Experimental Comparison of Blockchain Scaling Protocols Using Virtual Machines

Research Paper, Draft 4 (non-TeX version)
Michael Lin, Yorktown High School

Abstract

The Internet has enabled a vast amount of applications that require fast and near-instant communication across increasingly powerful computers. With new developments in both computation and cryptography, an experimental yet proven technology has emerged: the blockchain. The blockchain enables timestamped, replicated histories of computation stored across the Internet with succinct proofs of truth. Trust is no longer required in a single party as computation is verified many times over by independent, uncoordinated parties. Trustless finances and verifiable computation have resulted from this new substrate but has not proven to be scalable enough to replace existing centralized finances and cloud services. The fault tolerance that is provided by a decentralized perpetual truth machine is an incentive to move computing, storage, and market applications to it, and provide an ecosystem for various applications. These are also known as smart contracts, enabling any automated action to be completed deterministically without dispute. In order to achieve this ideal, the scalability problem must be met without sacrificing the current benefits of using centralized systems. Because of the problems faced with this scalability trilemma, researchers have taken to creating protocols that attempt to maintain security while increasing scalability. Finding the best protocols and their optimal implementations, as well as their future-proofness is necessary for strengthening application infrastructure. Due to their decentralized nature, protocols can be compared using virtual machines in an experimental setting. It is expected that testing protocols that modify the blockchain structure itself will be less scalable but more secure than an external layer protocol.

An Introduction — Structure of the Internet

The history of the Internet was also the history of growing demands in data. Every time something needed to be accessed, modified, created, or destroyed, computers would send raw data to each other and performed tasks within milliseconds, but this great convenience came with a cost. Computers connected to the Internet could be meters apart, or halfway across the Earth. With the large variability in how data moved across the Internet, connections suffered from unreliability, faults, and attacks. As these technologies became better, modern cryptography was implemented to mitigate attacks, and redundancy in server architectures became the norm for any service provider on the Internet. However, problems still existed. Every time private data was sent out to an institution or company to be stored or acted upon, some form of trust was involved, and issues become apparent when a service such as online banking was compromised by an

References available in *Research Plan*

attacker, leaving many users in financial ruin. Furthermore, there was no guarantee that these services would always exist. All these problems boiled down to a general rule: When the masses trust data in the hands of few, disaster will ensue.

Fortunately, a new development aimed to defeat the issue of trust, run by the users, for the users. It was termed "the blockchain", a data structure that could hold any data for any purpose, using three properties: immutability, fault tolerance, and determinism. Arising from its simplistic nature of appending state, blockchains also held the property of decentralization.

Review of Literature

Elliptic Curves — Succinct Digital Signatures

Verifying that information is authentic was done differently on computers than on paper. The concept of digital signatures was first introduced when the RSA cryptosystem was published (Rivest & Shamir & Adleman, 1978). RSA was an asymmetric algorithm that relied on prime numbers and modular exponentiation to accomplish two goals: encrypting information with a publicly available key without revealing any secrets to be later decrypted and to produce a mathematical signature using a secret key and prove its validity without leaking secret data. All these operations involved a public and private key pair. RSA, though still in use today, was an inefficient algorithm. The basis of its security was derived from the size of its prime numbers. As a result, encrypted information (ciphertexts) and signatures tended to be very large and inefficient to verify. Using large numbers for security was not optimal for embedded systems, and a solution was soon to be reached. The concept of using elliptic curves, a type of algebraic plane curve, in cryptography was first introduced (Koblitz, 2001). Elliptic curves allowed for much more efficient, and succinct (smaller) ciphertexts and signatures, and used much smaller keys for equivalent security as RSA. Algorithms developed using elliptic curves soon found uses in a variety of services where security was considered paramount, including blockchains.

