The Performance of Byzantine Fault Tolerant Blockchains

Gary Shapiro
University of Sydney
Sydney, Australia
gsha5095@uni.sydney.edu.au

Christopher Natoli
University of Sydney
Sydney, Australia
christopher.natoli@sydney.edu.au

Vincent Gramoli
University of Sydney and EPFL
Lausanne, Switzerland
vincent.gramoli@epfl.ch

Abstract—**Blockchains have captured the attention of many, resulting in an abundance of new systems available for use. However, selecting an appropriate blockchain for an application is challenging due to the lack of comparative information discussing core metrics such as throughput, latency and scalability. Although a number of efforts have been devoted to performance evaluation, there is limited work dedicated to blockchains that are both efficient, due to avoiding complex Proof-of-Work cryptopuzzles, and secure, because they solve consensus deterministically despite Byzantine failures. In this paper, we evaluate the performance of three permissioned blockchains that cope with such malicious behavior, namely Burrow, Quorum and Red Belly Blockchain. To this end, we modified the Hyperledger Caliper benchmark to solve three main limitations: unnecessary overheads, offloading cryptographic signatures and multiplying clients. Our results identify the maximum send rate that Burrow and Quorum can process, and that Red Belly Blockchain can offer an 8-times higher throughput than the other blockchains.**

Index Terms—**Benchmark, Byzantine fault tolerance, performance**

I. Introduction

The plethora of blockchain proposals and their evaluation in isolation of one another raised questions on the suitability of a particular proposal for a targeted application. As of today, there are approximately 1045 blockchains available[1] but very little information on their benefits in terms of throughput, latency and scalability. When this information is provided [1], [2], [3], it often results from tests run by the developers of the blockchain, in isolation of other blockchains, and within a very specific experimental environment.

As a result, the open source community [4] as well as the database community [5] have started designing benchmarks that would allow the performance comparison of various blockchains on the same grounds. The former attempt targets only blockchains offering smart contracts and prevents comparison against transaction-based blockchains (e.g., Bitcoin [6]). The latter attempt compares blockchains that are made inherently slow to cope with potentially malicious permissionless users (e.g., Ethereum [7]) to blockchains that cannot tolerate malicious participants (e.g., Hyperledger Fabric [8]).

Despite these preliminary efforts, to our knowledge there is no thorough evaluation of secure blockchain systems. In this paper, we tackle this problem by thoroughly evaluating the performance of three blockchain systems that are secure in that they tolerate Byzantine failures, namely Burrow [9], Quorum [10] and Red Belly Blockchain [11]. These blockchain systems were precisely designed to either support financial workloads, offer scalability or be efficient. Interestingly, evaluating the performance of blockchains led us to identify and remedy three major drawbacks that have plagued previous evaluations of blockchain systems:

1) **Distributed workload generation.** Some benchmarks have been designed in a centralized fashion, having a single client machine, potentially running multiple concurrent client software, to send requests to the blockchain nodes [12]. As blockchains are distributed systems that aim at scaling to large numbers of participating nodes, it is crucial to generate a workload sufficiently high to stress test these numerous blockchain nodes. Without provisioning enough physical client resources, the risk is that an experiment may report performance that is misleadingly capped by the client resources rather than representing the true capacity of the blockchain itself.
2) **Limiting misleading overheads.** For the sake of simplifying deployment and monitoring, several blockchain systems come with packaging options that limit their access to physical resources. Caliper [4] requires a docker container for ease of deployment[2], yet containers are known to induce unnecessary overheads [13]. Distributed synchronization services help monitoring blockchains, like Zookeeper, but these services are known to bottleneck at the leader network interface, hence offering results that could be misleading [14]. It is thus crucial to limit the effects of these levels of indirections on the observed performance.
3) **Offline cryptographic preprocessing.** The cryptographic functions to guarantee the authentication in a blockchain system were found particularly CPU-intensive at various occasions [11]. Part of these CPU-intensive tasks are produced on the client side: the benchmark should sign sufficiently many transactions to generate a workload. Benchmarks that measure the time needed to produce

[1]Number of coins in existence according to https://coinmarketcap.com/coins/views/all/.

[2]https://github.com/hyperledger/caliper/blob/master/packages/caliper-tests-integration/ethereum_tests/networkconfig.json.

this CPU-intensive workload, like [4], return a biased throughput that does not represent the capacity of the blockchain system but that is limited by the time it took for the benchmark to sign the transactions to be sent.

Addressing these issues and deploying up to 32 Amazon Web Service (AWS) c4.xlarge virtual machines (VMs) was sufficient to outline important differences between Byzantine fault tolerant (BFT) blockchains. In order to fairly evaluate these blockchain systems, we modified Caliper to avoid the aforementioned limitations. Interestingly, we identified that Burrow would fail at high workloads and Quorum would offer performance almost one order of magnitude lower than Red Belly Blockchain.

We do not pretend that our proposal aims at evaluating all blockchains. First, proof-of-work blockchains can be made arbitrarily slow in order to increase their security. Although proof-of-authority blockchains can be compared to BFT ones, they often require parameterizing the block period that dictates their performance as well. Second, we observed that many BFT blockchains are not stable and at too early stages to be evaluated. Finally, the performance evaluation of smart contracts is out of the scope of this paper and left to future work.

