

Parallel and Distributed Computing in E-voting using Blockchain Mining System

A PROJECT REPORT

Submitted by

Vrishab V Srivatsa (18BCI0074)

Vishnu Shetty B (19BCE2116)

Pratyush Kumar (19BCE0506)

CSE4001 PARALLEL AND DISTRIBUTED COMPUTING

Slot –B1

Computer Science and Engineering

DECEMBER 2021

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	LIST OF FIGURES	2
1.	ABSTRACT	3
2.	INTRODUCTION 2.1. Blockchain 2.2. Mining 2.3. Distributed System 2.4. Parallelisation	4
3.	LITERATURE SURVEY	7
4.	ARCHITECTURE	9
5.	METHODOLOGY 5.1. Distributed Database 5.2. Simulation of votes/transactions 5.3. Parallelisation	10
6.	PROCEDURE 6.1. Mining Code using Multiprocessing (Python)	14
7.	RESULTS 7.1. Serial Mining of Blocks 7.2. Parallel Mining of Blocks	29
8.	ANALYSIS 8.1. Comparison 8.1.1. Pre-conditions 8.2. Analysis 8.3. Statistical Analysis	31
8.	CONCLUSION	34
9.	REFERENCES	35

LIST OF FIGURES:

1. Fig 4.1 Architecture diagram
2. Fig 5.1 Distributed database architecture
3. Fig 5.2 Nonce Value Calculation
4. Fig 5.3 Static Scheduling
5. Fig 6.1 Landing Page of the application
6. Fig 6.2 Signup Page of the application
7. Fig 6.3 Main Page of the application
8. Fig 6.4 Candidate selection and Voting Page
9. Fig 6.5 Admin Page of the application - Approving eligible voters
10. Fig 6.6 Administration and management of elections and candidates in the application
11. Fig 6.7 Admin Main Page
12. Fig 6.8 Generating votes and Mining into Blocks
13. Fig 6.9 Mined Blocks and Details
14. Fig 6.10 Verifying Votes by recalculating Merkle Hash
15. Fig 6.11 Verifying Votes by recalculating Merkle Hash (Data is tampered)
16. Fig 6.12 Syncing Vote data either from Local Backup or Firebase
17. Fig 6.13 Voter Data in Firebase
18. Fig 6.13 Voter Data (Confirmed Transactions in Block)
19. Fig 6.14 Block Data (Mined Blocks with n Confirmations)
20. Fig 7.1 CPU and Memory consumption (Serial mining)
21. Fig 7.2 Mining status and Time Complexity (Serial)
22. Fig 7.3 CPU and Memory consumption (Parallel Mining)
23. Fig 7.4 Mining status and Time complexity (Parallel)
24. Fig 8.1 Performance Chart (Blockchain difficulty vs Mining Time for n processes)
25. Fig 8.2 Performance Improvement using 4 processes vs Blockchain difficulty
26. Fig 8.3 Performance Improvement using 2 Processes vs Blockchain difficulty

1. ABSTRACT:

In our project, we will be demonstrating the use of Parallel mining and the distributed nature of Blockchain through a secure voting website. On the voting website, the users get to vote in a secure manner by using the properties of the blockchain. Whenever a user casts his/her vote the transaction is added to a block, based on timestamp. At the same time, this transaction is stored on multiple distributed remote databases. This block is then mined at the server side while counting the votes. Here we use game theory to generate nonce value in an inefficient and faster way using multiple threads with the help of parallel mining. In our project, we show the difference between sequential mining and parallel mining. Parallel mining helps to increase the hash rate to mine new and valid blocks for blockchain. We further improve this using parallel processing for mining the blockchain using multi-threading combined with game theory to improve the efficiency of mining. In case during the mining we realise that the block has been tampered with, we immediately request the data from the distributed database. This way the voting of the citizens is extremely secure and transparent as the transaction data is transparent to all users. Unless the hackers access and manipulate greater than 51% of the distributed devices the transaction cannot be tampered with. Thus in our project, we demonstrate the use of parallel mining and distributed storage of blockchain through a secure voting platform.

2.INTRODUCTION:

2.1.Blockchain:

Blockchain is a decentralized peer-to-peer network that allows data to be stored on thousands of servers following a collective trust model among these unknown peers. Blockchain is mainly used for cryptocurrencies and the initial release of the concept was with the introduction of Bitcoin. Presently the Blockchain technology is used in various other domains such as Global Trade and Commerce, Real Estate, Capital markets, and Asset Management. In Blockchain all the transactions are recorded in a distributed immutable ledger which ensures that all the transactions that are done among nodes are available at each node and the transactions cannot be deleted or edited. blockchain technology offers a way to securely and efficiently create a tamper-proof log of sensitive activity. This includes anything from international money transfers to shareholder records. Financial processes are radically upgraded to offer companies a secure, digital alternative to processes run by a clearinghouse. Altogether avoiding these often bureaucratic, time-consuming, paper-heavy, and expensive processes. When you write data to a blockchain, it gets etched on the network. When you have a series of transactions over time, you gain an accurate and immutable audit trail. This is very useful for financial audits. Having data stored in a place where no single entity owns or controls it, and no one can change what's already written, gives you benefits similar to double-entry book-keeping. Ultimately, this means that there are fewer chances of errors or fraud.

2.2.Mining:

When using bitcoins or other cryptocurrencies, blockchain mining is a process that verifies each stage of the transaction. The people participating are known as blockchain miners, and their primary goal is to confirm the movement of cash from one computer in the network to another through a maze of computing gear and software. The 'blocks' and 'chain' structure of blockchains give them their name. The blocks are made up of several bitcoins, which are small units that contain all of the data code separately. The connections that connect one neighbourhood block to the next are referred to as the chain. Each blockchain represents a unique code authentication that is stored in the network software and is expressly encrypted. The phrase "blockchain mining" describes the process of adding transaction data to the

bitcoin blockchain at its most basic level. These Blockchain miners set up and run specialised Blockchain mining software on their computers, which allows them to safely interact with one another. When a machine downloads the software, joins the network, and starts mining bitcoins, it is referred to as a 'node.'

All of these nodes work together to interact with one another and process transactions in order to add new blocks to the blockchain, also known as the bitcoin network. This bitcoin network is active 24 hours a day. Since its inception in 2009, it has never been hacked or encountered downtime while processing millions of dollars in bitcoin transactions. There is nothing like a centralised authority—a regulatory agency, a governing body, a bank—in Blockchain to ensure bitcoin transactions are completed. Any user with mining hardware and Internet connection may join the mining community and participate. Proof of work is a tough mathematical challenge that is used to solve the procedure. Proof of work is required to validate the transaction and receive a payment for the miner. The miners compete amongst themselves to mine a specific transaction, with the miner who solves the riddle first receiving the prize. Miners are members in the network who have the requisite hardware and processing capacity to validate transactions.

2.3.Distributed System:

The management of the distributed immutable ledger, the backbone of Blockchain, is one of the main use cases of Distributed Computing in Blockchain. Distributed Computing ensures that every node in the network has a copy of the blockchain and that it is updated in real-time when a transaction takes place. Use of Distributed Computing along with the peer-to-peer architecture, blockchain also eliminates the use of a central authority to manage the network thus making the blockchain secure, transparent, and immutable.

