Performance Modeling of Hyperledger Sawtooth Blockchain

Benjamin Ampel
Management Information Systems
University of Arizona
Tucson, AZ
bampel@email.arizona.edu

Dr. Mark Patton
Management Information Systems
University of Arizona
Tucson, AZ
mpatton@email.arizona.edu

Dr Hsinchun Chen
Management Information Systems
University of Arizona
Tucson, AZ
hchen@email.arizona.edu

***Abstract* – With the rapid development of blockchain platforms, it is important that different implementations are tested and analyzed for comparative purposes. One such implementation is Hyperledger Sawtooth, a new member of the Hyperledger family. Sawtooth blockchain is a permissioned implementation developed in part by Intel. While research has been done on Hyperledger Fabric, research on Sawtooth is not well documented. Using the Hyperledger Caliper benchmarking tool, we aim to test the performance of the blockchain and identify potential issues.**

Index – Blockchain, Hyperledger Sawtooth, Hyperledger Caliper, Performance Modeling

I. INTRODUCTION

A blockchain is an immutable ledger system with the goal of decentralization. Each block is encrypted with a hash, which is referred to by the following block in the sequence, forming a chain. Since the release of the Bitcoin blockchain in 2008, hundreds of different blockchains have been created for a multitude of purposes. These include cryptocurrency, supply chain management, decentralized applications, and many others. To perform these tasks, blockchains use smart contracts, which are programs that take an input and create an output. The most common use for smart contracts is to validate transactions before they are written into the blockchain.

Bitcoin is known as a permissionless blockchain [1], meaning that it is open to whomever wants to use it, and that the nodes are implicitly not trusted. A consensus algorithm is used to try to stop malicious actors from writing to the blockchain and keeping every block legitimate. The throughput, usually measured in transactions per second (tx/sec) of permissionless chains currently is much slower than standard payment options, and transactions can be backed up for hours during heavy usage time. This has led to the popularity of permissioned blockchains.

Permissioned blockchains [2] aim to scale throughput by making the consensus algorithm less computationally intensive. This is done by trusting all nodes and adding a barrier of entry to the blockchain. Only those with permission can interact with the blockchain or see transaction history. Since the consensus algorithm simply needs to determine the next block, and not determine if the node is legitimate, throughput can increase drastically. While research has been done on many permissioned blockchains like Hyperledger Fabric, we believe we are the first to do testing of the Sawtooth permissioned blockchain.

II. HYPERLEDGER SAWTOOTH

Hyperledger Sawtooth [3] is built to be an open source distributed ledger for the modern enterprise. Unlike many popular blockchains, Sawtooth is not built for cryptocurrency, but instead for business supply chain management. The transaction flow begins with the client placing all transactions into a block, and then signing the batch and sending it to a validator. The validator uses its transaction processor to ensure the integrity of the batch, and then commits it. Sawtooth executes transactions in parallel, instead of in serial, when possible through a REST API to improve performance. It also contains the novel feature of being modular, which includes consensus algorithms, rule sets, coding language, and smart contracts. This allows it to efficiently change depending on the business need. Programmers can use Python, JavaScript, Go, C++, Java, and Rust to build and interact with the Sawtooth blockchain.

Currently, four different consensus algorithms are supported by Sawtooth. These are Dev_mode, PoET, PoET-Simulator, and RAFT. Dev_Mode is a random generator algorithm used purely for developer testing. Proof of Elapsed Time (PoET) [4] is built specifically for Sawtooth and does not follow byzantine fault tolerance (BFT), allowing it to reach higher throughput than other models. BFT is a type of accepted failure systems in case of a malicious actor and can define the amount of fault a system can tolerate. This is because PoET assumes that everyone that interacts with the blockchain is a trusted member. PoET has each node randomly generate a timer. The node that has the timer run out first is made leader, and the leader appends a new block to the end of the chain. The calculation process is done within a private, Intel CPU environment that cannot be accessed by the node to prevent tampering, known as Intel Software Guard Extensions. This simple lottery system cuts down on computational processes to increase throughput and reduce latency. RAFT [5] is an election-style algorithm where each node can

become a candidate each term if it does not hear back from a leader after a certain amount of time. Candidates then request votes from others, and if they get more than half of the votes, they become leader and get to append a new block to the end of the chain. The leader also has the job of replicating the new log to all other nodes to maintain consistency. All nodes are given a term number and will only accept logs from leaders that have a term number greater than the current term that they know.