II. Background and Related Work

Blockchain benchmarks are often developed and tailored specifically for one blockchain. It thus makes it hard to compare the performance of blockchains as they are usually obtained in different benchmarks. To solve this issue, a number of benchmarking frameworks have been developed to compare different blockchains or its consensus component on the same ground.

a) Caliper: Hyperledger Caliper [4] is a benchmarking framework aimed at measuring blockchain performance through defined use-cases. Currently, Caliper supports Hyperledger's blockchains such as Fabric, Sawtooth, Iroha, Burrow and Besu, but is also planned to support Ethereum. Caliper's pre-defined workloads specify which contract is called, which functions are used and the rate at which the transactions are sent. It outputs success rate, transaction and read throughput, transaction and read latency and resource consumption. At the time of writing, Caliper has minimal support for distributed worker client machines.

b) Blockbench: Blockbench [5] provides a number of workloads ranging from blockchain specific use-cases to more traditional database system benchmarks such as YCSB [15]. Currently, Blockbench compares a crash fault tolerant blockchain (Hyperledger Fabric) to a BFT blockchain (Quorum) and to a proof-of-work blockchain (Ethereum). Our focus is on comparing exclusively BFT blockchains as they are appealing to critical applications.

c) Chainhammer: Chainhammer [12] provides a benchmark framework supporting only Ethereum-compatible blockchains, based on different consensus algorithms (geth clique [16], parity aura [17], and Quorum's [10] RAFT and IBFT). This benchmark provides metrics of transactions per second, block information. Chainhammer was used to experiment blockchains under varying machine sizes, but not to test scalability.

d) BFT-Bench: BFT-Bench [18] measures the performance of various BFT State Machine Replications (SMRs) executing a sequence of consensus instances for distributed replicas to agree on a common state, similar to blockchain nodes agreeing on a block. However, SMRs do not support signature and verification of transaction requests that are necessary in blockchain systems. These verifications are known to significantly impact blockchain performance [11], this is why we take these into account in our evaluation.

e) Simulators: BFTSim [19] simulates the execution of BFT consensus algorithms and run on top of an `ns-2` network simulator. It allows researchers to rapidly test a BFT algorithm after writing it in a declarative specification language and simulates its cryptographic operations. BlockSim [20] aims at simulating different blockchains using a discrete-event model to scale to a large number of nodes. BlockSim was used to simulate Ethereum and Bitcoin and demonstrated that the cost of encrypting communications is costly. Our evaluation does not rely on a specific model and assess the performance a BFT blockchains user can experience in a real environment.

f) Crash fault tolerant distributed ledgers: Hyperledger Fabric [8] and Corda [24] are two permissioned distributed ledger technologies. Unfortunately, there is no full fledged solution of these distributed ledgers offering Byzantine fault tolerance, so we excluded them from our evaluation. Although a research prototype of the orderer of Hyperledger Fabric builds upon BFT-SMaRt and tolerates Byzantine behaviors, the full system of Fabric v1.x has not been designed towards this goal. An initial version of Corda was originally designed to build upon BFT-SMaRt, however, this version is not stable and the developers recommend to use the version based on Raft that cannot tolerate intrusions [25].

g) Blockchains requiring synchrony: Other blockchain systems do not tolerate unexpected message delays. They assume synchrony [26] and an attacker may double spend if some messages get unexpectedly delayed [27]. These blockchains are thus vulnerable in large networks to attacks, natural disasters or human misconfigurations, like in the Internet. Solida [28] assumes synchrony even though it builds upon PBFT [29]. Dfinity [27] is proved under synchrony but executes in asynchronous rounds. Avalanche [30] is analyzed under synchrony but is conjectured to work under partially synchrony. OmniLedger [31] needs synchrony to assign participants to shards.

h) Ethereum for permissioned settings: Even though Ethereum [7] is originally a proof-of-work blockchain, it has been deployed in permissioned environments, like the environment we consider. A key difference between Ethereum and the BFT blockchains we study here, is that Ethereum assumes synchrony [32] whereas the blockchains we consider here simply assume partial synchrony [26], hence the latter blockchains do not need to know the time it will take to deliver a message. Proof-of-authority has been proposed in

TABLE I: Comparison of selected BFT blockchains

Blockchain	Consensus	Fault tolerance	Developer	Goals
Burrow [9]	Tendermint [21]	$n > 3f$	Hyperledger	Simplicity and speed[3]
Quorum [10]	IBFT [22]	$n > 3f$	J.P. Morgan	Financial applications[4]
Red Belly [11]	DBFT [23]	$n > 3f$	U. Sydney and CSIRO	Security and scalability[5]

Ethereum as an alternative to avoid the difficulty parameter of proof-of-work. It was shown to offer a throughput of up to 84 transactions per second [33], however, it requires selecting a block period and it has proven vulnerable in partially synchronous settings [34].

i) Permissioned Blockchain Performance: There have been a number of other works performing analysis on permissioned blockchains. Pongnumkul et al. [35] perform an analysis of Ethereum against Hyperledger Fabric in a permissioned context. The benchmark results utilize smart contract invocations to compare throughput and latency, highlighting that Hyperledger outperforms Ethereum with throughput and latency. Leppelsak [36] presents a thesis performing analysis of private blockchains utilizing Hyperledger Caliper. The paper highlights Hyperledger Fabric's performance with varying levels of transaction requests and interacts with smart contracts. Similarly, Thakkar et al. [37] presents experimental analysis of Hyperledger Fabric and optimizations made to increase overall throughput.

