.5 Proof of Initial Concept	23 - 24
3.6 Potential Used of Proposed Solution	25
CHAPTER 4 IMPLEMENTATION, RESULT & DISCUSSION	
4.1 Introduction	26
4.2 Implementation Process	26
4.2.1 Collecting Open Resources Code	27
4.2.2 Test Run Potential Code	28
4.2.3 Code Modification	29 – 31
4.2.4 Train Data	32
4.3 Result (Data Visualization)	33
4.3.1 Performance	33
4.3.2 Scalability	34
4.3.3 Latency	35
4.4 Discussion	36
4.4.1 Performance	36 – 37
4.4.2 Scalability	38
4.4.3 Latency	39
CHAPTER 5 CONCLUSION	
5.1 Introduction	40
5.2 Discussion on the Result	41

5.3 Limitation and Constraint	42
5.4 Future Work	42
REFERENCE	43 – 44
APPENDIX	45 - 56

LIST OF FIGURES

Figure 2.1	Flow of blockchain system in bitcoin	10
Figure 3.2	Basic Flow of the Research	18
Figure 3.4	Full flow of the Research	21
Figure 4.1	Apply the filter function to browse the potential code.	27
Figure 4.2	Test Run Proof-of-Work	28
Figure 4.3	Test Run Proof-of-Stake	28
Figure 4.4	Completely run for 50 sets of random data in Proof-of-Work	32
Figure 4.5	Completely run for 50 sets of random data in Proof-of-Stake	32
Figure 4.6	The graph of Number of Process vs Runtime in Proof-of-Work	33
Figure 4.7	The graph of Number of Process vs Runtime in Proof-of-Stake	33
Figure 4.8	The graph of Cumulative Run Time for 50 sets of process for both algorithms	34
Figure 4.9	The graph of Cumulative Run Time for 200 sets of process for both algorithms	34
Figure 4.10	The graph of the Latency of Proof-of-Work and Proof-of-Stakes	35
Figure 4.11	The process takes more time to solve an -digit nonce	36
Figure 4.12	The process repeats for assign a random winner to own the block	37

LIST OF TABLES

Table 2.1	Advantages and Disadvantages of Bitcoin	12
Table 2.2	Advantages and Disadvantages of Ethereum	13
Table 2.3	Advantages and Disadvantages of Chia	14
Table 3.3	Software Requirements of the Research	20
Table 3.4	Hardware Requirements of the Research	20
Table 3.6	Comparative Analysis of the Algorithms used	23 - 24

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW

With the improvement of blockchain technologies and the trend of cryptocurrencies, it is getting more ordinary to use NFTs. NFTs are a Non-Fungible Token digital assets which are built on blockchain. Every NFT own its unique identity number to make user easy to trace them (Sharma, 2022). Using blockchain technology, NFTs occurs is used to make copyright more easily recognized and protect artwork from loss. For instance, Monalisa by Leonardo da Vinci is a well-known painting in the world. However, nobody knows which is real and counterfeit. By using NFTs, all artworks could be identified by using serial number. Therefore, only the artwork with registered serial numbers is the real and actual artwork with copyright of the artist. The existence of NFTs help give artists a safety way to keep and sell their works, and it could kindly protect their work from copy or stolen.

This chapter discusses the background of NFT, problem statement, research objective, etc. All general content and basic explanation regarding this project proposed.

1.2 BACKGROUND

In the digital transformation generation, it is better for us to keep our documents in a digital way. For example, Dropbox is used to upload any document and files for user to keep, Google Photo may focus on keeping images and videos digitally. It is safer for us to upload and store our important documents in a trusted platform. NFTs could be a format for user to store their artwork online. To prove it is better than a physical form, we may talk about the “Water Lilies” of Claude Monet. He is a French painter who draw a lot of amazing paint, but his artwork is getting burn in The Museum of Modern Art, New York. Therefore, a digital form of asset can ensure the hand down of artwork. A graphic designer, Mike Winkelmann is using NFTs to manage his artwork.

Non-fungible token, also called as NFT is an economic security made up of digital format recorded in a public ledger called a blockchain system with specific identity serial and metadata to differentiate them. By using cryptocurrency, which is ETH coin, user may purchase an NFT, and it could be encoded with mostly software provided (Clark, 2021). According to Statista Research Department, NFTs sold in one year (April 2021 to March 2022) are more than 565 thousand no matter it is primary sales, or secondary sales (*Statistic_id1265353_daily_nft_sales_value_worldwide_up_until_january.Pdf*, 2022). To ensure all NFT is unique, each of them have its specific identity code. Generally, most of the NFTs are using an Ethereum blockchain, which is a cryptocurrency that could support NFTs functionality. By using this technology, NFTs may only be owned by one person at a time, it will never occur duplicated ownership exist in this system. All NFTs will only be verified by the person who owns the ownership, and it could transfer the token between owners. It also allows the creator to hide some specific information in the token to prove the copyright of the digital asset. It is different between NFT and cryptocurrency since cryptocurrency is a fungible, and it could be exchange. Unfortunately, NFTs are unable to replace, and it is linked with a digital asset permanently.

In the field of data security, NFTs is a safe platform due to its blockchain technologies. It can be considered impossible for cybercriminals to hack, change, and remove any asset that is stored in this platform. The reason for the good security is because blockchain technologies will record all the transaction history duplicate to all the participants that joining in the network. Everyone will own the same transaction history and it is hard to change or remove all records in million billion accounts. Back to NFTs, all digital assets

will be saved on blockchain system and distributed the authentic record to every user. Theoretically, it can avoid any thefts of missing issue happened. From these, NFTs may declared that have improved the private and confidentiality of artwork to a new level and it is specific in the market in this generation.

Theoretically, NFTs are safe enough to keep all the digital assets. However, it is still occurring some hidden risk between it. Commonly, all the images could be easily copy and duplicate, and it has a big possibility to spread to any online platform and social media. This illegal spreading of artwork will never credit their original creators, most of the online surfers will never know who owns the copyright of that media. Besides that, the genuine copyright and ownership of NFTs are not verified and enforced since there is no legal justification or evidence. It is a crucial problem because everyone may keep doing other people's artworks as NFTs and make their own profits. The buyer who purchased this kind of NFTs might need to take legal responsibility of getting sued by the person who owns the real copyright.

NFTs bring security concerns to the user which is the private keys may link to all digital assets. Therefore, variants of NFTs might be happened in the platform and it could interrupt the NFTs and cause loss of digital asset due to scam or fraud. On either side, effective security measures on NFT exchanges may also be endorsed the idea favor. Unfortunately, Strong industry security procedures was not enough to overcome NFT security vulnerabilities when using centralized markets. Platform participants may be to blame for uncovering a slew of additional security flaws in NFT markets. Users might forfeit their crucial NFTs for a range of factors, such as uncomplicated password combination or a failure of apply two-factor authentication (Brock, 2022).

To summarize, NFTs is a new technology which enhances the copyright protection and transaction of ownership for artwork in digital form. However, its own advantages and disadvantages have a lot of hidden risks that the developer needs to overcome. For instance, NFTs has brought some ownerships concerns which everyone could copy other artworks and sell as his own in this platform. NFTs must become a more safety platform to avoid any commercial loss occurring to every participant. In additional, it should be more focus on the security layers to avoid any hacked, thefts, and cybercrime issue happened in this platform.

1.3 PROBLEM STATEMENTS

Two problem statements have been identified. These problems happened because of the performance of NFT. it contains 2 problem statements cause the low stability occurs in NFTs, which is:

1. Performance of NFT

Blockchain systems that adopt the classical Byzantine Fault Tolerant (BFT) protocols and the newly enacted Nakamoto consensus (NC) such as Proof-of-Work, Proof-of-Stakes, and Proof-of-Authority may experience low-performance problems as a result of the extensive communication or intensive computation (Wang & Li, 2021). In order to achieve dependable and transparent administration, NFT users often communicate the transactions to the blockchain network. Nevertheless, present NFT systems have poor performance since they are tightly connected with the underlying blockchain technology. The blockchain topology has to be modified, its structure needs to be optimised, or the consensus processes need to be improved in order to tackle this issue. Such conditions cannot be satisfied by the current blockchain systems, and consumers may have a lengthy delay for each NFT trade (*NFT Challenges - NFT*, n.d.). Without a confusion, the performance problem has emerged as the main barrier to the adoption of blockchain in NFTs, particularly for systems requiring high speed.

2. Stability and Scalability

NFTs are traded anonymously, like other virtual currency and asset transactions, and their sale indicates market volatility. NFTs are good platforms for financial fraud because of their obscurity and volatility, but they also constantly run the risk of being used to support terrorism (*Why Would 99% of NFTs Fail?. In Economics, the Concepts of Fungible... | by BSN | Blockchain Thought Leadership | Medium*, n.d.). The subgroup of blocks that are confirmed must organically expand over time when new blocks are added as this is a fundamental referral specifically of a blockchain system. This is not assured, though, as blocks are generated throughout time at various points around the system and must subsequently be distributed. Communications on the network suffer delays and may not be immediate due to capacity restrictions (Gopalan et al., 2020). A blockchain system may be unable to confirm blocks if blocks are produced too fast network latency. The scaling issues that occur when blockchain technology develops and grows need to be defined.

1.4 OBJECTIVES

There are three objectives in this project which are:

- To study blockchain algorithms and identify the most suitable algorithms which may enhance the performance of NFTs.
- To implement the relevant blockchain algorithms in NFTs which could ensure the stability of the blockchain process.
- To evaluate the scalability of blockchain using different algorithms in NFTs.

1.5 SCOPE

We have 2 scopes include in this project which is:

- Digital Asset.

In blockchain system, all tokens are unpredictable and unique. There is impossible for collecting and reuse the same token for testing multiple time. In this case, a token randomisation will be done. Multiple random tokens will be created by the system following the number of testing needs. These tokens will be created and assigned by the code to ensure valid and eligible data are provided.

- Suitable Blockchain Algorithms.

Two suitable algorithms will be chosen for the comparison and the result will be shown to evaluate the most suitable techniques applied in NFTs. The algorithms chosen must be the most potential algorithms that could be apply in NFTs. A prediction must be done and the reason of algorithm chosen should be stated to prove the research value.

1.6 HYPOTHESIS

Based on the block arrival rate, network capacity restrictions, and network structure, the constraints on the stability region of blockchain systems are calculated. The highest block arrival rate that allows a blockchain system utilizing a tree or throughput policy to confirm an unlimited number of blocks at once (Gopalan et al., 2020). Therefore, the performance of blockchain may be affected by the algorithms use and the data size for processing. As a summary, the project outcome might be investigated:

- The best blockchain implementation which can cover the heavy workload of NFTs function and able to run in a most stable time with less latency. The blockchain algorithms should ensure that the pending time for every use in NFTs are relevant and acceptable.
- Suitable algorithms to ensure all artwork uploaded to NFTs is able to have their own address and there is no duplicate blocks happened.

1.7 RESEARCH CONTRIBUTION

There are some main points need to focus on in this research:

- The most promising blockchain algorithms have to evaluate which could make sure that is suitable to apply on NFTs.
- This research finds out the most suitable testing method of blockchains algorithms to prove the performance when it applies in NFTs.

1.8 ORGANIZATION OF THESIS

This thesis is including 6 chapter as followed:

Chapter 1 briefly introduces the existing of NFT technologies. This research also discusses the needs of NFT nowadays. However, it could be occurred some issue that make NFT less trusted (Ownership problem and Uniquely problem). It also highlights the Objectives, Scope, and Hypothesis of this research.

Chapter 2 explains about NFTs includes the comparison of research from other sources and state out the suitable algorithms could be applied in NFTs. It also includes the related work with blockchain technologies.

