The Standardization of Blockchain Benchmarking

by

Jefferson Van Buskirk

Presented to the Graduate and Research Committee

of Lehigh University

in Candidacy for the Degree of

Master of Science

in

Computer Science

Lehigh University

May 2023

Thesis Signature Sheet

Thesis is accepted and approved in partial fulfillment of the requirements for the Master of Science in Computer Science .

Thesis Title:
The Standardization of Blockchain Benchmarking

Name: Jefferson Van Buskirk

LIN: 817923105

5/4/2023
Date Approved

DocuSigned by:
Henry F Korth
567E318F6EB4440...
Director/Advisor Henry F Korth

Co-Advisor

DocuSigned by:
Brian D. Davison
EDC163969825469...
Department Chair/Second Reader Brian D. Davisor

Committee Member

Acknowledgements

This thesis covers research I performed independently as well as the preliminary development of a blockchain benchmarking suite. The benchmarking suite was a project developed by a larger group of students from National University of Singapore, University of San Diego, and Lehigh University. The students included Ren Kunpeng (NUS), Miguel Monares (UCSD), Zheng Yong (NUS), and Shizheng Hou (NUS), and myself. While I was very involved in the design of the benchmark driver, Section 3.3, as well as systems to create inputs for the driver, the NUS students were the primary developers for the benchmark driver. I would like to thank the professors who made this project possible. Hank Korth and Roberto Palmeri facilitated the project from Lehigh as well as Beng Chin Ooi and Dumitrel Loghin from National University of Singapore. I would like to specifically thank Hank Korth, who acted as my mentor and advisor during my graduate degree.

Contents

List of Tables

List of Figures

Abstract

A benchmark is a measurement of a systems performance against another systems performance. A proper workload is relevant, reproducible, fair, verifiable, and usable [24]. A proper workload is also standardized between the systems being measured, as to produce comparable results. The Transaction Processing Performance Council (TPC) provides a range of database benchmarks including standardized benchmarks for transaction processing, decision support, and virtualization. These benchmarks provide external parties with results that are comparable between systems and lead to informed decisions when choosing the best system for a given task. This standardization of benchmarks across the blockchain industry is necessary for the adoption of blockchain technology. Currently, blockchain organizations benchmark in-house using their own workloads, runtime environments, and definition of metrics. As such, the results of a given blockchain organization are not comparable to the results of another blockchain organization. In this thesis we explore the problems within blockchain benchmarking, our proposed framework for developing standardized blockchain benchmarks, and our implementation of this framework in a web3 focused benchmark for layer-1 blockchains.

1 Introduction

Since the introduction of blockchain with the proposal of Bitcoin in 2008 [19], blockchain technology has experienced rapid growth in popularity and development, with numerous blockchain applications emerging in domains such as financial services, supply-chain management, IoT, and healthcare [18]. With the increasing popularity and demand for decentralized systems, the number of blockchains has skyrocketed, most with their own unique features, protocols, and design choices.

However, the abundance of blockchain choices poses a challenge for developers and businesses seeking to implement blockchain solutions. It is difficult to determine which blockchain is most appropriate for a particular use case, as each blockchain cites its own performance characteristics, distinctive features, and unique strengths. Their published benchmark data give at least the appearance of being biased towards the blockchain being promoted and is usually not subject to independent verification. In addition to being unverified, the benchmarks are highly incompatible. Most blockchains report their throughput in transactions per second (TPS), but fail to define transaction in a standardized manner. Without a consistent definition of transaction, the numbers produced cannot be compared and are thus useless for actual decision making. The in-house benchmarking currently presented by blockchain organizations is only useful for marketing purposes.

This thesis presents a standardized framework for developing blockchain benchmarks (BBSF). This framework addresses the complications of blockchain benchmarking and requires explicit definitions of all aspects of the benchmark. This framework is accompanied with a driver to execute the benchmarks in a standardized manner, ensuring that all changes in performance measured are a result of the blockchain being measured. The framework also includes a standardized reporting format for easy comparison of results and replication of the benchmark for independent verification.

After presenting the framework, we present our implementation of this framework: Blockbench v3, a blockchain benchmark for public layer 1 blockchains composed of workloads relating to common web3 applications. This benchmark, and the results produced, show the viability of the BBSF and the possible future of a standardized blockchain benchmark.

This thesis assumes the reader has some background in the basic concepts underlying blockchain technology including the use of hashes and digital signatures to achieve immutability and irrefutability. This thesis also assumes at least a high-level understanding of the internal operation of Bitcoin's proof-of-work consensus protocol, and the basic concepts of proof-of-stake as used in Ethereum and other chains. An introduction to these concepts at a high level appears in a variety of sources including [16, 22].

The type of code that can run on a blockchain is dependent on the features that the blockchain has for code execution. Bitcoin has a stack-based scripting language offering no loops and no recursion and, thus, is not Turing complete. Ethereum and most other major blockchains offer a Turing-complete framework for writing code that runs on-chain. Such code is referred to in the blockchain community as a *smart contract*, but those more familiar with database systems will find this concept similar to a stored procedure. The power of a blockchain's smart-contract language and the strength of its built-in features play a significant role in what workloads the blockchain can run and how effective it will be in running them.

2 Initial Research

2.1 Defining a Proper benchmark

Before looking at blockchains, it is first important to understand the foundation of a good benchmark. A proper benchmark provides quantifiable metrics that can be used to compare systems against each other, identify the strengths and weaknesses of a system, and ensure that a system is applicable to a given project. Kistowski, et al. [24] define the desirable characteristics of a proper benchmark:

- *"**Relevance**: How closely the benchmark behavior correlates to behaviors that are of interest to consumers of the results.*
- ***Reproducibility**: The ability to produce consistently similar results when the benchmark is run with the same test configuration.*
- ***Fairness**: Allowing different test configurations to compete on their merits without artificial limitations.*
- ***Verifiability**: Providing confidence that a benchmark result is accurate.*
- ***Usability**: Avoiding roadblocks for users to run the benchmark in their test environments." [24]*

These characteristics sound obvious, but are easily forgettable in a decentralized environment. Two independent organizations could independently create their own proper benchmarks, but their results may not be comparable. Benchmarking two cars can be done in many different ways. If two car companies want to compare the acceleration of their cars they must measure the same metrics, in the same environment, under the same workload for the results to be useful. If they develop their benchmarks independently, it is unlikely that all of these requirement will be met.

If Company A measures the 0-45mph acceleration time of their car and Company B measures the 0-60mph acceleration time of their car, they are both measuring acceleration, but the two numbers measured are not comparable. Despite both measuring acceleration in the same units, the Company A benchmark cannot be extrapolated and estimated for comparison against Company B without making assumptions about the time to shift gears, maximum revolutions per minute, and top speed.

Standard metrics are important, but only under a standardized workload. The workload is the work that a system performs while the metrics are measured. If Company A measures their 0-60mph acceleration time with wind at their back or down a hill then their results will be skewed in their favor. This will allow for better marketing, and technically their measurement will be true, but clearly removes the ability compare their results with companies measuring on flat ground without wind. Explicitly stating the environment, processes, and requirements of the workload ensure that both systems are being benchmarked equivalently, producing comparable results.

Tesla advertises the Model S Plaid acceleration from 0-60mph less than 1.99 seconds [13]. They have a tiny asterisk that explains this is excluding the first foot of acceleration, effectively making the measurement a 6-60mph acceleration. This asterisk only exists on their fastest version of their models, misleading all of their customers and advertising benchmarks that are not measured under standardized workloads. In addition to cutting the first foot of acceleration, they also coat the track they are testing on with a resin to allow the car to accelerate faster [9]. This is a clear advantage for Tesla and generates times that are real, but not measured under relevant, standardized workloads. Cars drive on asphalt, and customers looking to purchase a car will be misled for assuming relevant testing conditions.

The same challenges present in car benchmarking are present within the blockchain industry. Differences in metrics, workloads, and environments present a plethora of challenges in creating proper performance benchmarks. The car industry has found the agreed upon standard of acceleration to be 0-60mph, and for the most part companies adhere to this standard. Blockchain organizations need this type of standardization for the industry to properly develop. Right now, every company is Tesla, creating their own benchmarks with adhesive tracks, as there is no standard for well behaved companies to use.

