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

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 coordinator 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 [**?**]. This became the reason why blockchain is often associated with cryptocurrencies but potencial of this technology is actually much higher.

One of the big problems that prevents blockchain-based technologies from entering our everyday life is limited throughput. For example, limited block size in case of Bitcoin, caused the so-called "Bitcoin scalability problem" which resulted in intense research [**?**] 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 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 two widely used blockchain systems, Ethereum and Hyperledger, 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 [**?**].

Proof-of-Stake is a consensus algorithm where decisions are made based on weighted

3

random choice. It is calculated by participation duration in combination with the size of the stake. It's byzantine fault tolerant [**?**].

2   Method

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

2.1   Transaction throughput

In out 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 [**?**]. 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 to 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 [**?**]

In case of Ethereum, we created a private network with the Proof-of-Authority consensus algorithm. There are 2 nodes running on Geth [**?**]: 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 [**?**]. The block frequency F was set to 2 seconds. In order to guarantee that all transactions are non-conflicting, i.e. 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 ETH block period was setup to 0. That means that miner starts to mine a new blocks only When there are a new transactions. If there are no transactions in a blockchain mining frequency drops to 0 and the mining nodes wait for the new transaction to be authorised and added to the blockchain.

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 2 EC2 instances running. One for the Blockchain system running with one registered validator and one normal participant. The second instance is used to send the transactions to the first node.
To create a usable configurate 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 participants 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 switched 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 was implemented by Burrow via TCP. Because this resulted in a round-trip time of about one second, a thread pool was used to increase the number of parallel transactions.

3 Results

Blockchain is not a normal BASE or ACID database, where transactions are executed immediately after submitting. In case of blockchain transactions are stored in a blocks and than commonly added to a blockchain. As it is clearly visible in a 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.

Throughput was measured consequently with 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 on the level 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 a similar numbers - average filtered throughput between 75-80 tx/s.
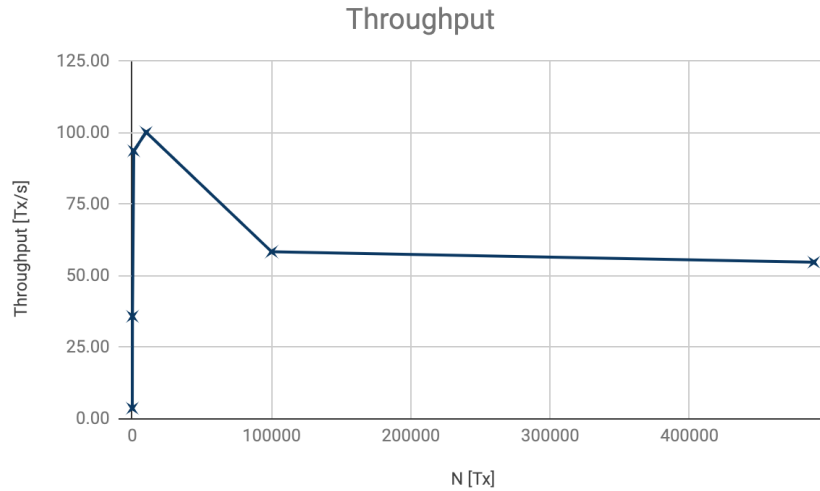

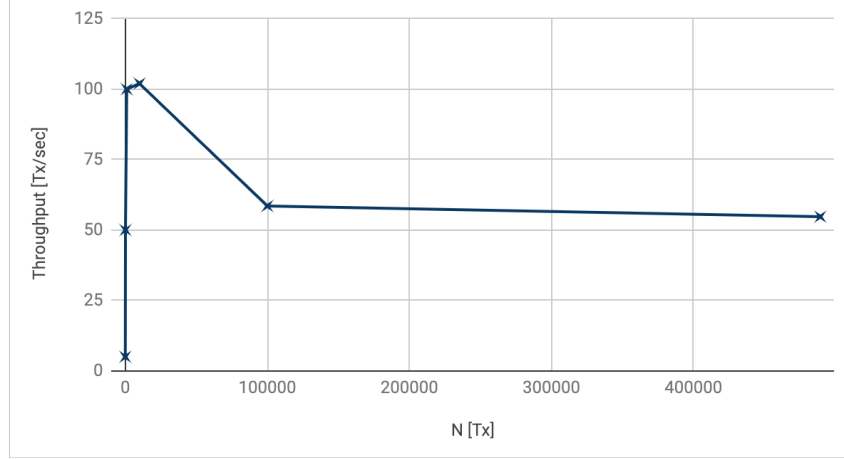Figure 1: ETH: Throughput as average number of transactions in a block divided by the block frequency