Baliga et al [38] present the most comparable performance analysis to ours, but utilizes Caliper to evaluate exclusively the Quorum blockchain. This analysis provides insights into how Quorum's configurations effect the overall throughput and latency, focusing directly on the impact of changes to Quorum. In this paper, we utilize the Quorum extension of Caliper to provide comparisons against other blockchains; namely Burrow and Red Belly Blockchain. We analyze how the performance of Quorum with default configurations compares against others with similar environmental setups.

III. Byzantine Fault Tolerant Blockchains

In this section, we present the BFT blockchain systems that we evaluate and the ones we could not evaluate. A summary of the blockchains we evaluate can be seen in Table I where n is the number of nodes and f is the number of Byzantine failures they tolerate.

A. Burrow

Burrow [9] is a blockchain optimized for the proof-of-stake setting. It does not run proof-of-work, which allows for higher performance and can run as a consortium or private blockchain that tolerates Byzantine failures. It supports both the Ethereum Virtual Machine (EVM) and the Web Assembly (WASM) programming language. Its builds upon the Tendermint algorithm [21] to reach consensus.

B. Quorum

Quorum [10] is a permissioned blockchain system that builds upon the Istanbul Byzantine Fault Tolerant (IBFT) consensus protocol [22]. Although the initial version of IBFT suffered from a liveness issue, some fixes were proposed [22]. Quorum targets financial applications and can treat EVM-based smart contracts.

C. Red Belly Blockchain

Red Belly Blockchain [11] builds upon the leaderless DBFT consensus protocol [23] that was designed for blockchains. Instead of having a leader imposing its proposal, it combines multiple proposals into a superblock to scale to large numbers of nodes. It spawns multiple instances of a binary consensus algorithm that was recently proved correct using model checking [39]. In addition to avoiding the leader, Red Belly Blockchain supports a UTXO model and exploits verification sharding to avoid having all nodes verifying the exact same sets of transactions in order to scale to hundreds of geo-distributed nodes. We interpret scalability as the ability for performance not to deteriorate as we increase the system size.

D. Unsupported blockchains

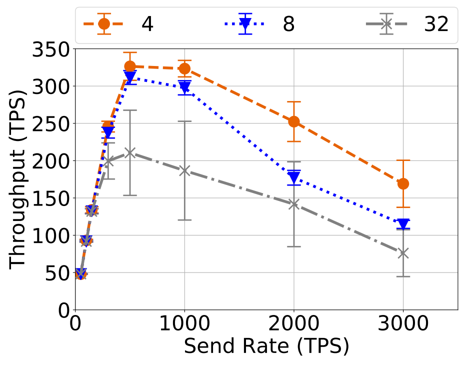
Most BFT blockchains are either no longer supported or their code was made proprietary, and offer at best a non-stable version. Ethermint [40] is the combination of Tendermint, a BFT consensus algorithm [21], with the Ethereum Virtual Machine (EVM). At the time of writing it is at the pre-alpha development stage[6] and despite our efforts we could not deploy it on a distributed system.

Concord[7] is a permissioned blockchain that builds upon the Byzantine fault tolerant consensus algorithm SBFT [41] and is compatible with the EVM. The consensus algorithm was evaluated within a key-value store application and a blockchain application, however, the developers could not help us make the publicly available version of SBFT run on multiple machines. While it might be possible to deploy a proprietary version, its code is not available so we could not compare it to other blockchains.

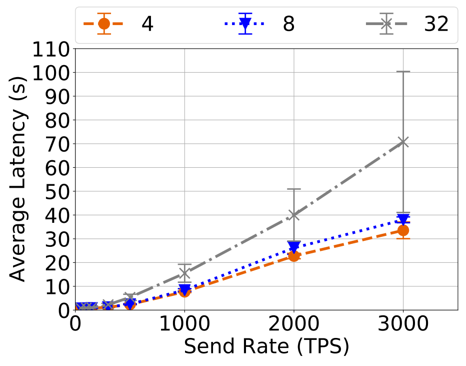
SMaRtChain [42] presents a permissioned blockchain utilizing BFT-SMaRt with reconfigurable consensus to maintain state machine replication, showing promising results. The blockchain layer provides an example execution by supporting value transfer transactions and the consensus layer is compatible with state execution models such as the EVM. Although

[6]https://github.com/ChainSafe/ethermint.

[7]https://github.com/vmware/concord-bft

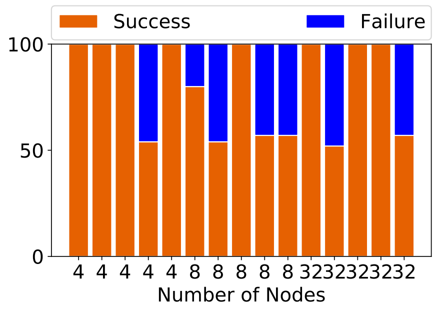


(a) Throughput

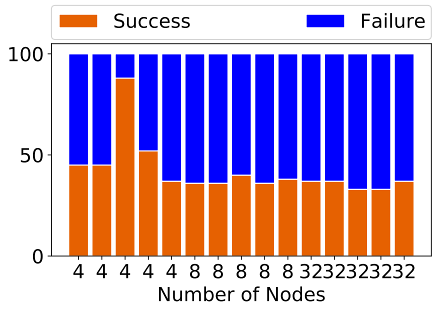


(b) Latency

Fig. 1: 4,8 and 32 Burrow nodes - 1 Caliper client machine



(a) Burrow - send rate 2,000 TPS



(b) Burrow - send rate 3,000 TPS

Fig. 2: Failed transactions for Burrow

SMaRtChain provides a permissioned BFT blockchain, it does not provide methods to interact with the chain that can be utilized within Caliper's benchmark model.