III. HYPERLEDGER CALIPER

Supported by the Linux Foundation [6], this is a tool that is used as a performance benchmark framework for permissioned blockchains. The tool allows for testing different blockchains with similar environments for direct comparison. The tool can track metrics such as throughput, latency, success rate, and CPU / Memory resource consumption. This is done by listening to transaction timestamps and then calculating the metrics based on those. Throughput is measured as tx/sec and is how fast transactions are committed to the ledger successfully. Latency is measured in seconds and is the amount of time between transactions being sent and them being received. Success rate is how many of the transactions were successfully committed against how many were sent. The goal for this metric should be 100%. CPU / Memory resource consumption gives information on the minimum, maximum, and average usage of those metrics. CPU is measured as a percentage, and memory is measured in megabytes.

For Sawtooth, Caliper has a chaincode test called simple, which we utilized for our research. The simple test keeps track of account balances over time with a series of random transactions. Chaincode [7] is a type of program that implements the business logic of the blockchain. This simple chaincode test can be preset with the number of transactions, the input transaction speed, and the batch size you want to test. Caliper runs the chaincode and outputs an HTML file with the results.

IV. RESULTS

Throughout testing, throughput was able to reach a maximum of about 2300 tx/sec. This is much higher than any researched permissionless blockchain, and also ahead of the results found for Hyperledger Fabric. The other three Hyperledger projects have not yet been researched.

Testing Environment: The tests are run on a VMWare Workstation running Ubuntu 16.04, installed on a 500GB NVMe SSD. The virtual machine has access to 16GB DDR4 RAM, a Ryzen 1600 6-core 3.2 GHz CPU, and an NVIDIA GTX 2060 GPU. Every test consisted of sending 30,000 transactions from two peers at a fixed rate, which are built within docker containers. Sawtooth version 1.0 was tested.

Observation 1: *Impact of input transaction rate on throughput.* Changing input transaction rate, and holding batch size constant, saw a linear increase from throughput until about 1000 tx/sec, where performance would degrade, and transactions would fail at a 100% rate due to queue timeout. This is because Sawtooth uses an all-or-nothing approach to batches. Either they all succeed in being committed, or all fail. The only way to move past this point was to increase the batch size during testing.

Observation 2: *Impact of batch size on throughput.* Batch size is equivalent to block size, as it is the number of transactions that are in every block of the blockchain. Multiple tests were run by changing both batch size and input transaction rate to see if they remained linear with throughput over different sizes. In Figure 1, Batch size is shown to almost linearly increase with throughput in the implementation. This holds true until about 2300 tx/sec, where transactions begin to fail due to a timeout. It is also important to note that if batch size is placed much higher than send rate, transactions will fail, regardless of the total number of transactions No combination of batch size and input transaction rate could get a higher throughput than 2300 tx/sec in the current model.

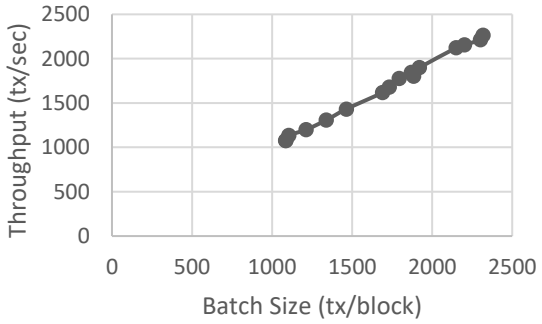

Figure 1: Effect of Batch Size on Throughput

Observation 3: *Impact of input transaction rate on memory usage.* The CPU usage and the memory usages also spike exponentially to much higher levels than previously seen as the input transaction rate is increased, which is shown in Figure 2. This could become a potential bottleneck in transaction speed, as memory could continue to rise and cause issues in transaction success rate and latency.

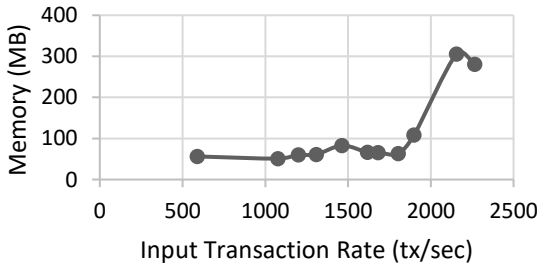

Figure 2: Effect of Input Transaction Rate on Memory

Observation 4: *Impact of Throughput on Latency.* While there is a linear increase between batch size and throughput, latency scales exponentially with increasing throughput, as shown in Figure 3.

