Experimental Analysis of the Processing Cost of Ethereum Blockchain in a Private Network

Iago S. Ochôa[1, 2], Rafael A. Piemontez[1], Lucas A. Martins[1]
Valderi R. Q. Leithardt[1], and Cesar A. Zeferino[1]

[1]Laboratory of Embedded and Distributed Systems – LEDS
University of Vale do Itajaí – UNIVALI
Rua Uruguai, 458 – C. P. 360 – 88302-901 – Itajaí– SC – Brazil

[2]Serviço Nacional de Aprendizagem Industrial – SENAI
Av. 1 de maio, 670 – 88353-202 – Brusque – SC – Brazil

{iago.ochoa, piemontez, lucasmartins}@edu.univali.br

{valderi, zeferino}@univali.br

Abstract. *Blockchain technology has aroused the interest of researchers as it ensures security and privacy in decentralized applications. One of the platforms currently used for the development of new blockchain-based solutions is the Ethereum network. This article presents an experimental analysis of this network to identify its gas cost and performance for several contracts in a simulated private Ethereum network. The results obtained demonstrate that storage operations for large amounts of data can prevent the use of blockchain due to the high gas cost charged for this type of operation. Also, the experiments have pointed out that processing operations which do not store information have a small gas cost.*

1. Introduction

The blockchain is a technology that applies community validation to maintain the ledger content synced and replicated through all the nodes. The blockchain nodes consent on the validity of a sequence of blocks which have the record of a series of transactions, making the blockchain a record of ordered transactions [Dinh et al. 2018, Saraf and Sabadra 2018, Aste et al. 2017]. This characteristic of replication provides support to the central premises of the blockchain, which include security, trustability, transparency, and decentralization. These premises are highly attractive for the development of several applications, and the cryptocurrencies are the most known.

In addition to applications for the cryptocurrency market, several other applications from different domains have employed the blockchain technology to develop more robust solutions. Privacy control and identity management are examples of the services that have been used blockchain in the development of new solutions [Saraf and Sabadra 2018] and have found in the blockchain a way to manage user identities independently of a centralizing authority. We can also cite the supply chain management area as another major use of this technology [Miller 2018]. Because a supply chain has multiple links with several authorities, which are responsible for the flow of processes, there has been a great interest in applying blockchain to offer more transparency to the

customers. The employment of the blockchain in the development of new solutions has brought a great focus to this technology, perhaps more than the one expected at the time it was designed, and some of its features are reaching their physical limitations.

With the increasing popularity and adoption of cryptocurrencies, such as Bitcoin and Ethereum, these applications require that a fundamental concept present in the initial design of this technology be rethought, i.e., its scalability. As digital currencies become more commonplace on a day-to-day basis, the number of transactions has increased exponentially [Chauhan et al. 2018]. With the number of transactions increasing day by day, the blockchain becomes too bulky, and each node needs to store all transactions to validate if those transactions have not been "spent" before. This blockchain characteristic directly influences the ability of nodes to replicate information in a short period to not spend a very high amount of time in the data replication process instead of the transaction validation process, which also tends to be a current bottleneck of this technology.

In addition to the problem of scalability due to the volume of data, constraints related to the block size and the time interval for generating a new block influence the capacity and processing time within the network. According to [Zheng et al. 2017], in the Bitcoin blockchain, the processing capacity is only seven blocks per second, while [Blom and Farahmand 2018] shows that the Ethereum blockchain reaches a capacity close to twenty transactions per second. These numbers are remarkably lower than the major electronic payment methods currently used, such as Paypal and Visa, which deal with thousands of transactions per second [Chauhan et al. 2018]. For the continuation of this technology as a viable base for large applications, it is necessary to develop new techniques that allow the blockchain to overcome the problems being identified today, which has brought considerable attention to this subject.

As mentioned before, the need for data replication on all blockchain nodes is one of the critical features that compromise network scalability. This feature is assessed in the work of [Decker and Wattenhofer 2013], in which the authors evaluate how the delay in the data replication and process validation may be the bottlenecks causing the forks in the Bitcoin blockchain. The authors evaluate these points through unilateral simulations, forcing the limits of the Bitcoin protocol by merely varying the behavior of the network nodes. This work serves as a reference for several other studies focused on the scalability analysis of blockchain-based networks.

A framework for the performance analysis of private blockchains is presented in [Dinh et al. 2017]. This framework simulates different workloads applied to a network with a fixed number of nodes for throughput. For the scalability tests based on network growth, the authors defined a fixed frequency of requests and increased the number of nodes. As results, they presented the behavior of three private blockchains for validation purposes, namely: Ethereum, Parity, and Hyperledger. This work also served as a basis for the development of the work presented by [Pongnumkul et al. 2017], in which the authors similarly performed an analysis of two networks through diverse workloads. In this last work, three different types of operation were used to evaluate how networks behave through operations with different levels of complexity. Similarly, we have defined the proposal for our work.

