Technische Universität Berlin

Experiment report

Ethereum vs. Hyperledger

Comparison of throughput for non-conflicting transactions

Authors:
Jacek Janczura
Igor Molcean
Julian Valentino Weigel
Kim Janik Jasun Herter

Contents

Abstract

In this experiment, we compared Ethereum and Hyperledger private networks with regard to throughput for non-conflicting transactions, i.e. transactions that do not cause double-spending. We proposed two mathematical definitions of throughput and used them to compare both systems. We set up the networks in similar conditions and provided them with synthetically generated workload. We used custom tools to collect relevant data from systems' logs. Finally, we came to the conclusion that Ethereum, generally, shows more promising results. However, further research will be needed for more profound analysis.

1 Introduction

Authors: Igor Molcean, Julian Weigel (Hyperledger-specific parts)

Blockchain is a relatively new concept. It represents a distributed, uncentralized and immutable data storage where every single change can be seen by anyone and the whole amount of data is stored at every node that participates in the network. This makes blockchain quite different from traditional databases that try to restrict access as far as possible and require centralized coordinators for replication. Unlike many distributed storages, blockchain does not use redundancy to increase performance but rather to improve resilience of the system. For the first time, the term was used in context of Bitcoin in 2008 [8]. This became the reason why blockchain is often associated with cryptocurrencies but the potencial of this technology is actually much higher.

One of the big problems that prevents blockchain-based technologies from entering our everyday life is the limited throughput. For example, limited block size in case of Bitcoin, caused the so-called ”Bitcoin scalability problem” which resulted in intense research [7] and numerous proposals on how to increase throughput of the system. Being able to perform operations fast, is important for a system that is used by a big number of people worldwide. This is why throughput becomes so critical in context of blockchain systems.

The goal of our experiment is to measure the throughput for Ethereum and Hyperledger, which are two widely used blockchain systems. This is done in order to learn how effective their underlying algorithms process big numbers of submitted transactions in short periods of time.

1.1 Main concepts

Throughput is the maximum rate at which transactions can be successfully processed by the blockchain, i.e. end up in a block of the canonical chain. Not to be confused with **Bandwidth**, which is the maximum possible rate at which transactions can be processed theoretically. Both are measured in transactions per second (Tx/s).

Proof-of-Authority is an algorithm where only authorized nodes, called sealers or signers, can add blocks to chain. To make sure a malicious node can't do harm to the network, any signer can sign at most one of a number $\lfloor S/2 \rfloor + 1$ of consecutive blocks (where S is the number of sealers). PoA algorithm used in Ethereum is called Clique [9].

Proof-of-Stake is a consensus algorithm where decisions, whether to accept a transaction or not, are made based on weighted random choice. It is calculated by participation duration in combination with the size of the stake. The proof of stake algorithm is fully byzantine fault tolerant [5]. A high transaction throughput is gained via the used Tendermint consensus engine [6].

2 Method

Authors: Igor Molcean, Julian Weigel (Hyperledger-specific parts)

This sections will describe materials and procedures that were used during the experiment. We will start with the formal mathematical definition of the main metric, continue with the experimental set up and finish with the procedures for data collection.

2.1 Transaction throughput

In our experiment, we defined throughput as average number of transactions in a block divided by the block frequency:

$$Throughput = \frac{1/N \sum_{b=1}^{N} T_b}{F}$$

where

- N is total number of transactions submitted
- T_b is number of transactions contained in block b
- F is block frequency

Other definitions can also be found in literature. We decided to compare one of them with ours. It is formulated as follows: transaction throughput is the rate at which valid transactions are committed by the blockchain System Under Test (further SUT) in a defined time period [4]. This definition may be presented as the following formula:

$$Throughput_{alt} = \frac{N}{t_{b_N} - t_{T_0}}$$

where

- N is total number of submitted transactions submitted
- t_{b_N} is the commit time of the last block
- t_{T_0} is the submission time of the initial transaction

2.2 Experimental set up

The experimental setup consists of the two private blockchain networks deployed in a controlled distributed environment. As such environment we used several Amazon AWS EC2 instances, their parameters are presented in Table 1. Each node runs on its own Virtual Machine (further VM) but all VMs are located in the same subnetwork to minimize the effect of network latencies within the experiment. For simplicity, both networks were deployed with just one mining node. We conducted the same experiment several times with different values of N.