2.2 Blockchain vs. Database Benchmarking

The database community has a robust and mature set of methods for benchmarking. The TPC (Transaction Processing Performance Council[1]) database benchmark is a widely used industry standard for measuring the performance and scalability of database systems [11]. TPC has a variety of benchmarks that are used by major companies like IBM, Oracle, Microsoft, Intel, Dell, Cisco, Nvidia, and AMD. These benchmarks cover a wide range of database use cases, the most relevant being TPC-C and TPC-E, two workloads that simulate OLTP (Online Transaction Processing) businesses with multiple types of transactions aggregated into one transaction mix. The TPC-C and TPC-E benchmarks measure throughput, transactions completed in a set amount of time, with a constraint on individual transaction latency. As most databases are specialized for specific usage, developers can choose the TPC benchmarks that most closely match their intended use. This leads to simpler, more relevant benchmarks across varied database use cases. In the blockchain setting, the variance in use cases is much larger.

[1] tpc.org

While it may seem trivial to port TPC benchmarks to the blockchain environment and use them as the standard, developing a standardized benchmark for blockchains is significantly more challenging than it is for databases. First, the transactions being executed are not the same in both settings. In a database setting, the majority of transactions are data processing in the form of read, write, update, and delete. Under the TPC-C Benchmark, these database transactions are required to support the ACID (Atomicity, Consistency, Isolation, and Durability) properties [11]. Atomicity refers to either executing the entire transaction or no part of it. If the system crashes while a transaction is being executed, the whole transaction should be voided. Consistency refers to the idea that any update done to a consistent database state (all restrictions and conditions of the database are met) should produce a new consistent database state. Isolation requires transactions to properly support "time of check, time of use" requirements that prevent a transaction's execution from being influenced by transactions executing concurrently. Lastly, durability ensures that committed transactions are final and can be considered complete. These ACID requirements ensure that all transactions are similar and easily testable. The TPC-C transaction mix contains "business transactions" composed of one or more of these actions, creating transactions that are predictable and similar in composure. For these actions to execute, the code simply executes and commits.

In a blockchain setting, a transaction is much more ambiguous. Transactions may be simple debit/credit functions between wallets, but can also be much more complex. Transactions include minting new tokens or publishing contracts to the blockchain and creating new digital entities and assets. Transactions may execute smart-contract code that may, in turn, call other smart contracts, which is far more complex than a simple database function. This leads to measurements that claim to measure the same metric, but produce questionable results.

In addition to these differences, block finality adds another challenge. When a database transaction commits, it is done executing and is durable. On a blockchain, blocks that are newly added to the chain may not be considered final. Accidental forks of the blockchain happen often, sometimes requiring a several-block race for a fork to win. The blocks on the losing side of the fork are nullified ("orphaned"), making it so their transactions never happened. These transactions will eventually be attempted at a later point within the winning fork, which ensures that the chain operates properly but creates a benchmarking issue as it is not always obvious when a transaction is complete.

In contrast to the database concept of transaction commit, in which a specific atomic action makes the transaction durable, block finality, in many blockchains, can be considered as a probability distribution of how likely a transaction is final, and different users may set different acceptable thresholds for a transaction being considered complete.[2] A 90% probability that a transaction is completed will mark transactions complete sooner than a 98% probability, as more time will need to pass to solidify the transaction's completion. Two benchmarks measuring transaction latency with the same workloads will produce vastly different results if the finality thresholds are different.

Some blockchains, for example, Algorand [6], claim to support instant finality, a property resulting from the absence of forks under certain assumptions about the degree of dishonest node behavior. Instant finality makes benchmarking much easier, as it is easy to identify when transactions are complete, however not all blockchains support this property.

[2]The Bitcoin community accepts a notion of finality based on a block being followed by 6 more blocks, a process that takes approximately an hour. The exact probability of finality that this represents is dependent on many features of the block mining ecosystem and is thus difficult to compute with any degree of precision.

The differences in transaction execution are only a portion of the benchmark that require standardization. Representing a running blockchain is very difficult. Database environments can represent the expected hardware of a fully deployed database. In contrast, a fully deployed chain cannot be easily represented. Bitcoin is currently running with 50,000 validator nodes, which is not a model that can be created in a testing environment. A testing environment representing a chain cannot simulate the noise of other transactions running concurrently on the chain. Blockbench, a blockchain benchmark from 2017, benchmarks their workloads with blockchains ranging from 4 to 32 nodes [8].

A metric missing from most database benchmarking is fault tolerance. If nodes fail to act and consensus is not reached, actions are taken to fix the situation and get the system to where it should be. While this does happen, and recovery times are important, failures are less common. All nodes in a private database system are properly acting nodes and failure occurs only in the case of hardware malfunctions or software bugs. In a blockchain scenario, there is an incentive for nodes to attack the chain, and in a public scenario, there can be no expectation that all nodes are acting properly. Attacks are composed of nefarious nodes choosing to act in unpredictable ways that aim to prevent proper consensus and stop the chain from operating. Different consensus algorithms have different tolerances for attacks, a threshold at which nothing happens (denial-of-service), and a threshold at which the chain is taken over. It is important to benchmark both this threshold for total chain stoppage and the slowdown achieved by a smaller attack. Measuring fault tolerance provides a metric for the resilience of a chain under attack, a metric typically missing from database benchmarking since external security is not within the scope of a database system.

Blockchain	Throughput (TPS)	Blocktime (s)	Transaction Finality (s)
Ethereum (Pre-Merge)	30_1	12_1	900_8
Ethereum (Post-Merge)	$100{,}000_2$	12_3	N/A
Solana	$400{,}000_4$	400_5	5_4
BNB Chain (Binance)	5000_9	1_6	1_6
Celo	$140{,}000_7$	1_7	0_{10}
Algorand	$46{,}000_{12}$	2.5_{12}	0_{11}
Stellar	$2{,}000_{13}$	5_{13}	0_{14}

Table 2.1: Blockchain Organizations' Claims [Sources in Bibliography]

2.3 Current Blockchain Benchmarks

Blockchain Organizations' Claims

The blockchain industry is not void of benchmarking results. The results are primarily used as marketing resources to claim superiority over other blockchains, but the results produced are not comparable. The lack of standardization between benchmarks as well as no reporting requirements produces numbers without proper data to support them. Table 2.1 compiles some of the current blockchain measurements and explains the problems associated with them.

The claims shown in the table exemplify the problems with in-house benchmarking. The metrics presented for Ethereum are measurements produced by the live Ethereum chain under the workload of all Ethereum transactions, while "Solana's testnet has demonstrated 400,000 TPS on a single machine without any networking" [21]. Ethereum's results show the performance of their deployed, scaled, and active network; while Solana's results are highly specialized, engineered, results. While Ethereum's results appear to be less advertisement focused, their results are not any more useful than Solana's. Other blockchains looking to compare their performance to Ethereum cannot be sure of the amount of nodes in the network, the hardware the nodes are running, the actual workload being measured, or the definition of transaction being reported. For all blockchains in Table 2.1, the numbers are non-comparable and irreproducible, thus providing no use for developers.

Current Benchmarking Solutions

There are few available blockchain benchmarking solutions, however, they are limited in scope, functionality, or design. Others were designed specifically for older versions of blockchains and not constructed so as to be easily deployed on new systems.

Our proposal rests on the foundation of the prior Blockbench (and related) work discussed in [3, 8, 15]. Blockbench is a blockchain benchmarking framework released in 2017 that focuses on the evaluation of micro/macro metrics for private blockchains. Blockbench evaluates chains on workloads such as Smallbank and key-value storage. Since its release, the complexity and breadth of blockchain applications and workloads have dramatically increased, creating a need for a benchmarking solution that is relevant to modern blockchain use cases. Hyperledger Caliper [2] is a blockchain benchmarking framework that supports performance evaluations of transaction/read throughput, latency, and resource consumption using synthetic workloads. Another benchmarking solution is Gromit [20], which uses fixed asset transfer as its workload in its evaluation of blockchains' performance and scalability. BCTMark [23], a framework that benchmarks blockchains with an emphasis on system metrics, conducts its evaluations using workloads such as varied sorting algorithms. The Diablo Benchmark Suite [10] benchmarks blockchains with smart contracts inspired by Web2 workloads, such as Dota (gaming), Uber (mobility service), and YouTube (video sharing). While these workloads attempt to capture the nature of applications in the field, these Web2 workloads are not characteristic of applications run on a blockchain.