Chapter 3 presents the methodologies of testing blockchains algorithms. This chapter also explains the way to prove the performance of blockchains using the methodologies stated.

Chapter 4 is the whole process of the research conduct, and the testing will be run on. All modification of code will be explained here. Furthermore, the results which have visualize are discussed in this chapter.

Chapter 5 discusses the conclusion of the research. There are contain the limitation and constraint of this project and briefly discuss on the future work that can be apply in this research.

CHAPTER 2

LITERATURE REVIEW

2.1 Blockchain Algorithms

There is few of blockchain algorithms can be evaluated. Based on the study, these are the common algorithms used (Krishnamurthi & Shree, 2020):

2.1.1 Proof of Work (POW)

Proof of Work is a common algorithm used in cryptocurrency. PoW may ensure all the transaction success and a new block in a blockchain is created. This algorithm makes participants to mine block harder and harder and need to compete with each other's to make sure they could get something. PoW may meet highest energy consumption when it is running. However, it has more fairness since all the coins will be equally separated to all nodes who are mining. Furthermore, it has more advance creditability in the performance and mining of coins because of its super accuracy of block generation. For the concerns of PoW, this algorithm must face the risk of DoS attack, Selfish Mining and Sybil Attack.

2.1.2 Proof of Stake (POS)

Proof of Stakes have been used for validating transaction in a system. Basically, this algorithm will choose the miners who have the higher balance to give priority. Therefore, the richest man in the selection will be possible to become a permanent miner since he has the highest balance and always given priority by system. In the performance site, PoS also required high energy consumption and it is quite consistent in the stated energy levels. Based on the logic of richest man get priority, it is unfair to the person who own less balance since they only could be leads by the person who at the top. Rich man will getting richer and poor man will still maintained. Besides that, over long periods, the reliability of PoS systems changes rather a lot. For the security concerns, it faces a big problem since it has a high risk of getting malware attack in Dos, Short-Range Attack, Long-Range Attack, Coin-Age accumulation, Pre-Computation Attack and Sybil Attack(Hackernoon, 2018).

2.1.3 Proof of Space

Proof of space (PoS) is a consensual process that involves dedicating a non-trivial quantity of memory or disc space to fulfill a task posed by the service provider in order to demonstrate one's genuine interest in a system . Proofs of space are similar to proofs of work, the only difference is the storage is utilized to earn bitcoin instead of compute. Proof-of-space functions vary from recollection functions in that the barrier is the quantity of memory required, not the number of main memory instances. Due to a general structure of capacity and the reduced energy cost required by memory, blockchain supporters consider proofs of space as a kinder and better option, however they have been challenged for greater storage demand (*Proof of Space / Owlapps, n.d.*).

2.2 Related Works

The related work in this research is Bitcoin, Ethereum and CHIA. These three systems are represented the three of algorithms will be used respectively which are Proof-of-Work, Proof-of-Stake and Proof-of-Space. These systems are currently activated, and it can be proved that the algorithms selected are works and the advantages and disadvantages could be bringing the benefits to the research and easier the evaluation of result. Nevertheless, these related works may refer as a comparison to justify the potential of each algorithm and minimize the scope of testing.

2.2.1 Bitcoin

Bitcoin is a virtual currency using blockchain technologies as their basic technique. This technique may enhance the security of the system which could encrypts a user's profile to secure their personal data. Every Bitcoin node record and validates the authenticity in any transaction on the blockchain, which would be a distributed shared-data database that comprises a block documenting all created transactions over an amount of time (Zhu et al., 2018). Only transactions with a variety of input activities and output destinations are added to the blockchain. The transaction may be verified by comparing the total of the output values of the input operations to the total of the target value of that kind of transaction. A bitcoin address is the base58 encoding of a participant's public key hashing, which specifically identifies the client. However, a person can hold many Bitcoin addresses by generating a new address or changing the address provided by the system, which provides security in Bitcoin. Others are unable to precisely determine the transfer of cash with a particular Bitcoin address due to the various incoming and outgoing addresses of an operation and the system's automatic creation between several addresses.

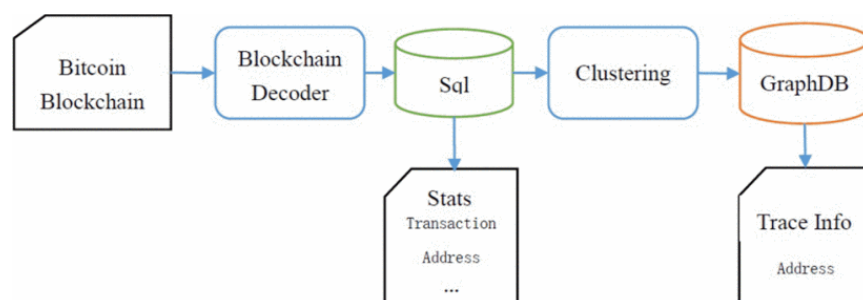


Figure 2.1: the Flow of blockchain system in bitcoin.

a. Blockchain Decoder

To isolate and obtain data on asset and address information, that is require the decoder to decode the blockchain data. The input and output of each operation will be retrieved from the binary blockchain data following this approach.

b. MySQL

It is necessary to keep Bitcoin blocks in order to study the patterns of Bitcoin exchanges and identifies. Due to the large number of transactions in blockchain, an index is required to greatly minimize lead times. It is possible to figure out another comparative statistic of operations and addresses using a database system as the preservation of bitcoin network.

c. Address Clustering

Due to the system generating a large number of addresses, it is important to connect the addresses to a summary in order to de-anonymize them. As a result, the number of Bitcoin addresses can be lowered, removing some anonymity.

d. Graph Database Store

This requirement gives the accounting process of a specific address and returns the transaction's connected chain. To successfully carry out this job, a graph database is employed to record all transaction information, which may be more effective and unique than a database system in terms of searching and displaying social relationships. Thus, by describing the node as a cluster formed in the previous stage and the feature as a transaction, store the operation in a graph database. The cluster-id is stored in the datatype variable, whereas the date, target output, and transaction id are stored in the edge's asset. Finally, to illustrate the performance and eventually, a massive, directed graph will be created.

Author	Satoshi Nakamoto
Year	2009
Advantages	<ul style="list-style-type: none"> • High-returning ability. • Fraudulent Payment Security. • Foreign Transactions, Instant Settlement. • Greater Stability and Flexibility.
Disadvantages	<ul style="list-style-type: none"> • High risk of big losses due to high unpredictability. • Operations in the black market. • Cyber intrusion is uncontrolled and unsupported. • Not refundable.

Table 2.1: Advantages and Disadvantages of Bitcoin

2.2.2 Ethereum

Ethereum is a popular blockchain platform without a block size restriction. Unfortunately, executing infinite transactions per second has significant drawbacks. Varying users execute the Ethereum blockchain code at varied speeds and with different levels of performance. There is two type of Ethereum Account which is:

- a. Externally Owned Accounts (EOA): User may send the transaction directly.
- b. Contract Accounts: Depends on the contract for possibility of call another contract to send the transaction.

The Transaction is a single input code that transmits a message from an Externally Owned Account to another account. The Ethereum blockchain begins with a blockchain network, after which further transactions are made, resulting in the creation of new blocks and a new state. Any modification within that status of the blockchain begins with a trade submitted by EOA. This transaction might either be a direct transfer of Ethereum digital money to another account or a contractual trigger. The sender's account private key authorizes that transaction. Ethereum might well be thought of as a control structure that is built on transactions (Saskatoon, 2017). Ethereum uses the transition probability mechanism to ensure that transitioning from one state to another is successful. Based on the Ethereum Yellow Paper, the formula of blockchain used in Ethereum are shown as below (Wood, 2019):

$$\sigma_{t+1} \equiv Y(\sigma_t, T)$$

Ethereum state transition function takes responsibilities to progress several tasks. For instance, it must evaluate if such transaction becomes well, resetting general ledger, refunding the reminder cost, and compensating miners for processing.

Author	Vitalik Buterin, Gavin Wood
Year	2015
Advantages	<ul style="list-style-type: none"> • Expertise and confidence are successfully distributed across network members. • Set up and manage private blockchain networks easily. • There are a lot of available protocol layers that may be used. • Not required a huge and efficient network. • Private transaction layers.
Disadvantages	<ul style="list-style-type: none"> • Utilizes a Difficult Programming Language. • Defects, failures, and hackers are all possible outcomes. • Investment seems to be a risky business.

Table 2.2: Advantages and Disadvantages of Ethereum

2.2.3 Chia

Miners fill data storage with randomized integers in new crypto currencies like Chia. The Chia blockchain generates its own random figure, and the participant with the closest is the winner. Chia calls it “proof of space and proof of time,” however the more hard disk drive capacity you own, ever more randomized integers you own, and the higher chances that you will win (Moss, 2022):

Chialisp is a sophisticated and reliable LISP-like language with smart-contract features for weighing down and transferring assets. The code below is the sample of Chialisp (*Chialisp*, n.d.):

```
(mod (password new_puzhash amount)
  (defconstant CREATE_COIN 51)
  (if(=(sha256password)(q.
    0x2cf24dba5fb0a30e26e83b2ac5b9e29e1b161e5c1fa7425e7304336
    2938b9824))
    (list (list CREATE_COIN new_puzhash amount))
    (x)
  )
)
```

Chia has using Chia Asset Tokens (CATs) which can create or trade among Chia’s Blockchain. Using a Token and Asset Issuance Limiter (TAIL), the owners of these assets specify the conditions for their tokens and withdrawal. The holders of these assets have complete control regarding their use. CATs could be used as stable currency, stock distribution tokens, shares outstanding, and like anything else.

Author	Bram Cohen
Year	2021
Advantages	<ul style="list-style-type: none">• There’s no need for high-capacity power supply.• The installation of Disk hardware is uncomplicated.• Setting up a software to mine coins does not need any specific skills.
Disadvantages	<ul style="list-style-type: none">• Different hard drive models have a broad range of durability.• PC components that are required are in low supply.• Purchase of large hard discs comes at a high expense.

Table 2.3: Advantages and Disadvantages of Chia

2.3 Related Resources

From the open resource, most of the study are related to comparison between blockchain algorithms only. Basically, there is no one testing the algorithms performance and relate into NFTs application. However, there are some aspects that can be applied on this study to make this research more efficient.

This section lists the maximum theoretical number of transactions per second (TPS) that a blockchain can process. The TPS values of Proof-of-Work, Proof-of-Stakes, and Proof-of-Space are 7, 7 and 15 respectively (L. M. Bach, B. Mihaljevic, 2018). There is some variation across systems that employ the same technology, even though the algorithm that powers a cryptocurrency determines the optimum TPS that can be reached. Larger blocks appear to be the main barrier for transactions in a PoW system; for instance, Bitcoin's TPS statistics are low because the block size has a difficult limit of one mb of data. Moreover, it is challenging to offer a really exact comparison of TPS for each currency's protocol since not all resources are easily verified. Thus, instead of the performance of blockchain, scalability of blockchain also a important things in this study.

2.4 Summary

	Proof-of-Work	Proof-of-Stakes	Proof-of-Space
Mechanism	Mining system, unlimited token provided.	Limited token assigned in the system.	Mining system depends on the device storage.
Example of Application	Bitcoin	Ethereum	Chia
Advantages	<ul style="list-style-type: none"> • High-returning ability. • Fraudulent Payment Security. • Foreign Transactions, Instant Settlement. Greater Stability and Flexibility.	<ul style="list-style-type: none"> • Expertise and confidence are successfully distributed across network members. • Set up and manage private blockchain networks easily. • There are a lot of available protocol layers that may be used. • Not required a huge and efficient network. Private transaction layers.	<ul style="list-style-type: none"> • There's no need for a high-capacity power supply. • The installation of Disk hardware is uncomplicated. Setting up a software to mine coins does not need any specific skills.
Disadvantages	<ul style="list-style-type: none"> • High risk of big losses due to high unpredictability. • Operations in the black market. • Cyber intrusion is uncontrolled and unsupported. • Not refundable. 	<ul style="list-style-type: none"> • Utilizes a Difficult Programming Language. • Defects, failures, and hackers are all possible outcomes. • Investment seems to be a risky business. 	<ul style="list-style-type: none"> • Different hard drive models have a broad range of durability. • PC components that are required are in low supply. • Purchase of large hard discs comes at a high expense.