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

In a 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 as well data for 1 million of transactions. 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.
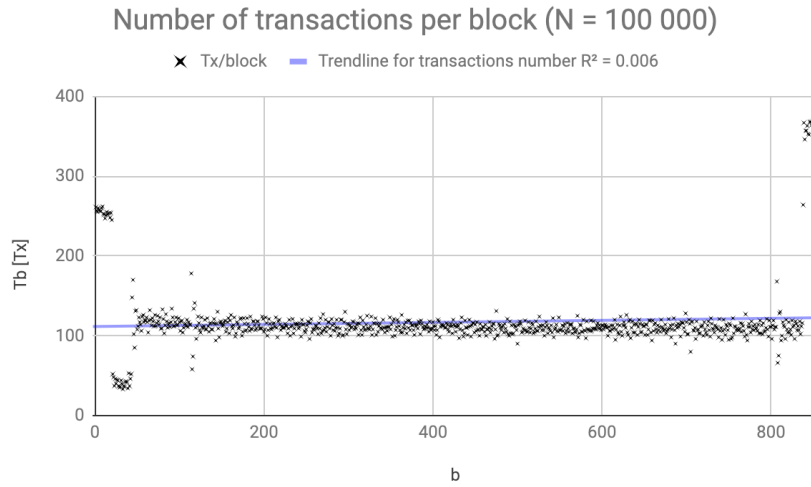

Figure 3: ETH: Block capacity in a higher blockchain load conditions

What is interesting we wanted to observe the drop and in a block capacity. This dorp is presented in a Fig. 4. We think that that might me result of an internal blockchain

balancing but we could not proof that. This problem needs further research.
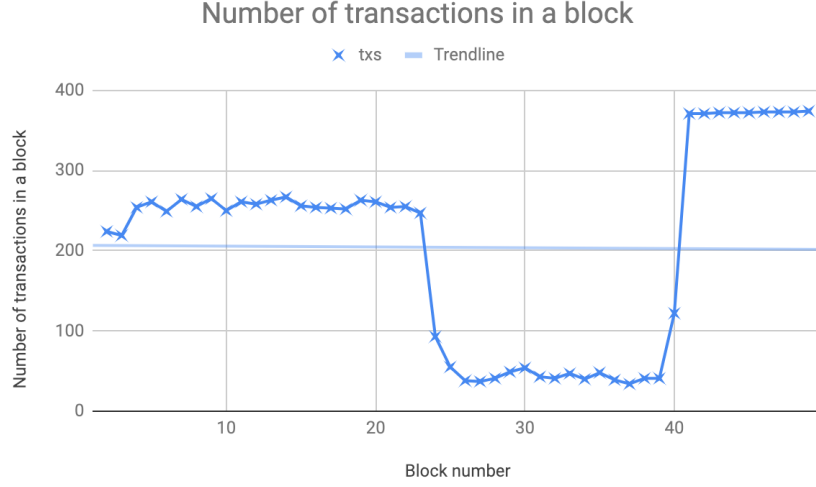
Figure 4: ETH: Drop in the block capacity

3.2 ETH measurements with block mining period 0s.

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 an presence of a transaction that is expected to be added to the blockchain. Using that method multiple transactions still might be fitted 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.

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

In a 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 first method. raised to in average 112,2 transactions per second with an 1,02 average number of transactions per block
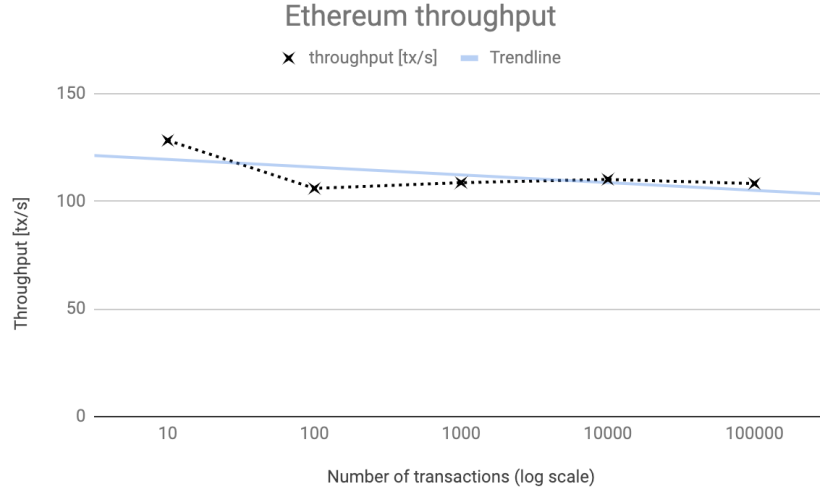

Figure 5: ETH: throughput on a log scale

In this setup in a 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 whole experiment.