The prevailing issue among current blockchain benchmarking solutions is the lack of relevant workloads that are representative of realistic blockchain applications. This leads to evaluations that fail to characterize completely modern blockchains' true workloads and use cases. Section 4.2 further explores relevant workloads for blockchain benchmarking.

2.4 Problem Statement

Given the sparse ecosystem and wide architectural variance of blockchains, there is a need for methods to evaluate and compare different blockchains based on objective and standardized methods, in order to empower well-informed choices of blockchains for applications.

In the state of blockchains today, benchmarking of blockchain performance is mostly performed in-house. In-house benchmarking is great for advertising a blockchain's capabilities but lacks the verifiability and comparability of a standardized benchmark. Many current blockchain performance evaluations lack the transparency of methodology, workload, and testing environment, often leading to irreproducible claims.

Blockchain foundations, supporters, and others often cite figures presuming their proposed transactional models and workloads as the "correct" ones. This is far from the enterprise-focused benchmarking performed in the database industry by TPC, where workloads match what real users do and the benchmarks prescribe testing details from that workload perspective.

One difficulty in comparing the declared assessment of benchmarked blockchains is the lack of standardization in the metrics being measured, such as transactions per second (TPS). For the results of a benchmark to be useful for a fair comparison, they must be measuring the same metrics under the same workloads and environment.

Without a standardized system, the evaluation of blockchain performance is invalid and, therefore, largely uninformative to developers.

Standardized metrics are important, but only under a standardized workload. The workload is the set of processes that a system is performing while the metrics are measured. In the state of blockchain benchmarking today, workloads are heavily unstructured between evaluations, and there is a tremendous variety of possible workloads that are executed on a blockchain. For example, Solana claims to perform tens of thousands of transactions per second, an astronomical difference compared to Ethereum's 10s of transactions per second, but the definition of "transaction" is ambiguous. In general, blockchain transactions can be simple, such as debit/credit transactions for simple payments, or complex and demanding, such as NFT minting or smart contract execution. Classifying all of these actions as "transactions" allows companies to make claims that may technically be true, but not fair and relevant when comparing blockchains. Explicitly stating the environment, processes, and requirements of the workload ensures that systems are being benchmarked fairly. A comparison of performance claims by various chains published in early 2021 [17] illustrates the lack of clarity in terms of definitions and workload.

In addition to standardized metrics and workloads, a proper benchmark requires standardized environments. Blockchain environments may differ in hardware, number of nodes, and percentage of nefarious nodes; lack of standardized environments will lead to incomparable results.

A proper benchmark must explicitly specify how metrics are defined and measured, workload parameters for adjustable elements, and require the benchmark to list full details of the hardware environment (nodes, network, etc.). Such a benchmark must contain a variety of relevant workloads that explore the strengths and weaknesses of blockchains robustly so that each chain is tested not only on its strong points but also areas of weakness. In the following sections, we propose a framework for developing proper benchmarks (Blockchain Benchmark Standardized Framework), that contains requirements for workload definition, a structure for useful metrics with explicit definitions, a driver for executing these workloads, and a reporting format for presenting results.

3 BBSF Framework

The Blockchain Benchmark Standardized Framework (BBSF) provides standardization across all aspects of blockchain benchmarking [14]. It is designed to facilitate the design and implementation of standardized, relevant, and transparent blockchain benchmarks. The BBSF is composed of:

1. A standardized workload framework that contains explicit definitions of all aspects of a workload ensuring proper implementation across all blockchains.
2. Standardized, workload-specific, micro metrics aggregated into a set of macro metrics that are easily comparable between blockchains.
3. A standardized driver that interfaces with a fully deployed blockchain, generates the workloads, calls the transactions, and measures metrics in a standardized manner.
4. A standardized reporting format that ensures that all metrics measured are transparently communicated in an easily comparable and reproducible format.

3.1 Standardized Workload Framework

Each workload is composed of standard components to ensure that all implementations of the workloads provide comparable results. The workload framework starts with an overview of each of the major sections. Firstly, a summary of the type of activity represented by the workload is given as well as current applications that generate this type of activity. Next, an overview of the transactions that make up this activity is given. Workloads may include multiple transaction types as most applications have multiple types of actions. Having multiple transaction types means that the transaction throughput (TPS) measured in the workloads represents an “average transaction” that contains the average amount of work among the transaction types and their frequencies. Following the transaction overview are sections outlining the smart-contract functions, wallets required, and external structures used in the workload that may be "off-chain".

After the overview of the workload, the next section outlines workload sizing. For each workload, each worker client (more details in Section 3.3) is given a number of transactions determined by the transaction mix. Adding more clients increases the total number of transactions and thus the total workload size, however any number of clients that properly stress the system suffices. The number of nodes used is reported in the standardized results reporting framework. This workload section includes a statement of the transaction mix and the arrival distribution of transactions. The transaction mix contains the list of transaction types and their contributing portion to the overall transaction mix. While some workloads have only one transaction type, others will have multiple. It is important to ensure that workloads with multiple transaction types use the same mix across all implementations. As the transactions are pseudo-randomized, the exact proportion of each transaction to the total mix may slightly vary. The transaction mix has minima and maxima for the proportion of each transaction type permitted. Arrival distribution refers to the times at which transactions are given to the workload. Some workloads will provide all of the transactions at time 0, while other workloads create new transactions throughout the workload, with workload-dependent distribution characteristics (BBSF does not mandate any specific distribution such as uniform or Poisson).

Following the workload section, the setup section outlines the starting conditions for the workload. The wallet and contract sections outline the starting balances of wallets and contracts, as well as the initial state of the contract. If certain information needs to be in the contract before the workload begins, that need is stated here, including the information required and the contract functions that need to be called to reach this state. The setup section also contains the starting values for external structures that may be used for assisting the workload. It is important to note that if these external structures are accessed by a smart contract associated with the workload the performance of this structure may impact results. To ensure that this impact is standardized, implementations of these structures must be explicitly defined.

The contracts section of the workload explicitly defines each contract used in the workload. Each contract has an overview of the wallets, structures, and other contracts called within its functions and then outlines each variable, struct, event, and function. The variables, structs, and events are straightforward definitions of what each contains as well as a description of the usage and purpose of each. The functions section lists each function's parameters, return type, processes, wallets accessed, events emitted, and external contracts called. It is important to define each function in as detailed a manner as possible to ensure that the transactions are implemented by different blockchains in similar fashions. While functions could be defined line by line, we chose to take a less controlling approach. Some blockchains support different tools within their programming language that possibly can affect performance. While this does seem like an unfair advantage in what should be a standardized process, it is important to remember the goal of providing results that are relevant to real developers. If specific blockchain languages support different functionality, this functionality will be used by developers building on top of said blockchain and will thus see the performance increase as a result.

The results of a blockchain's performance under a workload are determined through measurements called metrics. Each workload is structured with a list of workload-specific micro metrics that directly measure the performance of the blockchain under the workload, usually relating to throughput, latency, and time. Each micro metric has sections explaining what units are being measured, what transactions are associated with the metric, and how to measure[1] the metric. These micro metrics serve as intermediate measurements that can be aggregated to determine the macro metrics of a blockchain, the overall, cross-functional scores. The macro metrics can describe the performance of the blockchain in 3 scores without requiring any understanding of the underlying workloads. This allows developers to make quick decisions without needing to read entire performance reports.

3.2 Standardized Metrics Framework

In addition to standardizing the transactions being called, BBSF has a standardized framework for the metrics being measured. Each workload has specific micro metrics used to compose macro metrics by changing the parameters of the blockchain being measured. Changing blockchain parameters and plotting the changes in micro metrics leads to a broader analysis of the performance. For example, running the same workload using a blockchain with 1, 4, 8, 16, 32, and 64, nodes and plotting the micro metric throughput against the number of nodes will provide insight into the scalability of the blockchain. This macro metric can be used to extrapolate how efficiently the chain will run at full scale and conveys valuable information that cannot be measured by one workload. Although individual workload results can be compared, macro metrics provide a zoomed-out view of these results that is more easily understood by an external viewer. A developer can look at these macro metrics without needing to understand the underlying workloads and still make accurate comparisons about their blockchains of interest.