In this paper, we analyze the processing cost of the Ethereum blockchain in a smaller private network. We investigate the behavior of this network using three types of operation with different complexities. We have divided the operations into Linear Operations, Data Manipulation, and Sorting Algorithms. Our work evaluates the gas consumption, as well as the impact of the input parameter value of the algorithm in that consumption. In this sense, we are mainly evaluating the consumption of the selected algorithms in the Ethereum network and the limits of the blockchain to process these algorithms. This experimental analysis is the main contribution of the work and in this context lies our analysis of network scalability. From these tests, it was possible to identify the behavior of the Ethereum network for different types of algorithm. It was also possible to obtain the average gas cost for each operation performed in the network.

The remainder of this paper is structured as follows. Section 2 describes briefly the Ethereum blockchain, while Section 3 presents the proposed analysis used to evaluate the blockchain, including the simulation and test applied. Section 4 presents and discusses the experimental results. Concluding, Section 5 gives the final remarks.

2. Ethereum Blockchain

The Ethereum platform was introduced to the world by [Buterin 2014]. Ethereum is not just a cryptocurrency, but it is also a programming language running in a blockchain. This platform allows users to create smart contracts and perform the transfer of cryptocurrency to other users by validating these contracts. According to [Mohanta et al. 2018], a smart contract is a computer program that self-checks and self-executes, and is anti-fraud. Such smart contracts are intended to facilitate negotiation between unknown persons, eliminating the need for a third-party to audit the negotiation. The existing applications for smart contracts are diverse, such as prevention of violation, financial services, and application of credit, among others. The validation of contracts and information contained in the blockchain employs mining processes.

The mining process consists of the execution of a consensus algorithm. This type of algorithm is used to validate the information transmitted in a distributed network. As an example, consider the double spending problem. A user named Bob, who has only ten coins, intends to send these coins to Alice and Ana at the same time. If there is no adequate consensus algorithm, Ana and Alice will each receive ten coins, generating double spending.

The Ethereum platform employs the PoW (Proof-of-Work) consensus algorithm, which was proposed in [Dwork and Naor 1992]. This algorithm consists of solving a cryptographic problem that satisfies all the data present in the blocks of the blockchain. In a consolidated network, the PoW algorithm improves security, but it is energy inefficient. According to [Kim and Kim 2018], the energy consumption generated by the Ethereum and Bitcoin blockchains already surpasses the energy consumption of Syria, which is the 72^{nd} country with the highest consumption of electricity in the world.

Another consensus algorithm is the PoS (Proof-of-Stake), which was proposed by [Sunny King 2012] to solve the energy issues and guarantee security to the information contained in the blockchain. This algorithm defines the miner of the next block by the

amount of cryptocurrency that a node has. The user with the highest number of cryptocurrency could be defined as the miner of the next block. However, to avoid the monopoly of the network by whoever has the higher amount of cryptocurrency, the process uses a random number generator to define the mining node. This algorithm becomes advantageous over PoW because it spends less electricity on the mining process. Currently, the Ethereum platform uses an optimized version of PoW, but it is estimated that it will be completely migrated to PoS in 2019.

When the network confirms the validation of a block, the miners of the block receive a specific value in Ether for the work done. It is necessary to mention that the cryptocurrency of the Ethereum platform has pseudonyms (see Table 1) that are also used to charge fees for users using the Ethereum blockchain. Each operation performed on the Ethereum platform has a cost called *gas* [Wood 2014]. The cost of gas varies for each operation. For instance, storing a contract on the Ethereum platform has a cost of 32,000 gas, while storing a value with 256-bit size has a cost of 20,000 gas.

Table 1. Ethereum pseudonyms

Multiplier	Pseudonym
10^0	Wei
10^{12}	Szabo
10^{15}	Finne
10^{18}	Ether

The users of the network set the gas price. At the time of the development of this work, the cost of a gas corresponded to 3.6 GWei. It is worth noting that the gas price varies according to the operation of the network. The amount previously shown is the standard gas price, which means that the user paying that price for the used gas will have their transaction confirmed on the Ethereum platform within 30 minutes. If he/she needs to confirm the transaction be more quickly in the blockchain, he/she should pay 15 GWei perused gas, and the transaction will be confirmed in less than 2 minutes. Information regarding the Ethereum platform values can be obtained in [EthStats 2018].

With the popularization of the platform, the number of users in the Ethereum network has been growing exponentially. Figure 1 shows the increase of addresses in the platform since 2015. Nowadays (Dec. 2018), the platform has about 48 million active addresses.

3. Implementation

This section describes the methods used to evaluate the processing costs of the Ethereum network.

We have implemented eight contracts of three different classes: Linear Operations, Data Manipulation, and Sorting Algorithms. As development and experimental platform, we have employed Truffle Suite to implement, compile, and run the tests, and Ganache software to build the private blockchain network.