Type	Cores	CPU	RAM	Network performance
t2.micro	1	3.3 GHz	1 GiB	Low to Moderate

Table 1: Parameters of the EC2 instances [1]

In case of Ethereum, we created a private network with the Proof-of-Authority consensus algorithm. There are two nodes running on Geth [3]: Node 1 is sealing blocks and Node 2 is submitting party-to-party transactions to the blockchain. In order to enable nodes' communication, we used the so-called Bootnode [2]. The block frequency F was set to 2 seconds. In order to guarantee that all transactions are non-conflicting, i.e. they do not potentially cause double spending, we preallocated enough Ether on the account used as a transaction sender.

Additionally, we have tested another mining setup. In that second experimental version, block period was set up to 0. That means that the sealer starts to seal new blocks only if there are new transactions. If there are no transactions in the transaction pool, sealing frequency drops to 0 and the sealer will wait for the new transaction to be authorised and added to the pool.

For Hyperledger we used the Burrow framework. Burrow was selected, because it uses the same EVM as Etherium. Therefore the same language for contracts could be used. There are two EC2 instances running. One for the Blockchain system with one registered validator and with one normal participant. The second instance is used to send the transactions to the first node.
To create a usable configuration for Burrow to run, the following code was used.

```
   burrow spec -p1 -f1 | burrow configure -s- > burrow.toml
```

The key pairs for one validator and one participant were generated. The automatic generated config files listening addresses had to be changed from *127.0.0.1:port* to *0.0.0.0:port* for it to work on two different machines. To avaid double spending, the initial funds were set to a high amount and the transaction amount during the testing phase changed to one.

2.3 Methods of data collection

In case of Ethereum, all necessary data could be found directly in the logs generated by Geth. For this purpose, we developed a parser that went through the log files extracting relevant information and storing it in a CSV file. By relevant information, we mean the number of transactions for each block (T_b) as well as timestamps of the first submitted transaction (t_{T_0}) and last committed block (t_{b_N}). Knowing block frequency (F), total number of transactions (N) and being able to parse everything else, it became trivial to calculate throughput according to the definitions presented in subsection 2.1.

For Hyperledger Burrow, the logs were used as well. The arrival and execution timestamps as well as the hash of a received transaction were logged. With an own written parser, these timestamps could be extracted and used for the calculation of the throughput. The sending of multiple transactions at once was done by another self-written tool. The communication of the Blockchain and the clients were implemented by Burrow via TCP. Because this resulted in a round-trip time close to one second, a thread pool was used to increase the number of parallel transactions.

3 Results

Author: Jacek Janczura
Blockchain is not a normal BASE or ACID database, where transactions are executed immediately after submitting. In case of blockchain transactions are stored in blocks and than they are commonly added to a blockchain. As it is clearly visible in Table 2, measuring throughput for the number of transactions less then the block capacity is pointless. That is why we present AVG filtered values where we do not include the measurements for $tx < 500$.