CHAPTER 3

METHODOLOGY

3.1 INTRODUCTION

This chapter briefly explained the testing framework and project requirements. All the methodologies to ensure the guidelines of testing are fully evaluated. The general process of the testing is derived as 4 parts which are Data Collection & Token Randomization, Code Modification, Experimental Algorithms and Result Evaluation.

3.2 PROJECT MANAGEMENT METHODOLOGY

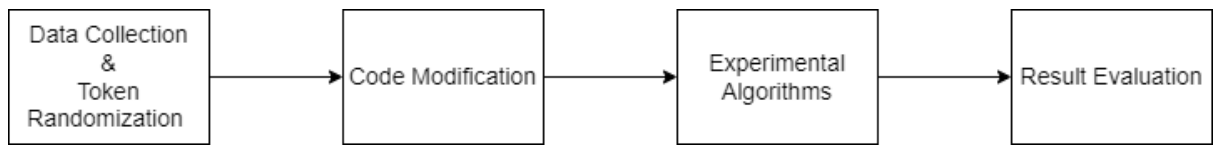


Figure 3.2: Basic Flow of the Research

3.2.1 Data Collection & Token Randomization

In this performance testing of blockchain algorithms, first step is data collection. The code that uses for testing must collect by each algorithm. The input needed is the random token that generated by the code.

3.2.2 Code Modification

To make sure the open sources code fulfill the requirement of the testing result, a modification have to done to set all the testing can be reach the parameter needs in the testing.

3.2.3 Experimental Algorithms

During the experiment, result will be collected, and a calculation will be done to compare the performance of different algorithms.

3.2.4 Result Evaluation

A graph will be generated by using data in multiple testing to ensure the accuracy of the result.

3.3 PROJECT REQUIREMENT

3.3.1 Input

Input data is the randomization token of the tools. There are impossible to create or collect the blocks in blockchain and use the same thing to test since all blocks are unique. Therefore, only random token could be taken as the input of the project.

3.3.2 Output

Output is the data after processing. All the result after testing will be recorded and processed to determine the best algorithms of NFTs. It will be evaluated into a graphical format.

3.3.3 Process Description

The whole testing process include 4 steps. First, the data collection will be done with taken the code from any open source's tools. Hence, tokenization will be done to ensure all data is generated by the code for the input. Furthermore, the code that collect must modified until it hit the requirement of the blockchain testing. All manipulation variables should be considered as great as possible to minimize the inaccuracy of data. Finally, an analyzing of data should be done to view the proper result of testing.

3.3.4 Constraint and Limitation

The constraint of this research is it does not include all algorithms. There is a lot of algorithms in blockchain which include Proof-of-Work, Proof-of-Sakes, Proof-of-Space, Delegated Proof-of-Stake, Proof of Importance, Practical Byzantine Fault Tolerance and Ripple Transaction. It is impossible to us to do experiment on each algorithm, the time spend will be long and waste energy. As a result, determining which algorithms is the best and most accurate for this study has become a restriction.

The limitation of this research could be the size of the dataset. When a huge dataset occurs, it could limit the performance of the testing and cause inaccurate data. Additionally, many attributes and predictor data can be lengthening the time taken for the testing, an error could be happened when the output are stimulated.

3.3.5 Software Requirements

The table below show the software required in this research:

Software	Specification	Purpose
Microsoft Office Word	Version 2019	Used for report documentation.
Microsoft Office PowerPoint	Version 2019	Used for preparing presentation material.
Canva	Version 2021	Used for preparing presentation material.
Draw.io	Version 13.9.9	Used to draw flow chart for the research.
Google Chrome	Version 102.0.5005.61/63	Assist with the search for research regarding to the project
Google Collaboratory	Python 3.7.13	To process data and run the source code for testing algorithms on research purpose.

Table 3.3: Software Requirements of the Research

3.3.6 Hardware Requirements

This research is determining some specific hardware for the testing. The efficiency of hardware and usability of hardware facing high-load data are crucial and highly needed. The table below show the hardware required in this research:

Software	Specification	Purpose
Laptop	ASUS Vivobook A412DA CPU: AMD Ryzen 5 3500U with Radeon Vega Mobile Gfx 2.10 GHz GPU: AMD Radeon Vega 8 Graphic RAM: 12GB SODIMM 2400MHz	Used for development, documentation, and testing algorithms in the research.
Smartphone	Apple iPhone Xs Max CPU: Hexa-core (2x2.5 GHz Vortex + 4x1.6 GHz Tempest) GPU: Apple GPU (4-core graphics) OS: iOS 15.5	Used For Searching information that required to complete this research.
Tablet	Apple iPad 9 th Generation CPU: Hexa-core (2x2.65 GHz Lightning + 4x1.8 GHz Thunder) GPU: Apple GPU (4-core graphics) OS: iPadOS 15.5	Used to record data and analytic. It also included produce chart and diagrams.

Table 3.4: Hardware Requirements of the Research

3.4 PROPOSED DESIGN

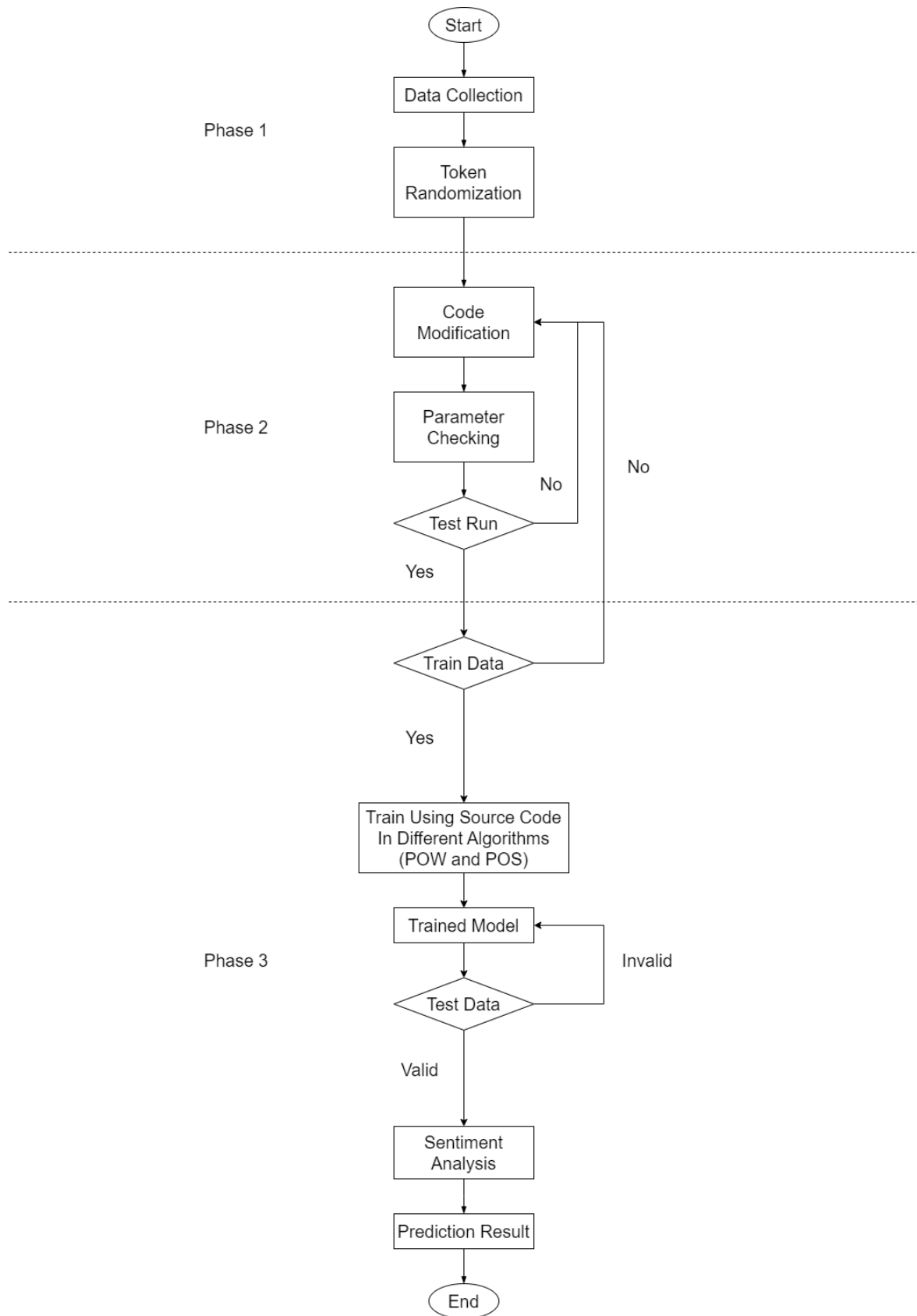


Figure 3.4: Full flow of the Research

Phase 1:

Data Collection is a procedure for browsing from any open-source programme or tools which has potential to do testing in this project. Since there is 2 algorithms have to test in this project, therefore we need to make sure each algorithm has at least one of the testing tools in order to ensure all the result is valid. It is impossible to us to create the blockchain token by our own and collect a big class of data for testing. To make the testing less complicated, a random sample will be chosen which all the blocks will be generated by the testing code. This is what randomization proposed.

Phase 2:

The majority code of the open source given might be not achieved the testing parameter we set. Therefore, a code modification has to done to obtain all the testing parameter that stated in this project can be reached. The Code Modification are not necessarily modified using individual knowledge, but it could be using other sources and combine it as long as the parameter are achieved. After modified the code, all the parameters have been checking and established the performance, security, and stability of the algorithm can be tested. Before the code running and get the real result, a test run must be done to verify any bugs or error occurs after we modified the code.


Phase 3:

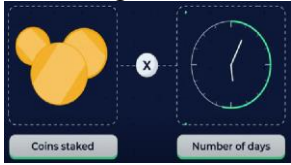
During the last phase, data have been trained with the random token. By using each algorithm stated in the project, an amount of token will be processing, and the trained data will be existed in the procedure. After the data trained, an actual data testing will be conducted, and the result will be recorded. A sentiment analysis will be hand out to filter the needed data and result. To produce a summary of result, all testing data will propose in diagram either graph, table, or chart. Last but not least, a data prediction has to done and justify the result of the testing.

3.5 PROOF OF INITIAL CONCEPT

In this research, three algorithms are filtering from all blockchain algorithms, the algorithms will be experimented in this research is Proof-of-Work (PoW) and Proof-of-Stake (PoS). The reason of used PoW is due to its mining capacity which could produce unlimited nodes for user. PoS are considered in this research because it is a less work and energy consumption on the validation of data blocks.