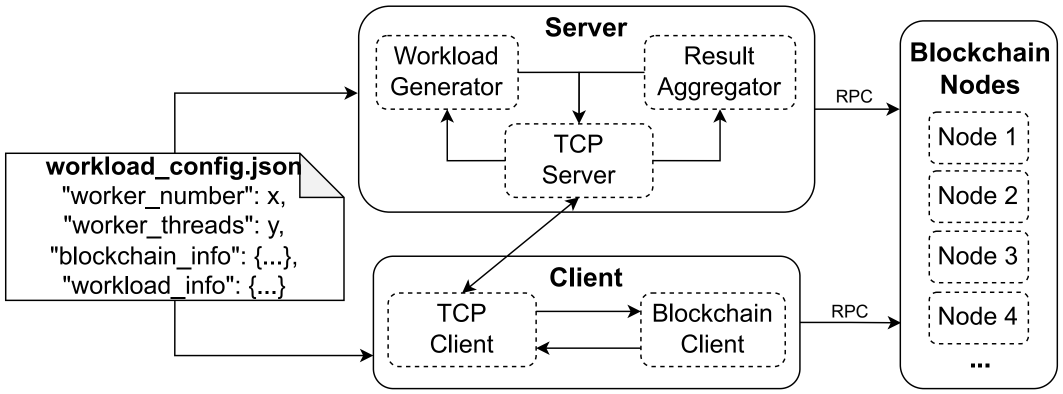

Figure 3.1: Driver Design

3.3 Standardized Benchmark Driver

We created a driver that executes the workload on the respective blockchain. Similar to the case for workloads, a proper driver must interface with all blockchains in a standard manner, so as to not have driver performance impact blockchain performance. The main goals for the driver are that it should not affect the performance of the blockchain, semi-idempotent execution that preserves real-world transaction ordering constraints so that running the same workload multiple times yields the same metric scores, and an implementation that makes it easy to benchmark a variety of blockchains. Our driver implementation is a server/client separate from the measured blockchain that calls transactions from clients without any direct connection to the blockchain itself. By keeping the blockchain a separate system, the integrity of the benchmark is preserved as the system that is being measured is the same system that will be deployed.

[1]Recall that we noted earlier that measuring these metrics in a blockchain setting may be less precise than in database benchmarking, since, for example, blockchain transaction finality may be only probabilistic, while database transaction commit is deterministic.

Structure

The driver is structured as a server that connects to multiple worker clients that act as users connecting to and using the blockchain. The server is responsible for compiling and deploying the contract, generating the workload from the transaction list and wallet list, signing each transaction, and sending transactions (invocation to smart contracts) to the clients. The clients call these transactions as normal users would and measure the metrics for each individual transaction. This structure ensures that the limiting factor in the benchmark, and thus the process being measured, is the execution of the transactions on the blockchain. Without multiple clients, a fast blockchain could outpace a slow driver and process transactions faster than a single source could call them. Multiple workers allow the driver to scale infinitely and eventually call as many transactions as the blockchain can handle.

Functionality

The driver takes inputs of the workload contract, a wallet list, a transaction list, and a finality parameter. The wallet list is the set of wallets used during the workload execution, represented as a file with the address and private key pairs. The transaction list is the set of transactions used in the workload composed of the type of transaction, the parameters of the transaction, the time the transaction is to be called, and the wallet calling the transaction. The finality parameter is determined by the user running the benchmark, as the process for determining this number is external to the benchmark. Setting this variable to 1 marks all transactions as final after they are one block "deep" onto the chain while setting it to n defers marking the transaction as final until it is n blocks deep. This variable is reported with the results to ensure the benchmark is reproducible.

The process starts with the server compiling and deploying the contract and signing all of the transactions using the wallet list. The server then sends the transactions to the clients, and then simultaneously signals the nodes to begin the workload. Once the clients start calling the transactions, the clients log when each transaction was called. While the workload is running, the server monitors each block and measures the throughput. Throughput is measured by taking the current completed transactions and measuring the current block timestamp and subtracting the time the first transaction was called. Dividing the completed transactions by the total time gives the current average throughput. When the workload is complete, this will represent the average throughput for the whole workload. When the workload is complete, the clients send their transaction timestamps to the server. The server uses these timestamps as well as the appropriate block timestamp to find the latency for each transaction by subtracting the block timestamp from the sending timestamp. Adding these differences together and dividing by the total transactions aggregates the average latency.

Workload Generation

While two of the driver goals were satisfied through a blockchain-neutral driver, semi-idempotency is achieved through the usage of a transaction-list file rather than randomized workloads. Every workload contains an arrival distribution and a transaction mix. These components are useful for understanding the transactions a workload contains, however, they could be interpreted slightly differently upon implementation. Varying interpretations could lead to changes in performance. For example, in the NFT marketplace workload, if the transaction mix is 50% listing and 50% buying and there are 100 transactions total, there will be 50 lists and 50 purchases that need to be measured. If all of the lists happen before the purchases, then everything will work fine. However, if the purchases happen before the listing of the NFT occurs, the purchasing transactions will fail as they are trying to purchase NFTs that have not been listed yet. This ordering discrepancy will cause the performance metrics to be significantly different, despite the individual transactions being exactly the same. To address this, we use pseudo-random pre-generated lists of individual transactions. To generate these transaction lists, an unordered list of transactions is generated to match the transaction mix related to the given workload. This transaction mix is generated from the relevant historical data relating to the given workload. The ordering of these transactions is then randomized while maintaining a proper ordering of related transactions. A list for NFT A will always come before the purchase of NFT A, however, there will be a random number of randomly chosen transactions between them. This pseudo-random ordering ensures that transactions occur in the correct order without allowing custom benchmark-specific code to exploit a set transaction ordering.

Using multiple clients ensures that the blockchain is the limiting factor, but causes idempotency problems. If the transactions are to be called in a specific order and there are multiple clients calling transactions at a time, the ordering of the transactions cannot be guaranteed. Client 1 could call the transaction to purchase NFT A before client 2 could call the transaction to list NFT A. To accommodate the concurrent calling structure, the transaction list represents a portion of transactions that represent a portion of the total workload. When this set of transactions is sent to each client, it is given a prefix to distinguish it from the transactions in the other clients. For example, client 1 may receive transactions regarding NFTs 1A, 1B, 1C, and 1D, and client 2 would receive transactions regarding NFTs 2A, 2B, 2C, and 2D. The transactions mix would be the same for each client however each node would be dealing with its own set of NFTs. As long as each client has ordered calls, then the overall workload's integrity is preserved. Client 2 can never call a purchase for an NFT that client 1 has not listed, as client 2 and client 1 deal with completely different NFTs.

3.4 Standardized Reporting Framework

Comparability among benchmark runs depends not only on standard workloads, metrics, and driver, but also on a result-reporting format that simplifies comparison among experimental runs by a variety of organizations. While the main goal is to provide easily understood macro metrics, it is important that all driver inputs, workload specifications, and environmental specifications are properly reported to maintain transparency. Reporting every aspect of the benchmark allows independent users to verify the results, by running the benchmarks themselves. Although each benchmark will be reporting different macro metrics, the reports from all benchmarks should follow the BBSF reporting framework.

To properly report the results from the benchmark, one must report the results of their experiment and the parameters used to recreate the experiment. Reporting the measurement is as simple as reporting the macro metric measured, as well as the individual micro metrics used to generate the macro metric. The macro metric is what will be compared and advertised, while the micro metric reporting gives credibility to the macro metric. Users who wish to verify the results can match their micro metrics with the reported ones, ensuring that everything being reported is correct.

In addition to reporting the results, the workload parameters, runtime environment configuration, and hardware must also be reported. Workload parameters include the workload being measured, the size of the workload used, and any other variables within the workload. Runtime environment specifications include parameters such as the amount of nodes used and the threshold for finality. These parameters represent the configuration of the blockchain being measured. Finally, the hardware being used must be reported as better hardware will yield better results. In conjunction, the workload parameters, runtime environment configuration, and hardware provide all information required to replicate the experiment and validate the reported results. This allows for third party verification of results as well as better comparability between reported results, as users can be sure that the environment that tests occur in are the same.

4 Blockbench v3

Our implementation of the BBSF, Blockbench v3 [1], is a benchmarking platform focused on Web3 applications running on layer-1 blockchains. The workloads proposed in Blockbench v3 cover a variety of transaction types seen in the Web3 space, as well as a range of arrival distributions to provide full insight into a blockchain's performance on Web3 applications. The results of this benchmark serve to both help developers make decisions regarding web3 performance on layer-1 blockchains and as a proof of concept for the BBSF as a whole.