IV. Evaluation on a Distributed System

In this section we present the experimental results achieved using Burrow, Quorum and Red Belly Blockchain on a distributed system. To this end, we modified the Hyperledger Caliper benchmark to interact with Red Belly Blockchain and to overcome three limitations by (1) removing unnecessary overheads, (2) offloading cryptographic signatures and (3) multiplying clients. Section IV-A describes the experimental settings, Section IV-B presents the initial experiments run with Burrow and Red Belly utilising a single Caliper client machine, Section IV-C introduces multiple Caliper client machines while Section IV-D shows the comparative results achieved for Burrow, Red Belly and Quorum.

A. *Experimental Settings*

In the remainder, we refer to *client* as the benchmark runtime sending the transaction requests, and to *server* as the blockchain runtime receiving and treating the requests. We deployed each of the servers as well as each of clients on separate Amazon Web Services (AWS) EC2 virtual machines (VMs). Each considered VM is a c4.xlarge instance with 4 Intel Xeon E5-2666v3 vCPUs running at 2.9 GHz with 7.5 GiB of memory and provided with "High" network performance as defined by Amazon. Each experiment is run at least 5 times for a duration of 20 seconds with a minimum send rate of 200 transactions per second (TPS). As a result each individually plotted data point in our graphs corresponds to the performance computed over at least 20,000 transactions. The error bars, when present, indicate the standard deviations.

In order to minimize the cryptographic overheads, we pre-generate the signatures of the transactions offline before collecting the statistics on the performance: once the transactions are pre-generated and correctly signed, the clients start sending them with specific *send rate* and measuring the performance of the blockchain servers. This decoupling between signing and evaluating the performance is necessary in order to avoid client bottlenecks induced by the CPU-intensive signing process at the client side. In order to minimize unnecessary overheads, we do not make use of additional middleware or virtualization layers like Zookeeper or docker.

B. *A Workload that does not Overload the Blockchain*

We start our series of experiments with a simple configuration where we dedicate one VM located in Oregon to the benchmark client and between 4 and 32 VMs to the blockchain servers. This client VM sends all its requests to the same blockchain server during the entire experiment. In order to achieve fault tolerance, this server must communicate with

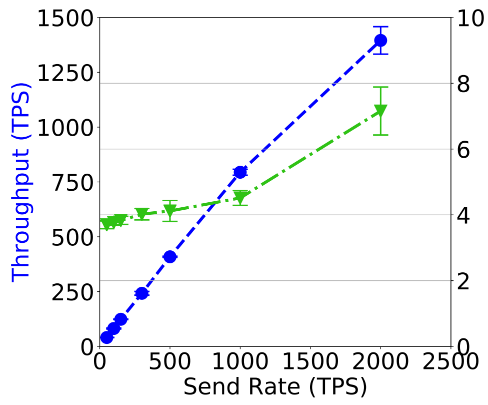


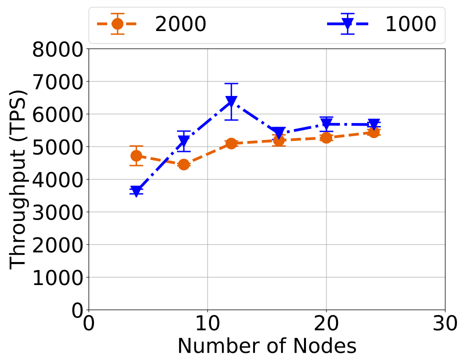
Fig. 3: 4 Red Belly nodes - 1 Caliper client machine

other servers before the requests can be fully treated by the service.

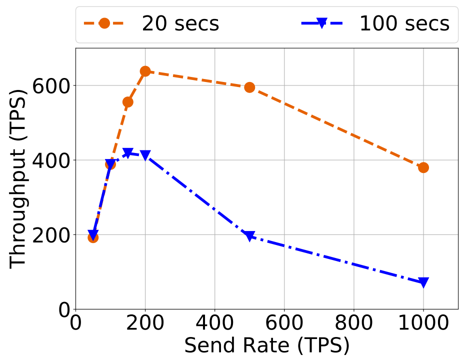
1) The limited capacity of Hyperledger Burrow: Fig. 1a depicts the performance of the Burrow blockchain with $n \in \{4, 8, 32\}$ blockchain servers or Burrow nodes. To this end, we measure the throughput—expressed as the number of transactions treated by the blockchain service per second (TPS)—as we increase the client send rate from 100 to 3000 TPS. We observe that the peak throughput of Burrow is 311 TPS, which is achieved with a send rate of 500 TPS. When the send rate increases further, the throughput decreases as the server struggles to keep up with the send rate increase. We further observe that once the send rate reaches 2,000 TPS, Burrow begins to fail. Interestingly, we observed that a benchmark containing numerous rounds resulted in more frequent system crashes than one-round benchmarks.

Fig. 2 depicts the corresponding failure rate as the percentage of failed transactions for each benchmark run with a send rate of 2,000 TPS or 3,000 TPS. At 2,000 TPS the blockchain handles the load most of the time at 4 server nodes but begins to fail most of the time at 8 and 32 nodes. Once the send rate reaches 3,000 TPS, the blockchain fails transactions on every run of the benchmark regardless of the number of nodes. Throughout the execution of the benchmark, we observed that a benchmark containing numerous rounds resulted in system crashes more frequently than one-round benchmarks.