3.1 ETH measurements with block mining period 2s.

Author: Jacek Janczura
Throughput was measured consequently with the two previously aforementioned definitions. In the Fig.1 throughput was measured as average number of transactions in a block divided by the block frequency and in the Fig.2 throughput was measured using our own formula.

tx	**tbn-t0** [s]	**Thr. alt.** [tx/s]	**Avg(tx/block)**	**Thr.** [tx/s]
10	*2.69*	*3.72*	*10*	*5*
100	*2.79*	*35.80*	*100*	*50*
1000	*10.67*	*93.72*	*200*	*100*
10000	*99.82*	*100.18*	*204.08*	*102.04*
100000	*1711.73*	*58.42*	*117.1*	*58.55*
489290	*8935.53*	*54.76*	*109.53*	*54.765*
	AVG	***57.77***	***123.45***	***61.73***
	AVG filtered	***76.77***	***157.68***	***78.84***

Table 2: ETH: Throughput measurement results. Block mined every 2s.

Table 2 presents the results of the throughput measurements. In ethereum private blockchain we have observed that in case of lower blockchain load ($tx < 10000$) throughput is in average around 100 transactions per second and the average number of transactions in a block is then at the level of around 200 tx/block. Unfortunately in case of higher blockchain load throughput falls to the level of 50 tx/s and stays on that level even in a really overloaded blockchain.

The results from measuring the throughput in both ways Fig.1 and Fig.2 results in similar numbers - average filtered throughput between 75-80 tx/s.

In Fig. 3 we present the number of transactions per block. It is clearly visible that for higher load average $tx/block$ stays in the level of 100 ts/block. We have data for 1 million transactions as well. The results are the same as for 100 000 but the chart is less clear. That is why we have decided to present this one.

What is interesting is, that we wanted to observe the drop in the block capacity. This drop is presented in Fig. 4. We think that this might be the result of an internal blockchain balancing but we could not proof that. That is why this problem needs further research.

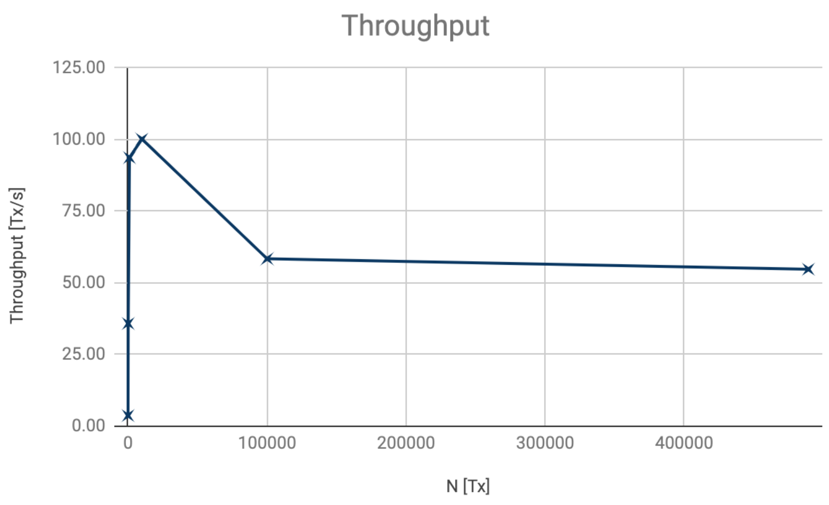

Figure 1: ETH: Throughput as average number of transactions in a block divided by the block frequency

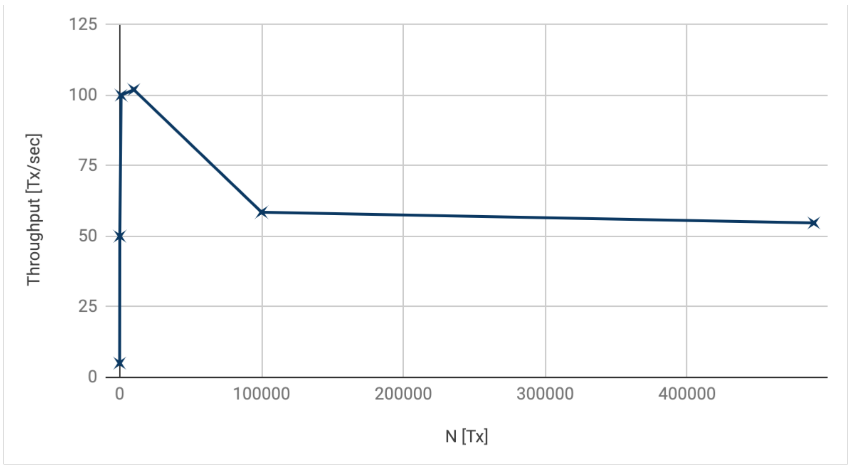

Figure 2: ETH: Throughput as total number of transactions divided by time from submitting the first transaction until mining the last block