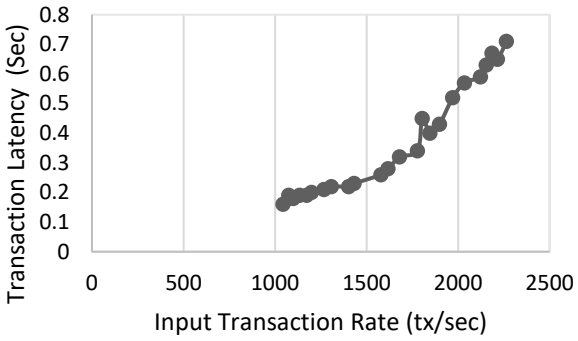

Figure 3: Effect of Input Transaction Rate on Latency

V. RELATED WORK

Performance modelling of Hyperledger Fabric has been done before [8] with interesting results that were like the results found in our research on Sawtooth. Caliper has also been used in research for Hyperledger Fabric and has proven successful to test various parameters and the performance of their implementation.

One study from Sukhwani et al [9] found that in testing Fabric, Caliper was an effective tool. The authors were able to leverage the simple chaincode to suit their needs. This study found the same results in increased latency as block size was increased, latency could exponentially increase. Their solution was to attempt to add multiple endorsers.

Another study from Baliga et al [10] found that latency could skyrocket at certain transaction rates, and especially if multiple chaincodes were used instead of just one. Multiple chaincodes deployed on the same channel also worked about as well as a single chaincode for input transaction rate until about 900 tx/sec, where the latency of the model then sharply increased from about 1 second to 28 seconds. They also found around this area that throughput rapid dropped off from 700 tx/sec to about 250 tx/sec when trying to increase input transaction rate past 900 tx/sec. Finally, they ran tests with 4, 8, 12, and 16 peers in their model, and found that with more peers, latency increases, and throughput decreases, with a more pronounced effect at higher input transaction rates.

VI. CONCLUSION & FUTURE WORK

In this paper, we presented a study to model the performance of the permissioned Hyperledger Sawtooth blockchain. Various parameters were tested, including transaction send rate, batch size, throughput, CPU / memory usage, and latency. With the benchmarking tool, Hyperledger Caliper, we provided information on the performance of our model that can be used as a framework for future discussion. Bottlenecks were identified in latency, CPU, and memory. Sawtooth is a relatively new blockchain, under active development, and will add different methods to improve modifiability and efficiency of the blockchain. This research not only creates a jumping block towards more in-depth Sawtooth testing, but also shows how the Caliper tool can be used for other implementations than just Hyperledger Fabric. In the future, we aim to test potential solutions to the bottlenecks using different consensus algorithms, different endorsers, differing databases, and more powerful hardware.

VII. ACKNOWLEDGMENT

This paper is written with the support of the National Science Foundation under grant number DGE-1303362.

References

1. M. Cash and M. Bassiouni, "Two-Tier Permission-ed and Permission-Less Blockchain for Secure Data Sharing," *2018 IEEE International Conference on Smart Cloud (SmartCloud)*, New York, NY, 2018, pp. 138-144.
2. X. Min, Q. Li, L. Liu and L. Cui, "A Permissioned Blockchain Framework for Supporting Instant Transaction and Dynamic Block Size," *2016 IEEE Trustcom/BigDataSE/ISPA*, Tianjin, 2016, pp. 90-96.
3. *Hyperledger Sawtooth*, [online] Available: https://www.hyperledger.org/projects/sawtooth.
4. *PoET Specification*, [online] Available: https://sawtooth.hyperledger.org/docs/core/releases/latest/architecture/poet.html.
5. C. Saraf and S. Sabadra, "Blockchain platforms: A compendium," *2018 IEEE International Conference on Innovative Research and Development (ICIRD)*, Bangkok, 2018, pp. 1-6.
6. *Hyperldger Caliper Documentation*, [online] Available: https://github.com/hyperledger/caliper
7. *Hyperledger Fabric Chaincode*, [online] Available: https://hyperledger-fabric.readthedocs.io/en/release-1.3/chaincode.html.
8. P. Thakkar, S. Nathan and B. Viswanathan, "Performance Benchmarking and Optimizing Hyperledger Fabric Blockchain Platform," *2018 IEEE 26th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*, Milwaukee, WI, 2018, pp. 264-276.
9. H. Sukhwani, N. Wang, K. S. Trivedi and A. Rindos, "Performance Modeling of Hyperledger Fabric (Permissioned Blockchain Network)," *2018 IEEE 17th International Symposium on Network Computing and Applications (NCA)*, Cambridge, MA, 2018, pp. 1-8.
10. A. Baliga, N. Solanki, S. Verekar, A. Pednekar, P. Kamat and S. Chatterjee, "Performance Characterization of Hyperledger Fabric," *2018 Crypto Valley Conference on Blockchain Technology (CVCBT)*, Zug, 2018, pp. 65-74