Another key feature in the Distributed Network of Blockchain is maintaining the present state of the ledger. Once a set of transactions is validated and a block is created by a miner that block is added to the Blockchain. Even though there are many miners working on block creation the final block that is being added to the blockchain will be created by a single winning miner and all the other nodes in the blockchain should agree and synchronize with the present state of the Blockchain. This is known as the consensus protocol followed by the

Blockchain and it helps the network to achieve reliability and maintain trust within the peers in the Blockchain. Distributed Computing plays a major role in maintaining this consensus protocol as every node has to communicate with its neighbour node in order to arrive at a common decision.

Blockchain is nothing but another Distributed System that heavily uses the concepts and elements of Distributed Systems and every computation that takes place in the blockchain can be stated as Distributed System Computing.

2.4.Parallelisation:

Parallelization is the act of designing a computer program or system to process data in parallel. Normally, computer programs compute data serially: they solve one problem, and then the next, then the next. If a computer program or system is parallelized, it breaks a problem down into smaller pieces to be independently solved simultaneously by discrete computing resources. When optimized for this type of computation, parallelized programs can arrive at a solution much faster than programs executing processes in serial.

Parallelization has become a common technique in scientific computing to fasten the execution time of existing problems and to compute larger and more resource-intensive problems. The efficiency of parallel computing depends mainly on several factors. Such as the distributed search, allocation of resources and election process of a coordinator. To do these, one of the popular algorithms is the election algorithm. Here, a coordinator can be changed after a certain period or the coordinator failed to perform his/her responsibility. Our proposed algorithm is based on a similar type of technique which is discussed in the following section.

A blockchain is a distributed ledger forming a distributed consensus on a history of transactions and is the underlying technology for the Bitcoin cryptocurrency. Its applications are far beyond the financial sector. The transaction verification process for cryptocurrencies is much slower than traditional digital transaction systems. One approach to scalability or the speed at which transactions are processed is to design a solution that offers faster Proof of Work. In this project, we propose a method for accelerating the process of Proof of Work based on parallel mining rather than solo mining. The goal is to ensure that no more than two or more miners put the same effort into solving a specific block. The proposed method

includes the distribution of work. This method has been implemented in a test environment that contains all the characteristics needed to perform Proof of Work for blockchain and has been tested, using a variety of case scenarios, by varying the difficulty level and a number of validators.

3.LITERATURE SURVEY

S. No.	Title of the Paper and year	Algorithms	Scope for future work
1.	Parallel Mining in Blockchain for Bitcoin using Game Theory (2019)	Generate various blocks using different nonce values in order a final constrained block as per the blockchain model can be generated. The block can be generated using a game theory approach and evaluation of performance can be evaluated. Multi-Threading tools are used for creation and generation of blocks in order to improve the performance of the system	Take into account strategic decisions related to punishment and cooperation between miners over repeated games. Those games naturally emerge in the sequential solution of multiple puzzles.
2.	A Parallel Proof of Work to Improve Transaction Speed and Scalability in Blockchain Systems (2019)	To implement this approach on the web, a Docker container has been used. Docker provides a Linux based container with its own network interface. A dedicated network has been created in Docker where all peers will be connected.	Multiple miners solve the hash at the same time. When two miners solve a hash at the same time, one of their solutions will be selected by the manager for the next block as the previous hash. That data will also be broadcast to all miners by the manager, along with transaction data and range of nonces.
3.	Improving Transaction Speed and Scalability of Blockchain Systems via Parallel Proof of Work (2020).	A peer-to-peer network has been developed by using the GX library of Golang [21]. This is a decentralized package manager that is used to distribute the same program to different nodes. In order to perform the Proof of Work, a SHA-256 cryptographic hash algorithm has been used.	In future work, we will evaluate this aspect, based on a real-time network

4.	Design and Development of a Parallel Proof of Work for Permissionless Blockchain Systems (2019)	As the miner works separately, it may happen that multiple miners can use the same transaction data and nonce to create the next block. As the miners do not share the data they are using to find the cryptographic solution, there is no way to know if they are using the same data.	The different dynamic factor of the network such as network latency, throughput, and network bandwidth are not considered during the evaluation.
5.	A Proof-of-Work Parallel-Chain Architecture for Massive Throughput (2019)	The Chainweb design embraces parallelism as a way to linearly increase throughput with each additional chain added to the network.	The energy consumption needs to be evaluated.
6.	IoT Meets Blockchain: Parallel Distributed Architecture for Data Storage and Sharing (2018)	PDash is a new architecture for blockchain systems which aims to solve the scalability problem at the data level. PDash separates the data and transaction on blockchain to release storage burden of nodes	Deploying the solution in Bitcoin testnet to compare the scalability with Bitcoin in the real-life network. The parallel distributed architecture has the full potential to be the infrastructure of the Internet of Things.
7.	DiPETrans: A Framework for Distributed Parallel Execution of Transactions of Blocks in Blockchain (2019)	Use a cluster of machines that form a community to execute or validate the transactions in a distributed manner based on a static analysis of the transactions. This improves the performance of both block mining and validation. The community has a leader machine and a set of follower machines	Incorporating nesting is in the proposed framework. Adopt a wholly distributed approach within the community instead of the leader-follower approach that is resilient to failures and other faults. Integrate it with Ethereum blockchain by deploying a DiPETrans community smart contract.
8.	Efficient Parallel Machine Learning-based Blockchain Framework (2021)	To show what the combination of blockchain and machine learning is capable of, in this paper, we proposed a parallel framework to find out suitable hyperparameters of deep learning in a blockchain environment by using a metaheuristic algorithm.	Solve open issues to such as taking into account the additional costs to transfer the data, models and parameters, unnecessary waiting for the synchronization and communication, to further enhance the performance of the proposed framework

9.	SlimChain: Scaling Blockchain Transactions through Off-Chain Storage and Parallel Processing (2020)	We implement an end-to-end prototype of SlimChain in the Rust programming language. It includes functionalities of node discovery, gossip broadcast, and P2P RPC messaging;6 (iv) SGX enclave. The Rust SGX SDK is used to implement the secure enclave	Propose optimizations to reduce network transmissions and a new sharding technique.
10.	Increasing TPS rate of state-based blockchains by parallel mining (2020)	Ethereum block-chain, obtained by using the Google Big Query framework.13 A block consist of at most 250 transactions, which is same as the Ethereum block size. In the following experiments, we vary the number of regions from 3 to 11 and study the effect of parallel mining on TPS rates.	Increasing the TPS rate of a state-based block-chain, by refining a parallel mining model.