4.1 Workload Design

Designing workloads for Blockbench v3 focused on choosing relevant applications, a variety transaction types and arrival distributions, and a range of measurable metrics. Section 2.3 discusses the most pressing issue with current benchmarking solutions: the relevancy of their workloads. The Diablo Benchmark Suite [10] is one of the most advanced blockchain benchmarking solutions, but their workloads fail to represent actual blockchain use cases. The Diablo Benchmark Suite workloads include a live gaming application representing the activity of a game of Dota 2, and a video sharing app representing YouTube. A live game requires large transfer of data in a live environment. Given the nature of mempools, gas fees, and mining, the information sent would not be received in a live, in-order fashion. In addition to the issues with their live game, the idea of storing video data on a blockchain is unreasonable. Although there are no official numbers for the total amount of data stored on YouTube, an independent user estimated a total of 2500 PB [1]. While there are decentralized storage organizations that utilize blockchains, these systems

[1]https://www.quora.com/What-is-the-total-size-storage-capacity-of-YouTube-and-at-what-rate-is-it-increasing-How-is-Google-keeping-up-with-the-increasing-demands-of-Youtube

do not store the information on chain. The leading storage oriented blockchain, Filecoin[2], allows users to purchase storage space on other nodes' computers and uses the blockchain to validate the storage. Filecoin does not store one's data on chain, as this would require every node in the network to store this data. Benchmark design needs to consider cases where a blockchain application may interoperate with off-chain data that is not stored on-chain but only secured on-chain. Treating such data as fully on-chain does not produce useful benchmark results since economic forces would drive real users to the off-chain-stored, on-chain-secured approach.

In addition to choosing relevant workloads, when designing Blockbench v3, we strived for a variety of transaction types and arrival distributions. Blockbench (v1) [3, 8, 15] was composed of relevant workloads, but lacked variety. Transactions all represent similar actions, primarily sending and receiving tokens. When Blockbench (v1) was designed, there were not as many developed transaction types as the blockchain industry was much younger. Blockbench v3 focuses on a broader interpretation of "transaction". Workloads are composed of a mix of different transaction types associated with the application represented by the workload. For example, the Token Exchange workload is composed of transactions including supplying liquidity, withdrawing liquidity, and token trades. This mix of transactions is averaged together to form the measured "transaction". In addition to a variety of transactions, we wanted a variety of arrival distributions. Arrival distribution refers to the time in which the blockchain is "made aware" of each transaction. For the Token Exchange workload, the arrival distribution is somewhat constant, with X transactions being called per second. To achieve a variety, we included a sports betting workload. As soon as the big game is over a sports betting application knows exactly who needs to be paid out. In this instance, the arrival distribution has all of the transactions

[2] https://file.app/

called at time 0. This variety ensures that the results produced from each workload are different from each other and provide a wider range of information. By choosing a variety of relevant use cases, the transactions, arrival distributions, and metrics associated with each workload are naturally a variety. Transactions include liquidity supply and withdrawal, token trading, NFT minting, NFT lists and sells, as well as simple sending of tokens between wallets.

4.2 Blockbench v3 Workloads

Blockbench v3 is composed of 4 workloads: a decentralized token exchange, an NFT marketplace, NFT minting, and a sports betting site. These workloads use a variety of transaction types and arrival distributions to cover a range of blockchain use cases. The workload sizing, transaction mix, and arrival distribution of the workloads are determined through historical data from Web3 applications that perform similar tasks to the workload, or historical demand from Web2 equivalents. For the sake of benchmarking, these workloads are simplified, representing the core functionality required to complete these tasks.

Token Exchange

The first Blockbench v3 workload is a decentralized token exchange roughly based on Uniswap-V2 [12], an application that allows users to trade tokens of type A for tokens of type B for a small fee. The exchange is an automated market maker that calculates exchange rates automatically and removes the need for a central party to create a market. In addition to using the exchange to swap tokens, users can earn rewards by providing liquidity to the exchange. When a user is done providing liquidity, that user can retrieve the tokens and the share of the fees generated by swaps using liquidity that had been provided. This workload is composed of transactions for providing liquidity, retrieving liquidity, and swapping tokens. To simplify the smart contract, we do not implement the reward function of the liquidity pool, due to the complex economic model. The workload sizing, transaction mix, and arrival distribution are generated from Uniswap-V2 historical data.

NFT Marketplace

The next Blockbench v3 workload represents the trading volume of an NFT marketplace. NFT marketplaces allow users to list, sell, and buy NFTs. Non-fungible tokens (NFTs) are one-of-a-kind tokens that are tied to a digital asset. Owning the token proves ownership of the tied digital asset. NFTs have a very high potential for practical real-world application. NFTs can be used for titles/deeds of a car or house, tickets for concerts or flights, or licensing of music or other creative works. Despite this potential, most NFT usage right now is digital art. Our inclusion of this workload is motivated not by recent NFT fads, but rather by the long-term application potential in business and government. NFT transactions require different protocols than those for simple debit/credit transactions, broadening the pool of features measured by Blockbench v3. This workload is composed of listing transactions, posting of an NFT, sale transactions, and the transfer of funds and NFT. These features represent the core functionality of an NFT marketplace. Workload sizing, transaction mix, and arrival distribution for this workload are based on OpenSea historical data.

NFT Minting

The third Blockbench v3 workload is an NFT-minting workload. NFT minting is the process of generating an NFT. This process is interesting as a unique virtual asset is created. As mentioned above, NFTs have many potential uses but the current digital-art use case creates a particularly interesting transaction type. In a digital-art collection, each piece has unique traits. These traits allow each piece to be part of the whole collection but unique in its own way. To mint an NFT in this style, a random number is generated to determine the traits of the NFT being minted. This generated number needs to be different from every NFT minted in the past to avoid duplicate pieces being created. Once a unique random number has been generated the associated artwork needs to be tied to a token. This is done using Interplanetary File Storage (IPFS)[3]. After the artwork is tied to this token, the token is given to the user. This workload does not require an arrival distribution as the structure of the workload provides all transactions at time 0 (because the typical NFT project does most, if not all, minting at the initial deployment of the NFT project). The sizing for the workload is based on common NFT project sizes.

Sports Betting

The last workload of the Blockbench v3 benchmark represents the traffic of a sports-betting website. Sports betting is a large industry that revolves around a middleman bookie that could easily be replaced with a smart contract. In the time leading up to the game, users will place bets on the game and then once the game ends all of the winners need to be paid out. This workload represents the calculation of winners and the payment to these winners. As soon as the game ends, all information is available to the contract and thus all winners are known. This creates an interesting arrival distribution where all transactions start at time 0, allowing for the blockchain to operate at maximum throughput. Although we are modeling a Web3 sports-betting environment, we have sized the workload using data from current Web2 sports-betting sites such as Fanduel on games of varying sizes.

[3]The actual piece of digital art is likely a large data item. Because on-chain storage is expensive, our workload uses IPFS for off-chain storage of the NFT itself with that off-chain data bound to the generated token

4.3 Blockbench v3 Macro Metrics

The macro metrics proposed by Blockbench v3 are designed around the central trilemma of blockchain. The trilemma states that a successful blockchain needs to be decentralized, scalable, and secure. However, in practice, a blockchain struggles to maximize all three of these pillars. A decentralized blockchain's main feature is the lack of a central authority. Without a central authority to control consensus, blockchains use decentralized consensus algorithms to ensure security in the decentralized environment; however, these algorithms tend to scale poorly. Bitcoin has high security and is completely decentralized, but has very slow performance. If the blockchain aims to prioritize scalability while maintaining its decentralized status, then the requirements for consensus will be lessened, possibly leading to worse security. If the blockchain has high security and high performance, then a central authority is required to process the transactions. Visa, although not a blockchain, has very high security and also very high performance, but is obviously a centralized organization. This trilemma is a triangular spectrum in which moving towards two of the pillars brings us away from the third. While these features are at odds with each other, the Blockbench v3 macro metrics allow developers to choose which features suit their requirements and which blockchains maintain these features the best.