2) Red Belly performance increases with the workload: We evaluate Red Belly Blockchain similar to the previous Burrow experiment, with one client sending requests to the same blockchain server. Fig. 3 depicts the throughput of Red Belly as we increase the send rate with $n = 4$ server nodes. As opposed to Burrow, Red Belly highest throughput reaches 1,395 TPS with a send rate of 2,000 TPS. As the throughput keeps increasing with the send rate indicates that it is probably not the peak capacity of Red Belly. Moreover, the fact that no transactions fail confirms a previous observation that Red Belly Blockchain offers starvation freedom: every correctly submitted transaction gets eventually committed.



(a) Red Belly - Throughput with varying send rate of 1000 and 2000 TPS



(b) 4 Burrow nodes - Throughput for the same send rate for varying duration of 20s and 100s

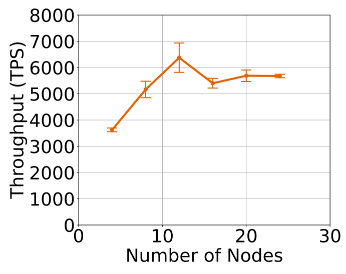
Fig. 4: Equal number of blockchain nodes and Caliper client machines

3) Adjusting the workloads: The previous results illustrated the problem of stress testing a single server node with a single client node. First, as the performance of Red Belly keeps increasing with the send rate, one can provision more client resources to identify faster the Red Belly peak capacity. Second, the instability of Burrow indicates that each benchmark should only be run with a single test round to minimize failures. This is the reason why we multiply the number of clients and balance their send rate to multiple servers in Section IV-C.

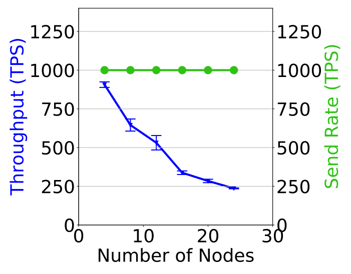
C. *On the Need of Multiple Benchmarking Clients*

We now explore whether a higher throughput can be achieved by multiplying the client machines. As before, we experiment with Burrow and Red Belly. At the time of implementation, Caliper had minimal support for distributed worker client machines, so we first modified Caliper.

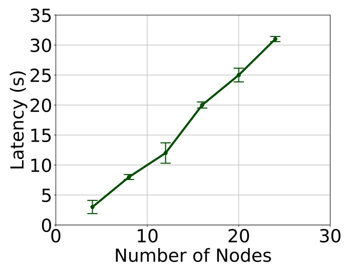
1) Caliper modifications: We wrote scripts to configure and launch multiple client instances on separate VMs and orchestrate the benchmark. One of the scripts launches the specified number of client and server nodes through AWS CLI. It then configures Caliper with network parameters and launches simultaneously the benchmark on multiple VMs,



(a) Total Throughput

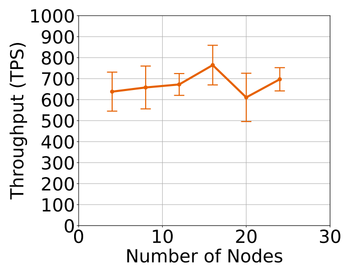


(b) Average Throughput Per Node

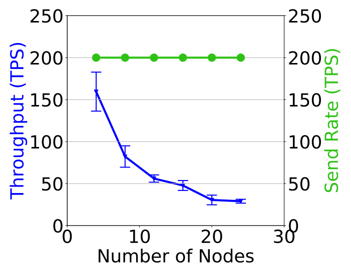


(c) Latency

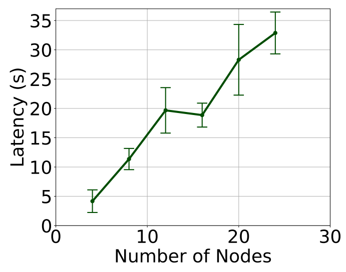
Fig. 5: Red Belly - send rate 1,000 TPS per Caliper client machine



(a) Total Throughput

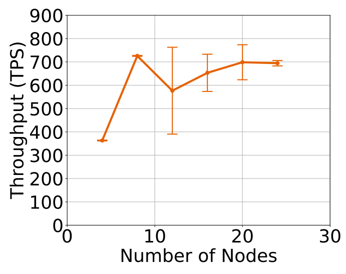


(b) Average Throughput Per Node

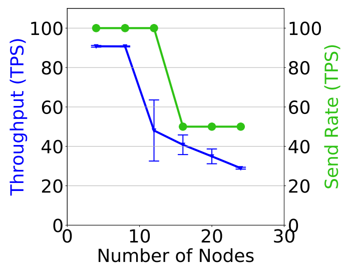


(c) Latency

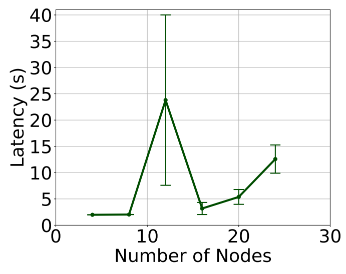
Fig. 6: Burrow - send rate 200 TPS per Caliper client machine