4.ARCHITECTURE:

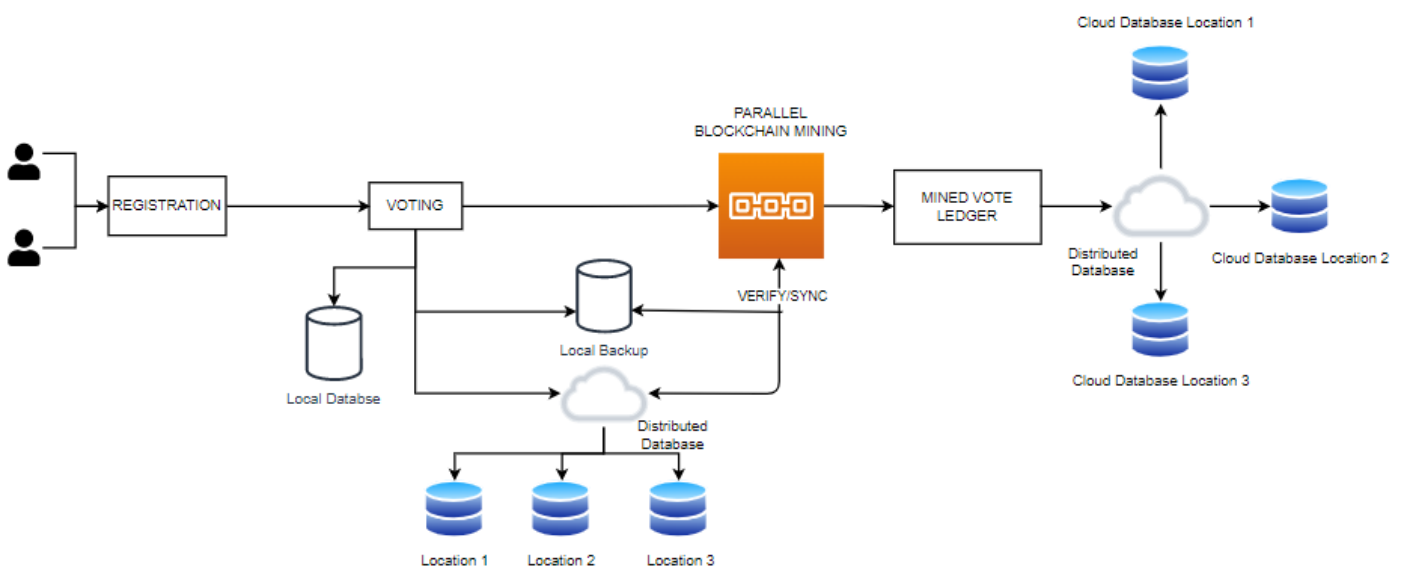


Fig 4.1 Architecture diagram

5.METHODOLOGY

We are proposing a secure e-voting system that harnesses the advantages of blockchain mining, parallel processing as well as distributed database networks. In our project, the people can vote securely from a comfortable location as long as they are connected to the internet. When a user sign's up we give them a unique key that can be used while voting, this makes it secure. Votes are synonymous with transactions that happen across a blockchain network, and in our project, whenever a user has cast their vote, the unconfirmed vote is temporarily stored outside the blockchain across a distributed network of databases to avoid tampering. Once all the votes are cast in the election, and the election has been closed for result tabulation, miners inside the blockchain network (election commission and others) will be able to validate the votes, in blocks of 5 votes (or more). In our proposed method we aim to parallelise this process of mining to reduce the time taken by the verification. Once the votes are confirmed and the complex math problem of mining blocks in the blockchain is completed (depending on the difficulty of the hash), the confirmed blocks are added to the blockchain. Once all the unconfirmed votes have been mined in the blockchain, the results of the election will be made publicly available and stored in a distributed network. Each member in the network has a copy of the exact same data in the form of a distributed ledger. If a member's ledger is altered or corrupted in any way, it will be rejected by the majority of the members in the network. This ensures that there is no voter fraud.

Our project has been divided into three different parts:

5.1.Distributed Database:

When the vote is cast the transaction is temporarily stored in a distributed network of databases. We also create a local backup and store this data to add an extra layer of security. This is to prevent tampering before the block has been mined. When the transactions are being mined, we once again compare the server database with the distributed network to check for any tampering. In case the value is different we sync with the distributed network. This way by making use of a distributed network of databases it makes our application more secure and reliable. Once the block is mined, the data is stored across the distributed network and is made public. Each member in the network has a copy of the exact same data in the

form of a distributed ledger. If a member's ledger is altered or corrupted in any way, it will be rejected by the majority of the members in the network. This ensures that there is no voter fraud.

Distributed database serves as the backbone of our project and increases the reliability of our voting system. By using a distributed system it is no longer easy to tamper with the voter data as it would mean that the hacker needs to hack more than 50% of the network to tamper with the data. Tampering with the whole network is very difficult and next to impossible as all the databases are located in different locations. In case the local database is tampered with before the mining, it can be easily detected when we compare the values with the distributed network. If the data is tampered then we can recover the original data by syncing the local database with the distributed database. Once the transaction has been mined into a block it is again distributed over the network and is made public to prevent any kind of tampering. Our proposed system is thus providing multiple backups of data to increase the security.

In our project we make use of a Local Backup as well as a Firebase located at a different location. This makes our voting system very secure and less vulnerable to tampering.

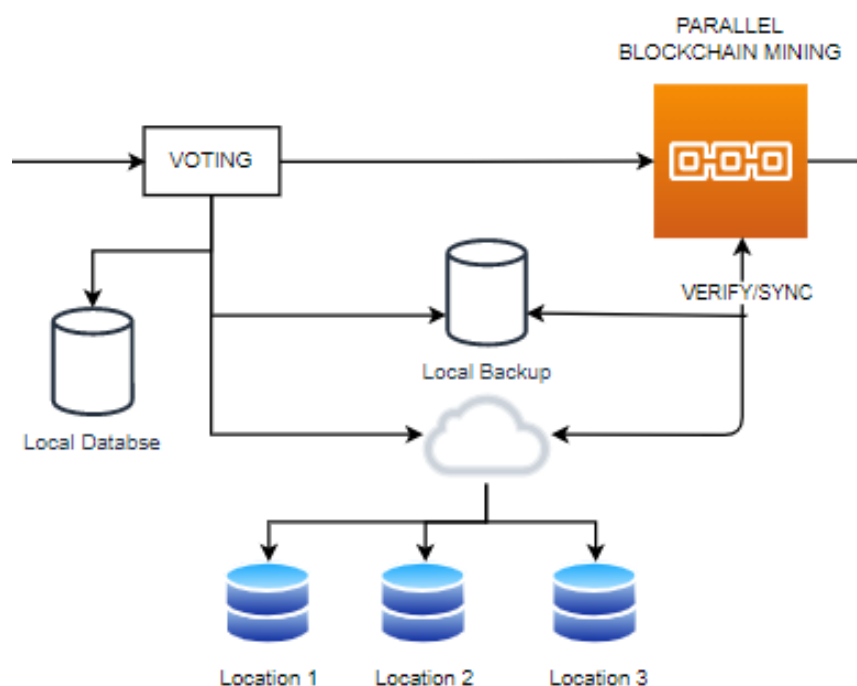


Fig 5.1 Distributed database architecture

5.2.Simulation of votes/transactions

In our project in order to demonstrate the parallel mining of the blockchain we need to simulate the votes. Although in real life people would be voting, for our project implementation we will be simulating the voting. In our simulation algorithm, whenever a vote is cast it is first stored in the local database, then a copy is stored as a local backup, and finally one more backup is stored in the distributed database.

While simulating the votes we had to use serial simulation, this is because sql does not support concurrent insertion of data. But in order to save time we turned off autocommit and created the simulation. In the simulation the automated function first casts the vote. The vote is stored in the local database. After this a copy is stored in the local backup, and also in the distributed database network. We run a loop for this voting process to obtain the desired number of votes. Once the vote simulation is over we can then proceed to the parallel mining of the blocks. The simulation function is very efficient and simulates the live voting scenario by voting based on a random function. This way the simulated results will be very similar to the live voting process.

