Improving Transaction Speed and Scalability in Blockchain Systems

Joshua Aaron DeNio
Department of Computer Science
North Dakota State University
Fargo, North Dakota
joshua.denio@ndus.edu

Simone A. Ludwig
Department of Computer Science
North Dakota State University
Fargo, North Dakota, USA
simone.ludwig@ndsu.edu

***Abstract*—This paper presents a parallel mining architecture model intended to be used in blockchain systems to improve transaction speed and network scalability while maintaining decentralization. Typical blockchain validation times are significantly slower than traditional digital transaction systems. The model proposed is intended to allow devices with limited computational power to make meaningful contributions to the blockchain system by introducing parallel proof of work, managed by automated manager nodes. This will allow blockchain systems to be integrated into cloud environments and the internet of things. The proposed model is also intended to address and reduce power consumption problems current blockchain systems face, by allowing the network to validate transactions without the need of high-powered specialty mining machines. Automation and virtualization of network nodes is intended to utilize hardware already online to preform parallel proof of work together in contrast to nodes all competing against each other and ultimately wasting electrical power.**

***Index Terms*—Blockchain, internet of things, parallelism, power consumption, energy efficiency, parallel proof of work, manager automation**

I. Introduction

Blockchain technology has gained popularity in recent years causing a large influx of new interests in the potential application of blockchain technology into real world systems. Blockchain systems serve as the foundational technology behind cryptocurrencies and smart contract systems. In recent years blockchain systems have gained international attention particularly in the field of cryptocurrencies. This renewed interest has caused stresses on the current mining pools and in the cases where mining pools cannot keep up with the demand has even temporarily caused outages on some block chains [1], [2]. This paper intends to introduce a change in the underlying architecture of blockchain mining that will be able to support the vast expansion and scalability while not wasting valuable resources. Current mining approaches are very resource intensive and compete with one another over mining the same blocks. In the current model only one miner or mining pool can actually solve a proof of work on a block at a given time, and as a result the remaining miners and mining pools have wasted their resources trying to mine the block [3]. This waste of resources causes the cost of contributing to mining to go up significantly as to remain competitive, the miners must either join a large mining pool or devote intensive resources comparable to a mining pool in order to be capable of completing a proof of work before a mining pool can complete the same block [4]. Therefore, research is needed to find a more efficient way to process blockchain transactions that will not result in the waste of electricity.

II. Related Work

Bitcoin-NG, introduced in 2016, presented the concept of decoupling Bitcoins blockchain task into two planes: a leader selection and exchange serialization plane [5]. Bitcoin-NG also partitions time into epochs where each epoch has a solitary leader. In our application of managers, the leaders will form a team where one is active, and the remaining leaders are available to take over if something happens to the active manager. The solitary leader used in Bitcoin-NG introduced a single point of failure, which our solution addresses by adding redundancy to the role with the implementation of a team of backup managers in proportion to the network size. These back up managers will all be in synch with the active manager, and when the active manager fails the group will promote a new manager to active status randomly, then update the miners of the new active manager. Bitcoin-NG uses two types of blocks one being a Key Block and the other being a Microblock [5].

Another related approach is that presented by Xavier Boyen et al. also in 2016 [6]. In their approach each transaction is connected to two or more verified transactions and miners verify new transactions in a parallel network. The network used consists of a graph structure like the network structure used in Bitcoin, Tangle, and IOTA [7]. The Boyen model also utilized proof of stake rather than proof of work to validate blocks and append them to the blockchain. Boyen's approach has done away with the traditional blockchain and implements a lean graph of transactions making verification times much faster.

Hazari and Mahmoud presented a model in 2020 which the proposed approach is based on [3]. Their presented model makes use of a single manager node and a network of miner nodes. The manager is selected based on which node completed the last block making it easy to predict and target the manager. The network is a peer-to-peer network, which also poses potential vulnerabilities to attack as a single node loss could cause the network to temporarily loose communication

capabilities as nodes reconnect. The aim of this paper is to address the possible weaknesses of the Hazari-Mahmoud model and make a derived model that is more robust and resistant to node failures while maintaining the benefits of the Hazarti-Mahmoud model of distributed parallel proof of work. The proposed approach differs from the Hazarti-Mahmoud model in that it introduces multiple manager roles to account for redundancy and introduce more recoverability. Also, the proposed approach introduces a star like hybridization of the ring peer-to-peer network topology used in the Hazarti-Mahmoud model. This interconnects each node more tightly with other nodes in the network and allows for better data retention and less local forking of transactions entering the mempool as more nodes will be present locally to verify the incoming transactions.

The managers used in the proposed solution are based on the idea of coordinator selection, which was first implemented by Gerard Lelann in 1977 [8]. The proposed solution greatly improves their role and increases redundancy thus, reducing any potential for a point of failure. This role of process coordinator is a crucial part of improving the quality and performance of a distributed system. Dework et al. introduced a consensus protocol using coordinator election for a partially synchronous processor in 1988 [9]. In their work, the coordinator distributes work to peers in proportion to the number of peers within the network. When the work is completed, a final decision is made using the consensus protocol to validate the work.

III. Proposed Approach

In this paper, we present a method to improve the transaction speed and scalability by implementing parallel processing and validation of blocks across a decentralized network of peers. In the method proposed most of the nodes will perform work to validate the same block and the remaining nodes will be the active manager and backup managers, respectively. Miners will receive data from the active manager and the backup managers will receive updates from the active manager and take over if the active manager becomes unavailable. We will develop a consensus mechanism to ensure that managers are randomly selected, and nothing can be done to force a specific node to become a manager.