References available in *Research Plan*

Merkle Trees — Logarithmic Proofs

Bottlenecks could be encountered when large sets of data need to be signed, but space constraints were an issue. If Victor[1] the verifier wanted to verify a part of a set of data signed by Sam, he would have to get the data, hash it to check a signature, and ask Peggy the prover to fetch the signature from a large store of signatures to prove its validity. This system required the number of signatures to match the number of stored data objects. Schemes like this were often not efficient to implement. However, if a tree construction of hashes in a well-ordered set of data were used, the hash at the top of the tree is the only piece of information that would need to be signed, and any piece of data could be verified by providing the complement hashes, which would form a path up to the root (a "branch" of the tree). Proofs of this size would be logarithmically related to data and could save space in a setup. Data that is considered expired or void could be removed completely from storage without invalidating the tree, provided the other hashes were still stored as proof (Merkle, 1979).

bitcoin[2] **— The Genesis of a New Era**

Blockchains were a recently developed technology that provided permissioned data agreement across the Internet. The term was coined to describe a deterministically verifiable growing data structure that records changes to state. Blocks, as they were first referred to, were collections of state transitions, also called transactions, chained together with a public collision-resistant cryptographic hash function (CRHF). Using a method of proving a block's validity and canonical nature, such as a proof of minimum computational work, (Back, 2002), computers could rapidly agree on a history of collated state transitions in a decentralized, fault tolerant, and completely trustless manner. All transactions could be verified using Merkle branches.

[1] See *wikipedia:Alice and Bob* as quick reference. This is not a citation and is provided for convenience.
[2] Logo image and type font is in the public domain.

References available in *Research Plan*

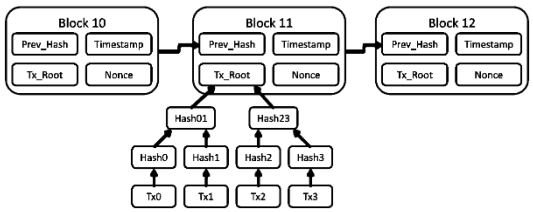

Fig. 1 Transactions, Tree, and Block Headers
Under Fair Use, source

The first novel use case of this technology was the application of currency on an immutable, trustless ledger of monetary transfer. These state transitions have found a use as transactions between accounts of changing balances. Invented and published by an unknown pseudonymous entity by the name of "Satoshi Nakamoto", Bitcoin was created as a transparent and peer-to-peer alternative to institutional banking (Nakamoto, 2008). The Genesis Block of Bitcoin was mined on January 3rd, 2009 and ushered in a new era of applications that could finally be realized without trust.

ethereum[3] — Persistent and Indisputable Computing

Blockchains since have found various other uses due to their generalization as indestructible and perpetual truth machines. Any code or agreement (or "contract") could be executed using computers, leveraging the fundamental nature of mathematics and computer science that one calculation done on one computer would always be the same on another computer. Another blockchain was be born to fulfill that purpose — Ethereum (Wood & Buterin, 2018[4]). Since its inception in 2014, many decentralized applications have been created and deployed for public use. These contracts required absolutely no trust, no arbiter, only computers. They could hold currency, send it on specific conditions, perform complex calculations, or anything that a normal computer could do. The possibilities of trustless computation for any purpose, financial or not, became endless in a literal sense.

[3] The Ethereum Visual Identity is licensed under CC Attribution 3.0 by Stiftung Ethereum.
[4] This refers to the Metropolis vByzantium revision of the paper. The original was published in 2013.

References available in *Research Plan*

The Rise of the Integrated Circuits

The Scalability Trilemma, which is discussed further in depth in *The Gap*, had become an apparent issue and subject of controversy in Bitcoin's governance. Bitcoin was inherently decentralized in its so-called "gossip protocol", computers would send transaction and block data around in a decentralized network, which would visually look like an irregular mesh. As the mining business grew, so did competition. The main hash function Bitcoin was based off, Secure Hash Algorithm 2, 256 bit (SHA256), was a secure algorithm, but miners have designed chips to perform specifically one function: Calculate SHA256 hashes as fast as possible, find a valid block solution and reap the rewards. These chips were not too hard to design, given that SHA256 was compute-heavy, not memory-heavy. These specialized chips ended up centralizing the entire mining business, which gave regular Bitcoin users no chance of mining unless they were willing to spend copious amounts of money on mining machines and electricity, and low chance of return on block rewards. The so-called "network difficulty" skyrocketed to the point where buying a few chips would be pointless, and entire computer farms were dedicated to calculating hashes. The select few who put in the investment became the *de facto* controllers of Bitcoin. This clearly shown one property now lost; decentralization. Satoshi also designed Bitcoin to have a 1 MB block size limit, and blocks would take approximately 10 minutes to produce. 1 MB was not enough to fit all the transactions spreading around the network and led to frequent congestion. Many transactions would take very long times to be confirmed by the network, not ideal when, for example, someone wanted to buy a cup of coffee and needed to wait 60 minutes for the merchant to become confident that the transaction was irreversible. Bitcoin had only one property that was still present; security. Due to the sheer amount of hashes calculated per second that secure the network, Bitcoin became the hardest blockchain to attack.