5.3.Parallelisation:

In this project we make use of parallel techniques to reduce the time of the Proof of work(nonce value) calculation. Whenever the transactions are mined, the Proof of work is calculated to verify the transactions. This process of finding the proof of work can be parallelised with the help of multiprocessing. In our proposed method we create multiple processes that mine a single block parallelly until the Proof of work is calculated. This reduces the time of mining significantly and also makes efficient use of resources. As soon as the Proof of work is achieved all the processes are stopped and they move on to the next block.

In this project we create a pool of processes that concurrently operate to find the nonce value to verify the transaction. If the hash of the calculated nonce is lesser than the pre-defined hash value the block is successfully mined and it is added to the block. The calculation of required Hash value is largely random. The calculation of the nonce value whose hash value is close to the defined hash value can take time. By parallelising this process of nonce value calculation, we can greatly reduce time taken.

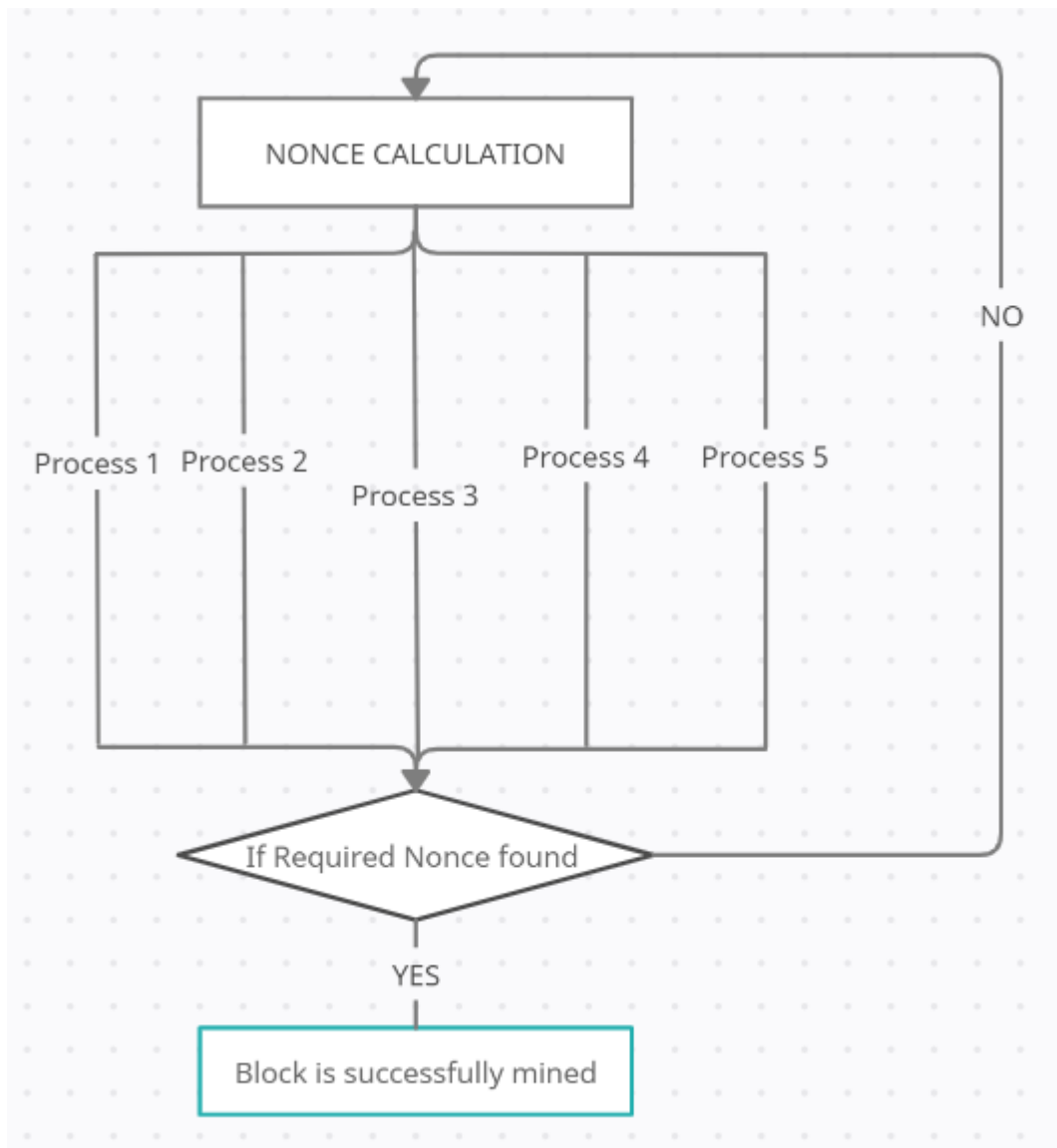


Fig 5.2 Nonce Value Calculation

In our project we use static scheduling of processes, iterations are divided into chunk sizes of N and each chunk is assigned to the process in a round robin fashion. Each process is given a range of nonce values to calculate the hash value. In our application the parallelisation is used to find a particular number whose hash value is equal to the predefined hash value. Hence there is no ceiling to our loop and it does not benefit from any other type of scheduling. Thus each process parallelly searches for the nonce value in the predefined range given to at compile time, and continues in a round robin fashion until the nonce value is found. Basically the nonce can lie in any range and there is no specific target/ceiling to the calculation required to find this. Thus static scheduling provides the optimum results for our application.

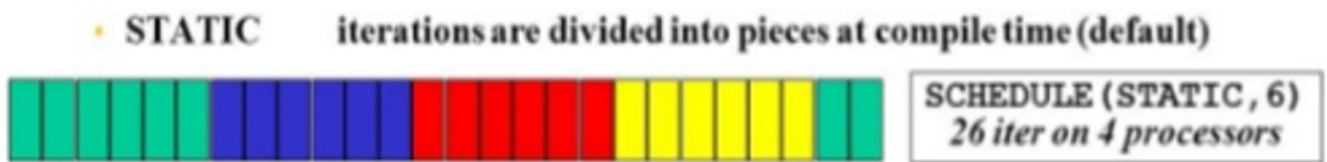


Fig 5.3 Static Scheduling

6. PROCEDURE:

Blockchain Based Voting [Sign Up](#)

Welcome back

Enter your email and password to sign in

Email or Username

Password

Enter 2FA OTP

SIGN IN

Don't have an account? [Sign up](#)

Fig 6.1 Landing Page of the application

Welcome back

Enter your email and password to sign in

Username

Required. 150 characters or fewer. Letters, digits and @/./+/-/_ only.

First name

Last name

Email

Required. Enter an existing email address.

Password

- Your password can't be too similar to your other personal information.
- Your password must contain at least 8 characters.
- Your password can't be a commonly used password.
- Your password can't be entirely numeric.

Password confirmation

Enter the same password as before, for verification.

SIGN UP

Already have an account? [Sign in](#)

Fig 6.2 Signup Page of the application

Welcome back

Blockchain-based e-voting simulation

Voting not Allowed

Please Update the following details to proceed:

- Aadhaar Card
- Voter ID
- Phone Number
- Address

PROCEED TO VERIFICATION


Fig 6.3 Main Page of the application

Election 1


UUID:

7aca30ed-e558-4114-8810-d60137a62bb9

Eligible candidates are shown below:



Alice



Bob

Your candidate choice:

Enter valid candidate ID



Your private key:

The matching public key must already be registered.