3.3 Hyperledger

In Hyperledger Burrow the throughput was measured using the $Throughput_{alt}$ method mentioned in 2.1. In 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 more than 1000 transactions drastically increases from 25 seconds to around 1000 seconds for 10000 transactions.

The throughput for the test with 10 transactions as shown is figure 7 shows a more or less stable curve with only few up and downs and not much variation. The result for 100 (figure 8) transactions has a similar outcome. Starting the benchmark with 1000 (figure 9) or even 10000 (figure 10) transactions, the variance of some of these
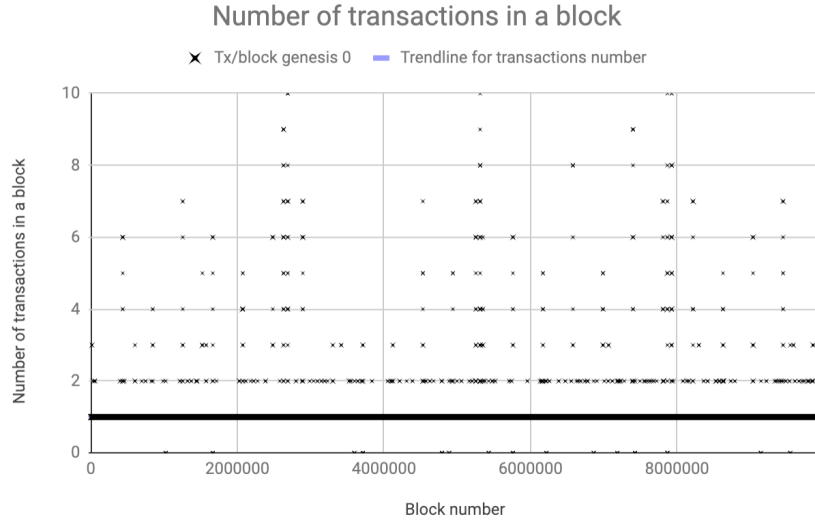
Figure 6: ETH: Block capacity

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

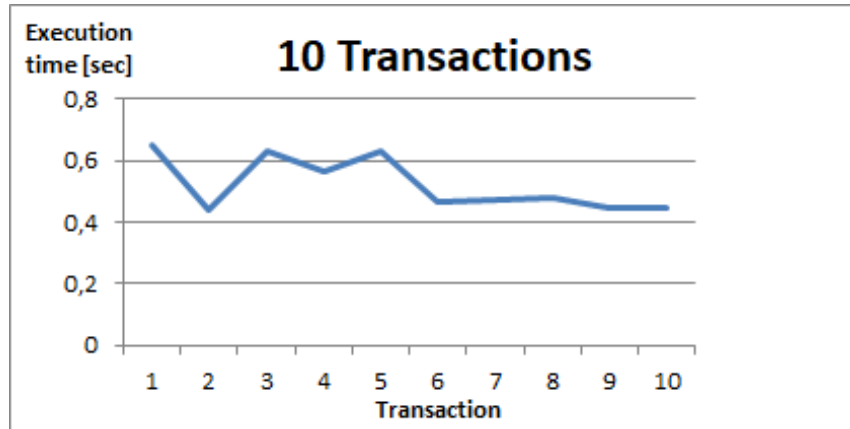measurements is extremly high. In the last test, the effect even intensifies with increased time passed.

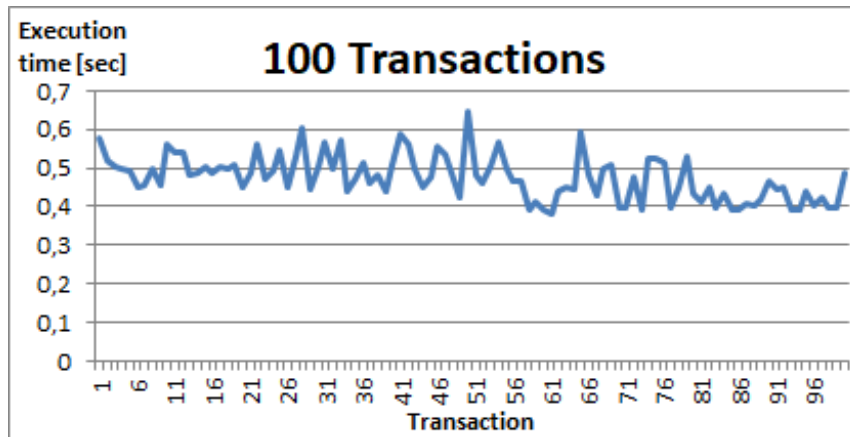
Figure 7: Burrow: 10 transactions and their execution time

