Performance and Scalability of Private Ethereum Blockchains

Markus Schäffer, Monika di Angelo(✉), and Gernot Salzer

TU Wien, Vienna, Austria
{markus.schaeffer,monika.di.angelo,gernot.salzer}@tuwien.ac.at

Abstract. Smart contracts provide promising use cases for the public and the private sector by combining cryptographically secure blockchains and the versatility of software. In contrast to public blockchains, private ones can be tailored by configuring blockchain-specific parameters like the time passing between two consecutive blocks, the size of blocks, the hardware of the nodes running the blockchain software, or simply the size of the network. However, the effects of parameters on the performance of private smart contract platforms are not well studied.

In this work, we systematically examine to which extent the performance of private Ethereum blockchains scales with various parameters, and which parameters constitute bottlenecks. We introduce a concept for measuring the performance and scalability of private Ethereum smart contract platforms, as well as a framework for the automatic deployment of differently configured private Ethereum blockchains on the cloud. Based on the collected performance-related data, we visualize the impact of parameter changes on performance. Our results show that the effect of variations in one parameter is highly dependent on the configuration of other parameters, especially when running the system near its limits. Moreover, we identify a structure for the bottlenecks of current private Ethereum smart contract platforms.

Keywords: Blockchain · Ethereum · Evaluation · Performance

1 Introduction

For blockchain systems, there is currently a trade-off between decentralization, security, and scalability. This is known as the scalability trilemma [9], which states that a blockchain can only have two of the three properties.

For the most prominent smart contract platform Ethereum, each mining node stores the entire state (i.e. for each account its balance, code and storage) and also processes all transactions sequentially. This approach provides a high amount of security, yet greatly restricts scalability. Since there is no parallel processing in Ethereum mining, the throughput is currently capped at around 15 transactions per second in the public network [9].

This paper is a condensed version of the measurements described in [19].

C. Di Ciccio et al. (Eds.): BPM 2019 Blockchain and CEE Forum, LNBIP 361, pp. 103–118, 2019.
https://doi.org/10.1007/978-3-030-30429-4_8

In order to drastically increase scalability of Ethereum, its key developers focus on a combination of two approaches: sharding [9] and side-chains [7]. The former approach requires only a small percentage of nodes to see and process every transaction of the network, thereby allowing transactions to be processed in parallel by means of horizontal partitioning. The latter involves creating additional chains for transactions to be executed off the main chain.

For private networks, the scalability trilemma can also be tackled by choosing non-default values for the chain parameters (e.g. block size, block interval, or power of a mining node) to improve performance. The choice of these blockchain parameters affects throughput and latency. Even though Ethereum is comparatively well studied, there has been only little discussion about the effects of the different chain parameters on performance in a private setting so far (see Sect. 2).

Goals and Methods. The overall aim of this work is to further extend the understanding of the effects of different parameters in private Ethereum smart contract platforms with respect to performance and scalability. In this context, we specifically address the following research questions:

- What are suitable means for measurements?
- What are the effects of different parameter settings?
- Which parameters represent bottlenecks?

In order to answer these questions, we first devise a concept for measuring the performance of differently configured Ethereum networks. This includes the identification of relevant parameters, the definition of evaluation metrics, and the development of a deployment mechanism. After the implementation of the formulated concept as a measurement framework, the actual performance measurements are carried out to collect data. For the data analysis, we visualize the effects of parameters on performance, and we provide interpretations. Furthermore, we identify a bottleneck structure for the measured parameters. Finally, we re-examine findings reported in related work.

Roadmap. Section 2 summarizes related work. Our measurement concept is detailed in Sect. 3, and we analyze the collected data in Sect. 4. Identified bottlenecks are presented Sect. 5. In Sect. 6, we discuss our results in relation to related work, while we draw our conclusions in Sect. 7.

2 Related Work

In the most closely related work [4,5], the authors present an evaluation framework where private blockchains can be benchmarked against pre-defined workloads. In order to compare different blockchains, four abstraction layers (application, execution engine, data model and consensus) and according workloads are defined. The evaluation of Ethereum used only the proof of work (PoW) consensus algorithm, while the configuration of the nodes was not varied.

The performance and limitations of Hyperledger Fabric and Ethereum with varying numbers of transactions are studied in [17]. Metrics measured execution time, latency, and throughput. However, the consensus mechanism was disabled and the analysis was performed on a single-node network. In addition, the configuration of the node was not varied, and other parameters such as the mining difficulty of Ethereum were not investigated either.

The effect of different Ethereum clients (Geth and Parity) with respect to performance was studied in [18]. Even though the analysis was conducted with different types of nodes (different amount of RAM is mentioned), there is no information about the consensus algorithms and its parameters (e.g. mining difficulty or block frequency) used. Apparently, the number of nodes was not varied during the experiments.

