BBSF: Blockchain Benchmarking Standardized Framework

Kunpeng Ren
dcsrenk@nus.edu.sg
National University of Singapore
Singapore

Jefferson F.B.Van Buskirk
jef@lehigh.edu
Lehigh University
Bethlehem, PA, USA

Miguel Monares
mmonares@ucsd.edu
UC San Diego
La Jolla, CA, USA

Zheng Yong Ang
zhengyong@u.nus.edu
National University of Singapore
Singapore

Shizheng Hou
housz@nus.edu.sg
National University of Singapore
Singapore

Nathaniel R. Cable
nrc324@lehigh.edu
Lehigh University
Bethlehem, PA, USA

Henry F. Korth
hfk2@lehigh.edu
Lehigh University
Bethlehem, PA, USA

Beng Chin Ooi
ooibc@comp.nus.edu.sg
National University of Singapore
Singapore

ABSTRACT

We propose a framework for standardized, transparent, and fair benchmarks for blockchains, the Blockchain Benchmark Standardized Format (BBSF). BBSF enables blockchain users and developers to compare platforms using metrics derived from realistic workloads. We outline the challenges in developing a blockchain benchmark with this degree of breadth and flexibility. We contrast results using our approach with prior benchmark implementations and show why BBSF generates results that are more verifiable than prior published benchmark data. We present an implementation of our framework, Blockbench v3, a benchmark with web3-application-focused workloads, primarily to be used by layer-1 blockchains. Blockbench v3 serves as a test case for our framework's effectiveness as part of ongoing work in the characterization of blockchain performance.

CCS CONCEPTS

• **Networks** → *Network performance modeling*; *Network performance analysis*; • **Information systems** → *Distributed database transactions.*

KEYWORDS

blockchain, benchmarking, standardization, verifiable, performance

ACM Reference Format:
Kunpeng Ren, Jefferson F.B.Van Buskirk, Miguel Monares, Zheng Yong Ang, Shizheng Hou, Nathaniel R. Cable, Henry F. Korth, and Beng Chin Ooi. 2023. BBSF: Blockchain Benchmarking Standardized Framework. In *Proceedings of June 23, 2023 (VDBS).* ACM, New York, NY, USA, 9 pages. https://doi.org/XXXXXXX.XXXXXXX

VDBS, June 23, 2023, Seattle, WA

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Since the introduction of blockchain with the proposal of Bitcoin in 2008 [13], 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 [12]. 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. Our proposed framework seeks to meet the need for a reliable means of benchmarking and comparing the performance of blockchains.

1.1 Blockchain Technology Overview

We assume the reader has a basic background in the basic concepts underlying blockchain technology including the use of hashes and digital signatures to achieve immutability and irrefutability. We also assume 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 [10].

The type of code that can run on a blockchain is dependent on the features that 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.

1.2 Benchmarking Overview

A proper benchmark provides quantifiable metrics that can be used to compare systems against each other, identify strengths and weaknesses of a system, and ensure that a system is applicable to a given project. Kistowski, et al. [19] defines 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

The database community has a robust and mature set of methods of 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 [6]. TPC has a variety of benchmarks that are used by major companies like IBM, 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.

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 [6]. Atomicity refers to executing all of a transaction or none 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, 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.

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. Some blockchains, for example, Algorand [3], 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.

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 a chain. Blockbench, a blockchain benchmark from 2017, benchmarks their workloads with blockchains ranging from 4-32 nodes [4].

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

[1] tpc.org

[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.

hardware malfunction or software bugs. In a blockchain scenario, there is 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 in 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.

1.3 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.

Chain foundations, supporters, and others often cite figures presuming their proposed transactional models and workloads as the "correct" ones. Contrast that with enterprise-focused benchmarking typified in the DB world 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 [11] illustrates the lack of clarity in terms of definitions and workload.

While many blockchain organizations may measure the same metric, the environments in which they are performing in may differ. The environments may differ in hardware, number of nodes, and percentage of nefarious nodes, resulting in incomparable results. A proper benchmark must specify 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 workloads that explores the strengths and weaknesses of blockchains robustly so that each chain is tested not only on its strong points but also areas of weakness.