Table 3.6: Comparative Analysis of the Algorithms used

	Proof of Work	Proof of Stake
Energy Consumption	<ul style="list-style-type: none"> • Required a lot of energy in mining. • The profitability of mining using Proof of Work are decreasingly follow the time. 	<ul style="list-style-type: none"> • Required less work to validate data blocks • Required less energy.
Working Criteria	<p>Virtual miners from all around the world compete to be the first to solve a math challenge to protect and verify proof-of-work blockchains. The winner will receive to update the blockchain with the most recent confirmed transactions and is paid with a certain amount of coin by the network.</p>	<p>No race against nodes to validate a transaction. All user is taking part in a lucky draw and the winner will be selected by blockchain itself. The winner validates the transaction and get smaller reward (it consume less electricity).</p>
How the Algorithms Work	<p>The network will provide a difficulty rate for the nodes (it could be a long prefix like prefix with 20 zeros).</p> <p>Miner frames the header with the following information:</p> <ul style="list-style-type: none"> - Cryptocurrency Version - Previous Blockhash - Merkle Root - Timestamp - Difficulty Target - Nonce <p>Once solved, Miner hash the blockchain twice SHA256 and it becomes current blockhash.</p> <p>Challenge String is what miners must solve which it has a string with guess number: “CS” + <Guess “”> = 000...</p> <p>Example:</p>	<p>Select nodes be a validator by using pseudo-random election process based on:</p> <ul style="list-style-type: none"> - Staking Age - Randomization - Node’s Wealth <p>Nodes chosen by the algorithms is from a pool of candidates. When nodes are choosing to forge the next block, it verifies the validity of the transaction, signed it and add into blockchain.</p> <p>Two technique of Proof of Stakes choose Validators:</p> <p>a. Randomized Block Selection</p> 

	$0 (CS) + 1 (Guess) = \dots 000$ $0 (CS) + 1 (Guess) = \dots 001$	<p>Stake sizes are public, the next forger can typically be predicted by other nodes.</p> <p>b. Coin Age Selection</p>  <p>After forging block, a node's currency age reset to zero. It must wait a specific of time before forging another block. This is used for avoiding the large stakes nodes control consensus mechanism.</p>
Features	<p>a. Mining Capacity depends on computational power.</p> <p>b. Block rewards received by solve a cryptographic puzzle.</p> <p>c. Impracticably of the 51% Attack. Hacker needs to insert malicious block in 51% of the user data.</p>	<p>a. Fixed Number of Coin.</p> <p>b. Transaction Fees as Rewards for Forgers.</p> <p>c. Impracticably of the 51% Attack. Hacker needs to own 51% of all the stakes in network.</p>

3.6 POTENTIAL USE OF PROPOSED SOLUTION

Among this research, it is helpful for NFTs to enhance their platform in different aspect such as performance, security, and stability. In terms of the performance of NFTs, a suitable algorithm may improve the data structure of blockchain, and it could cause a decreasing of the run time on every NFTs. The aims of the performance improve might be make NFTs required less work and energy on validate the data. Nevertheless, a preferable working criterion could help the participant of NFTs use this technology smoothen and fairly.

Equally important, determined an appropriate algorithm are able to provide higher security protection to NFTs. The main issue for NFTs to undergo is solving the security concerns and minimize the risk of fraud occurs in the system. Therefore, a relevant algorithm may use for avoiding any cheats and stolen asset occurs in NFTs. The protection of privacy must include the ownership issue of the system. A safe system should be included a mechanism which may prevent any counterfeit and fake data appear, especially NFTs which have linkage with economy values. As a result, this research are needs for NFTs as an evaluation of the best algorithms to ensure that their blockchain may recognized invalid digital asset and block any illegal transaction happened between users.

Meanwhile, stability might be considered as one of the crucial points of NFTs. If the blockchain system are less stable, it will crack of occurs error while the validation or transaction are process. It could burden the user and extended the system downtime. NFTs may applied a satisfied algorithm to enhance the complexity of the system but decrease the overload of module. An algorithm which can fit NFTs can reduce the overall downtime of the system and increase the availability to the level which can match the majority participant usage.

CHAPTER 4

IMPLEMENTATION, RESULT, AND DISCUSSION

4.1 INTRODUCTION

Chapter 4 describes the implementation and testing of the potential algorithms chosen in methodology. All code used for testing are received from open resources and modified to fulfill the testing parameter. This chapter contains the analysis of data which visualize in a better view for understanding. Data set used and explanation of result are fully discussed in this chapter.

4.2 IMPLEMENTATION PROCESS

The implementation process records all the steps for collecting open sources code and investigates the usability and functionality of the code. By using google collaboration, an online environment of python programming, the code is modified to ensure the output is scalable and the result are available to testing the parameter stated on previous chapter. To visualize the data, Microsoft excel is used by manually entering all the single data and producing a graph to validate the comparative between both algorithms.

4.2.1 Collecting Open Resource Code

In this process, looking for any potential code for different algorithms will be processed. GitHub is an open resource platform which provides a lot of code with different programming languages such as C++, Java, Python etc. Since Python is chosen as the main programming language in this testing, therefore a filter of programming language will be applied during browse potential code.

The screenshot shows the GitHub search interface for the query "proof of work". The search results are filtered by the programming language Python. The left sidebar shows the number of repositories for various categories, and the "Languages" section is expanded to show the count for each language. The main content area displays a list of repository results, including "indutny/proof-of-work", "tevador/RandomX", "tromp/cuckoo", "pk910/PoWFaucet", and "NakamotoInstitute/RPOW".

Category	Count
Repositories	2K
Code	?
Commits	494K+
Issues	274K
Discussions	3K
Packages	2
Marketplace	0
Topics	10
Wikis	15K
Users	58

Language	Count
JavaScript	449
Python	401
Java	138
Go	133
C++	109
Rust	84
C	82

Repository	Stars	Language	Updated
indutny/proof-of-work	108	JavaScript	Oct 4, 2020
tevador/RandomX	1.2k	C++	20 days ago
tromp/cuckoo	785	C++	Dec 5, 2021
pk910/PoWFaucet	256	TypeScript	10 hours ago
NakamotoInstitute/RPOW	123	C	17 days ago

Figure 4.1: How to apply the filter function to browse the potential code.

4.2.2 Test Run Potential Code

Potential code selected must run once in google collaboration to make sure the environment can support the algorithms provided. Test run for the code also helps to verify the output of code and ensure that the data provided can fulfill our testing plan.



```
if __name__ == '__main__':
    powser = Powser(db_path='./pow.sqlites')
    ip = '240.240.240.240'
    prefix, time_remain = powser.get_challenge(ip)
    print(f'...
sha256({prefix} + ???) == {'0'*powser.difficulty}{(powser.difficulty)}...
```

IP: {ip}
Time remain: {time_remain} seconds
You need to await {time_remain - powser.min_refresh_time} seconds to get a new challenge.
...

```
last = int(time())
i = 0
while not powser._verify_hash(prefix, str(i)):
    i += 1
print(int(time()) - last, 'seconds')
print(f"sha256({prefix} + {i}) == {'0'*powser.difficulty}{(powser.difficulty)}")
print(powser._verify_client(ip, str(i), with_msg=True))
powser.close()
```

sha256(aE#pLzQdD133C9X + ???) == 000000000000000000000000(22)...

IP: 240.240.240.240
Time remain: 600 seconds
You need to await 300 seconds to get a new challenge.

34 seconds
sha256(aE#pLzQdD133C9X + 2414542) == 000000000000000000000000(22)
(True, 'Okay.-')

Figure 4.2: Test Run Proof-of-Work



```
Coming from =====> {'Address': 'account4', 'Weight': 16, 'Age': 0}
***Generating new stake block***
***Exchanging temporary blocks with other nodes***
***Picking a winner***
Winner is =====> ['eltneg', '50', '2', 100]
***calling other nodes to announce theirs***
Coming from =====> {'Address': 'account2', 'Weight': 63, 'Age': 1}
***Generating new stake block***
***Exchanging temporary blocks with other nodes***
***Picking a winner***
Winner is =====> ['eltneg', '50', '2', 100]
Coming from =====> {'Address': 'account3', 'Weight': 63, 'Age': 0}
***Generating new stake block***
***Exchanging temporary blocks with other nodes***
***Picking a winner***
Winner is =====> ['eltneg', '50', '2', 100]
Coming from =====> {'Address': 'eltneg', 'Weight': 50, 'Age': 2}
```

Figure 4.3: Test Run Proof-of-Stakes

4.2.3 Code Modification

From the test run of previous steps, the code contains some incomplete parameters which have to fix before used for testing.

- Proof-of-Work
 - i. The process only will run once, it does not fulfill one of the criteria of testing which is “Run multiple set of data”.

- Proof-of-Stake
 - i. The process will run unlimited data without any stopping conditions.
 - ii. The run time of each process is not printed.

To solve this problem, code modification must be done by adding stopping conditions for each code and adding start and end time for Proof-of-Stake code. The stopping condition added must be scalable and easy to modify in case any changes or increasing data set is needed.

For Proof-of-Work:

```
if __name__ == '__main__':
    count = 0
    while count < 50 :
        powser = Powser(db_path='./pow.sqlite3')
        ip = '240.240.240.240'
        prefix, time_remain = powser.get_challenge(ip)
        print(f'''
sha256({prefix} + ???) == {'0'*powser.difficulty}({'powser.diff
iculty})...

IP: {ip}
Time remain: {time_remain} seconds
You need to await {time_remain - powser.min_refresh_time} seco
nds to get a new challenge.
''')
        last = int(time())
        i = 0
        while not powser._verify_hash(prefix, str(i)):
            i += 1
        print(int(time()) - last, 'seconds')
        print(f"sha256({prefix} + {i}) == {'0'*powser.difficulty}({
powser.difficulty})")
        print(powser.verify_client(ip, str(i), with_msg=True))
        count += 1
        print("Counter =" + str(count))
    powser.close()
```

Highlighted line is the modification of the code. By declaring a count number, the code can be controlled by setting the number of looping using while loop. Number 50 is declared on this testing since our data number needed is 50 set of processing. Counter print is added to verify the progress of the code running which how many set of data have been processed.

Proof-of-Stake

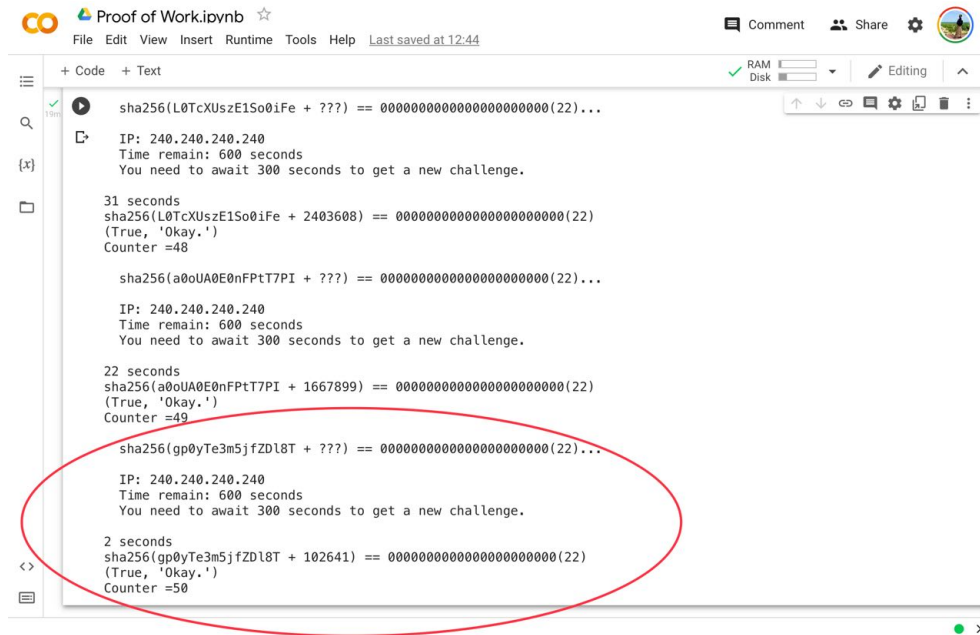
```
from datetime import datetime
import time
from hashlib import sha256
import json, requests
from random import randint
import timeit
...
...
...
...

count = 0
while count < 50:
    start = timeit.default_timer()
    print('===== \n
\n')
    client = clients[randint(0, 3)]
    client.pos()
    count += 1
    print("Counter =" + str(count))
    stop = timeit.default_timer()
    print('Time: ', stop - start)
```

Highlighted line is the modification of the code. The same method are use for modifying the stopping conditions which is by adding the count and set a count number in while loop. To fix the same comparison data, the same number of data is used which is 50 set of process. To solve another problem which is print time of each process, a start and stop time module are added. This is the original module which have been imported by added the first line “import timeit”. Therefore, the code are able to print every process runtime by using the stop time – start time of each process.