A comparison of blockchain and relational databases is presented in [2]. For testing purposes, Ethereum and MySQL were chosen. Although seven differently configured machines have been used, no information is provided on how the configuration of the machine affects the measured metrics. While it is stated that results vary from computer to computer, results are provided for a single configuration only. In addition, the configuration of the machines resembles consumer machines such as laptops. Moreover, the size of the private Ethereum blockchain was fixed to six nodes and not varied. Consensus algorithms and their parameters were not included in the study.

Overall performance metrics and a performance monitoring framework are proposed in [24]. The authors provide detailed metrics for measuring performance on the Ethereum blockchain. Still, their analysis lacks some parameters: only one consensus algorithm (PoW) is covered; mining difficulty or block size are not varied; the configuration of nodes is not addressed.

A quantitative framework for analyzing the security and performance tradeoffs of various consensus and network parameters of PoW blockchains is presented in [12]. However, it contains neither other consensus variants and their parameters nor any other number and type of node. A model predicting the performance and storage of executing contracts based on the transaction volume is proposed in [23]. It consists of formulas which were derived via regression analysis. The major drawback of this approach is that it only depends on the amount of transactions. The formulas do not include other variables such as consensus algorithm, or amount and configuration of the nodes. The authors of [11] deal with Bitcoin's scalability limits via proposing a new blockchain protocol which is designed to scale. Variables such as block frequency and block size were varied to make suggestions for a better blockchain protocol. For Hyperledger, there are a few tools and studies [13,16,21] that use similar ideas as already presented.

Bottlenecks in Bitcoin that limit throughput and latency are addressed in [3]. The results show that a re-parametrization of block size and block intervals may have a positive effect on performance and scalability. The authors of [22] proposed to improve scalability by optimization of block construction, block size and time control optimization, and transaction security mechanism optimization. Experiments were conducted that studied the relationship of block size

and block construction. The authors of [15] state that existing Byzantine tolerant permissioned blockchains only scale to a limited number of nodes. A different design of blockchains is proposed in [14] where the authors argue that, for improving the performance and scalability of blockchains to a significant level, simply tweaking blockchain parameters such as block size is not enough.

In conclusion, previous work primarily focused on block frequency and block size for blockchain systems in general. However, other parameters such as node configuration or network size were hardly studied. For Ethereum in a private setting, only very few researchers reported performance measurements with differently configured networks, while none included results with the proof of authority (PoA) consensus variant. As PoA may be better suited for private blockchains than PoW, an important part is missing in the existing literature.

3 Concept

For presenting our concept in this section, we first identify parameters that affect the performance and scalability of Ethereum in a private setting. Then, we define the metrics used in our analysis, and finally we introduce the experimental setup.

3.1 Identified Parameters

Table 1 lists the identified parameters, where the ones used for our experiment are set in bold. The increase of block frequency may have a positive effect on performance [3], while the block propagation time imposes a lower bound. Regarding the effect of the block size on performance, the related work provides contradictory statements. While a positive effect of block size increase on performance is reported in [3], no positive effect is implied in [5]. Concerning workload, different smart contracts may show different runtimes according to their instructions and storage access. For example, [5] and [24] report a difference in performance. The parameter characterizing the computational power of a node is called node

Table 1. Identified performance and scalability parameters

Parameter	Description
Block frequency	Time between two succeeding blocks
Block size	Amount of transactions fitting in a block
Workload type	Smart contract
Node configuration	CPU, RAM, network speed
Network size	Number of nodes
Network structure	Structure of the blockchain network
Workload quantity	Amount of transactions to be processed
Amount of miners/sealers	Actively participating nodes
Blockchain client and API	E.g. Geth or Parity, web3.js or web3.py

configuration; it includes hardware configuration like the amount and type of CPU and RAM. This parameter was studied only to some extent in [18] and [2]. Again, only a few sources [5,24] discuss the effects of network size.

3.2 Evaluation Metrics

Nearly all studies (see Sect. 2) share the concept of throughput measured in 'transactions per second' (tps), but no other metrics beyond that. Table 2 defines the metrics for our analyses. Due to the large number of parameters in Sect. 3.1, we keep the amount of metrics low and focus on simple and high-level metrics. Hence, metrics regarding the hardware layer (like the utilization of CPU or memory) and the security of blockchains were excluded.