Decentralization

The first pillar of the trilemma, decentralization, cannot be tested in a small test environment and does not necessarily connect with metrics like scalability and security. In addition, decentralization is difficult to quantify, especially in a fair and standard format. For this reason, our current implementation does not support any macro metrics regarding decentralization

Scalability

The main elements the macro metrics focus on are scalability and security. For scalability, the macro metric comes from measuring the micro metrics of a given workload with different numbers of nodes. Running the benchmark with 4, 8, 16, 24, and 32 nodes provides a set of micro metrics that can be graphed against the number of nodes. This graph can then be curve-fit to extrapolate how the blockchain will perform in a full-scale environment. The macro metric is the function provided by this curve-fitting process. Measuring with a larger number of nodes would provide more accurate results, however, it is unreasonable to expect every blockchain to have a test environment with more than 32 nodes. Scalability provides insight into how blockchain performance is affected as more nodes join the network. As more nodes join, the network is more decentralized and may have more computing power[4], however, responses from more nodes are needed to reach consensus. Users can use the results provided by scalability to see how two blockchains would perform at the same size, despite the deployed, live, chains being different sizes.

Fault Tolerance

For security, the macro metrics crash-fault-tolerance and nefarious-fault-tolerance explore the effects of performance when nodes are not behaving properly. Looking at the consensus algorithm used, it can easily be determined how many nodes are required to perform correctly for a system to work. A blockchain cannot function below that threshold. For fault tolerances, the macro metrics explore the impact on performance as this threshold is approached. Crash fault tolerance measures the performance change as the number of nodes that return nothing increases. To mea-

[4]In most blockchains, every node runs every transaction so the added computing power adds no significant ability to do more work. This remains true in general for sharded blockchains since the number of shards is normally fixed independently of the number of nodes.

sure crash fault tolerance, run a workload on a 32-node blockchain with 0, 4, and 8 faulty nodes. In this metric, faulty nodes return nothing during consensus, representing nodes that have crashed. With each amount of faulty nodes, we measure the performance of the relevant micro metrics. The crash fault tolerance macro metrics are composed of the ratios of the performance, with 0:4 faulty nodes and 0:8 faulty nodes. To get the nefarious fault tolerance, the same steps are performed, except the faulty nodes return random[5] responses, emulating an attack. These macro metrics provide insight into how well blockchains can handle attacks and how much their performance is affected during an attack.

[5]While the actual definition of malicious nodes is omnisciently evil behavior representing the worst possible case for a correctness proof, actual construction of such behavior for a workload is not possible as it is equivalent to the halting problem, which is proven to be undecidable.

5 Preliminary Results

Blockbench v3 is still a work in progress. The driver is up and running however there is still a lot of work to be done. For each blockchain benchmarked, workload smart contracts need to be coded, workload transaction lists need to be generated, an interface between the worker nodes and the blockchain needs to be developed, and a full blockchain network needs to be hosted to run the transactions within the benchmark. To prove that our framework is useful, and the driver is capable of executing workloads and measuring performance, we implemented the Token Exchange workload for Ethereum and Quorum.

Launched in 2015, Ethereum was the first platform to support smart contracts, decentralized applications, and layer-two tokens while using the blockchain to protect from fraud, downtime, and interference from third parties. Choosing Ethereum as a place to start was both acknowledgement to its industry dominance, but stemmed more from our familiarity with Solidity, the smart contract programming language, and ability to spin up an Ethereum network for testing. Ethereum currently runs a proof-of-stake consensus algorithm, however our tests use proof-of-authority.

We have also implemented support for Quorum[1], an "Enterprise-focused" fork of Ethereum with different protocols designed for a private environment. Quorum provides users with private transactions as well a permissioned mode for more network control. Most importantly Quorum uses RAFT consensus for fault-tolerance which boasts higher performance than traditional Ethereum consensus. Quorum conveniently uses the same smart contract language as Ethereum, Solidity, which allowed us to use the contracts coded for Ethereum again.

Although these blockchains are not as guilty for boasting large performance

[1]https://www.geeksforgeeks.org/quorum-blockchain/

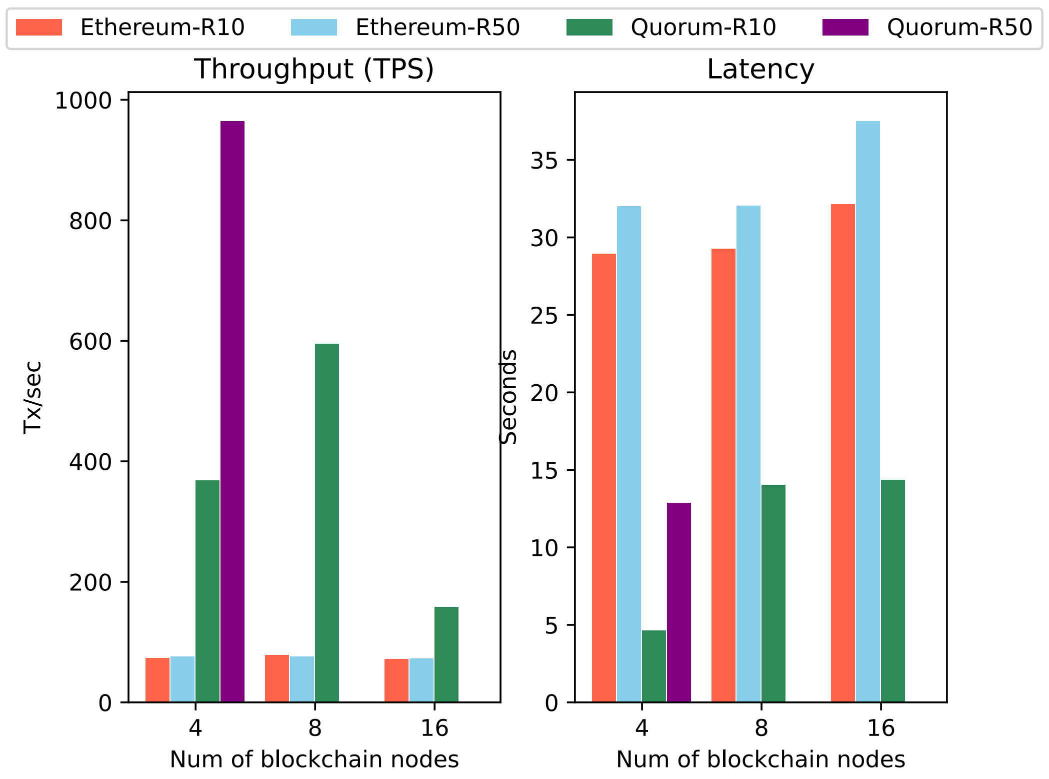

Figure 5.1: Token Exchange Workload Experiment Results

claims as other discussed blockchains, they were faster to develop and allowed us to run experiments faster. Figure 5.1 shows the results of running Backbench v3's Token Exchange workload Ethereum and Quorum. The results provided show the abilities of the benchmark driver and serve as a proof-of-concept for the BBSF and Blockbench v3.

For each blockchain, the experiment was run at two different sizes represented by R-10 and R-50. These numbers represent how many transactions each worker client received. For each experiment, we measured the throughput and latency with 4,8, and 16 nodes, with a finality of one block. While the wider blockchain industry will require more nodes to be used in testing for the results to be accepted, we can still draw some conclusions at these lower counts.

Starting with the R-10 experiments we can see that Quorum is significantly faster than Ethereum. Quorum has higher throughput at all node amounts and far less latency. Most blockchain companies would chose their highest throughput and publish this information without looking at the larger picture presented by the graphs. While Quorum does have better numbers, their trends are worse. Using the scalability macro metric, we can see that Ethereum appears to scale far better than Quorum (the actual macro metric draws curve fits with more amounts of nodes, drawing trends from only 3 data points is not very reliable). When increasing the size of the network, Ethereum's throughput and latency remain constant, implying that a fully deployed blockchain would be able to handle this workload. Quorum has higher throughput and lower latency, but the scalability macro metric shows that they scale much worse than Ethereum. As more nodes are added to the network, the throughput quickly falls off and the latency greatly increases.

The results of the R-50 experiments tell a similar story. Ethereum maintains almost the same performance with 5 times the amount of transactions being called, while Quorum is greatly affected. With a network of only 4 nodes, Quorum is able to reach an average throughput of 964 TPS, but cannot even handle the workload if there are 8 nodes in the network. Ethereum maintains its throughput at 76 TPS, dropping to 72 TPS with 16 nodes. A fully deployed Ethereum blockchain could handle this workload, the scalability metric, showing it should have around 70 TPS, while a Quorum network with only 8 nodes cannot.