References available in *Research Plan*

Failure of Scrypt & Memory-hard PoW

This mining problem was originally thought to be caused by SHA256's efficiency. Application Specific Integrated Circuit (ASIC) manufacturers leveraged this to make mining machines calculate many hashes at once. To solve this problem, more memory-heavy Proof-of-Work (PoW) schemes were implemented. Litecoin, a software fork of Bitcoin, aimed to solve this by using Scrypt (Percival & Josefsson, 2016), a more memory-intensive function originally intended to derive keys from short input data (a "key derivation function"). In effect, however, it only moved the competition to graphics processors (GPUs), and eventually ASICs again. While other PoW schemes based on DAGs and the generalized birthday problem existed, they still encountered the same problems. The centralization problem was yet to be solved for mining.

Tradeoffs of Intrinsic Properties

Other developers attempted a different method, increasing the block size. Theoretically, this would increase the scalability, but it would also prove to reduce the security of blockchains. Bigger blocks meant greedier storage requirements, meaning fewer users were willing to sustain networks. Attempting to reduce the block time too much also introduced issues in consensus due to the inherent latency of computers on the Internet (Buterin, 2015). Some proposed running multiple blockchains to increase scalability without increasing block size. The more blockchains, however, the less security for each chain.

A New Type of Proof

Another type of proof had also been introduced, Proof-of-Stake (King & Nadal, 2012) (Buterin & Griffith, 2017). PoS replaced the cost of electricity with the cost of staking currency into a security deposit that could not be spent until withdrawn. Stakers could make money with money while "forging" new blocks. The similarities between PoS protocols ended there, with the idea taken and adapted in many ways. Although PoS was still a new idea and provably more efficient than PoW, it was not provably secure without either encountering forking attacks (the

References available in *Research Plan*

"nothing at stake" problem), high centralization (Distributed PoS, or DPoS), or harsh penalties for honest stakers who accidentally forked (the Casper protocol). In effect, either security or decentralization had to be sacrificed, with few gains in scalability.

Gaining Consensus in Test Networks

At another extreme, authoritarian testing networks existed with different consensus mechanisms. Some utilized low difficulty PoW, others used PoS, but one notable exception was Proof-of-Authority, or PoA (Szilágyi, 2017). These used centralized validator nodes which were responsible for expanding the blockchain and confirming transactions. These networks were completely scalable, secure under normal circumstances, yet were completely centralized. For this reason, applications were only tested here, and no monetary activity would ever occur on these blockchains.

The Problems

The Scalability Roadblock — Where Blockchains Fall Short

Despite being so versatile, blockchains lacked the ability to perform at the same level as major centralized architectures achieved and could not scale to large numbers of users (Croman *et al.*, circa 2015). The ultimate goal of research in this field had become a race to optimize this technology and eliminate the bottleneck attached to decentralized consensus. Many systems have been tried and tested, attempting to do away with this bottleneck through various means that increase computational capacity, reduce cost, and prevent centralization. However, while the end goal was clearly known, research efforts have diverged in many directions and attempted to solve multiple problems with little to no project synergy. Meanwhile, the performance of production blockchains had met various impedances, with high fees and wait times for

References available in *Research Plan*

transaction confirmation that risen to unacceptably high levels. There is yet to be a widely and undisputedly acceptable solution to this growing problem.