SUBMIT BALLOT

Fig 6.4 Candidate selection and Voting Page

Change activation

User:	<div>vsrivatsa25 ▼  </div>
Code:	<div>hGTNjZ1Eb5MKwylyLEbG</div>
Email:	<div>vsrivatsa25@gmail.com</div>
Totp id:	<div>HBQ4JB62TO30XZRGL2BUKFX55W2MODRS</div>
Public key ecc:	<div>ecdsa-sha2-nistp256 AAAAE2VjZHNhLXNoY</div>
Private key ecc:	<div>PRIVATE</div>
Aadhaar id:	<div>321</div>
Voter id:	<div>13</div>
Voting allowed:	<div></div>
Phone number:	<div>123</div>
Address:	<div>123</div>

Delete

Fig 6.5 Admin Page of the application - Approving eligible voters

Home > Simulation > Candidates > Add candidate

ACCOUNTS	
Activations	+ Add



AUTHENTICATION AND AUTHORIZATION	
Groups	+ Add
Users	+ Add

SIMULATION	
Candidates	+ Add
Elections	+ Add

SITES	
Sites	+ Add

Add candidate

Name:

Election:  

Party:

Symbol:

Fig 6.6 Administration and management of elections and candidates in the application

Welcome back

Blockchain-based e-voting simulation

Verify the voting results

START

GENERATE VOTES

SEE TRANSACTIONS (VOTES)

SEE BLOCKS



Fig 6.7 Admin Main Page

List of unconfirmed votes

Showing last 100 transactions (or all if <=100)

Voter ID	Vote	Timestamp
4ca99b06-ab8b-462d-98e5-1525b3316f98	1	163830848415060.3
03035667-ef23-4c34-8ea8-82b44cb3afab	1	163830848414931.6
84fd22e6-d2bf-4def-9bd5-2f350e04a94b	1	163830848414930.6
b8774817-234c-4302-a8e4-b69d616aee00	2	163830848414449.5
4562a620-d85c-4a70-9e4b-3c8efa44ab84	2	163830848414449.5
f2bf3358-03a5-4429-ae9e-951a027188c1	2	163830848414449.5
06909f90-b312-4d1e-a30f-f11ff17f44f3	2	163830848414449.5
570c0f04-ec9b-4a30-b1c4-a693a7652183	1	163830848414448.5
cff5e9c9-d08b-456f-8201-582fa612932a	1	163830848414448.5
2c3f5e3d-a366-44f0-a263-d2ca8e709a41	2	163830848414448.5
7f306a11-1aa3-415e-b995-2bb4a04ede4d	2	163830848414448.5
dc912896-68a7-4b22-b14e-0b4f39e5b5d1	2	163830848414225.78
6e920fab-f42b-4919-9468-6f3a8a56be5f	1	163830848414224.78
e465f5fb-1675-4cb6-8da1-f73dfb989952	2	163830848414224.78

Fig 6.8 Generating votes and Mining into Blocks

List of blocks

SEE ALL TRANSACTIONS

BACK TO HOMEPAGE

#	Previous Hash	Block Hash	Merkle Root Hash	Nonce	Timestamp
1	00000000000000000000000000000000...	0000001c757e336115c65c534ae5...	a98068d6c5c81d7fa8f5293589c67...	25343797	1638312119.931674
2	0000001c757e336115c65c534ae5...	00000890e094d4adba1003808fe9...	dc85bc00d165d978a00a80f98cb9c...	8047828	1638312138.209692
3	00000890e094d4adba1003808fe9...	000003fc8645238ca4f68bc81114...	71f1284b199698b8a77b9de33224c...	32558541	1638312275.214843

VERIFY VOTES

SYNC LOCAL BACKUP

SYNC FIREBASE

Fig 6.9 Mined Blocks and Details

VERIFY VOTES

SYNC LOCAL BACKUP

SYNC FIREBASE

- All transactions in blocks are intact.

Fig 6.10 Verifying Votes by recalculating Merkle Hash

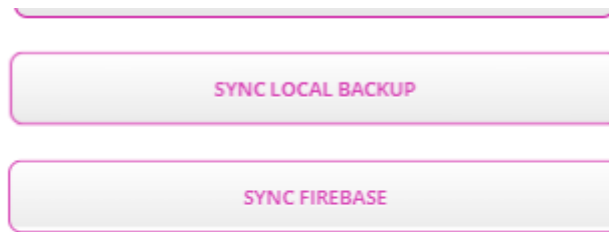
VERIFY VOTES

SYNC LOCAL BACKUP

SYNC FIREBASE

The following blocks have corrupted transactions: 1.
Vote ID: b15d1036-8327-4fd7-9809-cd1307dfa9d7 is tampered

Fig 6.11 Verifying Votes by recalculating Merkle Hash (Data is tampered)



- All blocks have been synced successfully.

Fig 6.12 Syncing Vote data either from Local Backup or Firebase

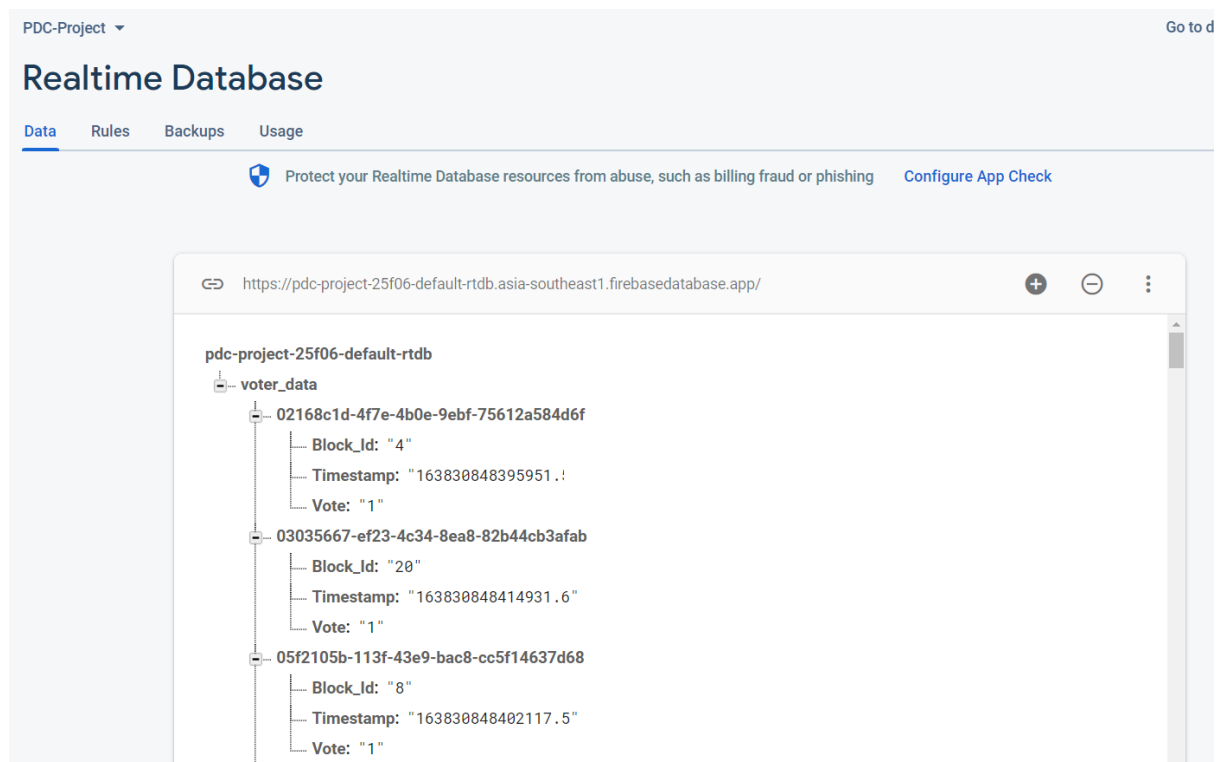


Fig 6.13 Voter Data in Firebase

List of confirmed votes - Election 1

SEE ALL BLOCKS

BACK TO HOMEPAGE

Candidate #1: Alice
Candidate #2: Bob

Candidate #1: 96 votes
Candidate #2: 104 votes

Showing page 1 of 2

NEXT

LAST »