1) Managers: Managers will be implemented in two forms: an active manager and a team of support managers we will call the management team. The active manager will distribute transaction data from the mempool to the miners and will issue each miner a set of nonce values. The active manager will ensure that no two miners receive the same nonce values for a given block. The manager is also responsible for creating the transaction hash that miners are to solve, along with the nonce value set they will apply to attempt to solve the hash.

When the nonce values are depleted, the active Manager will generate more and disperse them as needed. Another roll of the active Manager is to synch with the management team and keep them up to date with its activity as well as nonce ranges and data dispersed.

The role of the Support Managers is to form a team and if needed replace the active Manager in the event of a failure. The replacement should be randomly chosen to prevent a malicious user from gaining control over a manager and tampering with the network. The Support team will periodically elect new managers at random so long as they meet the system specifications required to fill the role. The management team will be a parallel network nested within the main network and should be continuously in contact with each other to ensure they are all in synch and contain the same information. If a single manager has differing information the others will expel the corrupted manager from the team and elect a replacement.

The management team will replace the active manager at given time intervals we will call epochs like those used in Bitcoin-NG [5]. This will be done to give rest to hardware components and ensure that the management role does not stay active on a specific hardware device for an extended amount of time. This is directed at reducing the ability of users from tampering with the active manager as they should have no way of knowing what node will become the active manager or for how long it will be active. The management team will monitor its members and elect new Support managers as needed and rotate them in and out of service depending upon the needs of the network. The level of redundancy required will need to dynamically scale with the size of the network with a percentage of the network being support managers to ensure that there will never be a failure that will remove them all at once. If possible, they will be geographically chosen to ensure that they are dispersed enough to evade failure even in the case of a continental power outage. The managers status will define its role and the support managers will be tasked with monitoring what manager is currently active and in the case that an incursion occurs, and the active manger is changed without the consensus of the group, the offending node will be removed from the network.

In the case that the manager is not an active manager it will be tasked with monitoring the activity status of the active manager and verifying security, optionally it could temporarily act as a worker if everything is up to date. If the active manager is not active the group will elect a new active manager based on the capabilities of the available managers in a semi random selection putting preference to those with the highest computational capability. It is important that it be difficult to predict what manager will step up to fill the role to reduce the likelihood an attacker is able to target the next active manager in advance. In the case of the node being the Active Manager, it will be tasked with distributing data and nonce values and reporting to the management team to verify that it is active and inform the team of the current data and nonce values in operation.

2) Miners: Miners in parallel mining will initially send a request to the management team, which will be accepted by the active manager. The Active Manager will then send a block of data and a set of nonce values to the miner. No two miners should be doing the same work at the same time. The miner will attempt to solve the puzzle and generate a suitable

hash value using the set of nonce values received. Once a suitable solution is found the successful miner will broadcast the completed block and the other miners will check to verify whether the solution is valid. If the solution is acceptable, they will update the manager and they will receive the next set of data to begin working on the next block. All miners will receive the same data set but each one will work on a separate set of nonce values. Once the set is depleted and if no solution was found a new request will be sent to the Managers and the miners will await a new set of nonce values.

At this point it will contact the management team and receive an update to its blockchain data. The miner will then request data to work as well as data relating to management team rolls used to validate managers as group consensus will be used to ensure that false managers cannot be injected into the management pool and only those elected are accepted as managers. The miner will then be given a roll by the management team. This role may be as a miner, or a supporting manager based on the needs of the network. In the case that the node is elected as a manager it will accept the role. Otherwise, the node will then receive data and or nonce values and will move on to mine the hash. If it finds a solution it will broadcast the solution to be validated by the other nodes. If another node broadcasts a solution the solution will be checked and if valid added to the blockchain and if invalid measures will be taken to address the issue. Next, a security and validation phase will be executed to ensure that only authorized nodes are serving as managers and in the case that a node is not abiding by its designated role or otherwise posing a threat to the network actions will be taken to remove the threat from the network.

A. Network Communication and Security

In traditional blockchain systems nodes are connected to one another via intermediate nodes. In our proposed approach to parallel mining, the nodes will still be connected via a peer-to-peer network, but they will also be connected to a team of managers that are also interconnected and in sync with each other. Each miner communicates to all managers and all managers communicate with all miners and other managers. Thus, each node is directly communicating with the managers and its peers, and the managers are communicating with each other collectively. Miners communicate with other miners using standard peer-to-peer ring topology communication but communicate with managers using a modified star topology resulting in a hybrid topology. This is to ensure redundancy and prevent a single point of failure. All communication should be verified from the other sources and rejected if the data is invalid. Consensus is required to validate any block and managers must agree upon the election of new managers. This consensus will ensure decentralization and prevent tampering with the network. This modified star topology differs from the ring topology used in existing approaches and introduces more resiliency while maintaining concurrency in communication.

1) Points Of Failure: When compared to existing techniques such as those presented by [5] and [6] the additional backup managers remove the points of failure present in both implementations as the before mentioned approaches both utilize a single manager that when targeted or removed have a noticeable impact on the network's functionality and operational capability. In the proposed approach the vulnerabilities presented by only having a single manager is addressed by providing backup managers that can step in and fill the role when needed. This will result in very little impact of the loss of a single manager. Also, the network itself is more robust than that used by [5] and [6] as a normal peer-to-peer network will suffer when nodes are removed, and interruptions can occur. In contrast the proposed approach will maintain more communication links and even with the removal of nodes no interruption can occur as the remaining links will serve to maintain communication across the network. Managers will also be more difficult to target as the roles will be changed for each block and there will not be an obvious way to predict what node will become the next Active Manager. In contrast to the approach presented by [6] where the manager is assigned based on the node that has solved the last block making targeting of the active manager more straightforward. Managers in the proposed approach will be elected semi randomly with a preference to those nodes with greater computational capabilities and even when a node becomes the active manager the specific address of the active manager is not directly logged in the blockchain for users to view. The active manager will not be directly communicated to instead the miners will send their requests to the manager pool and the communications will be accepted by the manager acting as the Active Manager.