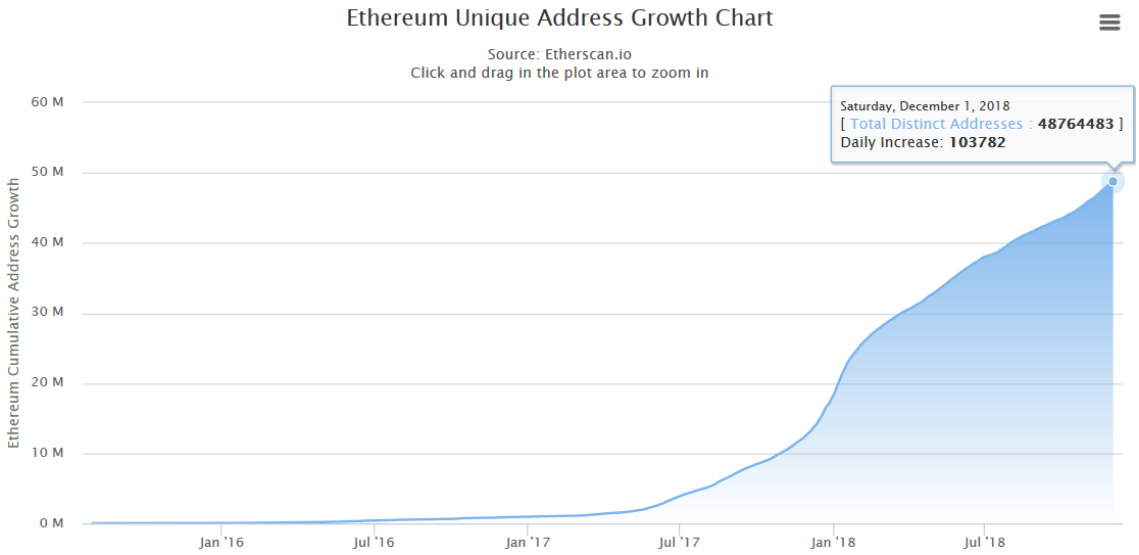

Figure 1. Address growth on the Ethereum platform from 2015 to 2018 (data obtained using [Etherscan 2018]).

We have employed the Linear Operations class for assessing and identifying whether the Ethereum network has a pattern of growth in the gas cost and the performance of data processing. This class is composed of two contracts named Single_Loop and Double_Loop. These contracts execute one and two nested loops, respectively, which do not perform any computation or access to the blockchain (see Algorithms 1 and 2).

Algorithm 1: Single_Loop

Input : n, Number of iterations
for $i \in \{0, \ldots, n-1\}$ **do**
end
return 1

Algorithm 2: Dual_Loop

Input : n, Number of iterations in each loop
for $i \in \{0, \ldots, n-1\}$ **do**
  for $j \in \{0, \ldots, n-1\}$ **do**
  end
end
return 1

The Data Manipulation class was intended to assess the gas cost and performance of operations that update the blockchain. This class comprises three contracts: (*i*) Store_nInt, which performs several stores of an integer variable into the blockchain; (*ii*) Store_nChar, which updates several times a char variable in the blockchain; and (*iii*) Store_String, which saves a string into the blockchain. Algorithms 3–5 illustrate these contracts.

Algorithm 3: Store_nInt

Input : n, Number of store operations in the blockchain
Data: x, Integer variable in the blockchain
$x \leftarrow 0$
for $i \in \{0, \ldots, n-1\}$ **do**
| $x \leftarrow x + i$
end
return 1

Algorithm 4: Store_nChar

Input : n, Number of store operations in the blockchain
Data: x, Char variable in the blockchain
for $i \in \{0, \ldots, n-1\}$ **do**
| $x \leftarrow Generate\_Char()$
end
return 1

Algorithm 5: Store_String

Input : n, Size of the string to be stored in the blockchain
Data: x, String variable in the blockchain
$x \leftarrow Generate\_Char\_Array(n)$
return 1

The contracts of the Ordering Algorithms class were planned to evaluate the performance of the network for operations requiring high processing. This class is composed of three algorithms (Bubble_Sort, Quick_Sort, and Merge_Sort) that perform the ordering of the values of a linear array. The contracts are also responsible for generating an unsorted array before ordering it.

Table 2 summarizes the computational complexity of each contract employed in the experiments. It is worth highlighting that all the contracts of the Data Manipulation class have low complexity because they focus on identifying the gas cost related to the updating of the blockchain.