1.4 Related Work

There are few available blockchain benchmarking solutions, and many 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 [2, 4, 8]. 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 [1] 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 [14], which uses fixed asset transfer as its workload in its evaluation of blockchains' performance and scalability. BCTMark [18], 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 [5] 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.

A 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.

1.5 Call for a New Framework

With these problems in mind, we propose a framework to standardize the benchmarking process for the blockchain industry. The goal of this framework is to create benchmarks that relate to realistic decentralized application use cases. The existence of such benchmarks can motivate blockchain system design not only to focus on user needs but to measure their success in that regard using independent metrics that are not biased towards a specific architecture. A standardized benchmark addresses the issues presented by in-house benchmarking and will push the industry towards more transparent, comparable results. This paper outlines our proposed framework, the Blockchain Benchmark Standardized Framework, a framework that requires explicit definition of all aspects of the benchmark to standardize the results produced. In addition to this framework this paper outlines our preliminary implementation of this framework, Blockbench v3, a benchmark focused on web3 applications on modern blockchains.

1.6 Overview of the Paper

In Section 2, we present the features of our new framework for blockchain benchmarking. These include manner in which to describe a workload, set of metrics measured by the framework, the benchmark driver that runs the actual benchmark, and a standout result-report format. Section 3 describes the 4 workloads in our Blockbench v3 instantiation of our framework: a token exchange, an NFT marketplace, NFT minting, and a sports-betting application. While many experiments remain a work-in-progress, we report results in Section 4 for Etherum and Quorum. As we note in Section 5, these initial results serve primarily as a demonstration of our framework and its operation. We plan to study additional blockchains and hope that others will use BBSF to do even more studies. A large set of experimental results will enable future research (about which we elaborate in Section 6) in comparing the suitability of various design features of blockchain systems for optimal performance in a variety of applications. That, in turn, can help blockchain systems designers design for practical performance and help developers choose wisely among available platforms.

2 BLOCKCHAIN BENCHMARK STANDARDIZED FRAMEWORK

The Blockchain Benchmark Standardized Framework (BBSF) provides standardization across all aspects of blockchain benchmarking.

(1) A standardized workload framework that contains explicit definitions of all aspects of a workload ensuring proper implementation across all blockchains.
(2) Standardized micro metrics that are aggregated into a set of macro metrics that are easily comparable among blockchains.
(3) A standard driver that interfaces with a fully deployed blockchain, calls the transactions in a standardized fashion, and uses standardized methods for measuring metrics.
(4) A standardized reporting format that ensures that all metrics measured are transparently communicated in an easily comparable format.

2.1 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 transactions 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 2.3) 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 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[3] 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.

2.2 Standardized Metric Framework

In addition to standardizing the transactions being called, the BBSF has a standardized framework for the metrics being measured. Each workload has specific micro metrics used to compose macro metrics by changing parameters of the blockchain being measured. Changing blockchain parameters and plotting the changes in micro metrics leads to 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 onto 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.

2.3 Standardized 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, as to not have driver performance impact blockchain performance. The main goals for the driver are that it not affect the performance of the blockchain, semi-idempotent execution 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.

2.3.1 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 allows the driver to scale infinitely and eventually call as many transactions as the blockchain can handle.

2.3.2 Functionality. The driver takes inputs of the workload contract, a wallet list, transaction list, and 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 with the sending timestamp. Adding these differences together and dividing by the total transactions aggregates the average latency.

2.3.3 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

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

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 the 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 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.

2.4 Standardized Reporting Format

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. Due to length constraints, the format is not fully outlined in this paper although it does contain all of the parameters to set up the same driver and blockchain environments, the hardware used, each specific micro metric measured, and the macro metrics extrapolated from these micro metrics.

3 BLOCKBENCH V3

Our implementation of the BBSF, Blockbench v3, is a benchmark focused around web3 applications running on layer-1 blockchains. The workloads proposed 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.

3.1 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.

3.1.1 Token Exchange. The first Blockbench v3 workload is a decentralized token exchange roughly based on Uniswap-V2 [7], 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.

3.1.2 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 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 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.

3.1.3 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)[4]. 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.

3.1.4 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.

3.2 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 moves 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.

3.2.1 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

3.2.2 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 on 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[5], however, responses from more nodes are needed to reach consensus. Users can use the results provided by scalability so see how two blockchains would perform at the same size, despite the deployed, live, chains being different sizes.