2) Denial Of Service Attacks (DDOS): Research into the security vulnerabilities of blockchain technology indicates that DDOS attacks have greater effect on blockchains than on other more traditional transaction networks [10]. In the event of a Denial of service attack the Active Manager will be overwhelmed and the management team will need to elect another active manager. At this point the management group should trigger a defensive feature to detect the reason that the active manager has failed. An effective detection mechanism should be developed and deployed on the manager nodes to detect the initiation and execution of a DDOS attack and impose countermeasures to mitigate the effectiveness of the attack. Once detected the management team or active manager can blacklist or expel an offending node from the network if requests from the node seem to indicate a possible DDOS attack. Each node serving as a supporting manager will be periodically monitoring the active manager and ensuring that it is responding to requests effectively. This monitoring process can be used to detect a DDOS attack as the number of requests will be significantly higher than normal and should have other characteristics that separate these requests from valid requests that can be used to identify an attack.

3) Malicious Managers and Detection Methods: Malicious managers are a possible threat that will be addressed by the management team by periodically verifying that only managers that have been elected are serving as a manager. If a node is detected that is acting as a manager, but its election is not traceable by the Support Manager pool it will be expelled

from the network and if needed any invalid data added to the blockchain by the expelled manager will be rolled back to prevent corruption of the system by malicious managers. Only nodes that are dynamically elected may serve their respective roles. Logs will be maintained in encrypted form to allow the managers to trace all the roles served and elected for each block.

4) Validation: Validation will follow existing methods of peer consensus used in blockchains such as Bitcoin. Forks that deviate from the accepted branch of the blockchain will be discarded in favor of the longer accepted valid blockchain preventing malicious nodes from attempting to insert transactions into the blockchain. This does not differ from the methods used currently in proof of work blockchains. Double spending verification will be handled like that of Bitcoin with no real difference in the block validation process, and the number of blocks used to validate transactions will not differ from validation methods employed by the Bitcoin network.

B. Genesis Block

The initial block will be created by the miners if the blockchain is empty. This first block will contain no transactions. Initially all managers will be elected at random using a test protocol to determine that they possess the required computational capabilities to fulfill the role. The node that completes the first block will become the active manager for the next block. Each new block will start a new period that we will call an epoch, with a new active manager managing each new epoch.

C. Manager Election

Support managers will be elected in an unpredictable fashion with consideration to their computational contribution to the network. Nodes that provide a greater computational contribution will have increased odds of being elected to the management team. This will ensure that each manager elected has the capability of fulfilling its role. Manager candidates can be selected based on their computational capabilities and added to an array. Managers can then be selected from the group of possibilities using random number generation from the management team to select the index of the newly elected manager. This should result in managers being capable of handling the loads required to fulfill their roles and ensure that the incoming managers cannot be predicted by outside observers. If for some reason the active manager is unresponsive or slows to an unsatisfactory response time the management team will elect a replacement to fill the active manager role. In cases where the system has scaled to the point that a single node can no longer support the tasks of the managers, nodes can be combined to divide the tasks and fill the role as if they were a single node. This can be done using virtualization to increase the computational capabilities of nodes within the network [11]. As managers require more computational capability than miners the management team will need to be capable of measuring and dynamically scaling to meet the networks requirements.

This application of rotating elected managers ensures that no single node can gain excessive influence over the network thereby maintaining the decentralized nature of the system.

D. Algorithms

In the proposed parallel mining system, each miner will have an equal opportunity to become a manager. Each miner will be compensated for their contribution to the network at the end of each epoch. This role of compensation will be monitored by the group of managers and only by full consensus will they distribute pay to the workers. Obviously, it will be beneficial to have the most powerful hardware involved in the management role. The miners who invest more resources into producing the greatest mining power will have higher likelihood of being elected to the management team. And as a reward will gain more revenue based on their network contribution. As the more powerful machines in the network solve more nonce values, they will in turn receive larger compensation than those who work less. Algorithm 1 and 2 show the basic algorithm chosen for block solving and block validation.

Algorithm 1 Block Solving Algorithm

```
Step 1: Initialization
Step 2: Record creation
Step 3: Solve puzzle
    for i = nonce_list[0] to nonce_list.length() − 1 do
        if blockchain.length() > block_index then
            Block was solved call validate block function
        end if
        Solution = Sha − 256(record + nonce_list[i])
        if Solution is valid then
            Broadcast the solution
        end if
        if Solution is invalid and block is not solved then
            Request new nonce_list and restart
        end if
    end for
```

Algorithm 1 shows the details of the algorithm used to solve blocks. This algorithm is written using pseudocode and is intended to convey the basic logic involved in the block solving process. Nonce values and data will be provided by the Active Manager and backed up to the supporting managers to preserve redundancy and recoverability in the case of a node failure.

E. Workload Size and Data Distribution

Workload size will be determined by the capabilities of the miner as well as its availability. The Active manager will distribute work in accordance with a given node's computational ability thus maximizing the effectiveness of the network and reducing the instances where a node receives more work than it can process in a reasonable period of time.

The active manager is responsible for transmitting data such as the transaction hash, as well as nonce values to miners. If

Algorithm 2 Block Validation Algorithm

```
if previous_block_index! = new_block_index then
    Return False
else if previous_block_hash! = new_block_hash then
    Return False
else if new_block.hash() > target then
    Return False
else
    Return True
end if
```

there are n miners active on the network the manager will distribute n distinct nonce sets. It is important that no two distributed nonce sets share any values. When any node has depleted its set of nonce values it will send a request for a new set to the management team and the active manager will distribute a set dependent upon the node's capacity. High throughput nodes will receive larger sets vs smaller nodes will receive smaller workloads. New miners joining the network will receive data (the hash) and a set of nonce values to work with.