Table 2. Computational complexity of the contracts

Contract name	Contract class	Best Case	Worst Case
Single_Loop	Linear Operations	$O(n)$	$O(n)$
Double_Loop	Linear Operations	$O(n^2)$	$O(n^2)$
Store_String	Data Manipulation	$O(n)$	$O(n)$
Store_nChar	Data Manipulation	$O(n)$	$O(n)$
Store_nInt	Data Manipulation	$O(n)$	$O(n)$
Bubble_Sort	Ordering Algorithms	$O(n)$	$O(n^2)$
Quick_Sort	Ordering Algorithms	$O(n.\log n)$	$O(n^2)$
Merge_Sort	Ordering Algorithms	$O(n.\log n)$	$O(n.\log n)$

3.1. Simulations and Tests

After defining the contracts, we have applied tests to evaluate their performance. Each contract has a single input parameter (n) whose meaning is defined in Table 3.

Table 3. Meaning of the input parameter (n) in each contract

Contract	Input parameter
Single_Loop	Number iterations of the loop
Double_Loop	Number of iterations in each loop
Store_String	Size of the string to be stored into the blockchain
Store_nChar	Number of times it stores a char into the blockchain
Store_nInt	Number of times it stores an integer into the blockchain
Bubble/Quick/Merge_Sort	Size of the array to be ordered

The algorithms listed above allowed to evaluate the results regarding the processing and gas cost of the network.

Each test was performed sequentially through a script written to request 12 operations of each contract. The parameter n was set to vary from 2 to 4096, in discrete steps based on powers of 2. All the tests were performed five times to obtain an average value in the processing time.

The Ethereum network allows building three types of function for contracts. Standard functions allow to read and store data in the blockchain. View functions allow only to view the data in the blockchain. Finally, Pure functions, which do not allow reading and storing in the blockchain, but allow to use the network as a processing source. Due to this difference, two test groups were developed, according to Table 4. The Pure group was built to evaluate contracts that do not make any modification in the database. The Standard group was developed to evaluate the contracts that make modifications in the database and to identify differences in the gas cost. As the contracts of the Data Manipulation class modify the database, we did not use it in the Pure group tests.

Table 4. Evaluation type x Test group

Evaluation type	Pure group	Default group
Linear Operations	•	•
Data Manipulation		•
Sorting Algorithms	•	•

The tests were performed on a computer with an Intel® i5-5200U processor, with 4 cores, and 8GB of RAM running Manjaro Linux distribution.

4. Experimental Results

This section presents the results of the experiments carried out with the Ethereum network for the three different classes of contract. The analysis is performed based on the gas cost and runtime (i.e., the execution time) of each contract.

4.1. Linear Operations

Figure 2 presents the gas cost and the runtime for the Linear Operations contracts. The contracts of the Pure group did not consume gas, and they are not presented in the Gas Cost chart. We observe that the standard Single_Loop contract has a linear growth pattern, while the Dual_Loop presents a quadratic growth, as described in Table 2. Also, we notice that the Dual_Loop contract exceeded the limit of gas cost and was not able to execute more than 256 iterations per loop. Moreover, it is worth noting that there is no significant difference in execution time between the Pure and the Standard groups.