#	Voter ID	Vote	Timestamp	Hash	Block
1	3cd824c8-8bff-45e8-84eb-056fcd0a9c03	2	163830848389550.6	192bd4393ad92fe044c...	GO TO BLOCK
2	09b60226-6d8d-4b5e-8d0a-7fbd299f22d0	1	163830848390350.38	5b7f539e68d211a2afb...	GO TO BLOCK
3	1ece5d48-0541-43c2-acc2-cf9da4867f46	2	163830848390350.38	3292c55653ea9cd89bc...	GO TO BLOCK
4	ca0c6d3d-8bea-4bcb-a893-6918929fe8e2	1	163830848390351.38	500e323983ed7721547...	GO TO BLOCK
5	7d9edf8c-565a-4e4f-ac56-91290d883ccb	2	163830848391057.4	e03bf32736ecd00943f...	GO TO BLOCK
6	cb273472-720e-4634-b0cf-b98ac5218683	2	163830848391057.4	779b3ff9686c740b7a4...	GO TO BLOCK

Fig 6.13 Voter Data (Confirmed Transactions in Block)

Block #1

[BACK TO BLOCKS LIST](#)

[BACK TO HOMEPAGE](#)

Previous

hash

00

Merkle

root

hash

a98068d6c5c81d7fa8f5293589c674dbdbd5ca8f49c952469eb7b976ca35b7a5

Block

hash

00000001c757e336115c65c534ae51948d79aa7086e5180baf77fa34284c3594

Nonce

25343797

Timestamp

1638312119.931674

Confirmations

3 confirmations

[NEXT BLOCK](#)

Transactions

Re-calculated merkle root hash:

a98068d6c5c81d7fa8f5293589c674dbdbd5ca8f49c952469eb7b976ca35b7a5

Transactions are not tampered

Showing page 1 of 1

#	Voter ID	Cand	Timestamp	Hash
1	3cd824c8-8bff-45e8-84eb-056fcd0a9c03	2	163830848389550.6	192bd4393ad92fe044cb15616192ab0...
2	1ece5d48-0541-43c2-acc2-cf9da4867f46	2	163830848390350.38	3292c55653ea9cd89bc26c267252c93...
3	09b60226-6d8d-4b5e-8d0a-7fbd299f22d0	1	163830848390350.38	5b7f539e68d211a2afb5fbab71f2c0f...
4	ca0c6d3d-8bea-4bcb-a893-6918929fe8e2	1	163830848390351.38	500e323983ed77215475aae5e080e18...
5	cb273472-720e-4634-b0cf-b98ac5218683	2	163830848391057.4	779b3ff9686c740b7a426785dca56cb...
6	7d9edf8c-565a-4e4f-ac56-91290d883ccb	2	163830848391057.4	e03bf32736ecd00943f0728d20a18c6...
7	88eb5953-322a-42c1-a39c-b0128b87d2d5	2	163830848391058.4	d05e5f2120cdc4aa3a743d37f17f8fd...
8	b15d1036-8327-4fd7-9809-cd1307dfa9d7	2	163830848391739.9	ca062451c6bae000f2e256a6e13dc55...
9	ad3503a1-fe19-4c3f-82b4-2f269f8d9b10	2	163830848391871.1	4aefa15973fec10afc5cfedcc72aa57...
10	860495c6-1677-48bd-8df3-53902eb02b94	1	163830848391872.1	0479e874be94e02d9de983d2b63744c...

Fig 6.14 Block Data (Mined Blocks with n Confirmations)

6.1.Mining Code (Python3) using Multiprocessing

```
from multiprocessing import Pool

from multiprocessing.context import Process

from django.conf import settings

from Crypto.Hash import SHA3_256

import datetime, time, math

import django

from django.db import transaction

from app.settings import DATABASES, INSTALLED_APPS

import os

os.environ.setdefault(

    "DJANGO_SETTINGS_MODULE",

    "app.settings"

)

django.setup()

from simulation.models import Vote, Block

from simulation.merkle.merkle_tool import MerkleTools

def Mine(i, prev_hash, merkle_h, nonce, timestamp):

    puzzle, pcount = settings.PUZZLE, settings.PLENGTH

    start_nonce = nonce

    while True:

        enc = ("{}{}{}{}{}").format(prev_hash, merkle_h, nonce,

timestamp)).encode('utf-8')
```

```

        h = SHA3_256.new(enc).hexdigest()

        if h[:pcount] == puzzle:

            block = Block(id=i, prev_h=prev_hash, merkle_h=merkle_h,
h=h, nonce=nonce, timestamp=timestamp)

            block.save()

            break

        nonce += 1

        if nonce - start_nonce == 10000:

            nonce += 40000

    return

if __name__ == '__main__':

    """Seal the transactions generated previously."""

    # Puzzle requirement: '0' * (n leading zeros)

    puzzle, pcount = settings.PUZZLE, settings.PLENGTH

    # Seal transactions into blocks, delete blocks from previous
simulations first

    deleted_old_blocks = Block.objects.all().delete()

    time_start = time.time()

    number_of_blocks =
math.ceil(Vote.objects.all().count()/settings.N_TX_PER_BLOCK)

    transaction.set_autocommit(False, using=None)

```

```

for i in range(1, number_of_blocks + 1):

    block_transactions =
Vote.objects.filter(block_id=i).order_by('timestamp')

    if(i == 1):

        prev_hash = '0' * 64

    else:

        prev_hash = Block.objects.filter(id = i - 1)[0].h

    root = MerkleTools()

    root.add_leaf([str(tx) for tx in block_transactions], True)

    root.make_tree()

    merkle_h = root.get_merkle_root()

    # Try to seal the block and generate valid hash

    nonce = 0

    timestamp = round(datetime.datetime.now().timestamp(), 12)

    p1 = Process(target = Mine, args=(i, prev_hash, merkle_h,
nonce, timestamp,))

    p2 = Process(target = Mine, args=(i, prev_hash, merkle_h, nonce
+ 10000, timestamp,))

    p3 = Process(target = Mine, args=(i, prev_hash, merkle_h, nonce
+ 20000, timestamp,))

    p4 = Process(target = Mine, args=(i, prev_hash, merkle_h, nonce
+ 30000, timestamp,))

    p1.start()

    p2.start()

    p3.start()

```

```

p4.start()

while(True):

    b = Block.objects.filter(id = i)

    #Kill all child processes and proceed to next block once a
nonce is found

    if(len(b)):

        p1.kill()

        p2.kill()

        p3.kill()

        p4.kill()

        break

    # Create the block

    print('\nBlock {} is mined\n'.format(i))

    transaction.commit()

    transaction.set_autocommit(True, using=None)

    time_end = time.time()

    print('\nSuccessfully      created      {}
blocks.\n'.format(number_of_blocks))

    print('\nFinished in {} seconds.\n'.format(time_end - time_start))

```

7.RESULTS AND DISCUSSION:

7.1.Serial mining of Blocks:

Name	Status	64% CPU	69% Memory
Visual Studio Code (21)		56.8%	866.5 MB
Python		23.5%	27.4 MB
Python		23.4%	33.1 MB
Visual Studio Code		5.7%	121.6 MB

Fig 7.1 CPU and Memory consumption (Serial mining)

```
Block 19 is mined

Block 20 is mined

Successfully created 20 blocks.

Finished in 370.3185155391693 seconds.
```

Fig 7.2 Mining Status and Time complexity (Serial)

The above pictures demonstrate the serial mining of the blocks. The blockchain difficulty is set to 5. There are 2 processes in the task manager screenshot, one process is executing the main application, while the other is mining the block. Thus only 1 process is mining the block. In serial mining a block is mined by a single process. Thus it is not very efficient and its performance can be improved by parallelizing the mining process.