4.2.4 Train Data

After the code is modified, an official process will be run. Both algorithms will be run until 50 sets of data are finished and the runtime will be record one by one. The output will be manually recorded in Microsoft Excel and the Data Visualization will be process for the further steps.



```
Proof of Work.ipvnb
File Edit View Insert Runtime Tools Help Last saved at 12:44

+ Code + Text
sha256(L0TcXUzE1So0iFe + ???) == 00000000000000000000(22)...
IP: 240.240.240.240
Time remain: 600 seconds
You need to await 300 seconds to get a new challenge.

31 seconds
sha256(L0TcXUzE1So0iFe + 2403608) == 00000000000000000000(22)
(True, 'Okay.')
Counter =48

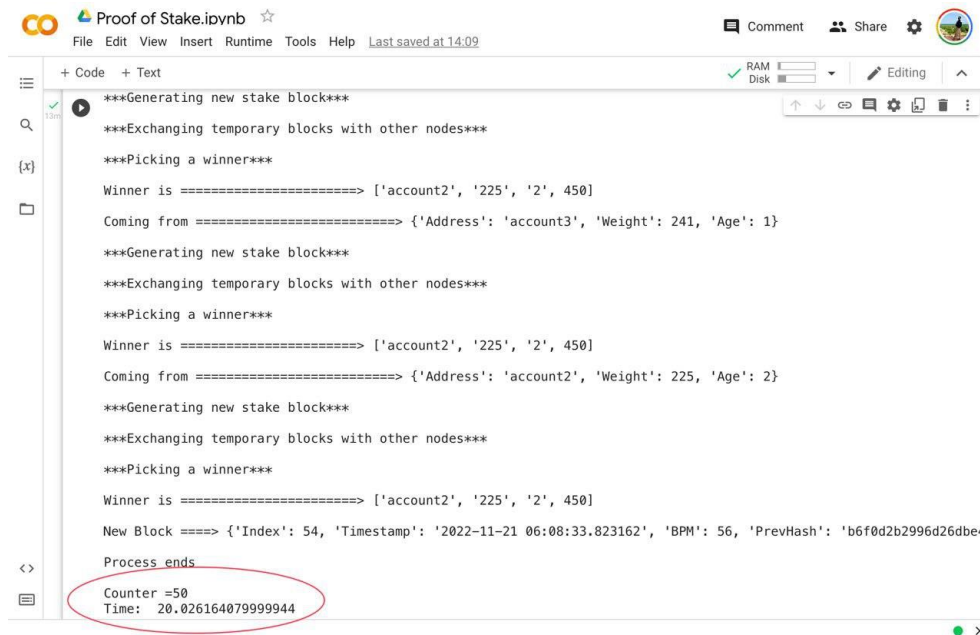
sha256(a0oUA0E0nFPtT7PI + ???) == 00000000000000000000(22)...
IP: 240.240.240.240
Time remain: 600 seconds
You need to await 300 seconds to get a new challenge.

22 seconds
sha256(a0oUA0E0nFPtT7PI + 1667899) == 00000000000000000000(22)
(True, 'Okay.')
Counter =49

sha256(gp0yTe3m5jfZDl8T + ???) == 00000000000000000000(22)...
IP: 240.240.240.240
Time remain: 600 seconds
You need to await 300 seconds to get a new challenge.

2 seconds
sha256(gp0yTe3m5jfZDl8T + 102641) == 00000000000000000000(22)
(True, 'Okay.')
Counter =50
```

Figure 4.4: Completely run for 50 sets of random data in Proof-of-Work.



```
Proof of Stake.ipvnb
File Edit View Insert Runtime Tools Help Last saved at 14:09

+ Code + Text
***Generating new stake block***
***Exchanging temporary blocks with other nodes***
***Picking a winner***
Winner is =====> ['account2', '225', '2', 450]
Coming from =====> {'Address': 'account3', 'Weight': 241, 'Age': 1}
***Generating new stake block***
***Exchanging temporary blocks with other nodes***
***Picking a winner***
Winner is =====> ['account2', '225', '2', 450]
Coming from =====> {'Address': 'account2', 'Weight': 225, 'Age': 2}
***Generating new stake block***
***Exchanging temporary blocks with other nodes***
***Picking a winner***
Winner is =====> ['account2', '225', '2', 450]
New Block =====> {'Index': 54, 'Timestamp': '2022-11-21 06:08:33.823162', 'BPM': 56, 'PrevHash': 'b6f0d2b2996d26dbe4'}
Process ends
Counter =50
Time: 20.026164079999944
```

Figure 4.5: Completely run for 50 sets of random data in Proof-of-Work.

4.3 RESULT (DATA VISUALIZATION)

4.3.1 PERFORMANCE

After all data is collected, two comparative graphs could be built to define the runtime of both algorithms.

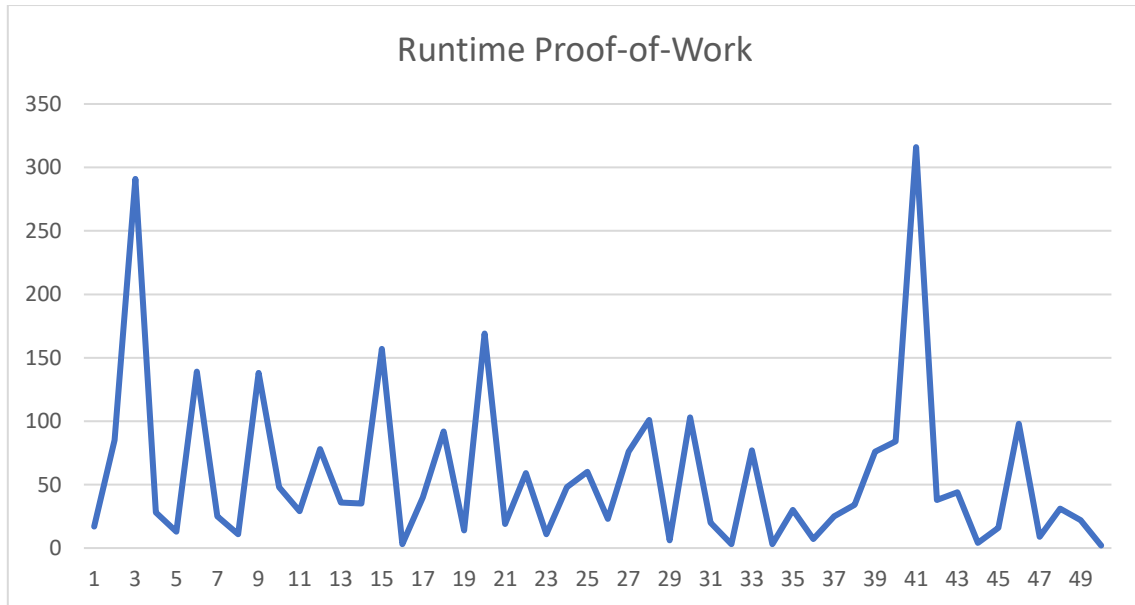


Figure 4.6: The graph of Number of Process vs Runtime in Proof-of-Work

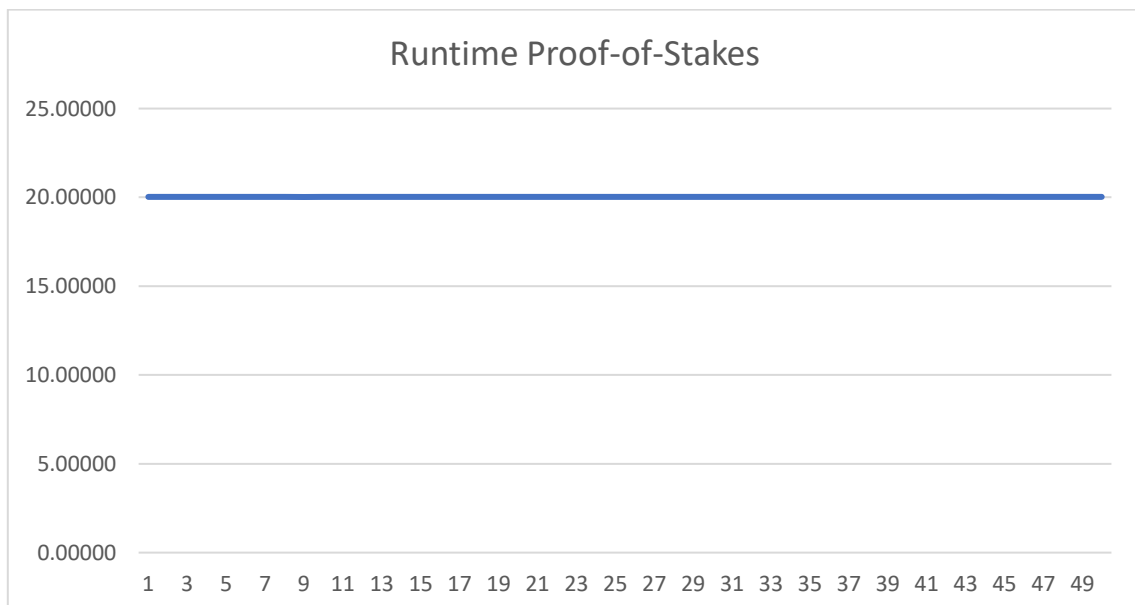


Figure 4.7: The graph of Number of Process vs Runtime in Proof-of-Stake

4.3.2 SCALABILITY

To test the scalability of the algorithms, data are increased from 50 sets to 200 sets. The result shows will be in the cumulative runtime to compare the total runtime of both algorithms in different number of data set.