F. Transaction Speed

The goal of implementing parallel mining is to dramatically improve the transaction and verification times as well as increase the overall scalability of the network. Using a parallel mining approach, miners can quickly reach consensus and verify transactions efficiently. This increase in efficiency and reduction in transaction times will improve the user experience and create a transactional environment that users can come to rely upon. In contrast to solo mining parallel mining provides a significant improvement in transaction speed and throughput. Examples of this improvement is seen in the implementation of mining pools as well as previous work on parallel mining [3], [5]. Later in this paper we will discuss some data comparing solo mining to parallel mining for more details.

G. Fees

Transaction fees are used on many blockchains, the most popular of which is the Ethereum blockchain and its use of gas to pay transaction fees. Fees can easily be incorporated into parallel mining, but it is important to note that fees should be scaled in relation to the transaction size. Ethereum has some drawbacks when processing small transactions as the fees can cost more than the transaction itself in some cases [12]. This is counterproductive to encouraging users to utilize a network so ensuring that fees remain affordable and proportional to the transaction size is crucial to maintain usability. A reasonable service fee would be somewhere between 1% and 2% of the overall transaction but there must also be an upper and lower bound to ensure that no customer is charged an unreasonable transaction fee. Say for example if a transaction is an exchange of $100,000,000 or 1 cent a percentage service fee will not make the transaction viable on the network and customers will go elsewhere to process their transactions. Thus, keeping transactions affordable is a key element to the overall success of the network.

H. Scalability

Parallel mining of proof of work is quite scalable as the more users using the network the more miners become a part of the network. Users can also opt to connect small computational devices to a single account making use of the internet of things to contribute to the blockchain and be rewarded based on the contribution utilized. If the network grows in size to the point that a single hardware node can no longer support service as the Active Manager, the Support manager pool may elect to elect multiple nodes as the Active manager and in this case the nodes will work in parallel to perform the work required of the Active Manager role. This creation of a virtual or composite node should extend the scalability of the system indefinitely as the more nodes are added to the network the more powerful and capable the management team will become. The algorithms required for this kind of dynamic scaling can be developed in future work.

I. Parallel Mining compared to Pool Mining

While pool mining is slightly more resource efficient as only one node serves as a manager it is vulnerable to attacks and introduces centralization as the pool manager is not a free part of the network like those used in parallel mining but controlled by the mining pool administrators. As such mining pools do away with the concept of decentralization. In contrast parallel mining maintains decentralization while making use of a distributed workforce to faster solve proof of work.

1) How Proposed Solution Addresses Tragedy Of Commons Problem: The tragedy of commons will be addressed by making even small computational devices capable of meaningful contribution to the network. This will ensure that there will always be a surplus of miners and so long as all miners that are being utilized receive a fair share of the reward based on the amount of work performed there will be a reason to keep miners available to the network to provide services and collect the rewards of their contribution.

2) Integration With Cloud: Cloud resources work well with the proposed parallel approach as nodes hosted on the cloud can contribute their resources when the resources are not being utilized and by setting a small workload size, they can request packets of nonce values that can be small enough for them to contribute while still being able to quickly transition to other work as required.

3) Integration With Internet Of Things: One of the most novel characteristics of this parallel decentralized blockchain network is its ability to integrate with the internet of things; allowing small devices to contribute their computational power to processing transactions on the network. This could dramatically change the way cryptocurrencies are used as transaction validation times could be reduced and transactions would be processed quite quickly.

4) Node Virtualization: Smaller nodes can be combined dynamically by the management team to produce virtual nodes that fill the requirements of the system by allotting resources from smaller nodes to work together as a single node. This will in theory allow computational devices with limited abilities to contribute to the network by joining together as a single virtual node [11]. Most likely this will provide the most utility when the network has reached a large size and single management nodes can no longer manage the large number of nodes in the network. By increasing the capability of the managers by merging node resources we can essentially create super computers using the collective capabilities of smaller nodes serving as a single node. This will allow the network to infinity scale as new nodes are added to the system.

J. Rewards

Block rewards will be distributed by the management team's consensus based on the miner's contribution to the block with payments being distributed each epoch. The collective of manager nodes will distribute the block reward based on the work performed by each miner so long as they made a meaningful contribution to the processing and validation of the block. Transaction fees will also be distributed based on the level of involvement a particular miner contributed to possessing the block. The most basic approach would be to sum up the transaction fees and add them to the block reward for distribution. Both the fees and block reward will benefit from being dynamically adjustable to ensure stability of the network and prevent inflation or scarcity issues that may arise in the future to unforeseen events.

K. System Events

1) Multiple Nodes Solving Hash Simultaneously: If multiple nodes solve the hash at the same time the first solution received by the active manager will be considered the first to be completed and be moved on to the block validation process. There is also the option of following traditional blockchain approaches and let forks occur and prune them after a set number of blocks choosing the longest chain as the valid path. This approach has been deemed effective and used in most mainstream blockchains in use at the time of this printing.

2) Nodes Entering Network: New nodes entering the network will make a request to the management team and will receive an updated dataset to ensure that the new node is concurrent with the current state of the blockchain. If there are no managers active the new node will receive its data from the other nodes on the network and the process of electing managers will be initialized by the collective.

3) Nodes Leaving Network: When a node leaves the network, it will have minimal impact on the network as workers can freely leave and if the nonce solution was in its set a new solution will be found using another nonce value. If the leaving node is the active manager, the management team will elect a replacement and the parallel mining will continue without any noticeable impact. A new manager will be added to the manager support pool to replace the manager that was elected to become the new active manager. The number of managers in the pool will be dynamically scaled in proportion to the total size of the network and optimal redundancy requirements.