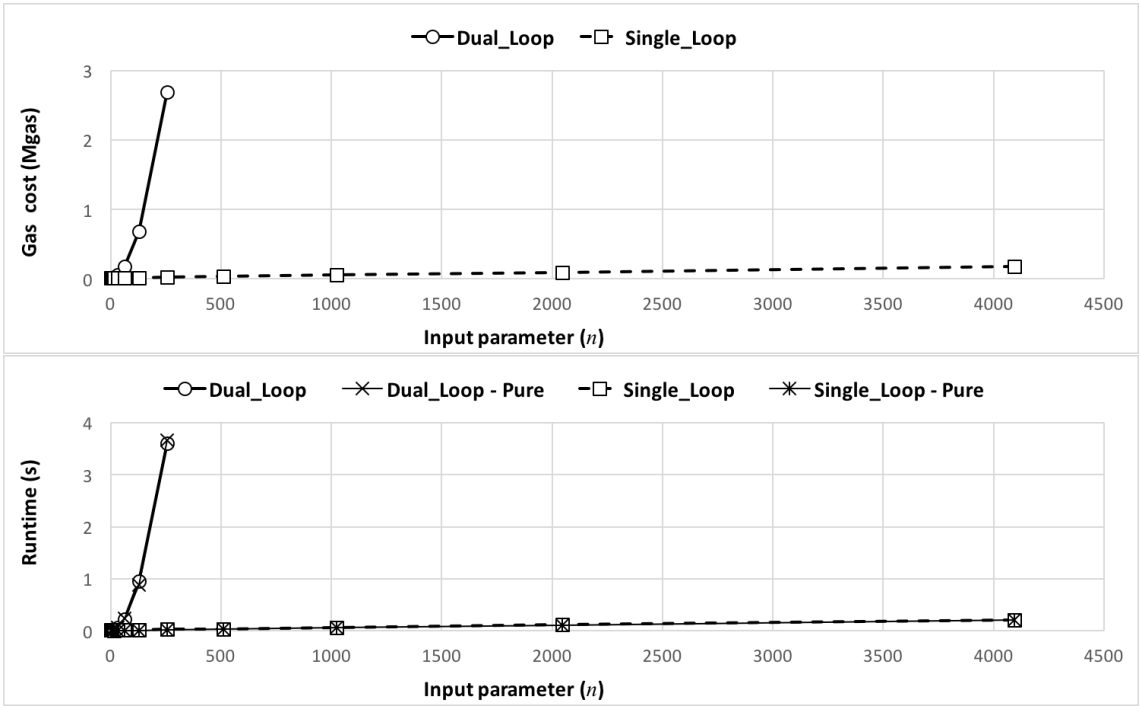

Figure 2. Gas cost and runtime of the Linear Operations contracts.

Figure 3 presents the runtime gas cost, which is a metric that expresses the ratio of gas consumption in time; it is given in Mgas per second (or Mgas/s). The average cost of the Single_Loop contract equals 0.87 Mgas/s, with a minimum of 0.66 Mgas/s and a maximum of 1.17 Mgas/s.

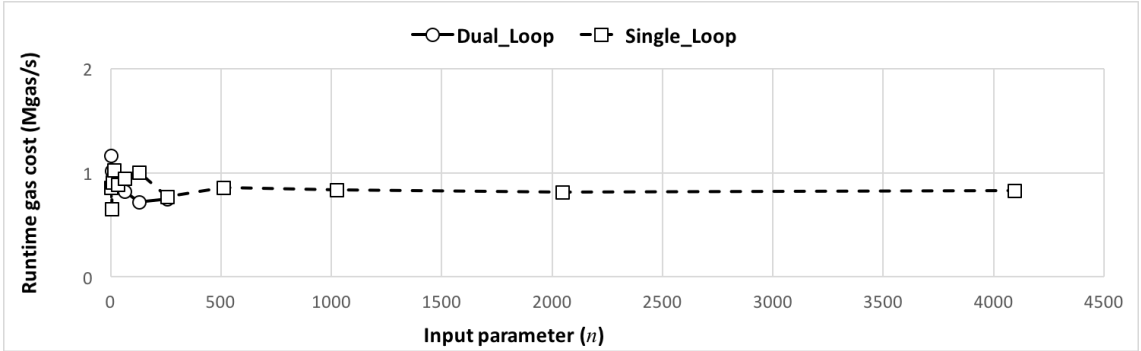

Figure 3. Runtime gas cost for Linear Operations contracts.

4.2. Data Manipulation

Figure 4 presents the gas cost and runtime results for the Data Manipulation contracts. Figure 5 identifies the cost per update for each one of these contracts. We observe that these algorithms also have a linear behavior in network processing and gas cost for the operations executed. We also notice that the contract that store the shorter data (Store_Char) consume more gas per update, but have better scalability.