(a) Total Throughput



(b) Average Throughput Per Node



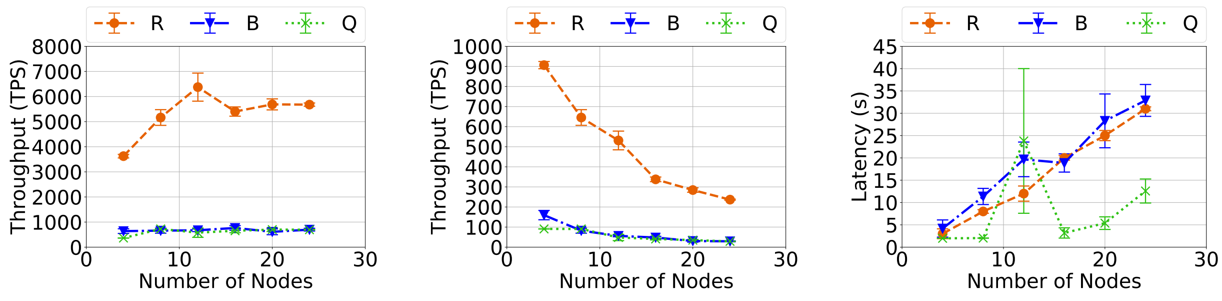
(c) Latency

Fig. 7: Quorum - send rate 100 TPS per Caliper client machine

running Ubuntu 18.04 LTS and NodeJS v10.13.0 and located in the AWS Seoul region. The benchmark was configured to run one test round with the simple benchmark configuration. The throughput is now computed as the sum of the throughputs reported by each client node.

2) The More Clients, the Higher the Load: Fig. 4a shows the performance of Red Belly Blockchain as the number of nodes increases for send rates of 1,000 and 2,000 TPS. The peak throughput of 6,375 TPS is achieved with 12 clients each sending 1,000 TPS to 12 Red Belly servers. Fig. 4a shows that when the send rate of each client is 2,000 TPS the throughput is not as high, indicating that there is an optimal send rate before the client overloads the server. Fig. 4b show the throughput of Burrow in two different length experiments with 4 servers and 4 clients as the send rate increases. The result indicates that for a sending rate above 100 TPS, Burrow achieves lower throughput in an experiment of 100s than in an experiment of 20s. This is mainly due to the blockchain system not being able to process transactions fast enough, which results in an increased number of failed transactions and hence a lower measured throughput.

3) Observation: Improvements during these experiments highlighted that better results are observed when using distributed Caliper clients, each sending simultaneous transactions to the blockchain servers. We observed that there exists an optimal send rate and optimal duration for a benchmark to observe peak performance, as blockchain nodes may become overloaded with high send rates, and longer benchmarks may stack unprocessed transactions leading to memory issues or failed transactions.



(a) Total Throughput (b) Average Throughput Per Node (c) Latency

Fig. 8: Comparison of Red Belly (R), Burrow (B) and Quorum (Q)

These observations led to benchmarks being standardized to 20 seconds, as well as utilizing multiple configurations to determine the optimal send rate prior to final metric gathering.

D. *Comparing Blockchain Performance*

The next set of experiments compares the three blockchain systems namely, Burrow, Red Belly and Quorum. To this end, we deployed multiple Caliper client machines, each sending transactions simultaneously to the same number of blockchain servers at an optimal send rate with an increasing number of blockchain and client nodes.

1) Large Performance Variations per Blockchain: We compare the performance of the three blockchains, Burrow, Red Belly Blockchain and Quorum, by measuring their throughput, latency and scalability as the system enlarges. We first present isolated results indicating the total throughput, the throughput per server as we change the send rate and the latency for each blockchain (Fig. 5, 6 and 7) before plotting the performance of the three blockchains on the same graph (Fig. 8).

Fig. 5 depicts the Red Belly performance with a send rate of 1,000 TPS. Fig. 5a shows that the throughput increases as the number of nodes increases and eventually plateaus to a constant throughput. The peak throughput of 6,375 TPS is achieved with 12 Red Belly nodes and a corresponding latency of 12s. Although the total throughput increases, the average throughput per node decreases from 906 TPS down to 237 TPS as the number of nodes increases, as can be seen from Fig. 5b. This is due to the superblock optimization of Red Belly that combines multiple proposals [11].

Fig. 6 depicts the Burrow performance with a send rate of 200 TPS. Recall that we chose this send rate to maximize Burrow's throughput as Burrow fails under high loads as explained in Section IV-C. Fig. 6a shows that the throughput remains relatively constant as the number of nodes increases. The peak throughput achieved at 16 nodes is 765 TPS with a corresponding latency of 18.86s. Fig. 6b shows that the average throughput per node decreases down to 29 TPS as the number of nodes increases.

Fig. 7 depicts the Quorum performance with a send rate of 200 TPS. Fig. 7b shows that the send rate was 100 TPS for 4, 8 and 12 nodes, however, we had to reduce it to 50 TPS as the blockchain system failed when the send rate was higher than this for more than 12 nodes. As can be seen from Fig. 7c the latency peaked to 24s with 12 nodes. Fig. 7a shows that the throughput peaks at 725 TPS with 8 nodes before dipping to 576 TPS and then coming back up to a maximum of 694 TPS with the lower send rate of 50 TPS. Fig. 7b shows the average throughput per node starts at 91 TPS with 4 Nodes but drops sharply to 29 TPS with 24 Nodes.