4) A Miner Requests New Nonce Range Before Completing Range: This is a highly unlikely situation and indicates a flaw in the operation of the miner. This will be counterproductive as the solution may be in one of the skipped values. While this will not cause damage to the networks functionality due to there being multiple possible solutions it will reduce the nodes chances of solving the puzzle thus, it is very unlikely that nodes would be altered to cause such a behavior.

5) Active Manager Goes offline and Manager From Backup Manager Pool Found Solution to Hash: Manager candidates will not be permitted to be elected if the manager in question has submitted a hash solution to the management pool in the absence of an active manager. The new active manager will be elected and the manager that found the solution will be treated as a normal miner until the block has been finalized.

IV. Experiment And Results

In this section experiments have been conducted on several test environments including both physical and cloud-based systems. The intended goals of these benchmarks are to illustrate the advantages of parallel proof of work using multiple manager nodes. To achieve this, we will start with the environment setup and network communication using SSH to establish a peer-to-peer network. The proposed modified star network was modeled using MPI communication via SSH. To establish the benefit of adding nodes to the network a benchmark program was made using the SHA-256 hashing algorithm to hash simple messages and the timer function was used to track the number of hashes that can be completed per second; by executing a set number of hashes and timing how long it takes to complete them with various numbers of nodes active as miners in the network. Several benchmarks will be implemented using the Go programming language and compared to that used in the work of Hazari and Mahmoud. Failures will be introduced, and block time will be measured using a timer to determine the changes in block time when manager nodes fail. These results will show the benefit of manager redundancy and the impact of a failed manager node on the parallel system. In the case of the proposed approach the backup managers will be set up to take over the active managers role in the event that a miner requests a nonce set, and the active manager fails to notify the management team that it has filled the request. In a real-world implementation, there will be a much more complicated manager election process that will fill this role which was not implemented during the testing process. This election process can be simulated by requiring the incoming manager to perform a small amount of work before taking over to better represent the time taken to elect a new manager, and to ensure that the Active manager actually needs to be replaced.

A. Environment

This paper research made use of several test environments including a physical Linux cluster, a cloud cluster, and Rasp-

berry Pi to collect benchmarks on parallel proof of work. The environments will be discussed in the following subsections in detail.

1) Physical Environment: Local benchmarks were taken by setting up a parallel computing environment consisting of 8 Linux machines running Ubuntu version 16.4 LTS. These machines were connected via ethernet cables and a switch with static IP addresses assigned to each machine to create a Linux cluster. Table I displays the environment specifications used to produce the benchmarks computed on the network. The network itself utilized password-less communication via ssh using stored key value pairs to connect 8 machines to form a Linux cluster on a LAN. The machines were connected via ethernet cables routed through a switch. The host files were edited to facilitate communication and NTFS was added to allow the machines to share programs across the LAN. Most of the benchmarks were executed using a Sha-256 hashing algorithm (Secure Hashing Algorithm 256). The reason that this hashing algorithm was chosen over others is that it is used in Bitcoin mining and the hash value is restricted in size. What this means is that for any given input message the output hash value will be 256 bits in length. This feature will greatly improve data storage capacity when the messages become large. The tests conducted on the Raspberry Pi platform were made utilizing the Blake2 hash within the random-x hashing algorithm used in mining Monero. This paper is focused on the increase in hashing capabilities provided with parallel proof of work so less focus will be given to the particularities of the hash functions themselves as the concept of parallel proof of work with multiple manager redundancy can be implemented with any blockchain that utilizes proof of work regardless of the hash function used. Sha-256 is a good starting point due to its use in the Bitcoin network and efficient data storage capabilities.

2) Cloud Environment: A cloud environment was also deployed using both Microsoft azure as well as Google Cloud Platform. Both deployments contained 8 nodes with 2 cores each. A virtual network was set up on Azure and communication between nodes was conducted via SSH. The performance of the two providers did not show any substantial differences between the two providers. Benchmarks were taken on the cloud environments to obtain block times as well as the possible hashes per second on the network with differing node counts.

TABLE I
ENVIRONMENT SPECIFICATIONS

Architecture	x86_64
CPU op-mode(s)	32-bit, 64-bit
Byte Order	Little Endian
CPU(s) Per node	4
Thread(s) per core	1
Core(s) per socket	4
Socket(s)	1
NUMA node(s)	1
Vendor ID	GenuineIntel
CPU family	6
Model	60
Model name	Intel(R) Core(TM) i5-4570S CPU @ 2.90GHz
Stepping	3
CPU MHz	3303.323
CPU max MHz	3600
CPU min MHz	800
BogoMIPS	5786.89
Virtualization	VT-x
L1d cache	32K
L1i cache	32K
L2 cache	256K
L3 cache	6144K

B. Benchmarks

Hash difficulty refers to the number of leading consecutive zeros of an acceptable hash. The greater the number the more difficult the hash is to solve and by extension the more work is required to solve it. Running a benchmark with a difficulty of 1 will be solved significantly faster than the same input with a hash difficulty of 10. The average time taken to solve a hash in seconds is used to measure the performance of the network. The average is calculated by measuring the time taken to solve a block a set number of times than dividing the sum of all times by the number of blocks solved. Hashes per second are calculated by timing the number of seconds taken to solve a block then dividing that time by the number of nonce values used to find the solution to the block. Optionally hashes per second can be measured by starting a timer, preforming a set number of hash attempts then stopping the timer and dividing the number of hashes executed by the number of seconds the timer has run. When measuring hashes per second we will see that different hash algorithms produce differing results and the hash difficulty IE the number of leading zeros will also have a significant effect on the hash speed of the benchmarks. Many CPU and GPU manufactures will opt to post hashes per second for their hardware that is excessively high in comparison to the hashes that will be seen when mining a crypto currency. For example, the Raspberry Pi is said to be able to produce an average of 108 hashes per second but at operational difficulty levels it only manages an average of 2.3 hashes per second as seen in Table II. The formula to determine the average number of hashes required to solve a block in the Bitcoin network can be seen in Equation (1):

$$Hashes\ per\ block = (Difficulty \times 2^{32}) \tag{1}$$

with a maximum difficulty of 2^{256-1}.