3.2.3 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 measure 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[6] 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.

[4] 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

[5] 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 in normally fixed independent of the number of nodes.

[6] 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.

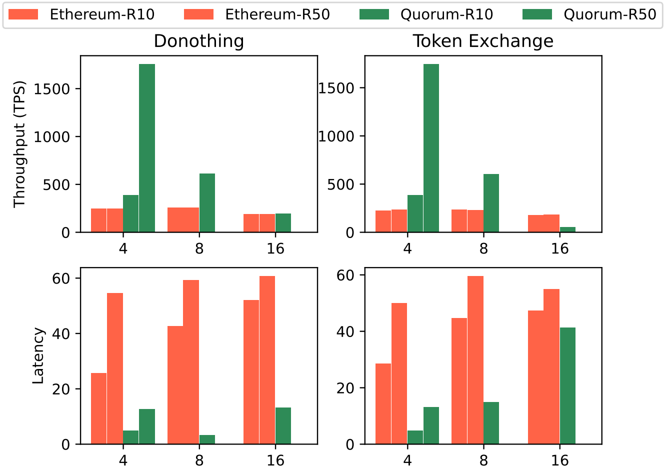

Figure 1: Evaluation results of two workloads on Ethereum and Quorum with different request rate and blockchain nodes

4 EXPERIMENTS

We report here preliminary results for two blockchains that we have benchmarked to prove the viability of the BBSF framework: Ethereum and Quorum. The experiments were run on a powerful machine with 2 AMD EPYC Milan 7513 (64 cores, 128 threads) and 1 TB RAM. Ethereum is using Clique for proof of authority consensus while Quorum is using IBFT. Both Ethereum and Quorum use the same block gas limit configuration of 30M and block period of 5 seconds.

Figure 1 presents the results of our evaluation of two workloads, Donothing and Token Exchange. Donothing represents empty contract calls and was used throughout the driver testing. Token Exchange is the workload outlined in 3.1.1. The evaluation was conducted by employing two different request rates, 10 and 50 tx/s (R10/R50) for each thread on each worker, using 10 threads per worker.

In the case of a 4-node network, our results indicate that Quorum outperforms Ethereum with a peak throughput of approximately 1700 TPS for both workloads and a small latency of around 12 seconds. Conversely, Ethereum's throughput is limited to 250 TPS due to the block gas limit. Notably, we also observed that the block gas limit does not apply to Quorum, allowing for the production of larger blocks when receiving high request rate workloads. Furthermore, our latency evaluation shows that Quorum exhibits faster response times for low-rate workloads, while Ethereum takes longer response times.

We also tested the performance of the two blockchains with 8 and 16 nodes. Our results show that Ethereum's performance slightly drops with increasing nodes, while Quorum's performance decreases significantly. Additionally, when the request rate is set to 50 with 8 and 16 nodes, Quorum only produces blocks with empty transactions, which prevented us from providing certain results. While our experiments can be used to create the scalability macro metric, we only have 3 sizes that are minuscule compared to a deployed chain. As we perform larger scale experiments using more blockchains and micro metrics, resulting in more data points, these macro metrics will help encapsulate these results.

5 DISCUSSION: LAYER-1 AND BEYOND

We have motivated the development of BBSF based on the need to compare layer-1 blockchains and the prevalence of irreproducible performance claims from advocates of specific blockchains. A wider domain of performance competition is emerging now at layer-2. In principle, our framework and workloads apply equally well at layer-2, but it will be interesting to consider designing specific workloads that gain insight into the performance of the alternative transaction aggregation approaches taken at layer-2. Among the issues faced in analyzing layer-2 are issues in the definition of finality when we compare optimistic rollups such as Optimism [16] and Arbitrum [15] with zero-knowledge(ZK) rollups such as Polygon ZK-EVM [17], Matter Labs' zkSync [20], and Loopring [9]. Layer-2 solutions compete on performance, security, and cost. That latter point, cost, is one we have not considered, and a difficult one to benchmark in a dynamic economic system.

6 FUTURE WORK

Although this paper focused on the framework itself, our medium-term goals are twofold: doing an extensive set of benchmarking experiments and encouraging others to use BBSF for their experiments. Our initial results show the feasibility of using BBSF. The long-term goal is to assemble a rich set of comparable benchmarking results and use those to gain insight into blockchain system design.