Fig. 2 Historically high fees on Bitcoin, peaking at $55.16 USD per transaction. Prices according to exchanges.
Under Fair Use, source

Governance — Fissures in Consensus

All blockchains behaved exactly as their code was written, even if the outcome was not desirable. As an effect, the irreversible nature of blockchains also meant that losses of funds, faulty contract code, stolen identities, etc. were treated the same as normal behavior to computers. This was unlike banks or credit card networks, where agencies could cooperate to reverse changes done by an attacker or by accident. To some users, the ability to reverse transactions seemed immoral and dangerous. This mindset was often termed "Code is Law" (Vicco, 2016)[5], which meant that whatever the result of a computation on the blockchain is said to have been, is what it should have been, and final. Others believed that changes should be allowed if the community deemed it necessary. Disagreement did not mix well with blockchains, as all users were meant to converge on a single chain; a single history and state that is canonical. When controversial changes that reverse damages or introduce protocol upgrades are not agreed upon, blockchains may split into two: one with the proposed changes, one with the old consensus rules. Many of these chain-splits have occurred on public blockchains, notably the Ethereum

[5] This is not a neutral source and should not be interpreted at face-value. Treat it as propaganda.

References available in *Research Plan*

DAO hard fork (a hard fork is a change in consensus rules that "loosens" the rules[6]), which was performed to return lost funds to users of an important smart contract drained of funds by attackers due to a programming error. Ethereum split into two chains, where Ethereum Classic used the old consensus rules and is maintained by its own community. New hard forks were still being introduced, but mainly to provide new network features and improve baseline security.

The Gap — Too many theories, too little results

The problem began with understanding three fundamental concepts in designing a blockchain. Security defined how resistant a blockchain is to an attack directly on a blockchain's network. Scalability described the amount of throughput that a network could encompass with no noticeable degradation in performance. In other words, the number of concurrent users that a blockchain could have before it becomes slow and unusable. Decentralization complemented security and described how equal nodes were treated in a network. If only a few powerful computers were providing all data to the network, it would not be decentralized. If each node provided as much data as it received and could connect to any other peer, the network would be completely decentralized. All three of these properties were desirable, however, as previous works have shown, incorporating all three properties was a difficult task, and often one property is left out. These three properties constituted what is commonly referred to as the Scalability Trilemma. The research field in this area was very large, and many efforts have been made to solve this trilemma. The problem lies in the fact that most research in this area was theoretical, and the pool of new ideas had not been put to the test. Understanding how different modifications in blockchain protocols affected the three properties was a core piece of information that missing in the research field. This data could only be gathered experimentally.

[6] Formally, a hard fork requires all clients to upgrade to a new version, lowering restrictions and adding previously incompatible rules, and move to the new blockchain, as old software will reject new blocks. This is not always how hard forks work.

References available in *Research Plan*

By gathering this data, research efforts could be put in the areas that count, and lead to a solution to the Scalability Trilemma much sooner than with fragmented research efforts.

Experiment Goal — Filling the Gap

Overview

The planned experiment was designed to measure the performance of blockchains using controlled methods. By using a completely sandboxed virtual machine, very precise data could be gathered about the three properties. Various blockchain implementations were tested. In this experiment, each blockchain implementation had its own distinct properties but mostly belong to one of two groups. Blockchain scaling protocols that focused on optimizing the blockchain itself to increase its scalability were called “on-chain” or “Layer 1”. Protocols that offloaded transactions and work from the blockchain and onto a helping peer-to-peer network(s) were called “off-chain”, or “Layer 2” (Stark, 2018).

Implications for Research

The results from this experiment aimed to focus research efforts on only one or two types of protocols, developing better versions faster than if efforts were fragmented on projects that may not provide an effective solution to the Trilemma. Further, emphasis on how protocols react to load differently would improve consensus on controversial protocol upgrades. As the Trilemma is finally solved in future research, the blockchain would be able to rival centralized architectures; eventually replacing them altogether.

Alternate Hypothesis — A Distinction Between Protocols

Overview

Both types of protocols had different use cases. By their fundamental nature, on-chain scaling protocols were more likely to have governance issues that made protocol upgrades difficult but tended to be more consistent and secure in making sure that transactions were sent and immutable. Off-chain scaling protocols were easier to upgrade, often had higher throughput due to the lack of immediate I/O with the parent blockchain but were susceptible to attacks on

References available in *Research Plan*

the helper network(s) due to their typically inferior security level, a tradeoff for increased scalability. In the event of an attack, helper networks would have to default to the parent blockchain to solve disputes deterministically (analogous to a hierarchical court system that makes rulings).