As of the time of this writing Bitcoins hash difficulty ranges significantly higher than that capable of being supported using CPU mining. The difficulty adjustment is directly related to the total mining power estimated by the Total Hash Rate (TH/s) chart [13]. This means that the hash difficulty is dynamically scaling to become more difficult over time. With our proposed approach the difficulty will also scale based on the networks capabilities to ensure against forking attacks, but care will be taken to ensure that the difficulty never exceeds the networks' ability to efficiently handle transactions. The difficulty used in our calculations refers to the number of leading zeroes the

resulting hashed value must have to be considered a valid solution.

Benchmarks for hashes per second were implemented using a difficulty level of 1 and the chrono library in C++ using mpich. Network communications used were broadcast, send and receive. Between nodes serving the designated roles, all roles were hard coded and dynamic node scaling was not implemented at the time of benchmarking. Other benchmarks were collected using the time library of the Go language with network communications provided by the go-libp2p library. Code relating to the benchmarking process can be found at [14].

Solo mining results in the speed of the fastest node being the average as the fastest node will always solve the hash before the others and the work of the others is wasted except during the validation step. During solo mining all the nodes are competing against one another to solve the hash before the others and only the fastest node will receive the reward for solving the proof of work.

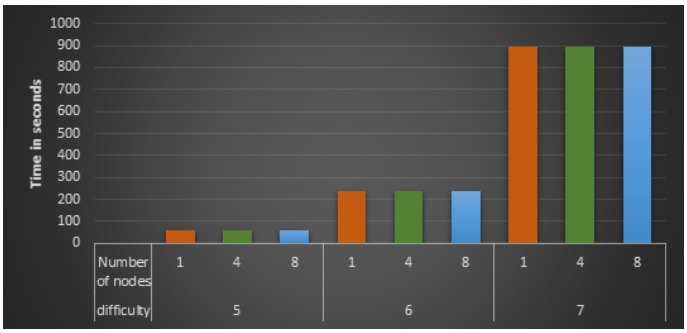

Fig. 1. Solo Mining Time in Seconds to Complete Hash

Figure 1 shows hashing times of a set of nodes solo mining. Note that the solution times are relatively the same regardless of the number of nodes as any advantage gained by introducing new nodes is only going to be visible if the new node has more computational ability than the others on the network. When the computational ability of the nodes is identical the hash times will be very close with minor deviations as other tasks run in the background. After averaging the runs, we get consistent results with no added benefit from the addition of more nodes.

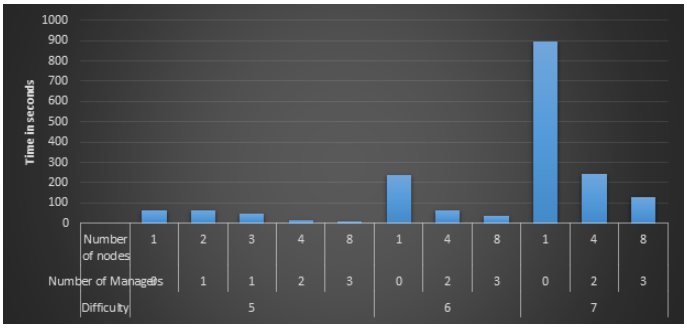

Fig. 2. Parallel Mining with Managers in Seconds to Completion

Figure 2 displays the benefit of additional nodes when using parallel proof of work. As seen in the figure the addition of new nodes has a visible effect on the hashing power of the network and drastically reduces the time taken to solve a block especially in the case of higher difficulty hashes. With a difficulty of 4 and below there is a relatively low amount of variance in the time taken to solve the hash due to its simplicity but as the difficulty increases the advantage of having more nodes begins to become greater. Once we reach a difficulty of 7 the benefit of parallel mining becomes obvious. There is little difference in hashing speeds with the addition of additional backup managers as the backup managers can still contribute to the mining process so long as they are not the designated active manager. If a manager finds the solution and the active manager is offline the management team will not permit the manager that found the solution to be elected as the active manager.

Figure 2 shows some interesting data relating to the hash difficulty and the number of nodes. Where the hash difficulty is low the time reduction the system sees is much lower than when the hash difficulty is increased. What this indicates is that in the presence of a large workforce the miners may become underutilized and if the nonce values are over spread the communication times may rise higher than the performance gained by dividing the work. Thus, Managers will need to monitor the work to worker ratio and divide the work accordingly leaving some workers idle if necessary. Idle workers will not be consuming the same power levels as working nodes thus this will result in energy savings across the network. As shown in Figure 2 networks with less than 3 nodes see no benefit from the addition of managers but any node count above 2 will benefit from additional nodes as even the manager nodes can dynamically scale their role back and serve as a miner when up to date creating added service to the network.