Fig. 8 shows a comparison of all three blockchains side by side. Fig. 8a and Fig. 8b show that Red Belly achieves a peak total throughput of 6,375 TPS and an average throughput per node of 906 TPS. This is almost one order of magnitude greater than both Burrow and Quorum. Fig. 8c shows that the Latency for Red Belly and Burrow are relatively similar as the number of nodes increases. The latency achieved by Quorum is relatively lower, however there is a spike at 12 nodes before the send rate decreases. The results achieved in terms of throughput are relatively similar for Burrow and Quorum, as can be seen in Fig. 8a and Fig. 8b, although Quorum has a lower latency as can be seen in Fig. 8c.

2) Red Belly Blockchain Scalability: Although testing showed the optimum send rate for Quorum was 100 TPS, once the number of nodes increased to 16, the Quorum blockchain system failed and Caliper was unable to measure results. As such the send rate was altered to 50 TPS per Caliper machine to accommodate for the higher node numbers in the network. We observed that both Quorum and Burrow only managed to achieve a throughput of under 800 TPS while Red Belly was able to achieve a peak throughput of 6,375 TPS as depicted in Table II.

TABLE II: Comparison summary of the blockchains, when deployed on c4.xlarge AWS instances

Blockchain	Consensus	Peak throughput
Burrow [9]	Tendermint [21]	765 TPS
Quorum [10]	IBFT [22]	726 TPS
Red Belly [11]	DBFT [23]	6375 TPS

V. Conclusion

The abundance of blockchains makes it hard to identify the most suitable blockchain. Despite recent efforts, there is no way to compare the performance of secure blockchains. We identified three frequent evaluation limitations: (1) the lack of distributed workload generation, (2) misleading overheads, and (3) processing of unnecessary computationally intensive tasks during testing. By addressing these limitations we were able to effectively evaluate three Byzantine Fault Tolerant blockchains: Burrow, Quorum and Red Belly Blockchain. This evaluation led to identify the maximum workload that Burrow and Quorum can tolerate before losing requests and that they are almost one order of magnitude slower than Red Belly Blockchain. Future work includes the evaluation of smart contracts.

Acknowledgments

This research is supported under Australian Research Council Future Fellowship funding scheme (project number 180100496) entitled "The Red Belly Blockchain: A Scalable Blockchain for Internet of Things".

References

[1] M. Cavicchioli, "EoS in the lead for TPS thanks to the new testnet," 2020. [Online]. Available: https://en.cryptonomist.ch/2020/02/12/eos-tps-testnet/

[2] S. O'Neal, "Who scales best? inside blockchains' ongoing transactions-per-second race," 2019. [Online]. Available: https://cointelegraph.com/news/who-scales-it-best-inside-blockchains-ongoing-transactions-per-second-race

[3] Unita, "Qtumx reaches 10,000 tps in benchmark tests," 2019. [Online]. Available: https://blog.qtum.org/qtumx-reaches-10-000-tps-in-benchmark-tests-cee6452166fd

[4] Hyperledger Foundation, "Hyperledger caliper," 2019. [Online]. Available: https://hyperledger.github.io/caliper/

[5] T. T. A. Dinh, J. Wang, G. Chen, R. Liu, B. C. Ooi, and K.-L. Tan, "Blockbench: A framework for analyzing private blockchains," in *SIGMOD*, 2017, pp. 1085–1100.

[6] S. Nakamoto, "Bitcoin: a peer-to-peer electronic cash system," 2008. [Online]. Available: http://www.bitcoin.org

[7] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," 2015, yellow paper.

[8] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. De Caro, D. Enyeart, C. Ferris, G. Laventman, Y. Manevich, S. Muralidharan, C. Murthy, B. Nguyen, M. Sethi, G. Singh, K. Smith, A. Sorniotti, C. Stathakopoulou, M. Vukolić, S. W. Cocco, and J. Yellick, "Hyperledger fabric: a distributed operating system for permissioned blockchains," in *EuroSys*, 2018, pp. 1–15.

[9] C. Kuhlman, B. Bollen, S. Davis, and D. Middleton, "Hyperledger burrow (formerly eris-db)," Mar. 2017. [Online]. Available: https://www.hyperledger.org/wp-content/uploads/2017/06/HIP_Burrowv2.pdf

[10] JP Morgan Chase, "Quorum whitepaper," Aug. 2018. [Online]. Available: https://github.com/jpmorganchase/quorum/blob/master/docs/QuorumWhitepaperv0.2.pdf

[11] T. Crain, C. Natoli, and V. Gramoli, "Evaluating the Red Belly Blockchain," arXiv, Tech. Rep. 1812.11747, 2018.

[12] A. Krüger, "Chainhammer: Ethereum benchmarking," 2017. [Online]. Available: https://github.com/drandreaskrueger/chainhammer

[13] C. Gorenflo, S. Lee, L. Golab, and S. Keshav, "Fastfabric: Scaling hyperledger fabric to 20,000 transactions per second," in *IEEE ICBC*, 2019, pp. 455–463.

[14] P. Hunt, M. Konar, F. P. Junqueira, and B. Reed, "Zookeeper: Wait-free coordination for internet-scale systems," in *USENIX ATC*, 2010, p. 11.

[15] B. F. Cooper, A. Silberstein, E. Tam, R. Ramakrishnan, and R. Sears, "Benchmarking cloud serving systems with YCSB," in *ACM SOCC*, J. M. Hellerstein, S. Chaudhuri, and M. Rosenblum, Eds., 2010, pp. 143–154.