7.2.Parallel mining of Blocks:

Name	Status	CPU	Memory
Visual Studio Code (25)		90.1%	1,012.2 MB
Python		14.6%	33.2 MB
Python		14.3%	27.4 MB
Python		14.2%	27.4 MB
Python		14.1%	27.3 MB
Python		14.1%	27.3 MB
Python		14.1%	27.4 MB

Fig 7.3 CPU and Memory consumption (Parallel mining)

```
Block 19 is mined

Block 20 is mined

Successfully created 20 blocks.

Finished in 249.26334381103516 seconds.
```

Fig 7.4 Mining status and Time Complexity (Parallel)

The above images depict the parallel execution of the mining procedure. In the Parallel execution we have created 4 processes to simultaneously mine the block. These 4 processes follow static scheduling and are assigned tasks in a round robin manner. It can be clearly seen that the parallel execution is faster than the serial execution. For this example we have mined 20 blocks.

8.ANALYSIS

8.1.Comparison:

8.1.1.Pre conditions:

Number of Blocks: 20

Number of transactions per block: 10

Sl no	Blockchain Difficulty	Mining Time (Serial) (in sec)	Mining Time (Parallel with 2 processes) (in sec)	Mining Time (Parallel with 4 processes) (in sec)
1	2	14.02	17.19	18.89
2	3	14.68	17.89	18.92
3	4	40.67	31.32	30.11
4	5	370.32	322.72	249.26

Table 8.1 Comparison of Serial and Parallel Execution Times

Performance Chart

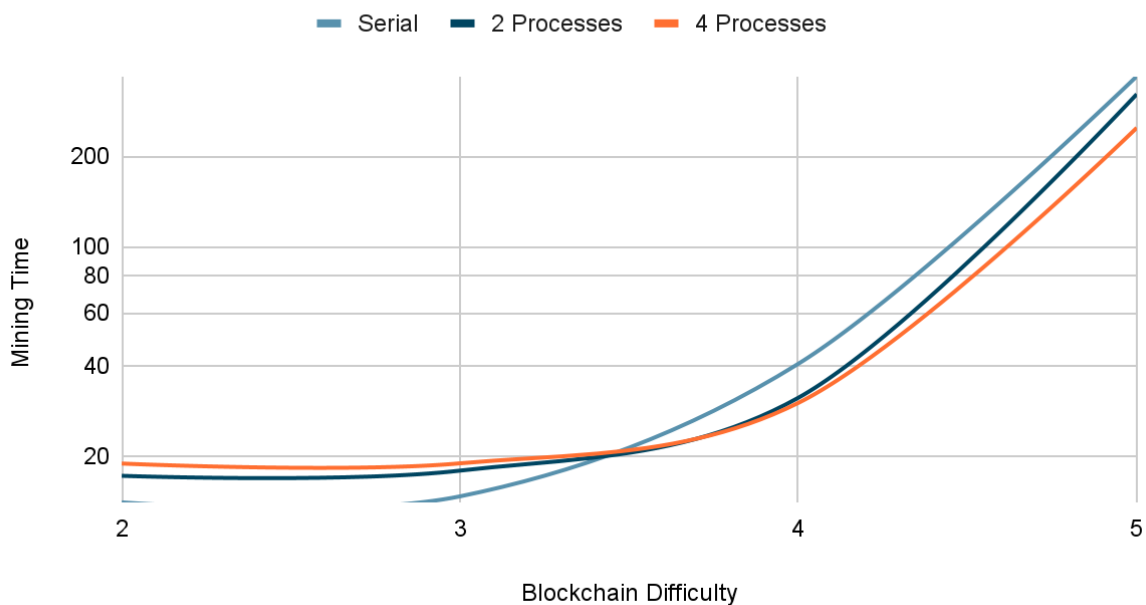


Fig 8.1 Performance Chart (Blockchain difficulty vs Mining Time for n processes)

8.2. Analysis:

From this table we can clearly comprehend the difference between serial and parallel mining.

Inference for SI No. 1,2: In the initial stages when the difficulty of the blockchain is less, serial is faster than parallel. This is due to the overhead that is involved in the creation of multiple processes. Thus when the difficulty is less the serial execution outperforms parallel execution. It can also be noticed that the more the number of processes the more time it takes at this level of difficulty.

Inference for SI No. 3: When the difficulty is increased we can gradually see that the parallel execution outperforms the serial execution. When the difficulty is equal to 4 we can slowly see a shift in performance. It can also be noticed that there is no significant difference between 2 and 4 processes. Serial execution takes nearly 10 seconds more to compute than 2 and 4 process parallel execution. This is a 25% increase in efficiency.

Inference for SI No. 4: As the difficulty is further increased, it can be seen that there is vast difference between serial and parallel mining. 4 processes far outperforms both serial and 2 processes. There is a 32% improvement in the efficiency compared to the serial execution, and a 22% increase in efficiency compared to 2 process execution. 2 processes also perform better than serial execution by 13%. Thus it can be clearly noticed that parallel execution outperforms serial execution when the difficulty of the mining is increased. Thus in real world applications where the difficulty is much more than the sample taken by us, the improvement seen can be huge.

8.3. Statistical analysis:

Efficiency of 4 process Parallel execution:

SI no	Blockchain Difficulty	Improvement compared to Serial	Improvement compared to 2 process
1	2	-34%	-9.8%
2	3	-28.8%	-5.75%