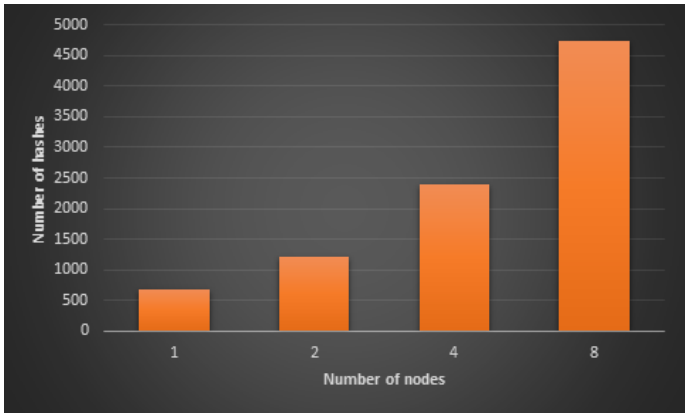

Fig. 3. Hashes per Second with Introduction of Additional Nodes

Figure 3 displays hashes per second that can be computed with the addition of more nodes to the network. As additional nodes are added the computational capability of the network increases allowing the network to compute higher numbers of

hashes as the node count increases. Note that the increase in hashes per second does not double when the node count is doubled; this is due to communication overhead required for communication between nodes across the network. Next, we will see how the data in Figure 3 contrasts with that in Figure 4 where the blocks difficulty is also considered.

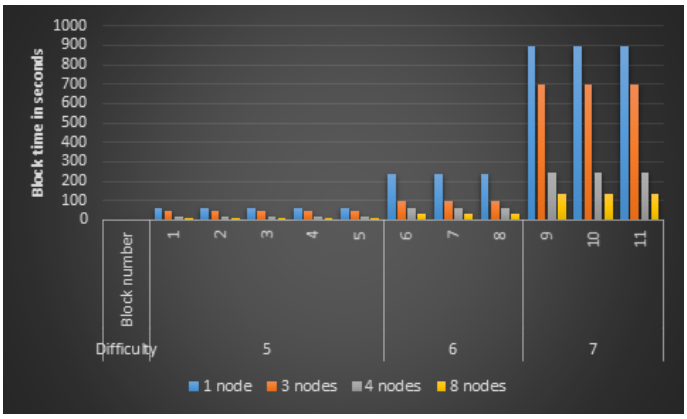

Fig. 4. Block Time with Increasing Hash Difficulty

Figure 4 shows block times in seconds as the difficulty increases with the lines indicating networks with differing node counts. Longer time periods to process a block are not ideal and we are aiming to achieve the lowest time possible to complete the block as with the use of higher difficulty levels the time period will increase exponentially. As we saw in Figure 3 the network with 8 nodes has the highest rate therefore produces the block in a fraction of the time required for the single node solo mining which is represented as the blue bars where the 8-node network parallel mining is the yellow bars in Figure 4.

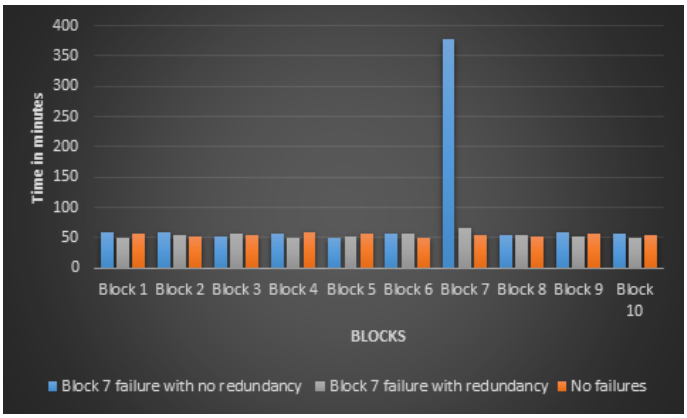

Fig. 5. Block Time Comparison with Node Failures

Figure 5 illustrates the impact of manager redundancy during the event of a node failure and network recovery. Note that the failure is only affecting block 7 and all other blocks are consistent. The difficulty level for this test was set at 10 leading zeros for an acceptable hash value. When the model running the Hazari-Mahmoud model with a single manager node represented by the blue line in our graph experiences a failure of the manager node parallel hashing stops and for the remainder of the block the network will perform at solo mining speeds. While in the case of the proposed approach the failed manager is replaced by a supporting manager and while the supporting manager is no longer able to contribute to mining hashes the parallel work continues through the remainder of the block causing only a small increase in time taken to process the block. Thus, in the rare event of a network attack or node failure the proposed approach provides a more robust and effective solution while introducing marginally higher communication costs to ensure network reliability.

TABLE II
MINING RESULTS FOR SINGLE RASPBERRY PI 4

CPU type	Raspberry Pi 4 Arm Cortex-A72
Coin type	Monero
Time	8 Hours
Difficulty	177,307,724,796
Hashes per Second	1 to 7
Average hashes per second	2.3
Blocks	o
Bad shares	1
Invalid shares	31
Good Shares	357
Total mined	0.000001410642

1) Raspberry Pi Tests: The Raspberry Pi is a good testing unit to consider when talking about the internet of things. If our network is to be connected to small devices, we should gather a baseline of what these devices are capable of in terms of hashes per second. Tests on the Raspberry Pi platform were done using the Raspberry Pi 4 with a Sandisk 32GB microSD card using the Raspberry Pi OS with Desktop. Heatsinks were added to aid in cooling the chips. The Raspberry pi was chosen because it has a low computational capability, and our aim is to develop a system where such devices being part of the internet of things may contribute in a meaningful way to the overall network. As we see in Table III the Raspberry Pi was not very capable when mining by itself resulting in an average hash rate of 2.3. With proper dynamic management and enough contributors, a smart network will be capable of sustaining a decentralized parallel mining system. One node by itself may not be very useful but together many of them could provide a scalable decentralized smart network to facilitate transactions true to Satoshi's vision of cryptocurrency [15].