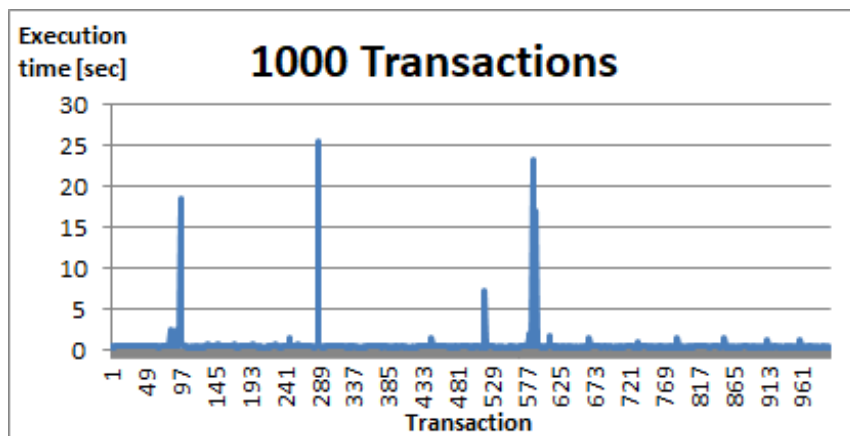Figure 8: Burrow: 100 transactions and their execution time

Figure 9: Burrow: 1000 transactions and their execution time
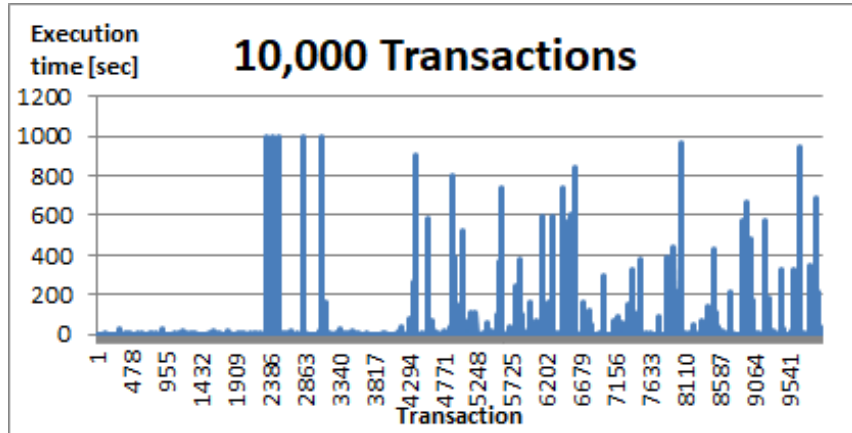
4 Future scope
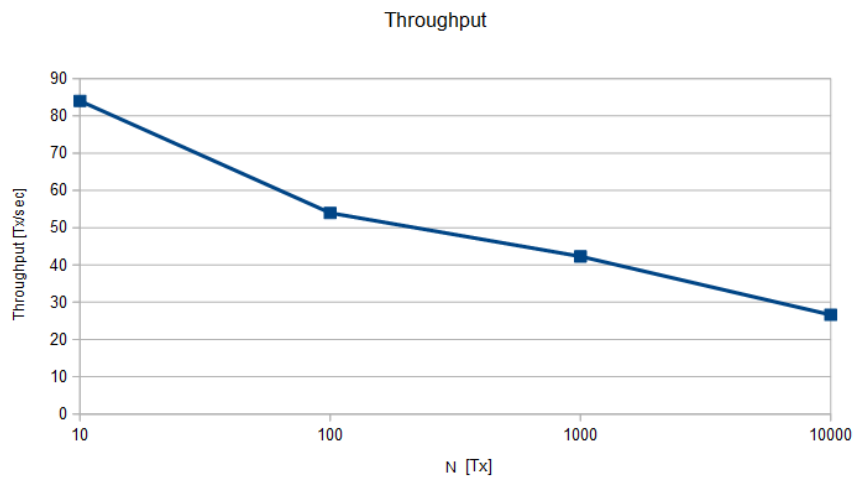
5 Conclusion

Figure 10: Burrow: 10000 transactions and their execution time

Figure 11: Burrow: Throughput