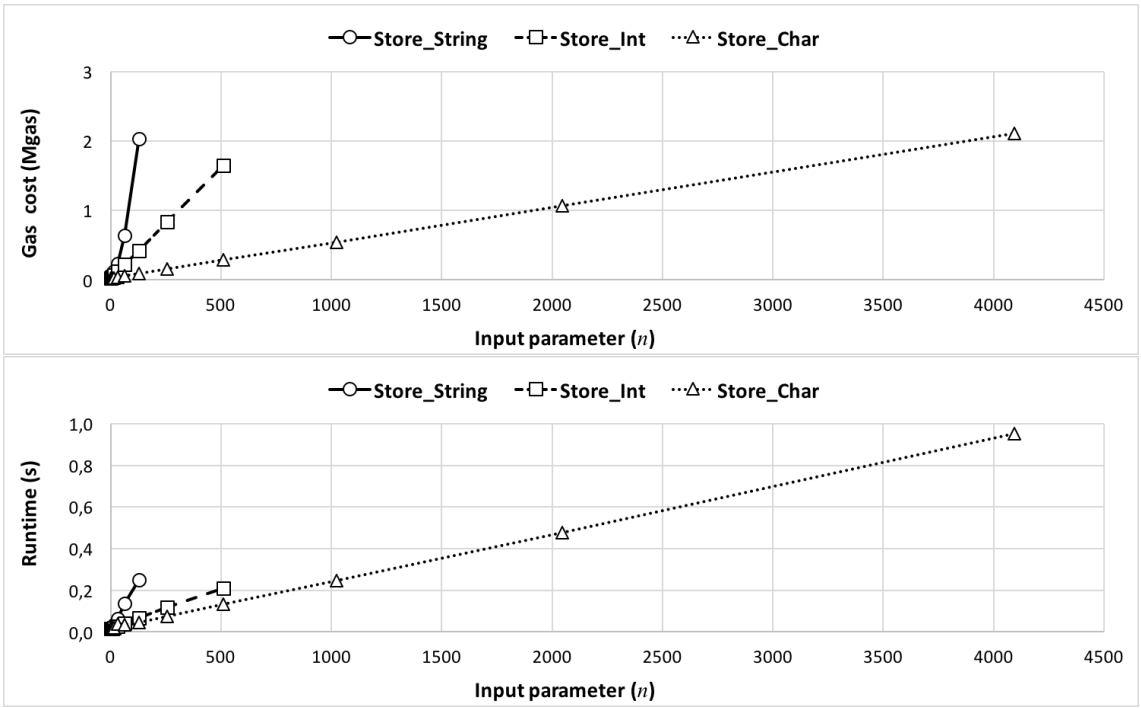

Figure 4. Gas cost and runtime of Data Manipulation contracts.

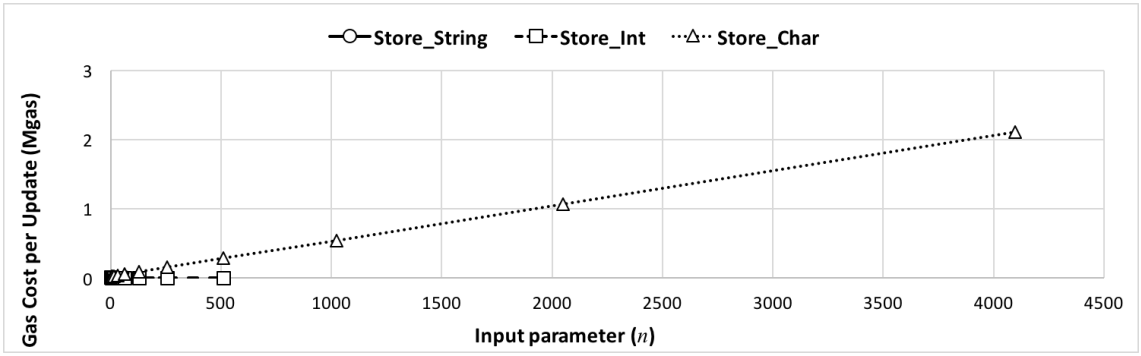

Figure 5. Gas cost per update.

The experiments shown that the Data Manipulation contracts have a high runtime gas cost (Figure 6). Its minimum cost (1.18 Mgas/s) is greater than the higher cost of the Linear Operation contracts (1.16 Mgas/s), reaching a maximum cost of 8.09 Mgas/s in StringChange operations.

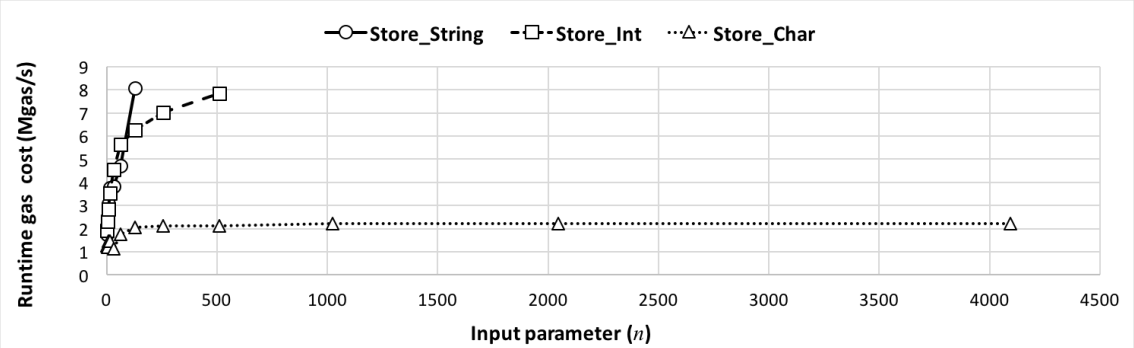

Figure 6. Runtime gas cost for Data Manipulation contracts.

4.3. Sorting Algorithms

As expected, the Sorting Algorithm contracts have the longest processing time, reaching the cost of 2.8 Mgas in 4 seconds of processing in BubbleSort ordering contracts. As Figure 7 shows, this same contract is only able to order vectors of up to 128 items because tests with more operations exceeded the gas limit, different from the much more complex QuickSort contract, which was able to perform all the orderings in the experiments. As in the Linear Operations class, the execution time of the Pure group implementations is similar to the one of the Standard implementations.