These results perfectly exemplify the problems with current blockchain benchmarking. Organizations can pick and choose what results they want to report, without reporting failures. Quorum could report a throughput of 964 TPS, over twelve times faster than Ethereum, without technically lying. Using the BBSF reporting format would ensure that these failures be reported and allow third parties to verify the claims being made. This would show that Quorum does have a higher maximum throughput, but the overall network cannot handle nearly the amount of work that Ethereum can when deployed at scale.

6 Summary of Contribution

The industry of blockchain is not standardized. Without proper standardization and transparency, the results marketed by blockchain companies cannot be compared or verified by independent users. We propose that the BBSF could solve these problems, by creating standardized, relevant, and transparent benchmarks[14]. The BBSF requires extensive descriptions of every aspect of workloads including standardized transaction definitions, transaction mix, arrival distribution, smart contract implementations, and micro-metric definitions. The BBSF provides a framework for metric structure that provides easy to compare, relevant, macro metrics that are generated and supported through the micro-metrics of many experiments. The BBSF provides a driver to execute and measure the workloads within the benchmark in a standardized fashion, ensuring no blockchain has any advantage over another. The reporting format proposed by the BBSF requires reporting of all aspects of the runtime environment, workload parameters, and the micro metrics used to support each macro metric. This transparency will allow for better comparison between results and promote third party verification of results.

Our implementation of this framework, Blockbench v3, is proposed benchmark for web3 applications on a layer 1 blockchain. This benchmark serves as both a relevant benchmark for layer 1 blockchains, but also as a proof of concept for the BBSF as a whole. Blockbench v3 proposes four relevant and different workloads with a variety of transaction types, transaction mixes, arrival distributions, and micro metrics. Blockbench v3 also proposes two macro metrics for easier comparison between blockchains that represent far more than the result of a single experiment. By generating relevant, comparable, and transparent results we hope to prove the validity of the BBSF as the new standard framework.

7 Future Work

This project is alive and continuing development after my graduation. The team I worked with, as well as new students are finishing the benchmarking we are pursuing currently and will likely expand the benchmark into new areas.

7.1 Short Term

The short term future of this project is focused on getting more numerical performance results. This will require the full set of workloads and metrics to be developed for each of the current set of blockchains that we are evaluating as well as creating the full set of workloads for more blockchains. More results will allow for more comparison, creating a network effect that increases the utility of each blockchain benchmarked. Currently we are focused on benchmarking Solana and Aptos.

Solana

Founded in 2017, Solana provides a similar platform to Ethereum in terms of computing capabilities but with a different approach to consensus. Solana supports the same smart contract and token functionality that Ethereum does using a language called Serum that closely resembles Ethereum's Solidity. Solana uses a proof of stake system similar to the one explained in the Ethereum section but utilizes a tool Solana calls "proof of history" to reduce the time to validate transaction ordering.

One problem Ethereum faces in transaction ordering stems from decentralized time. As the network is decentralized, there is no guarantee that different nodes are running with perfectly synchronized clocks. When a block is published, the timestamp used is created based on the clock of the node that created the block. In a decentralized setting, timestamps may occur at slightly different times due to the nodes being slightly out of sync. This sync difference comes from microsecond delays from processing transactions and passing this information to the rest of the network, as well as the possibility of the computers simply having a different set time. The Ethereum "Yellow Paper" only requires that a given block's timestamp comes after the previous block's timestamp, while popular protocol implementations Geth and Parity also require the timestamp to be within 15 seconds of the local node time [5]. These requirements have low impact on the slower Ethereum chain with blocktimes of around 12 seconds [Figure 2.1], but in a much faster system, millisecond sync differences can cause major problems for block ordering and transaction validation.

Solana uses proof of history[1] to build the timestamps into the blockchain itself. When performing a transaction, Solana appends the hash of the current state with all previous states. The state, inputs, and a count are then published on chain setting an upper bound on the time a transaction occurred. There is also a lower bound, the upper bound of the previous state, which can be verified through the previous hashes. This proof of history provides accurate times as well as all previous states for the Solana blockchain, allowing any user to validate the entire blockchain with a small amount of information. Proof of history allows Solana to achieve much faster performance than Ethereum. Solana claims it can support 400,000 TPS, a large step up from Ethereum's 25 [Table 2.1]. In addition to benchmarking different consensus algorithms clarification on the validity of Solana's claims through a third party, rather than the Solana organization, is a priority.

Aptos

In 2022, Meta abandoned their cryptocurrency project Diem (formerly known as Libra). The project was intended to be a low fee stablecoin to be used worldwide. While the idea of a worldwide stablecoin is good, many people were worried about

[1]https://solana.com/news/proof-of-history

giving Facebook this power. Throughout its development the tech side of the project appeared sound, while the fear was in the ethics behind a private company running the world's money supply. When the project was abandoned, the developers at Diem founded Aptos as a startup using the original code as a base.

Aptos claims to support 160k transactions using a parallel execution engine [4]. The Aptos team designed a blockchain specific version of the Software Transactional Memory (STM) library called Block-STM. STM is an approach to parallel transaction processing that processes all transactions in parallel while keeping track of memory accesses. After processing all transactions, transactions that have memory conflicts are aborted and re-executed [4]. This approach was adapted for blockchain use. As transactions are uploaded in blocks, transactions do not need to be committed individually and allows a lazy commit system to be implemented that avoids synchronization. The Aptos programming language, Move, runs on a virtual machine which also helps with safe memory access. Lastly, while most STM libraries target non-determinism, Block-STM uses a fixed ordering of transactions which allows for conflicts to be resolved easier and allow overall throughput to increase.

7.2 Long Term

After benchmarking enough chains to prove the effectiveness of Blockbench v3, there are other ideas that can be explored. While the workloads presented by Blockbench v3 measure metrics that are relevant to the usage of blockchains today, the structure of the layer 1 blockchain is evolving to be faster, more scalable, and easier to validate. Two intriguing areas to explore are layer 2 blockchains, and the sharding of layer 1 blockchains.

Layer 2

Layer 2 blockchains[2] are blockchains that run on top of layer 1 blockchains. Layer 2 blockchains use their own code and tokens to perform execution that a layer 1 blockchain may not support. Periodically they bundle the recent activity on their chain and publish this bundle on the layer 1 blockchain. By doing so, the layer 2 chain can use the robust security provided by the layer 1 chain while running code that the layer 1 chain may not support.

The primary support for layer 2 chains beyond their wider code functionality is the scalability that they bring to blockchain. Bringing transactions to layer 2 removes transactions from the layer 1 chain that may be congesting the system and slowing the performance. These transactions are periodically bundled together and recorded on the layer 1 chain to take advantage of the layer 1 chains security and decentralization. This bundling is referred to as a "rollup"[2]. By bundling these transactions, gas fees are reduced, fewer transactions are on the layer 1 blockchain, and the time to validate transactions greatly decreases. This is because a node validating a block treats the rollup as a single transaction while representing hundreds of transactions. Rollups are treated as either "optimistic" or use a zero-knowledge proofs to verify the validity of the transactions in the rollup. Optimistic rollups allow the node to assume that every transaction within the rollup is valid. The rollup can be challenged if needed, and a more rigorous process is performed to verify the transactions. Some rollups contain zero-knowledge proofs that prove the validity of the transactions contained in the rollup. For both types, the time to validate hundreds of transactions is greatly reduced, increasing the performance of the layer 1 chain.

Benchmarking layer-2 blockchains is both a complex coding challenge, but also

[2]https://ethereum.org/en/layer-2/

an algorithmically difficult one. There are many different metrics that can be explored and many different ways to approach benchmarking. The first metric to explore is the increase in the performance of the layer-1 chain by using a layer-2 chain. A workload could be run on layer-1 chain A using two different layer-2 blockchains B and C. The performance of A could be measured with no layer-2 chain, with layer-2 chain B, and layer 2 chain C. The change in performance by adding the different layer-2 chains could be compared to see which chain is “better”.

The metrics to measure performance will be quite different in this environment than in our current implementation. The main feature provided by layer 2 is the quicker validation of blocks due to rollup. Our current metrics measure the throughput and latency of the blockchain, but do not inspect the time for individual nodes to validate specific blocks. This process becomes more complicated a user wants to compare two layer-2 blockchains that run atop different layer-1 blockchains. Layer-2 chain B may appear worse than layer-2 chain C, but that may only be a result of the performance and interoperability of their underlying layer-1 chains. While ratios between the performance of the underlying chains may allow some comparison between the layer-2 chains, it will be important to find an equitable and standardized approach.