3	4	+25.96%	+3.8%
4	5	+32.69%	+22.76%

Table 8.2 Efficiency of 4 process Parallel execution

Performance Improvement

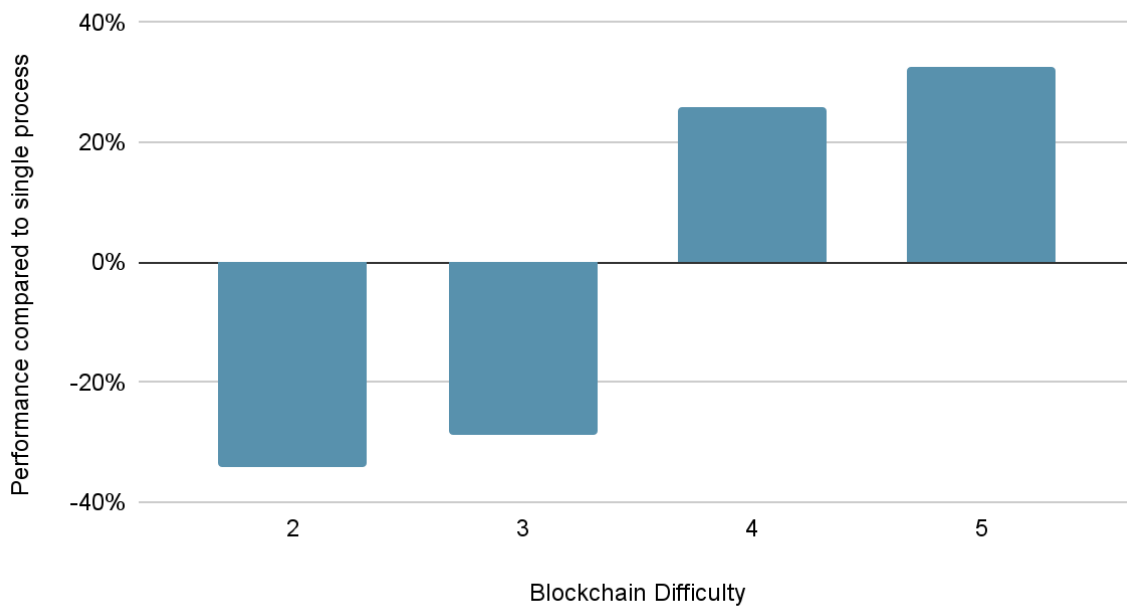


Fig 8.2 Performance Improvement using 4 processes vs Blockchain difficulty

Efficiency of 2 process Parallel execution:

Sl no	Blockchain Difficulty	Improvement compared to Serial	Improvement compared to 4 process
1	2	-22.6%	+8.9%
2	3	-21.86%	+5.43%
3	4	+22.98%	-4.0%
4	5	+12.69%	-29.47%

Table 8.3 Efficiency of 2 process Parallel execution:

Performance Improvement

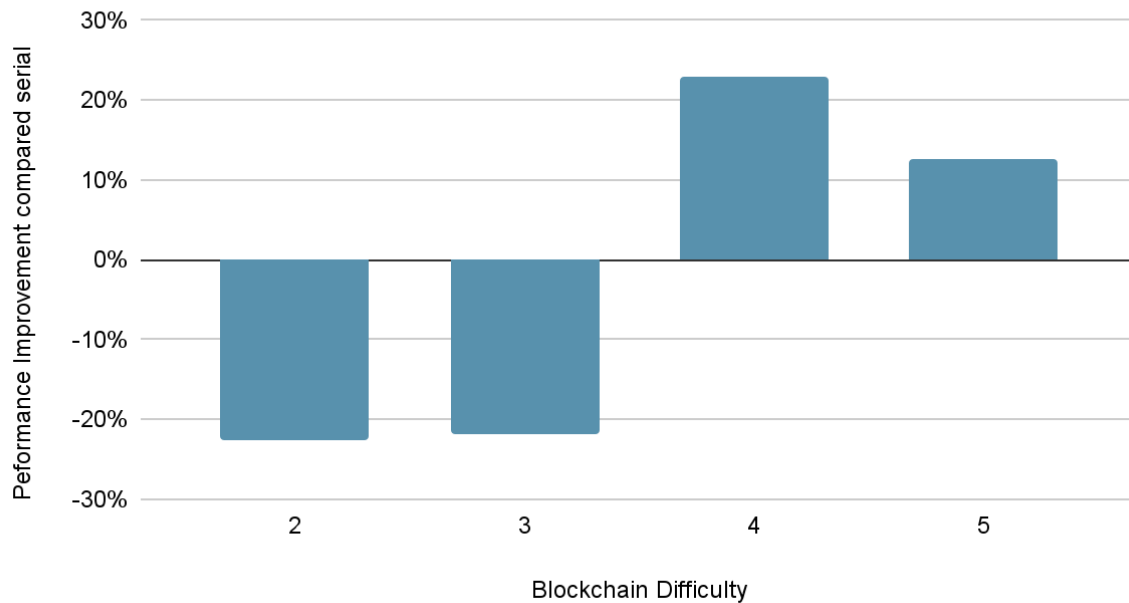


Fig 8.3 Performance Improvement using 2 Processes vs Blockchain difficulty

Thus it can be clearly seen from this statistical analysis that 4 processes perform drastically better than serial execution as the difficulty of the mining increases. Even 2 processes also perform better than serial execution but it can be seen that the increase in efficiency is better when 4 processes are used. As the difficulty is increased the overhead in creation of the processes is insignificant in comparison with overall work done. Thus in order to maximise the benefits of parallelism we must select a difficulty where the overhead of creation of the process does not exceed the task itself. For lesser difficulty serial execution performs better as there is no overhead in the creation of the threads. Thus depending on the scenario one must choose wisely between serial and parallel computing. For smaller tasks serial execution is faster, and as the difficulty increases parallel execution increases in efficiency.

9.CONCLUSION

Thus in our project we have delivered a safe and secure e-voting system that exploits the benefits of blockchain, parallel and distributed computing. By using Distributed computing we have increased the security of the application making it resistant to tampering. We have also shown the improvement in performance of the mining while using multiprocessing.

There is a clear increase in efficiency when we mine the blocks in parallel. Thus we have created an efficient and secure Mining system that uses parallel execution. Overall we deliver a simple e-voting system that employs distributed systems and parallelism techniques to improve security and performance.

10.REFERENCES

- [1] Milind Tote, Ankit Kumar, Mayank Mahankal, Siddhesh Khadse and Vrushabh Uprikar. "PARALLEL MINING IN BLOCKCHAIN FOR BITCOIN USING GAME THEORY" 2019 JETIR May 2019, Volume 6, Issue 5
- [2] Hazari, Shihab Shahriar, and Qusay H. Mahmoud. "A parallel proof of work to improve transaction speed and scalability in blockchain systems." 2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC). IEEE, 2019.
- [3] Shahriar Hazari, Shihab, and Qusay H. Mahmoud. "Improving Transaction Speed and Scalability of Blockchain Systems via Parallel Proof of Work." Future Internet 12.8 (2020): 125.
- [4] Hazari, Shihab Shahriar. Design and development of a parallel Proof of Work for permissionless blockchain systems. Diss. 2019.
- [5] Martino, Will, Monica Quaintance, and Stuart Popejoy. "Chainweb: A proof-of-work parallel-chain architecture for massive throughput." Chainweb Whitepaper 19 (2018).
- [6] Liu, Shaowei, Jing Wu, and Chengnian Long. "Iot meets blockchain: parallel distributed architecture for data storage and sharing." 2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData). IEEE, 2018.
- [7] Baheti, Shrey, et al. "DiPETrans: A Framework for Distributed Parallel Execution of Transactions of Blocks in Blockchain." arXiv preprint arXiv:1906.11721 (2019).

- [8] Tsai, Chun-Wei, et al. "An efficient parallel machine learning-based blockchain framework." *ICT Express* 7.3 (2021): 300-307.
- [9] Xu, Cheng, et al. "SlimChain: scaling blockchain transactions through off-chain storage and parallel processing." *Proceedings of the VLDB Endowment* 14.11 (2021): 2314-2326.
- [10] Chishti, Mohd Sameen, and Amit Banerjee. "Increasing TPS rate of state-based blockchains by parallel mining." *Internet Technology Letters* 4.2 (2021): e220.