Elements of the Trilemma: Security

Due to their design characteristics, Layer 1 protocols were more secure. This is due to the direct involvement of the blockchain that maintained consensus and solved disputes. Except in the event of catastrophic network failure (i.e. severe network partitioning[7], 51% attack[8]), all rules of the blockchain would hold true in its history and state. Although harder to govern and propose amendments in rules, Layer 1 protocols were, in general, much harder to attack than a network not always operating on a blockchain. Layer 2 protocols used the blockchain as a consensus substrate, but users expected most operations to take place on an external network. This external network would be secure to an extent — typically using a decentralized ad-hoc network with commitment structures (called "state channels"), or smaller blockchains for end users that have less activity than a typical public blockchain — but could be attacked with less effort than a blockchain. Layer 2 failed in this aspect; when the helping network fails, users must default to the parent blockchain for safety.

Elements of the Trilemma: Scalability

Layer 2 protocols were designed to be very scalable by nature. The helper networks were inherently better at handling throughput due to the lack of a block size/computation limit. They would be on built on top of a parent blockchain, which could handle any extra throughput but generally would not need to. Layer 2 maintains its immutability through condensed state changes

[7] An otherwise meshed network split into multiple subnets, unable to communicate
[8] An attack where a majority of proving power (hash power or stake) is controlled by a single actor

References available in *Research Plan*

or net state changes periodically published to the parent blockchain. Old state changes may not be published, entailing a penalty if attempted (i.e. an attempt to steal money).

Elements of the Trilemma: Decentralization

Layer 1 protocols were typically decentralized to an extent, but many suffered from centralization as they grew in userbase. This was due to a specialization of users into three main categories: block producers, full nodes, and light clients. Block producers were also full nodes but had special equipment/funds dedicated to generating new blocks. There were fewer block producers than the entire body of full nodes, meaning they were the most centralized body. There were also less full nodes than light clients, as full nodes stored the entire blockchain, and storage cost played a factor. Light clients only needed to store their own transactions, and it was trivial to become a light client. Layer 2 protocols had the same roles, just reorganized to move light clients into a separate network. For this reason, decentralization could only be determined on a per protocol basis.

Overview of Methodology

This experiment has been designed to determine trends in performance (scalability) in various blockchains by adjusting transaction rates, with the goal of determining how different blockchain protocols scale under various loads. Indirectly, security of these blockchains can also be calculated using the intrinsic properties of each protocol.

To maintain a controlled test environment that can be reproduced precisely with near-equivalent results, virtual machines (VMs) will be rented from a cloud provider, where blockchains will be run. By adjusting transaction rates and collecting data on the order of hours, dependent variables can be calculated and plotted using information from the blockchain itself (i.e. blocks, transactions, and their timestamps).

References available in *Research Plan*

Methodology — Extracting Scalability Data

This experiment was designed and undertaken by student Michael Lin, under the guidance of Mr. Jonah Benton to troubleshoot technical issues and design problems. All results are open to the public for future research. Consult the *Experiment Outline §7* in the *Research Plan* for any unknown symbols, functions, or variable names.

Experiment Design: Controls and Constants

VMs were chosen due to their precision in specifications; no VM is more powerful than another for this experiment. This is a constant to reduce bias towards certain chains. A size of 2000 full nodes has been chosen for every network (excluding block producers and the centralized oracle. This seemed to be a reasonably large network size without incurring a large operating cost. The official Bitcoin Core client was used to run the control blockchain. Bitcoin was chosen due to being the first blockchain; it is an ideal choice to gather baseline performance data. Latency was added to the gossip of blocks and transactions between full nodes, mainly to prevent instantaneous gossip (would not measure scalability properly), and to emulate the behavior of the actual Internet, where latency is always present. A chance of failure existed if the hash matched certain values, albeit a very low chance. This "failure flag" was designed on purpose. The latency was calculated before a transaction was sent pseudorandomly using a quick hash function. Before gossiping, a client would wait for the designated time. A centralized oracle existed to collect the true timestamps of transactions; no latency occurred between full nodes and the oracle.