3.2 ETH measurements with block mining period 0s.

Author: Jacek Janczura
The measurements from the previous experiment were repeated with a new genesis. In new setup we decided to mine the block on demand. In that case demand is a presence of a transaction that is expected to be added to the blockchain. Using that

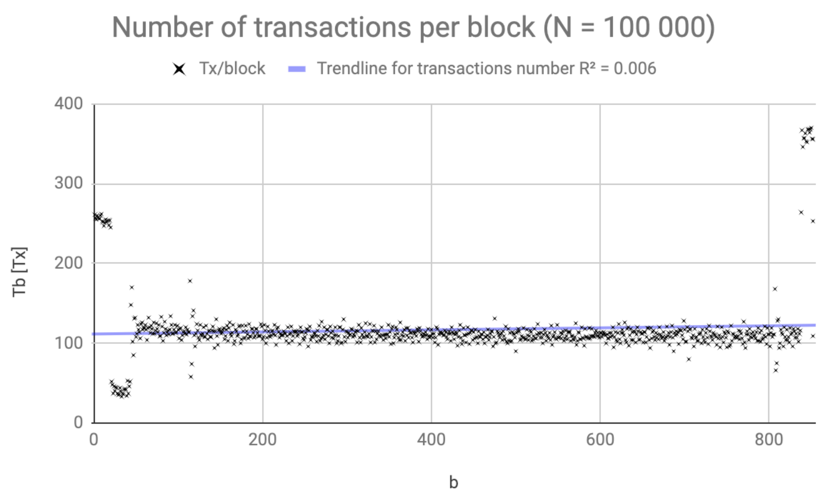

Figure 3: ETH: Block capacity in a higher blockchain load conditions

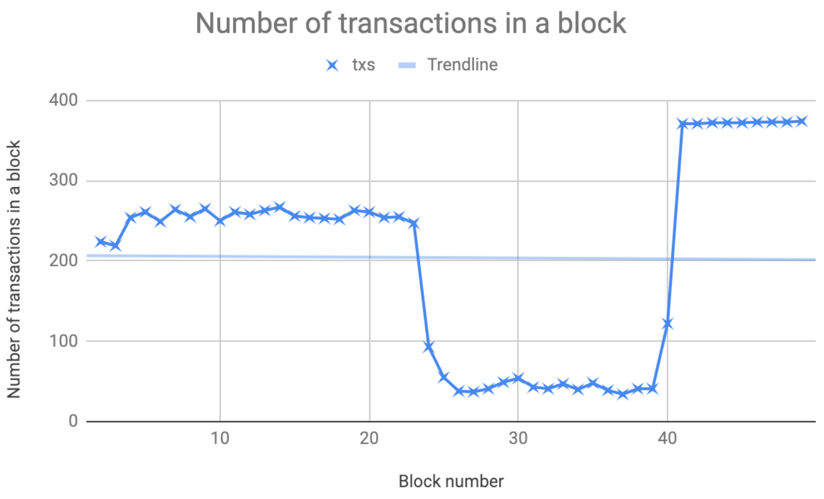

Figure 4: ETH: Drop in the block capacity

method multiple transactions still might be fitting in a block but the miner is not wasting its mining power to mine empty blocks. This setup looks more like BASE or ACID database where transactions are executed immediately.