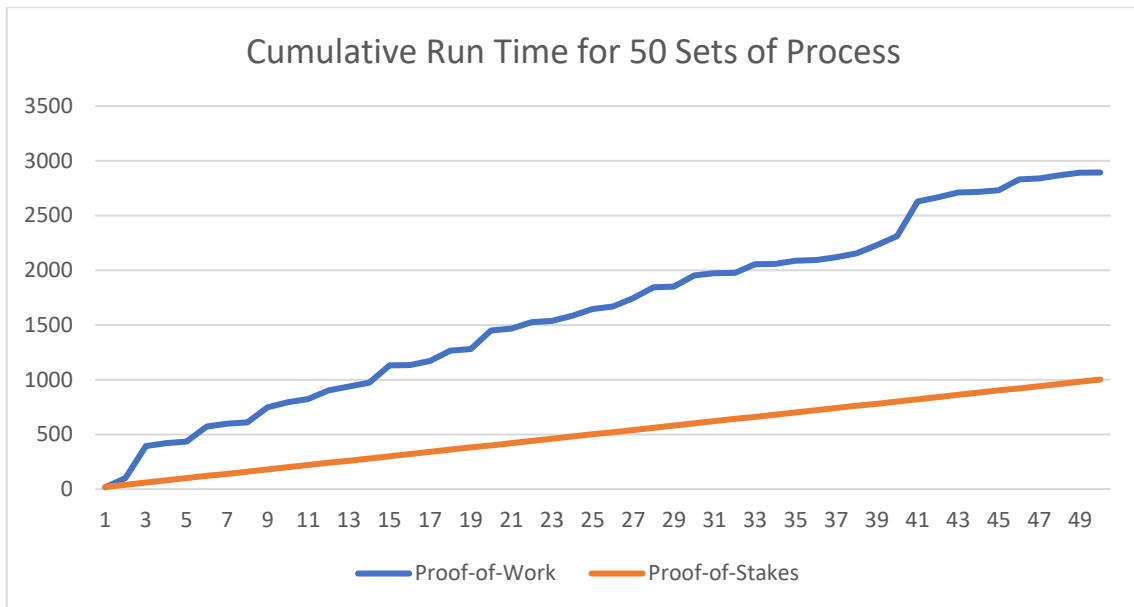


Figure 4.8: The graph of Cumulative Run Time for 50 sets of process for both algorithms

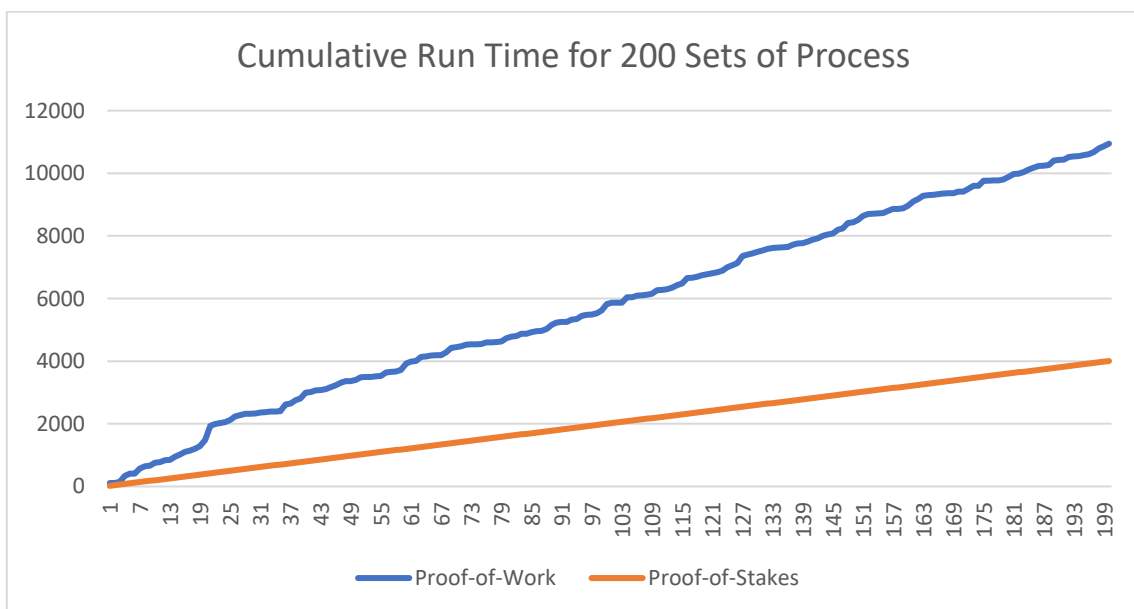


Figure 4.9: The graph of Cumulative Run Time for 200 sets of process for both algorithms

4.3.3 LATENCY

To estimate the latency time in more obvious units, the data record will be in a unit of microsecond (ms). The output will not be in cumulative value due to the comparison of each state and an observation of overlapping is needed.

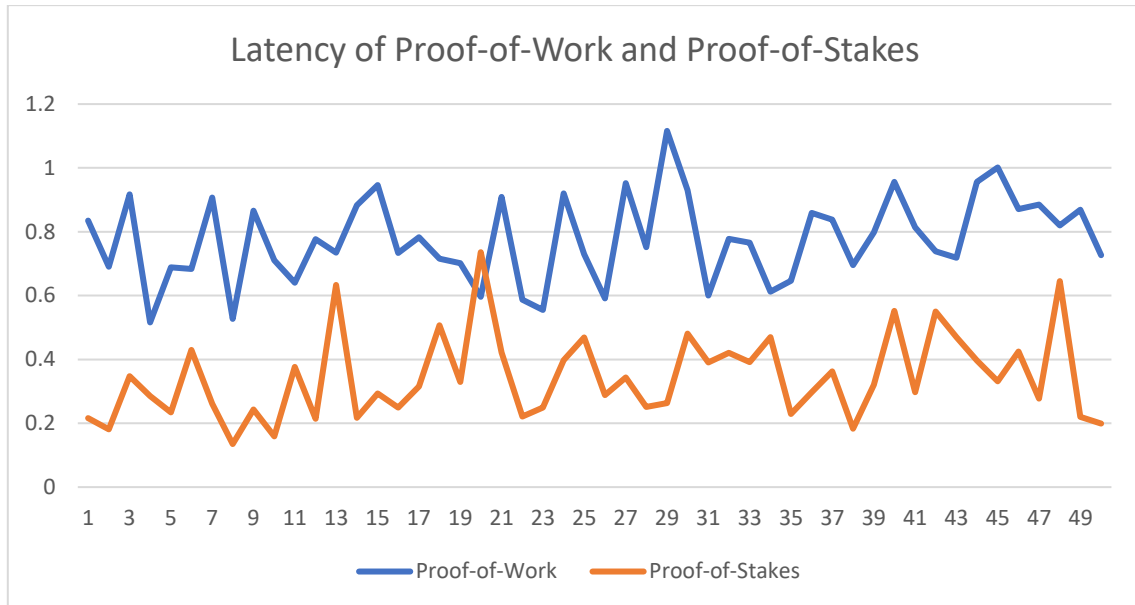


Figure 4.10: The graph of the Latency of Proof-of-Work and Proof-of-Stakes


4.4 DISCUSSION

4.4.1 PERFORMANCE

From the testing process, there is obviously may observed the shape of the graph from both algorithms output. 50 sets of data are good enough for the whole testing since the result have a critically fluctuation graph and an almost constant graph (non-constant are affected by decimal places). Therefore, there is unnecessary to upgrade the data set due to the number of data stated are satisfied and could fulfill the testing.

The graph above is the graph of Number of processes versus runtime in Proof-of-Work algorithm. Following the result shown above, proof-of-work own an unstable graph which have a strongly fluctuation in every processes. The runtime of proof-of-work is highly unstable. The maximum runtime is 316.00s and the minimum runtime is 2.00 second. From the peak of the graph, we may know that the solution is hard to solve due to the higher nonce, therefore the process takes more time to get the valid number for the certain process. For the bottom, since the process takes less time to count the nonce value, this process may declare as easier to solve. As conclusion, more digit of the nonce need to solve, longer of the time taken for the process to complete.


```
316 seconds  
sha256(tnXBA8j8GzeF7K4U + 23867672) == 00000000000000000000(22)  
(True, 'Okay.')
```



```
Counter =41
```

Figure 4.12: The process takes more time to solve an 8-digit nonce.

```
2 seconds  
sha256(gp0yTe3m5jfZDl8T + 102641) == 00000000000000000000(22)  
(True, 'Okay.')
```



```
Counter =50
```

Figure 4.11: The process takes less time to solve a 6-digit nonce.

The graph above is the graph of Number of processes versus runtime in Proof-of-Stake algorithm. Based on the graph above, proof-of-stake shows a nearly constant graph which has a constant runtime for the whole process. While rounding off all the data recorded in this graph, all the runtime is 20.00s which forms a perfect horizontal line in the testing. The maximum runtime is 20.03041s and minimum runtime is 20.02077s. As conclusion, Proof-of-stake own a stable runtime due to the validator are created completely. The algorithm only needs to assign the winner to each validator by random picking. Therefore, a fix process will be happened, but the validator will keep changing randomly due to the picking process.

```

***Calling other nodes to announce theirs***
Coming from =====> {'Address': 'account4', 'Weight': 16, 'Age':
***Generating new stake block***
***Exchanging temporary blocks with other nodes***
***Picking a winner***
Winner is =====> ['account2', '58', '2', 116]
Coming from =====> {'Address': 'eltneg', 'Weight': 50, 'Age':
***Generating new stake block***
***Exchanging temporary blocks with other nodes***
***Picking a winner***
Winner is =====> ['account2', '58', '2', 116]
Coming from =====> {'Address': 'account3', 'Weight': 53, 'Age':
***Generating new stake block***
***Exchanging temporary blocks with other nodes***
***Picking a winner***
Winner is =====> ['account2', '58', '2', 116]
New Block =====> {'Index': 5, 'Timestamp': '2022-11-21 05:52:11.551220', 'BPM': 56,
Process ends

Counter =1
Time: 20.025896180999553
=====

```

Figure 4.12: The process only repeats for assign a random winner to own the block.

4.4.2 SCALABILITY

To test the scalability, various data sets need to be used to compare the effect of data size on the runtime of algorithms. From the beginning, 50 sets of data have using for the testing. To make a big difference, 200 sets of input will be run to verify any delay happened in each algorithm. The reason for using 200 sets is because 100 sets is only double of the original data set, there is hard to make a conclusion on small changes of data size. Therefore, 4 times of the original data set, which is 200 sets are used in order to have a better and obvious observation on the cumulative runtime.

From the 50 sets of data, Proof-of-work algorithms owned an unstable graph due to its runtime are not fix and unpredictable. In contrast, Proof-of-stakes have a constant graph since all the runtime for every step is almost 20 seconds (if decimal places are not included). While looking at the total run time, proof-of-work takes longer time which is 2893 seconds compared to proof-of-stakes which only takes 1001 seconds for running 50 sets of data. The difference of both total runtimes is almost 3 times in 50 sets of data size. Therefore, from the 50 sets data testing, proof-of-stakes have better performance, stability and scalability compared to proof-of-work.

From the 200 sets of data, the same case happened in proof-of-work which an unforeseeable graph is exist in this testing. This issue happened because the data set does not affect the runtime of the single process which still depends on the nonce digit for every random token. However, Proof-of-stakes algorithms produce a sustainable data which could visualize a mostly perfect straight-line graph. The increase of data size makes both algorithms have a grow of total runtime. For proof-of-work, the runtime unpredictable, therefore there is no increased by 4 times (the increased of data set). There is still a long time that would not be acceptable if applied in NFTs which is 10946 seconds for 200 sets of data. Nevertheless, Proof-of stakes with a constant runtime for each process have increased nearly 4 times of the whole process which is 4005 second for 200 sets of data. Compared to the ratio of both increased total run time, proof-of-stakes have higher increased compared to proof-of-work. However, proof-of-work takes a longer time to process the same number data size. Therefore, even if the data size increases, proof-of-stake still take advantages compared to proof-of-work in the aspects of performance, stability, and scalability.

4.4.3 LATENCY

According to the graph above which display the latency in ms of both algorithms, there is unstable data that recorded. However, the average of latency for proof-of-stakes (orange line) algorithms is lower than proof-of-work (blue line). The maximum of latency happened in Proof-of-Work is 1.116 ms which is exceed 1.5 times of the maximum latency of Proof-of-Stakes which owned a value 0.736 ms. Furthermore, The average latency of Proof-of-Work is 0.77690 ms, this is a value that higher than latency of Proof-of-Stakes more than 2 times since there is only hold a average of 0.34354 ms. From the graph, there is only one spike of latency for proof-of-stakes overlapping another algorithm which is proof-of-work. Thus, a summary can be done which is the overall latency in Proof-of-Stakes are lower than Proof-of-Work. This might be brought back to the characteristics of Proof-of-Work which have a mining nonce. When a higher digit nonce are given to be solved, the algorithm takes more time to start the process. However, Proof-of-Stakes with a constant and fixed process will be takes less delay for each process started.