[16] P. Szilagyi, "Clique proof-of-authority consensus protocol," 2017. [Online]. Available: https://github.com/ethereum/EIPs/blob/master/EIPS/eip-225.md

[17] Parity, "Parity aura: Authority round." [Online]. Available: https://openethereum.github.io/wiki/Aura

[18] D. Gupta, L. Perronne, and S. Bouchenak, "BFT-Bench: A framework to evaluate BFT protocols," in *ACM/SPEC ICPE*, 2016, pp. 109–112.

[19] A. Singh, T. Das, P. Maniatis, P. Druschel, and T. Roscoe, "BFT protocols under fire." in *NSDI*, vol. 8, 2008, pp. 189–204.

[20] C. Faria and M. Correia, "Blocksim: Blockchain simulator," in *IEEE Blockchain*, 2019, pp. 439–446.

[21] E. Buchman, J. Kwon, and Z. Milosevic, "The latest gossip on BFT consensus," arXiv, Tech. Rep. 1807.04938, Nov. 2019. [Online]. Available: http://arxiv.org/abs/1807.04938

[22] R. Saltini and D. Hyland-Wood, "Correctness analysis of IBFT," arXiv, Tech. Rep. 1901.07160v2, Aug. 2019.

[23] T. Crain, V. Gramoli, M. Larrea, and M. Raynal, "DBFT: Efficient leaderless Byzantine consensus and its applications to blockchains," in *IEEE NCA*, 2018.

[24] R. G. Brown, J. Carlyle, I. Grigg, and M. Hearn, "Corda: An introduction," *R3 CEV, August*, 2016.

[25] R. Han, G. Shapiro, V. Gramoli, and X. Xu, "On the performance of distributed ledgers for internet of things," *Internet of Things*, Aug 2019.

[26] C. Dwork, N. Lynch, and L. Stockmeyer, "Consensus in the presence of partial synchrony," *J. ACM*, vol. 35, no. 2, Apr. 1988.

[27] T. Hanke, M. Movahedi, and D. Williams, "DFINITY technology overview series, consensus system," arXiv, Tech. Rep. 1805.04548, May 2018.

[28] I. Abraham, D. Malkhi, K. Nayak, L. Ren, and A. Spiegelman, "Solida: A blockchain protocol based on reconfigurable Byzantine consensus," in *OPODIS*, 2017, pp. 25:1–25:19.

[29] M. Castro and B. Liskov, "Practical Byzantine fault tolerance and proactive recovery," *ACM Trans. Comput. Syst.*, vol. 20, no. 4, pp. 398–461, Nov. 2002.

[30] T. Rocket, "Snowflake to avalanche: A novel metastable consensus protocol family for cryptocurrencies," 2018, unpublished manuscript.

[31] E. Kokoris-Kogias, P. Jovanovic, L. Gasser, N. Gailly, E. Syta, and B. Ford, "Omniledger: A secure, scale-out, decentralized ledger via sharding," in *IEEE S&P*, 2018, pp. 583–598.

[32] C. Natoli and V. Gramoli, "The balance attack or why forkable blockchains are ill-suited for consortium," in *IEEE/IFIP DSN*, 2017, pp. 579–590.

[33] F. Leal, A. E. Chis, and H. González-Vélez, "Performance evaluation of private ethereum networks," *SN Computer Science*, vol. 1, no. 285, 2020.

[34] P. Ekparinya, V. Gramoli, and G. Jourjon, "The Attack of the Clones against Proof-of-Authority," in *NDSS*. Internet Society, Feb 2020.

[35] S. Pongnumkul, C. Siripanpornchana, and S. Thajchayapong, "Performance analysis of private blockchain platforms in varying workloads," in *Proceedings of the 26th International Conference on Computer Communication and Networks (ICCCN)*, 2017, pp. 1–6.

[36] H. F. Leppelsack, "Experimental performance evaluation of private distributed ledger implementations," 2018.

[37] P. Thakkar, S. Nathan, and B. Viswanathan, "Performance benchmarking and optimizing hyperledger fabric blockchain platform," in *IEEE MASCOTS*, 2018, pp. 264–276.

[38] A. Baliga, I. Subhod, P. Kamat, and S. Chatterjee, "Performance evaluation of the quorum blockchain platform," arXiv, Tech. Rep. 1809.03421, 2018.

[39] P. Tholoniat and V. Gramoli, "Formal verification of blockchain Byzantine fault tolerance," in *6th Workshop on Formal Reasoning in Distributed Algorithms (FRIDA'19)*, Oct 2019. [Online]. Available: https://gramoli.redbellyblockchain.io/web/doc/pubs/frida19.pdf

[40] Interchain foundation, "A beginner?s guide to ethermint," 2017. [Online]. Available: https://blog.cosmos.network/a-beginners-guide-to-ethermint-38ee15f8a6f4

[41] G. Golan-Gueta, I. Abraham, S. Grossman, D. Malkhi, B. Pinkas, M. K. Reiter, D. Seredinschi, O. Tamir, and A. Tomescu, "SBFT: A scalable and decentralized trust infrastructure," in *IEEE/IFIP DSN*, 2019, pp. 568–580.

[42] A. Bessani, E. Alchieri, J. Sousa, A. Oliveira, and F. Pedone, "From Byzantine replication to blockchain: Consensus is only the beginning," in *IEEE/IFIP DSN*, 2020, pp. 424–436.