In Table 3 we present the results of this new approach. In this case we are measuring throughput as total number of transactions divided by time from submitting the first transaction until mining the last block. Since the block frequency is not fixed we cannot measure the throughput using the method with block frequency - 2.1.

tx	**tbn-t0 [s]**	**Thr. [tx/s]**
10	*0.08*	*128.21*
100	*0.94*	*105.93*
1000	*9.21*	*108.59*
10000	*90.84*	*110.09*
100000	*924.48*	*108.17*
	AVG	***112.20***

Table 3: ETH: Throughput measurement results. Mine on demand

Average throughput in that setup has raised to in average 112,2 transactions per second with an 1,02 average number of transactions per block

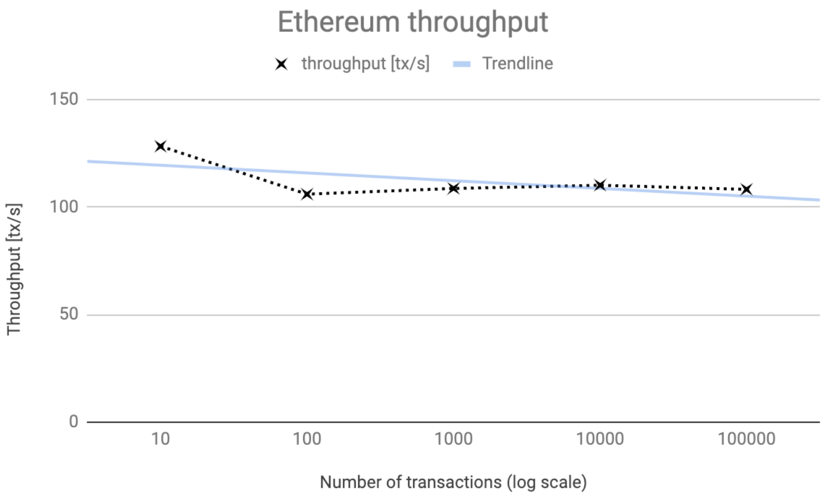

Figure 5: ETH: throughput on a log scale

In this setup in Fig. 5 we present the graph where we correlate the throughput with a number of transactions on a log scale. The throughput in that scenario is transaction independent and is in average around 100 tx/s.

Fig. 6 depicts block capacity. This experiment was held on a milion transactions. It's clearly visible that in that setup miner submits on average 1 transaction per block and that value is stable within the whole experiment.

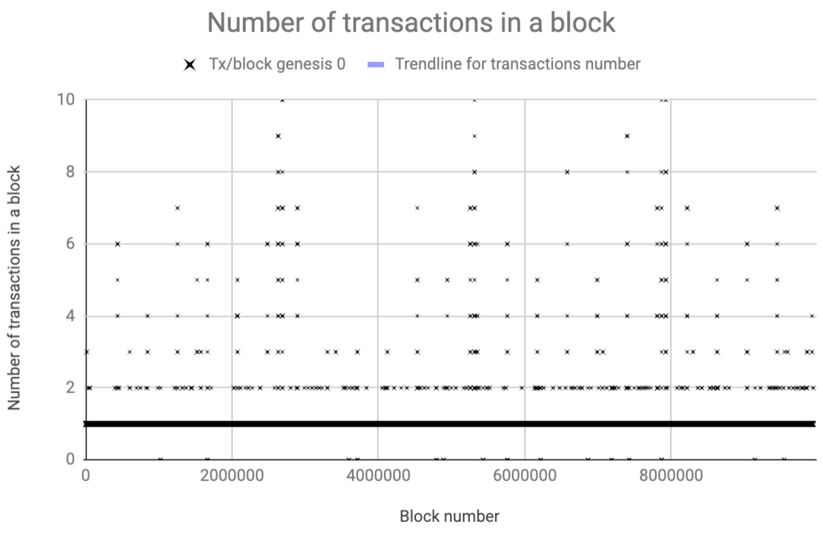

Figure 6: ETH: Block capacity

3.3 Hyperledger

Author: Julian
In Hyperledger Burrow the throughput was measured using the $Throughput_{alt}$ method mentioned in 2.1. In Table 4 the minimum, maximum and average time of a transaction is visualized. For 10 up to 1000 transactions the minimum and average time of one transaction is below 1 second. The maximum time for 1000 transactions is at around 25 secounds and it increases drastically to around 1000 seconds, if the transactions increase to 10000.