Sharding

Another interesting area to benchmark is sharded blockchains. Sharding[3] refers to the act of splitting a blockchain into multiple different parallel blockchains. By splitting the blockchain, transactions on one shard are only processed by nodes running that shard, rather than processing every transaction the way they do in a normal blockchain. This allows the overall blockchain to process more transactions

[3]https://education.district0x.io/general-topics/understanding-ethereum/ethereum-sharding-explained/

and may be a sustainable approach to scaling the blockchain's throughput. Each shard handles its own transactions and provide updates to the main blockchain when needed. Metrics for a sharded blockchain would look similar to a layer 1 blockchain, however they would represent additional features rather than just the consensus algorithm. A uniform distribution of accounts and data between shards will allow for better concurrency, whereas having shards with disproportionately more accounts and data negates the effectiveness of the sharding.

Benchmarking sharded blockchains is also more complicated than our current implementation. Transactions that stay within a shard are simple and fast, while transactions that cross between two shards require more complex protocols. If a blockchain organization were to publish performance metrics about their sharded blockchain without revealing the mix of cross-shard transactions users would not know whether the workload represents an actual sharded blockchain or if the results are just many blockchains in parallel. Finding a relevant standard for the amount of cross-shard transactions would be paramount to benchmarking sharded blockchains.

In addition to cross-shard transactions, the fault tolerance of a sharded system is very important. Normal blockchains face "51%" attacks in which an attacker controls 51% of the network hash rate and act nefariously with a controlling share of the network's votes. "In a 100 shards system, it only takes 1% of the network hash rate to dominate a shard" [7]. Building a robust system to deal with "1%" attacks is important for a sharded blockchain to succeed. Benchmarking this robustness through a metric similar to Blockbench v3's fault tolerance metric would require a different approach than the current implementation but would provide important results.

Bibliography

[1] https://github.com/KunPengRen/blockbench-3.0.

[2] Hyperledger caliper. Web document, 2023. https://hyperledger.github.io/caliper/.

[3] Dinh Tien Tuan Anh, Rui Liu, Meihui Zhang, Gang Chen, Beng Chin Ooi, and Ji Wang. Untangling blockchain: A data processing view of blockchain systems. *IEEE Transactions on Knowledge and Data Engineering*, 30(7):1366–1385, July 2018.

[4] Aptos. Block-stm: How we execute over 160k transactions per second on the aptos blockchain, Jun 2022.

[5] Rob Behnke. What is timestamp dependence?, 2023. https://www.halborn.com/blog/post/what-is-timestamp-dependence.

[6] Jing Chen, Sergey Gorbunov, Silvio Micali, and Georgios Vlachos. Algorand agreement: Super fast and partition resilient byzantine agreement. Cryptology ePrint Archive, Paper 2018/377, 2018. https://eprint.iacr.org/2018/377.

[7] Jordan Clifford. Crossing shards, Jul 2019.

[8] Tien Tuan Anh Dinh, Ji Wang, Gang Chen, Rui Liu, Beng Chin Ooi, and Kian-Lee Tan. Blockbench: A framework for analyzing private blockchains. In *Proc. ACM SIGMOD Conference on the Management of Data*, pages 1085–1100, 2017.

[9] Produced by Digital Editors. 2022 tesla model s plaid records unreal 0-60 mph time. Web document, Jul 2021. https://www.motorbiscuit.com/2022-tesla-model-s-plaid-records-unreal-0-60-mph-time/.

[10] Vincent Gramoli, Rachid Guerraoui, Andrei Lebedev, Chris Natoli, and Gauthier Voron. Diablo-v2: A benchmark for blockchain systems. page 14, 2022.

[11] Jim Gray and Andreas Reuter. *Transaction Processing: Concepts and Techniques*. Morgan Kaufmann, 1993.

[12] Hayden Adams and Noah Zinsmeister and Dan Robinson. Uniswap v2 core. Web document, 2020. https://blog.uniswap.org/whitepaper.pdf/.

[13] Mark Kane. Motortrend proves tesla can't truly do 0-60 in under 2 seconds, Jun 2021. https://insideevs.com/news/514979/tesla-cant-60mph-under-2seconds/.

[14] Ren Kunpeng and Jefferson Van Buskirk. Bbsf: Blockchain benchmarking standardized framework, 2023.

[15] Dumitrel Loghin, Tien Tuan Anh Dinh, Aung Maw, Gang Chen, Yong Meng Teo, and Beng Chin Ooi. Blockchain goes green? part II: characterizing the performance and cost of blockchains on the cloud and at the edge. *CoRR*, abs/2205.06941, 2022.

[16] Omid Malekan. *The Story of Blockchain*. Triple Smoke Stack, 2018.

[17] Mateusz Raczynkski. What is the fastest blockchain and why? analysis of 43 blockchains. Web document, 2021. https://alephzero.org/blog/what-is-the-fastest-blockchain-and-why-analysis-of-43-blockchains/.

[18] Bhabendu Kumar Mohanta, Soumyashree S Panda, and Debasish Jena. An overview of smart contract and use cases in blockchain technology. In *2018 9th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, pages 1–4, 2018.

[19] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. Web document, 2008. `https://bitcoin.org/bitcoin.pdf`.

[20] Bulat Nasrulin, Martijn De Vos, Georgy Ishmaev, and Johan Pouwelse. Gromit: Benchmarking the performance and scalability of blockchain systems, 2022.

[21] Crypto Research. The time to finality for solana, Sep 2022. `https://cryptoresearch.report/crypto-research/the-time-to-finality-for-solana/`.

[22] Pingcheng Ruan, Tien Tuan Anh Dinh, Dumitrel Loghin, Meihui Zhang, and Gang Chen. *Blockchains: Decentralized and Verifiable Data Systems*. Springer Nature, 2023.

[23] Dimitri Saingre, Thomas Ledoux, and Jean-Marc Menaud. Bctmark: a framework for benchmarking blockchain technologies. In *2020 IEEE/ACS 17th International Conference on Computer Systems and Applications (AICCSA)*, pages 1–8, 2020.

[24] Jóakim v. Kistowski, Jeremy A. Arnold, Karl Huppler, Klaus-Dieter Lange, John L. Henning, and Paul Cao. How to build a benchmark. In *Proceedings of the 6th ACM/SPEC International Conference on Performance Engineering*, ICPE '15, page 333–336, New York, NY, USA, 2015. Association for Computing Machinery.

Claims Table Sources

1. `etherscan.io/charts`
2. `decrypt.co/34204/ethereum-2-0-will-walk-and-roll-for-two-years-before-it-can-run`
3. `ethereum.org/en/developers/docs/blocks/`

4. cryptoresearch.report/crypto-research/the-time-to-finality-for-solana/

5. solana.com/

6. academy.binance.com/en/glossary/finality

7. cryptonews.com/news/celo-to-be-fastest-evm-chain-by-end-of-2022-co-founder-says.htm

8. ethereum.org/ph/roadmap/single-slot-finality/

9. coindesk.com/tech/2023/02/14/bnb-chain-aims-to-double-transaction-speed-targets-zk-tooling-in-2023-roadmap/

10. blog.celo.org/consensus-and-proof-of-stake-in-the-celo-protocol-3ff8eee331f6

11. developer.algorand.org/docs/get-started/basics/why_algorand/

12. https://algorand.com/resources/algorand-announcements/algorand-2021-performance

13. https://www.lumenauts.com/blog/how-many-transactions-per-second-can-stellar-process

14. https://blog.cryptostars.is/how-stellar-solves-the-blockchain-finality-challenge-b5678ebebd9d

Vita

Jefferson Van Buskirk grew up in Hingham, Massachusetts with his parents Peter and Patricia Van Buskirk. Jefferson attended Lehigh University from 2018 to 2022 where he studied Computer Science and Business, graduating with highest honors. in 2022, Jefferson was awarded the Gulden Memorial Award for excellence in algorithms and received the Lehigh University Presidential Scholarship. Jefferson then attended Lehigh University for his masters as a Presidential Scholar, graduating in May 2023. During his masters, Jefferson was a grader for "Blockchain Concepts and Applications" and "Blockchain Algorithms and Systems". In addition to his grading, Jefferson was a part of the Scalable Systems and Software research group as a student of Professor Hank Korth. Following graudate school, Jefferson is beginning work at FAST Enterprises, as a data engineer.