Experiment Design: Security Dependent Variables

On a per protocol basis, security DVs have been determined directly from each protocol's source code. For example, CT_s was 600 seconds, or 10 minutes for Bitcoin, 12 seconds for Ethereum, and so on. These variables are intrinsic properties of each protocol and have not been

References available in *Research Plan*

derived through experimentation. However, all other variables had been derived through experimentation.

Experiment Design: Bootstrapping the Networks

With the use of Kubernetes and Docker to automatically construct a network with containerized nodes, all nodes were able to communicate. This was achieved by populating each node's peer tables with other peer IP addresses. IP addresses were chosen at random to create a decentralized network topology, allowing gossip to occur naturally. While not trying to create a full mesh, node cliques were large enough that data could move efficiently. Additionally, block producer(s) and a centralized timestamp oracle were set up in separate containers. The use of these nodes was minimalized to prevent a bias in scalability data but introduced a slight amount of error. This is an unavoidable circumstance due to how blockchains are designed differently.

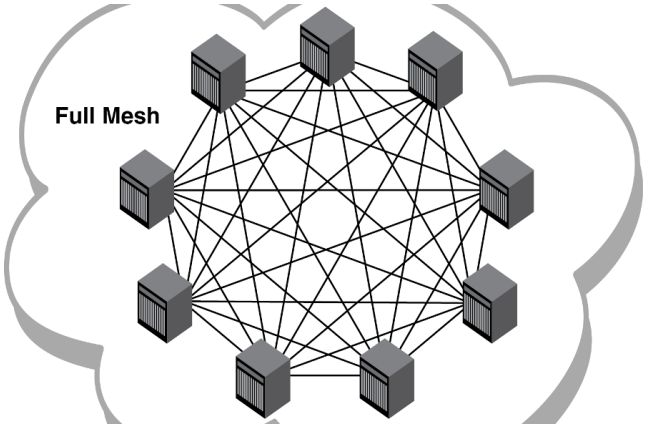

Fig. 3 A full mesh network topology, which P2P networks attempt to emulate at least partially.
Under Fair Use, source

Experiment Design: Generalized Procedure

Before the experiment was run on a blockchain, every node received a large endowment of funds to their addresses. *MinFee* was one of the dependent variables, and nodes had to be able to pay for these fees. Full nodes sent out transactions to random addresses in the set of full nodes in the blockchain (excluding its own address), at a specific rate. This was done by selecting nodes randomly to send a specific amount of transactions and wait. For example, if the rate was 3 TPS, 3 nodes would be selected randomly with a gap off 1/3rd of a second and send their

References available in *Research Plan*

transactions. This rate was measured for 2 hours, the length of a TPS interval. This was enough to gather the DVs and average them (*PoolSize* is not an average, but a regression function). Intervals started from 1 TPS all the way up to 120 TPS, a total of 10 days of uptime. Most of the scalability DVs were straightforward to derive, except for *PoolSize.* The mempool size had to be measured at various points during a TPS interval, and every time a block is produced, the mempool suddenly decreases. For this reason, a modified sawtooth wave was chosen as the best fit for representing *PoolSize* (See *Experiment Outline §7* for the general form).

Data Extraction

Timestamps and fees were of interest since they were used to calculate most of the DVs. Transactions and block timestamps were collected differently: block timestamps were collected directly from the network due to the presence of a block timestamp in the header. Transactions did not come with timestamps in their header and therefore had to be collected from an oracle with zero latency. Fees were present in transactions themselves, and the set of all fees in a TPS interval were all collected, where *MinFee* was derived. From this, *MaxBlock* and *MaxTime* were be derived. Every 5 seconds during the experiment, a peer was queried for the mempool size. These were stored in a separate file to derive *PoolSize*.

Additional Notes

The experiment was performed completely virtualized, meaning that all data originated from the experiment. No additional materials were needed besides the rented cloud VMs. No personal data needed to be collected, and all VMs operated in an isolated, sandboxed environment safe from real-world adversaries and had no consensus issues.

References available in *Research Plan*