tx	**Min** [s]	**Max** [s]	**Avg** [s]
10	*0.436*	*0.632*	*0.530*
100	*0.379*	*0.644*	*0.475*
1000	*0.387*	*25.557*	*0.572*
10000	*0.359*	*999.583*	*3.457*

Table 4: Burrow: Min, Avg and Max execution time of a transaction

The throughput for the test with 10 transactions is displayed in Fig. 7 the upper left graph shows a more or less stable curve with only few a ups and downs and not much variation. The result for 100 (Fig. 7 upper right) transactions has a similar outcome. Starting the benchmark with 1000 (Fig. 7 lower left) or even 10000 (Fig. 7 lower right) transactions, the variance of some of these measurements is extremly high. In the last test, the effect even intensifies with more time passing.

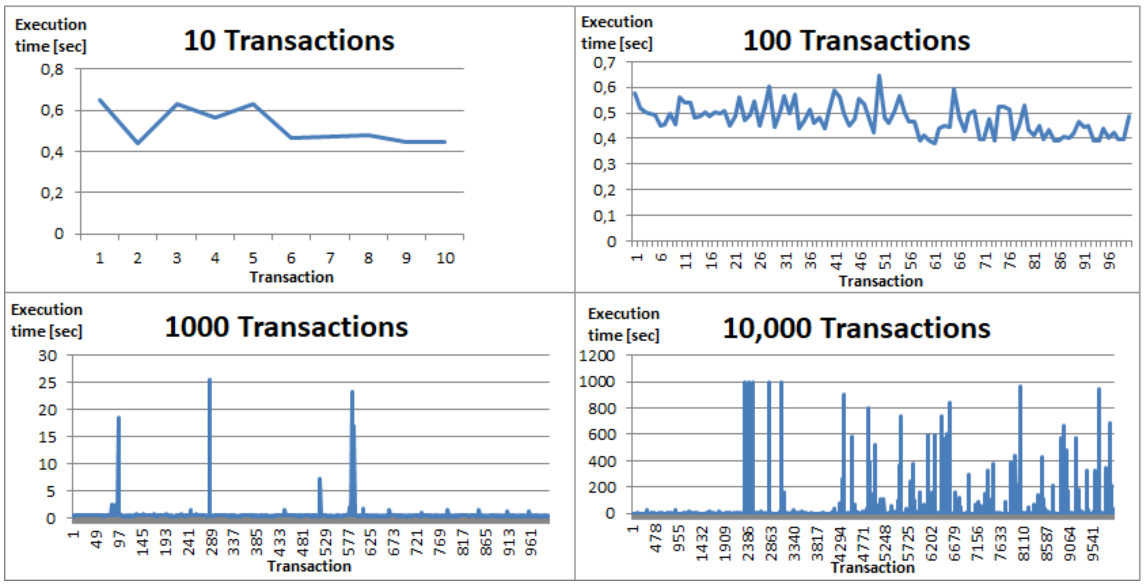

Figure 7: Burrow: Transactions and their execution time

The increase in the time needed to execute one transaction with a higher total number of transactions sent is part of the limitations of the used EC2 instance. The thread pool used to send multiple transactions at once were to many for the engine to handle and in the end the node ran out of free memory. This effect could be weakened by decreasing the thread pool size but would also result in a higher total execution time. A test was done with 1000 transaction and a pool size of 10 and 150. Where the total execution time of the first one was around 20 minutes, the second one was below 4 minutes.