CHAPTER 5

CONCLUSION

5.1 INTRODUCTION

Chapter 5 discuss the relationship of each algorithm and NFTs and link it to the application of blockchain algorithms in the entire implementation. In the others work that has been done, only blockchain algorithms are always compared and stated the advantages and disadvantages of the algorithms. However, there are leaks of research mentioning the implementation of NFTs and the importance of using suitable algorithms in the whole application. This comparative study could be one of the media to verify the suitable algorithms to provide an improvement to the existing NFTs system in order to gain a better performance in server site or user site.

5.2 DISCUSSION ON THE RESULT

From the testing result shown, Proof-of-Work algorithms have owned an unstable data no matter in performance or scalability. This is due to the algorithms characteristic which the system is required to do mining process. The mining process is unpredictable, and the puzzle solved time are highly depending on the digits of nonce. This is the main reason that makes the time of the whole process become longer and unexpected. Furthermore, proof-of-work algorithms provide unlimited tokens which make the token value decrease accordingly. The value of each token is directly proportional to the number of tokens provided on the platform. Therefore, unlimited tokens will make NFTs become worthless and decrease the market value of a digital asset. As a summary, Proof-of-Work algorithms are not suitable algorithms for NFT due to the mining processing and unlimited provided token specification.

Other than Proof-of-Work, Proof-of-Stakes is another algorithm that compared in this research. Proof-of-Stake displayed a definitely opposite part of Proof-of-Work. A stable performance, scalability and less latency data are evaluated after the testing is made. The reason for stability performance in Proof-of-Stakes algorithms can be refer to the restriction in number of tokens. Proof-of-Stakes have a limit on the token, this will cause the lucky draw of the algorithms to be more stable and faster since there is a limit of token provide in the platform. Since the tokens are limited, the issue of valueless digital assets will not occur. There are only such numbers of tokens that will be used for trading and every token is unique. This will make the token keep their value last longer.

As a conclusion, through the output given from this testing algorithms, Proof-of-Stake are the most suitable algorithms that can be apply in NFTs. A stable performance time and less latency is important due to the user having to know about the purchase immediately. It is impossible to let the user wait for a long term and mining the token for purchasing NFTs. Another reason that Proof-of-Stakes are suit to NFTs is the limited token provided. All the digital assets must be valuable, a fixed number of token can limit the transaction flow in NFTs platform and secure all the artwork selling in the website own a merit.

5.3 LIMITATION AND CONSTRAINT

The limitation of the testing is due to the algorithms only can provide an ideal performance for each process. It has no link to direct NFTs system and unable to set the real scenario which can fit the similarity of NFTs. In the reality case, it could be more possibility happened such as number of users, number of assets, mining criteria (for Proof-of-Work), number of tokens limited (for Proof-of-Stakes) and etc. Therefore, this comparison study only can proof the algorithms characteristic and apply the suitable algorithms in NFTs in the ideal case.

The constraint of the study comes from the algorithms used. It is impossible to test all the existing blockchain algorithms in the world. Therefore, only the most potential algorithms can be filtered and proceed to the testing. This does not mean the algorithms that are not applied in this testing have a worse performance compared to these algorithms selected. The selected algorithms only pick by the ideal criteria. Thus, there might be other algorithms that can fit NFTs but do not apply for testing in these comparison studies.

5.4 FUTURE WORK

From the limitation and constraint, there are 2 future works that can be applied in this study. To get more actual data which can meet the reality, the next study should include the real scenario event is virtual data. This should be involved in testing similar data. An idea can be applied which a stimulated NFTs platform can be done and fit the implemented algorithms into the platform to get data in the aspect of performance.

Not only the real problems have to be achieved, but a suggestion for the future work is also using other algorithms which have not been tested in this study to compare with Proof-of-Stake. From the conclusion, Proof-of-Stakes algorithm are the most suitable and ideal blockchain algorithms to fit in to NFTs, therefore Proof-of-Work are not required to testing anymore. Nevertheless, applying the other algorithms to compare with Proof-of-Stakes might get a different result and implemented new suitable algorithms to NFTs. In the opposite site, if Proof-of-Stakes still the best suit algorithms, then it can double clarify that Proof-of-Stakes is the highly recommended algorithms to NFTs applications.

REFERENCE

- statistic_id1265353_daily_nft_sales_value_worldwide_up_until_january.pdf*. (2022). Statista Research Department.
- Brock, W. B. T. J. (2022). *8 Pros and Cons of NFTs & How They Compare to Traditional Investments*. <https://www.annuity.org/2022/01/14/from-the-experts-8-pros-and-cons-of-nfts/>
- Chialisp*. (n.d.). Retrieved June 2, 2022, from <https://chialisp.com/>
- Clark, M. (2021). *NFTs, explained: what they are, and why they're suddenly worth millions - The Verge*. <https://www.theverge.com/22310188/nft-explainer-what-is-blockchain-crypto-art-faq>
- Gopalan, A., Sankararaman, A., Walid, A., & Vishwanath, S. (2020). Stability and Scalability of Blockchain Systems. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 4(2), 1–35. <https://doi.org/10.1145/3392153>
- Hackernoon. (2018). *Proof of Work, Proof of Stake and Proof of Burn*. *Iciv*, 279–283. <https://hackernoon.com/proof-of-work-proof-of-stake-and-proof-of-burn-6823eac2776e>
- Krishnamurthi, R., & Shree, T. (2020). A Brief Analysis of Blockchain Algorithms and Its Challenges. *Research Anthology on Blockchain Technology in Business, Healthcare, Education, and Government, January*, 23–39. <https://doi.org/10.4018/978-1-7998-5351-0.ch002>
- L. M. Bach, B. Mihaljevic, and M. Z. (2018). *Comparative Analysis of Blockchain Consensus Algorithms*. 1545–1550.
- Moss, S. (2022). *Understanding Chia, the cryptocurrency straining storage markets - DCD*. <https://www.datacenterdynamics.com/en/analysis/understanding-chia-the-cryptocurrency-straining-storage-markets/>
- NFT Challenges - NFT*. (n.d.). Retrieved November 29, 2022, from <https://thetokenizer.io/NFT/6-nft-challenges/>
- Proof of space | owlapps*. (n.d.). Retrieved June 2, 2022, from https://www.owlapps.net/owlapps_apps/articles?id=52214944&lang=en
- Saskatoon, S. (2017). *Performance Analysis of Ethereum Transaction in Private Blockchain*.

1–5.

Sharma, R. (2022). *Non-Fungible Token Definition: Understanding NFTs*.

<https://www.investopedia.com/non-fungible-tokens-nft-5115211>

Wang, Q., & Li, R. (2021). A weak consensus algorithm and its application to high-performance blockchain. *Proceedings - IEEE INFOCOM, 2021-May*.

<https://doi.org/10.1109/INFOCOM42981.2021.9488725>

Why would 99% of NFTs fail?. In economics, the concepts of fungible... | by BSN | Blockchain Thought Leadership | Medium. (n.d.). Retrieved November 29, 2022, from <https://medium.com/blockchain-thought-leadership/why-would-99-of-nfts-fail-123d91d66195>

Wood, G. (2019). Ethereum: a secure decentralised generalised transaction ledger. *Ethereum Project Yellow Paper*, 1–32.

Zhu, J., Liu, P., & He, L. (2018). Mining information on bitcoin network data. *Proceedings - 2017 IEEE International Conference on Internet of Things, IEEE Green Computing and Communications, IEEE Cyber, Physical and Social Computing, IEEE Smart Data, IThings-GreenCom-CPSCoM-SmartData 2017, 2018-Janua*, 999–1003.

<https://doi.org/10.1109/iThings-GreenCom-CPSCoM-SmartData.2017.153>

APPENDIX

Gantt Chart:

Task and Milestones	Week													
	1	2	3	4	5	6	7	8	9	10	11	12	13	14
First Meeting with Supervisor	█	█												
Chapter 1 - Introduction			█											
Second Meeting with Supervisor				█										
Chapter 2 - Literature Review					█									
Chapter 3 - Methodology						█	█	█	█	█	█	█	█	█
PSM 1 Presentation														█
PSM 1 Submission														█

Task and Milestones	Week													
	1	2	3	4	5	6	7	8	9	10	11	12	13	14
First Meeting with Supervisor	█													
Code Browsing		█	█											
Code Modification			█	█	█									
Discussion with Supervisor on Enhancement						█								
New Testing Parameter and Code Modification							█	█						
Chapter 4 – Implementation, Result & Discussion								█	█					
Chapter 5 - Conclusion											█			
Poster Design												█		
PSM 2 Presentation														█
PSM 2 Submission														█

Code for Proof-of-Work:

```
#!/usr/bin/env python3
import sqlite3
from time import time
from time import perf_counter
import hashlib
import secrets
'''
db_path: required, where to save the sqlite3 database
difficulty: how many leading zero bits
clean_expired_rows_per: clean expire rows after n inserts
prefix_length: the length of prefix
default_expired_time: how long can a challenger solve the challenge in seconds, default is 10 minutes
min_refresh_time: how long can a challenger get a new challenge, default is half of default_expired_time
'''
class Powser:
    def __init__(
        self,
        db_path,
        difficulty=22,
        clean_expired_rows_per=1000,
        prefix_length=16,
        default_expired_time=None,
        min_refresh_time=None
    ):
        self.db = sqlite3.connect(db_path)
        self.difficulty = difficulty
        self.clean_expired_rows_per = clean_expired_rows_per
        self.prefix_length = prefix_length
        self.default_expired_time = max(600, 2**(difficulty-16))
        if default_expired_time is None else default_expired_time
        self.min_refresh_time = self.default_expired_time // 2
        if min_refresh_time is None else min_refresh_time

        self._insert_count = 0
        self._create_table()

    def get_challenge(self, ip):
        row = self.db.execute('SELECT prefix, valid_until FROM pow WHERE ip = ?', (ip, )).fetchone()
        if row is None:
            return self._update_row(ip)

        prefix, valid_until = row
        time_remain = valid_until - int(time())
```

```

        if time_remain <= self.min_refresh_time:
            return self._update_row(ip)
        return prefix, time_remain

    def verify_client(self, ip, answer, with_msg=False):
        row = self.db.execute('SELECT valid_until, prefix FROM po
w WHERE ip=?', (ip, )).fetchone()
        if row is None:
            return (False, 'Please get a new PoW challenge.') if
with_msg else False
        valid_until, prefix = row
        if int(time()) > valid_until:
            return (False, 'This PoW challenge is expired.') if w
ith_msg else False
        result = self._verify_hash(prefix, answer)
        if not result:
            return (False, 'The answer is incorrect.') if with_ms
g else False
        self._update_row(ip)
        return (True, 'Okay.') if with_msg else True

    def clean_expired(self):
        self.db.execute('DELETE FROM pow WHERE valid_until < strf
time("%s", "now")')
        self.db.commit()

    def close(self):
        self.db.close()

    def _verify_hash(self, prefix, answer):
        h = hashlib.sha256()
        h.update((prefix + answer).encode())
        bits = ''.join(bin(i)[2:].zfill(8) for i in h.digest())
        return bits.startswith('0' * self.difficulty)

    def _update_row(self, ip):
        self._insert_count += 1
        if self.clean_expired_rows_per > 0 and self._insert_count
% self.clean_expired_rows_per == 0:
            self.clean_expired()
        prefix = secrets.token_urlsafe(self.prefix_length)[:self.
prefix_length].replace('-', 'B').replace('_', 'A')
        now = int(time())
        valid_until = now + self.default_expired_time
        data = {
            'ip': ip,
            'valid_until': valid_until,
            'prefix': prefix