In reaching our long-term goal, we expect to refine the framework, but hope quickly to reach a point where the framework is stable and focus all work into new workloads using the existing framework. Ultimately, we aim to enable the broad blockchain community to be able to make informed, reliable performance comparisons based on independently defined standards.

As we noted in Section 5, the need for independent and transparent benchmarking in blockchain is growing and the domain over which testing needs to be done expanding beyond layer-1 to layer-2 and the burgeoning ZK-EVM ecosystem.

ACKNOWLEDGMENTS

This work at Lehigh is supported by a gift from Steel Perlot. The research group from NUS is supported by the National Research Foundation, Singapore under its Emerging Areas Research Projects (EARP) Funding Initiative.

REFERENCES

[1] 2023. Hyperledger Caliper. Web document. https://hyperledger.github.io/caliper/.

[2] Dinh Tien Tuan Anh, Rui Liu, Meihui Zhang, Gang Chen, Beng Chin Ooi, and Ji Wang. 2018. Untangling Blockchain: A Data Processing View of Blockchain Systems. *IEEE Transactions on Knowledge and Data Engineering* 30, 7 (July 2018), 1366–1385.

[3] Jing Chen, Sergey Gorbunov, Silvio Micali, and Georgios Vlachos. 2018. ALGORAND AGREEMENT: Super Fast and Partition Resilient Byzantine Agreement. Cryptology ePrint Archive, Paper 2018/377. https://eprint.iacr.org/2018/377 https://eprint.iacr.org/2018/377.

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

[5] Vincent Gramoli, Rachid Guerraoui, Andrei Lebedev, Chris Natoli, and Gauthier Voron. 2022. Diablo-v2: A Benchmark for Blockchain Systems. (2022), 14. http://infoscience.epfl.ch/record/294268

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

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

[8] Dumitrel Loghin, Tien Tuan Anh Dinh, Aung Maw, Gang Chen, Yong Meng Teo, and Beng Chin Ooi. 2022. Blockchain Goes Green? Part II: Characterizing the Performance and Cost of Blockchains on the Cloud and at the Edge. *CoRR* abs/2205.06941 (2022). https://doi.org/10.48550/arXiv.2205.06941 arXiv:2205.06941

[9] Loopring. 2023. https://loopring.org/#/protocol

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

[11] Mateusz Raczynkski. 2021. What Is The Fastest Blockchain And Why? Analysis of 43 Blockchains. Web document.. https://alephzero.org/blog/what-is-the-fastest-blockchain-and-why-analysis-of-43-blockchains/.

[12] Bhabendu Kumar Mohanta, Soumyashree S Panda, and Debasish Jena. 2018. An Overview of Smart Contract and Use Cases in Blockchain Technology. In *2018 9th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*. 1–4. https://doi.org/10.1109/ICCCNT.2018.8494045

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

[14] Bulat Nasrulin, Martijn De Vos, Georgy Ishmaev, and Johan Pouwelse. 2022. Gromit: Benchmarking the Performance and Scalability of Blockchain Systems. arXiv:2208.11254 [cs.DC]

[15] OffchainLabs. 2023. https://offchainlabs.com

[16] Optimism. 2023. https://www.optimism.io

[17] Polygon. 2023. https://wiki.polygon.technology/docs/zkEVM/introduction

[18] Dimitri Saingre, Thomas Ledoux, and Jean-Marc Menaud. 2020. BCTMark: a Framework for Benchmarking Blockchain Technologies. In *2020 IEEE/ACS 17th International Conference on Computer Systems and Applications (AICCSA)*. 1–8. https://doi.org/10.1109/AICCSA50499.2020.9316536

[19] Jóakim v. Kistowski, Jeremy A. Arnold, Karl Huppler, Klaus-Dieter Lange, John L. Henning, and Paul Cao. 2015. How to Build a Benchmark. In *Proceedings of the 6th ACM/SPEC International Conference on Performance Engineering* (Austin, Texas, USA) *(ICPE '15)*. Association for Computing Machinery, New York, NY, USA, 333–336. https://doi.org/10.1145/2668930.2688819

[20] ZKsync. 2023. https://docs.zksync.io/userdocs/intro

Received 24 March 2023; revised n/a; accepted n/a