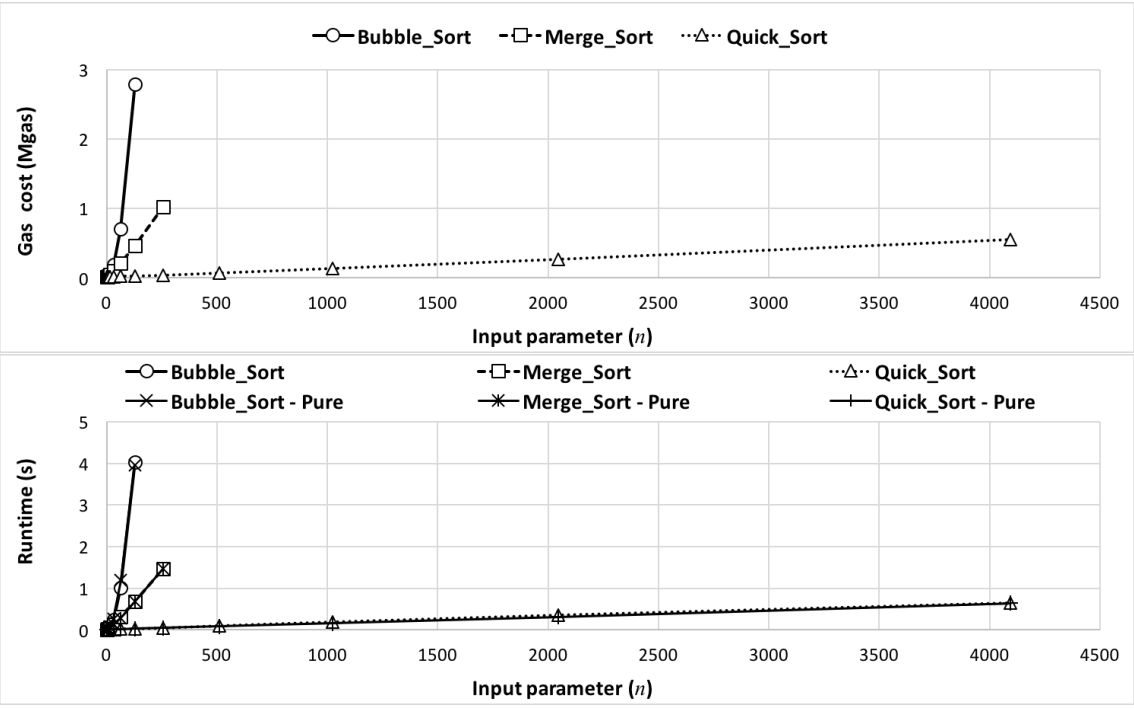

Figure 7. Gas cost and runtime for Ordering Algorithms contracts.

Although the gas cost and the processing time are higher than the ones observed in the other classes, this class has an average runtime gas of 0.6 Mgas/s. This cost is close to the one obtained with the Linear Operations algorithms and lower than the Data Manipulation class. These results are evidence that contracts that perform operations in the blockchain are much more costly, as Figure 8 shows.

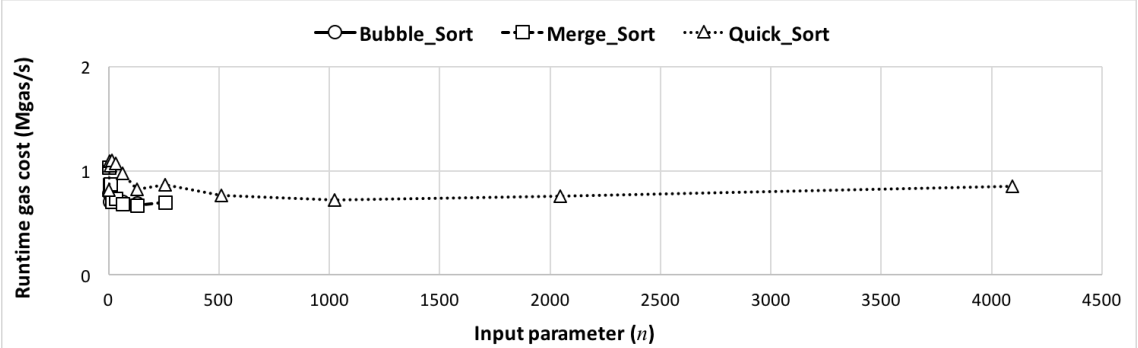

Figure 8. Runtime gas cost for Ordering Algorithms contracts.

The results of this study questioned the ability of a private network as a processing source and base for blockchain data storage, which, despite a solution for simple computational problems, brought a high data processing time in the tests performed and a high gas cost in operations that carry out large amounts of records in the blockchain.

5. Conclusions

In this paper, we have presented an experimental analysis of the processing cost of the Ethereum blockchain in a private network. From the tests performed, we have observed that algorithms that use the Ethereum blockchain for data processing are not efficient because some operations take too many time to process a request. For instance, it takes around four seconds to sort an array of 128 positions. While using it for linear data processing is not good, making use of the Ethereum blockchain to process data in parallel might be useful. However, further studies on this topic are necessary to confirm this hypothesis.