Table II shows the Raspberry Pi 4's mining capability when mining Monero as a member of a mining pool. Note that when mining in a pool the hash difficulty was at 177,307,724,796 which is an extremely com difficulty thus resulting in exceptionally low hashes per second. These kinds of difficulty levels are common in the normal operation of a cryptocurrency but rarely used when benchmarking as most CPU benchmark tests aim to achieve the highest results possible without regard to real world load. Monero is one of the most used CPU mined cryptocurrencies and uses the RandomX hashing algorithm.

TABLE III
HASHES PER SECOND IN CLOUD ENVIRONMENT

Number of nodes	Hashes per second
1	339.00
2	670.54
3	1005.47
4	1326.30
5	1659.87
6	1994.06
7	2329.66
8	2653.34

2) Scalability Of Distributed Work: As shown by increasing the number of worker nodes the work becomes easier. There will be a time when the number of available miners is higher than that optimally required to compute calculations most efficiently. Thus, managers must ensure that the nonce values being distributed are not below a set size in comparison to the number and capabilities of the miners. If the nonce sets are too small the communication overhead could be higher than any gain achieved by splitting the work resulting in a loss of efficiency. The managers should dynamically monitor the networks condition to ensure that miners are not used unless needed to maintain the best energy and network efficiency. With new nodes joining the network available for mining if needed, the system should be dynamically scalable and can easily support the needs of the users with performance increasing as the number of users increases.

Table III shows the scaling of hashing capability as new nodes are added to a cloud environment. With the addition of nodes, the collective hashing capability of the network will increase as will the difficulty required to solve the hashes. This will reduce the likelihood that a 51% attack could occur as there will be little chance of a single entity gaining such a dominant foothold in a large network. This approach supports both decentralization and dynamic network scaling.

V. CONCLUSION

Cryptocurrencies are a growing technology and have great potential to leave a lasting mark on civilization. Integrating a decentralized cryptocurrency into both the cloud and internet of things will provide great scalability and accessibility to the blockchain. This is achievable by integrating parallel computing into a decentralized blockchain protocol to create a smart network. Parallel computing has many advantages to offer and its integration into blockchain technology will increase the benefits of the distributed transactional system. A truly decentralized transaction system will benefit from parallel application over the cloud and across the internet of things. The computational resources required to run blockchain technology can be dramatically reduced and confirmation times will improve with the addition of new nodes into the network. The scalability of this system is substantially superior compared to traditional blockchains and the proposed approach solves several security vulnerabilities present in traditional blockchain applications. Most notably the 51% attack as it will be extremely difficult for a single party to gain a majority share in the network, especially if the network is globalized and integrated into the internet of things.

REFERENCES

[1] N. Marinoff. The Bitcoin Price Drop May Have Been Caused By a Power Outage, Live Bitcoin News, 20-Apr-2021. [Online]. Available: https://www.livebitcoinnews.com/the-bitcoin-price-drop-may-have-been-caused-by-a-power-outage/

[2] C. Baraniuk. Bitcoin's global energy use 'equals Switzerland'. BBC News, 03-Jul-2019. [Online]. Available: https://www.bbc.com/news/technology-48853230

[3] S. Hazari and Q. Mahmoud. Improving Transaction Speed and Scalability of Blockchain Systems via Parallel Proof of Work. Future internet 12.8 (2020): 125131. Web.

[4] A. Hern. US seizes $1bn in bitcoin linked to Silk Road site. The Guardian. 6 November 2020. [Online]. Available: https://www.theguardian.com/technology/2020/nov/06/us-seizes-1bn-in-bitcoin-linked-to-silk-road-site.

[5] I. Eyal, A. Gencer, E. Sirer, R. Renesse. Bitcoin-NG: A Scalable Blockchain Protocol. In Proceedings of the 13th USENIX Symposium on Networked Systems Design and Implementation (NSDI '16), Santa Clara, CA, USA, 16-18 March 2016; pp. 45-59.

[6] X. Boyen, C. Carr, T. Haines. Blockchain-Free Cryptocurrencies. A Rational Framework for Truly Decentralized Fast Transactions. IACR Cryptol. ePrint Arch. 2016, 2016, 871.

[7] S. Popov. The Tangle. White paper 1, no. 3 .2018. [Online]. Available: http://www.descryptions.com/Iota.pdf.

[8] L. Lann. Distributed Systems-Towards a Formal Approach. IFIP Congress. 1977, 7, 155160.

[9] C. Dework, N. Lynch, L. Stockmeyer. Consensus in the presence of partial synchrony. Journal of the ACM, 1988-04-01, Vol.35 (2), p.288-323.

[10] H. Hasanova, U. Baek, M. Shin, K. Cho, M. Kim. A survey on blockchain cybersecurity vulnerabilities and possible countermeasures. International Journal of Network Management, 29(2), e2060n/a. [Online]. Available: https://doi.org/10.1002/nem.2060.

[11] T. Kawakami. A Node Virtualization Scheme for Structured Overlay Networks Based on Multiple Different Time Intervals. Applied Sciences, 10(8596), 8596. 2020. [Online] Available: https://doi.org/10.3390/app10238596

[12] D. Carl, C. Ewerhart, Ethereum Gas Price Statistics. University of Zurich, Department of Economics, Working Paper No. 373, 22-December-2020, [Online]. Available: https://ssrn.com/abstract=3754217 or http://dx.doi.org/10.2139/ssrn.3754217.

[13] Blockchain.com. Network Difficulty, Blockchain.com. [Online]. Available: https://www.blockchain.com/charts/difficulty.

[14] J. DeNio. Benchmarking Code for Parallel Mining. 2021. Available: https://github.com/SliverOverlord/Masters_Paper.

[15] S. Nakamoto. Bitcoin: A Peer-To-Peer Electronic Cash System. Bitcoin.org. Bitcoin Project 2009-2021. [Online]. Available online: https://bitcoin.org/bitcoin.pdf.