```

```

    }
    self.db.execute('INSERT OR REPLACE INTO pow VALUES(:ip, :
valid_until, :prefix)', data)
    self.db.commit()
    return prefix, valid_until - now

def _create_table(self):
    self.db.execute('''
        CREATE TABLE IF NOT EXISTS pow (
            ip TEXT PRIMARY KEY,
            valid_until INTEGER,
            prefix TEXT
        )
    ''')
    self.db.commit()

if __name__ == '__main__':
    count = 0
    while count < 50 :
        powser = Powser(db_path='./pow.sqlite3')
        ip = '240.240.240.240'
        prefix, time_remain = powser.get_challenge(ip)
        print(f'''
sha256({prefix} + ???) == {'0'*powser.difficulty}({powser.diffi
culty})...

IP: {ip}
Time remain: {time_remain} seconds
You need to await {time_remain - powser.min_refresh_time} secon
ds to get a new challenge.
''')
        last = int(time())
        i = 0
        while not powser._verify_hash(prefix, str(i)):
            i += 1
        print(int(time()) - last, 'seconds')
        latency_start = perf_counter()
        latency_end = perf_counter() - latency_start
        print('Latency = {:.10f}s'.format(latency_end))

        print(f"sha256({prefix} + {i}) == {'0'*powser.difficulty}({
powser.difficulty})")
        print(powser.verify_client(ip, str(i), with_msg=True))
        count += 1
        print("Counter =" + str(count))
        powser.close()

```

Code for Proof-of-Stake:

```
"""
    Python implementation of POW
    Credit: https://github.com/mycoralhealth/blockchain-tutorial
"""
from datetime import datetime
import time
from hashlib import sha256
import json, requests
from random import randint
import timeit

DATE = datetime.now()
GENESIS_BLOCK = {
    "Index": 0,
    "Timestamp": str(DATE),
    "BPM": 0, #instead of transactions
    "PrevHash": "",
    "Validator": "" #address to receive the reward {validator, weight, age}
}
GENESIS_BLOCK2 = {
    "Index": 0,
    "Timestamp": str(DATE),
    "BPM": 0, #instead of transactions
    "PrevHash": "",
    "Validator": "" #address to receive the reward {validator, weight, age}
}
GENESIS_BLOCK3 = {
    "Index": 0,
    "Timestamp": str(DATE),
    "BPM": 0, #instead of transactions
    "PrevHash": "",
    "Validator": "" #address to receive the reward {validator, weight, age}
}
GENESIS_BLOCK4 = {
    "Index": 0,
    "Timestamp": str(DATE),
    "BPM": 0, #instead of transactions
    "PrevHash": "",
    "Validator": "" #address to receive the reward {validator, weight, age}
}
```

```

class Blockchain(object):

    def __init__(self, _genesisBlock, account):
        """
            If the genesis block is valid, create chain
        """
        self.blockChain = []
        self.tempBlocks = []
        #self.candidateBlocks = [] #constains block
        self.myCurrBlock = {}
        #self.announcements = []
        self.validators = set() # stakers and balance
        #self.unconfirmed_txns = []
        self.nodes = set()
        self.myAccount = {'Address': '', 'Weight': 0, 'Age': 0}
        self.myAccount['Address'] = account['Address']
        self.myAccount['Weight'] = account['Weight']
        try:
            genesisBlock = self.generate_genesis_block(_genesisBl
ock)

            if self.is_block_valid(genesisBlock):
                self.blockChain.append(genesisBlock)
            else:
                raise Exception('Unable to verify block')
        except Exception as e:
            print('Invalid genesis block.\nOR\n' + str(e))

    def is_block_valid(self, block, prevBlock={}):
        try:
            _hash = block.pop('Hash')
        except KeyError as e:
            return False
        try:
            hash2 = self.hasher(block)
            assert _hash == hash2
        except AssertionError as e:
            return False

        prevHash = prevBlock['Hash'] if prevBlock else ''
        block['Hash'] = _hash
        if self.blockChain:
            prevHash = self.blockChain[-
1]['Hash'] if not prevHash else prevHash
        try:
            assert prevHash == block["PrevHash"]
        except AssertionError as e:

```



```

        if prevHash == self.blockChain[0]['Hash']:
            block['Hash'] = _hash
            return True
        block['Hash'] = _hash
        return False
    block['Hash'] = _hash
    return True

    def generate_new_block(self, bpm=randint(53, 63), oldBlock='',
, address=''):
        if self.myCurrBlock:
            return self.myCurrBlock
        prevHash = self.blockChain[-1]['Hash']
        index = len(self.blockChain) if not oldBlock else oldBlock
k['Index'] + 1
        address = self.get_validator(self.myAccount) if not addr
ess else address
        newBlock = {
            "Index": index,
            "Timestamp": str(datetime.now()),
            "BPM": bpm, #instead of transactions
            "PrevHash": prevHash,
            "Validator": address
        }
        newBlock["Hash"] = self.hasher(newBlock)
        assert self.is_block_valid(newBlock)
        self.myCurrBlock = newBlock
        return newBlock

    def get_blocks_from_nodes(self):
        if self.nodes:
            for node in self.nodes:
                #resp = requests.get('http://{}/newblock'.format(
node))

                node.add_another_block(self.myCurrBlock)
                resp = node.generate_new_block()
                if self.is_block_valid(resp): #resp.json()
                    #self.tempBlocks.append(resp.json())
                    if not resp['Validator'] in self.validators:
                        self.tempBlocks.append(resp)
                        self.validators.add(resp['Validator'])

    def add_another_block(self, another_block):
        if self.is_block_valid(another_block):
            if not another_block['Validator'] in self.validators:
                self.tempBlocks.append(another_block)
                self.validators.add(another_block['Validator'])

```

```

def pick_winner(self):
    """Creates a lottery pool of validators and choose the va
lidator
        who gets to forge the next block. Random selection we
ighted by amount of token staked

        Do this every 30 seconds
    """
    winner = []

    self.tempBlocks.append(self.myCurrBlock)
    self.validators.add(self.myCurrBlock['Validator'])
    for validator in self.validators:
        acct = (validator.rsplitt(sep=', '))
        acct.append(int(acct[1]) * int(acct[2]))
        if winner and acct[-1]:
            winner = acct if winner[-1] < acct[-
1] else winner
        else:
            winner = acct if acct[-1] else winner
    if winner:
        return winner
    for validator in self.validators:
        acct = (validator.rsplitt(sep=', '))
        acct.append((int(acct[1]) + int(acct[2]))/len(acct[0]
))
        if winner:
            winner = acct if winner[-1] < acct[-
1] else winner
        else:
            winner = acct
    return winner

def pos(self):
    """
    #get other's stakes
    #add owns claim
    #pick winner
    """

    print(str(self.myAccount) + ' =====> Ge
tting Valid chain\n')
    self.resolve_conflict()
    time.sleep(1)
    self._pos()
    print('***Calling other nodes to announce theirs***' + "\
n")
    time.sleep(1)

```

```

    for node in self.nodes:
        node._pos()
    time.sleep(1)
    for block in self.tempBlocks:
        validator = block['Validator'].rsplit(', ')
        if validator[0] == self.pick_winner()[0]:
            new_block = block
            break
        else:
            pass
    print('New Block =====> ' + str(new_block) + "\n")
    time.sleep(1)
    self.add_new_block(new_block)
    for node in self.nodes:
        node.add_new_block(new_block)
    print('Process ends' + "\n")

def announce_winner(self):
    self.blockChain.append(self.myCurrBlock)

def add_new_block(self, block):
    if self.is_block_valid(block):
        #check index too
        self.blockChain.append(block)
        acct = block['Validator'].rsplit(', ')
        if self.myAccount['Address'] != acct[0]:
            self.myAccount['Age'] += 1
        else:
            self.myAccount['Weight'] += (randint(1, 10) * self
f.myAccount['Age'])
            self.myAccount['Age'] = 0
    self.tempBlocks = []
    self.myCurrBlock = {}
    self.validators = set()

def _pos(self):
    print("Coming from =====> " + str(se
lf.myAccount) + "\n")
    time.sleep(1)
    print('***Generating new stake block***' + "\n")
    time.sleep(1)
    self.generate_new_block()
    print('***Exchanging temporary blocks with other nodes***
' + "\n")
    time.sleep(1)
    self.get_blocks_from_nodes()

```

```

        print('***Picking a winner***' + "\n")
        time.sleep(1)
        print("Winner is =====> " + str(self.pick_winner()) + "\n")

    def resolve_conflict(self):
        for node in self.nodes:
            if len(node.blockChain) > len(self.blockChain):
                if self.is_chain_valid(node.blockChain):
                    print('***Replacing node***' + "\n")
                    self.blockChain = node.blockChain
                    return
        print('***My chain is authoritative***' + "\n")
        return

    def is_chain_valid(self, chain):
        _prevBlock = ''
        for block in chain:
            if self.is_block_valid(block, prevBlock=_prevBlock):
                _prevBlock = block
            else:
                return False
        return True

    def add_new_node(self, new_node):
        self.nodes.add(new_node)
        new_node.add_another_node(self)

    def add_another_node(self, another_node):
        self.nodes.add(another_node)

    @staticmethod
    def hasher(block):
        block_string = json.dumps(block, sort_keys=True).encode()
        return sha256(block_string).hexdigest()

    @staticmethod
    def get_validator(address):
        return ', '.join([address['Address'], str(address['Weight']), str(address['Age'])])

    def generate_genesis_block(self, genesisblock):
        address = {'Address': 'eltneg', 'Weight': 50, 'Age': 0}
        address = self.get_validator(address)
        genesisblock['Index'] = 0 if not genesisblock['Index'] else genesisblock['Index']
        genesisblock['Timestamp'] = str(datetime.now()) if not genesisblock['Timestamp'] else genesisblock['Timestamp']

```

```

        genesisblock['BPM'] = 0 if not genesisblock['BPM'] else g
genesisblock['BPM']
        genesisblock['PrevHash'] = '0000000000000000'
        genesisblock['Validator'] = address if not genesisblock['
Validator'] else genesisblock['Validator']
        genesisblock['Hash'] = self.hasher(genesisblock)
        return genesisblock

def main():
    """Run test"""
    account = {'Address': 'eltneg', 'Weight': 50}
    account2 = {'Address': 'account2', 'Weight': 55}
    account3 = {'Address': 'account3', 'Weight': 43}
    account4 = {'Address': 'account4', 'Weight': 16}
    blockchain = Blockchain(GENESIS_BLOCK, account)
    blockchain.generate_new_block(52)

    blockchain2 = Blockchain(GENESIS_BLOCK2, account2)
    blockchain3 = Blockchain(GENESIS_BLOCK3, account3)

    clients = [blockchain, blockchain2, blockchain3]

    blockchain.add_new_node(blockchain2)
    blockchain.add_new_node(blockchain3)

    blockchain2.add_new_node(blockchain)
    blockchain2.add_new_node(blockchain3)

    blockchain.get_blocks_from_nodes()
    blockchain2.get_blocks_from_nodes()

    blockchain.pick_winner()
    #check if temp blocks are same

    blockchain.pos()
    blockchain2.pos()
    blockchain3.pos()

    blockchain4 = Blockchain(GENESIS_BLOCK4, account4)
    blockchain4.add_new_node(blockchain)
    blockchain4.add_new_node(blockchain2)
    blockchain4.add_new_node(blockchain3)
    blockchain4.pos()
    clients.append(blockchain4)

    count = 0
    while count < 50:
        start = timeit.default_timer()

```

```
n')
    print('===== \n\

    client = clients[randint(0, 3)]
    client.pos()
    count += 1
    print("Counter =" + str(count))
    stop = timeit.default_timer()
    print('Time: ', stop - start)

    latency_start = time.perf_counter()
    latency_end = time.perf_counter() - latency_start
    print('Latency = {:.10f}s'.format(latency_end))

if __name__ == '__main__':
    main()
```