The process of designing and creating a smart contract must be carefully thought to decide which types of data are going to be used to store the information into the blockchain. From the tests performed using data manipulation, we have identified that requesting changes of a single character of a text variable is four times more expensive than requesting changes of integers. Furthermore, we have found that the cost for manipulating changes to a string of 128 characters is six times higher than to manipulate only two characters of text, costing more than two thousand gas for a single operation. Based on that, we can conclude that using the Ethereum blockchain to store a significant amount of data is not the ideal scenario, although we can use it to store thousands of small registers for the same price as a big one. As an overall conclusion, we can assume that the Ethereum blockchain presents a lack of scalability because the network struggled to handle a small number of complex operations. The source code used in the tests can be found in the LEDS repository provided in GitHub [Ochôa et al. 2019].

As future work, we intend to simulate cryptocurrency operations into the network, because they are supposed to be the core application for this network. After that, we are going to be able to compare if the complexity of the transactions used in this work relates to the expected by the Ethereum blockchain core design. Also, we plan to perform the same tests on the Ethereum public network to compare the gas and monetary costs.

Furthermore, we intend to perform an analysis of the gas costs for real-world applications to assess how costly these applications are in an Ethereum-based network.

Acknowledgments

This work was financed by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Finance Code 001 and by Fundação de Amaparo à Pesquisa e Inovação do Estado de Santa Catarina – Brasil (FAPESC) – Grant No. 2019TR169.

References

Aste, T., Tasca, P., and Di Matteo, T. (2017). Blockchain technologies: The foreseeable impact on society and industry. *Computer*, 50(9):18–28.

Blom, F. and Farahmand, H. (2018). On the scalability of blockchain-supported local energy markets. In *International Conference on Smart Energy Systems and Technologies*, SEST'18, pages 1–6.

Buterin, V. (2014). Ethereum: A next-generation smart contract and decentralized application platform. `https://etherscan.io/chart/address`. Accessed: 29-11-2018.

Chauhan, A., Malviya, O. P., Verma, M., and Mor, T. S. (2018). Blockchain and scalability. In *IEEE International Conference on Software Quality, Reliability and Security Companion*, QRS-C'18, pages 122–128.

Decker, C. and Wattenhofer, R. (2013). Information propagation in the bitcoin network. In *IEEE 13th International Conference on Peer-to-Peer Computing*, P2P'13, pages 1–10.

Dinh, T. T. A., Liu, R., Zhang, M., Chen, G., Ooi, B. C., and Wang, J. (2018). Untangling blockchain: A data processing view of blockchain systems. *IEEE Transactions on Knowledge and Data Engineering*, 30(7):1366–1385.

Dinh, T. T. A., Wang, J., Chen, G., Liu, R., Ooi, B. C., and Tan, K.-L. (2017). Blockbench: A framework for analyzing private blockchains. In *ACM International Conference on Management of Data*, SIGMOD/PODS'17, pages 1085–1100.

Dwork, C. and Naor, M. (1992). Pricing via processing or combatting junk mail. In *Annual International Cryptology Conference*, CRYPTO'92, pages 139–147.

Etherscan (2018). Ethereum unique adress growth chart. `https://etherscan.io/chart/address`. Accessed: 29-11-2018.

EthStats (2018). Ethereum statistics. `https://ethstats.net/`. Accessed: 29-11-2018.

Kim, S. and Kim, J. (2018). POSTER: Mining with Proof-of-Probability in blockchain. In *Asia Conference on Computer and Communications Security*, ASIACCS'18, pages 841–843.

Miller, D. (2018). Blockchain and the Internet of Things in the industrial sector. *IT Professional*, 20(3):15–18.

Mohanta, B. K., Panda, S. S., and Jena, D. (2018). An overview of smart contract and use cases in blockchain technology. In *9th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, pages 1–4.

Ochôa, I. S., Piemontez, R. A., Martins, L. A., Leithardt, V. R. Q., and Zeferino, C. A. (2019). Experimental analysis of the processing cost of Ethereum blockchain in a private network: source code. `https://github.com/UNIVALI-LEDS/WBlockchain-2019`.

Pongnumkul, S., Siripanpornchana, C., and Thajchayapong, S. (2017). Performance analysis of private blockchain platforms in varying workloads. In *26th International Conference on Computer Communication and Networks*, ICCN'17, pages 1–6.

Saraf, C. and Sabadra, S. (2018). Blockchain platforms: A compendium. In *IEEE International Conference on Innovative Research and Development (ICIRD)*, ICIRD'18, pages 1–6.

Sunny King, S. N. (2012). PPCoin: Peer-to-peer crypto-currency with proof-of-stake. *self-published paper*.

Wood, G. (2014). Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper*, 151:1–32.

Zheng, Z., Xie, S., Dai, H., Chen, X., and Wang, H. (2017). An overview of blockchain technology: Architecture, consensus, and future trends. In *IEEE International Congress on Big Data*, BigData Congress'17, pages 557–564.