The calculated throughput for these four benchmarks is shown in Fig. 8. It is apparent, that the throughput is decreasing the more transactions are sent. Where for the test with only 10 transactions it would sum up to above 80 transactions per second. The test with 10000 transactions resulted in less than 30 transactions per second.

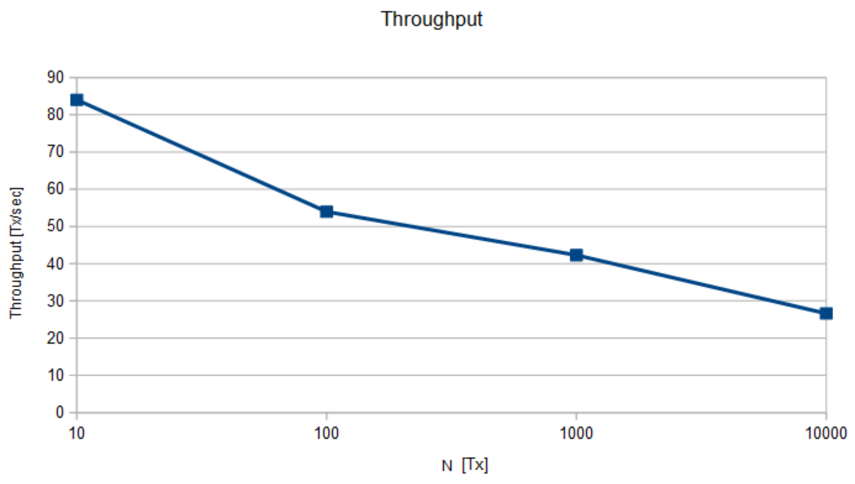

Figure 8: Burrow: Throughput

4 Conclusion

Author: Kim
For further research it would be necessary to investigate Hyperledger more intense. This is especially because of the lack of time that resulted in small numbers of transaction in the Hyperledger setup. To improve the results in that matter the number of transactions has to be increased.
In addition to that, Hyperledger was used with the Burrow framework only. That is why in further experiments the variety of frameworks that were tested has to be increased.

In our experiment the goal was to compare the two private blockchain networks Ethereum and Hyperledger with regard to its throughput for non-conflicting transactions.
To achieve this we proposed two mathematical definitions of throughput and used these to compare the two networks. For data collection we deployed several AWS EC2 instances and ran the networks as well as clients to send transactions on these. In both cases we could collect the data in a log-File and then had to run our own scripts to collect the necessary information. After collecting the data we were able to calculate the transaction throughputs for both, Ethereum and Hyperledger.
Comparing the results from Ethereum and Hyperledger leads to the conclusion, that Ethereum seems to be more efficient.
Nonetheless, this needs further research, especially to test it with higher number of transactions for Hyperledger.

References

[1] Amazon ec2 instance types. `https://aws.amazon.com/ec2/instance-types/`.

[2] Ethereum private network. `https://github.com/ethereum/go-ethereum/wiki/Private-network`.

[3] Geth. `https://github.com/ethereum/go-ethereum/wiki/geth`.

[4] Hyperledger Blockchain Performance Metrics. `https://www.hyperledger.org/resources/publications/blockchain-performance-metrics`.

[5] 3 things ethereum users should know about hyperledger burrow. `https://www.hyperledger.org/blog/2018/07/25/3-things-ethereum-users-should-know-about-hyperledger-burrow`, 2018.

[6] Hyperledger burrow git hub. `https://github.com/hyperledger/burrow`, 2018.

[7] Kyle Croman; Christian Decker; Ittay Eyal. On scaling decentralized blockchains, 2016.

[8] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2008.

[9] Péter Szilágyi. Clique poa protocol and rinkeby poa testnet. `https://github.com/ethereum/EIPs/issues/225`, 2017.