Table 2. Evaluation metrics

Metric	Description
Throughput	Number of successful transactions per second (tps)
Latency	Time difference in seconds between submission and completion of a transaction
Scalability	Changes of throughput and latency when altering a parameter (e.g. the network size or the hardware configuration of a node)

3.3 Experimental Setup

We employed the architecture in Fig. 1 to collect measurement data in an automated way. According to [18], the Parity client processes transactions significantly faster than the Geth client. Nevertheless, we decided to use the latter one as it is Ethereum's default client [8] and still is used more frequently than Parity [10]. For interaction, Geth offers an interactive JavaScript console, a JavaScript API for inclusion in applications and processes, and JSON-RPC[1] endpoints. The web3.js API [6] is the de facto standard for interacting with Ethereum clients. As a runtime environment for the application we selected Node.js since it is the most commonly used framework of all technologies included in the 2018 Stackoverflow developer survey [20].

As computing environment, we chose Amazon Web Services (AWS). Via the Elastic Compute Cloud (EC2) service one can rent computing power for different purposes on demand. Since the experiments required an Ethereum network with more than one node, we used AWS Cloudformation to start the nodes and install our mesaurement software in an automated manner. An additional

[1] JSON-RPC is a stateless, light-weight Remote Procedure Call (RPC) protocol that uses JavaScript Object Notation (JSON) as data format.

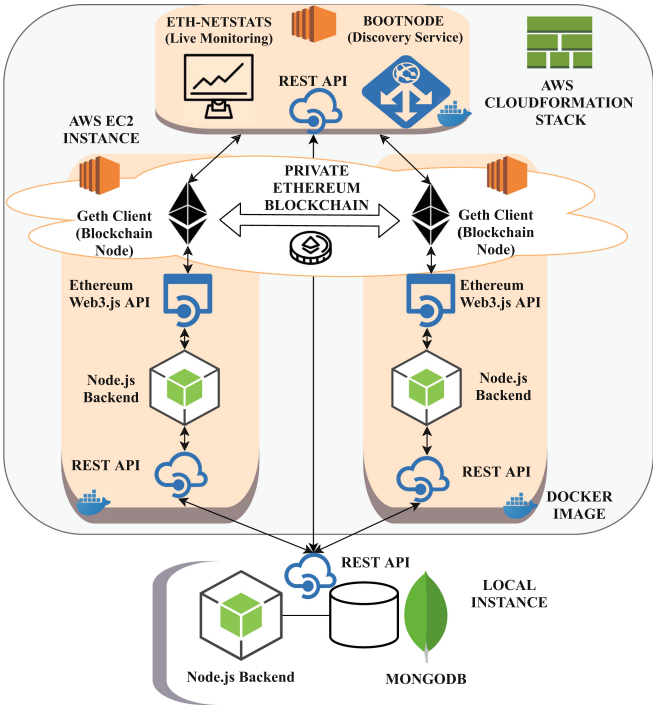

Fig. 1. Experimental setup

node ran the bootnode (a peer discovery service) and the monitoring software ETH-Netstats[2]. A local master node served as the command center for the experiments. The communication between the various nodes was done in REST-style (Representational State Transfer). For decoupling the major components, we used the Model-View-Controller design pattern. For storing the measurements for later analysis, we opted for MongoDB[3].

4 Data Analysis

In our experiments, we collected 4 000 data points ('measurements'), each corresponding to the average over several runs of the network with specific parameter settings. Each run comprised 1 000 transactions. In total, 4 million transactions were processed, which is roughly the volume of eight days on the Ethereum main chain. The computation time for all EC2 nodes and experiments was 380 h.

To measure the influence of node configuration and network size, we used the EC2 instances listed in Table 3. Unless stated otherwise, we varied only one variable at a time while the others were set to the default values in Table 4. The

[2] ETH-Netstats for the public Ethereum is available at https://ethstats.net/.
[3] MongoDB is a NoSQL database that uses JSON-like documents with schemata.

Table 3. EC2 instance types used for node configuration

Label	# CPUs	Speed	Memory	Network
c5.large	2	3 GHz	4 GB	10 Gbit
c5.xlarge	4	3 GHz	8 GB	10 Gbit
c5.2xlarge	8	3 GHz	16 GB	10 Gbit
c5.4xlarge	16	3 GHz	32 GB	10 Gbit
t2.xlarge	4	2.3 GHz	16 GB	Moderate

default values for difficulty and block period are those proposed by puppeth, the configuration utility of Geth.

Table 4. Default values for the variables included in the measurements

Difficulty	Block period	Gas limit	Workload	Instance type	# Nodes
524 288	15 s	4 700 000	Account	c5.xlarge	5

4.1 Block Frequency

Block frequency is inversely proportional to the time between blocks. In case of PoW, we varied the mining difficulty to obtain average block periods between 1 s and 2.5 s (0.25 to 4 times the default difficulty). For PoA, we used block periods of 2, 4, 8, 12, and 15 s. The experiments confirmed: throughput decreases and latency increases linearly with increasing block period.

4.2 Block Size

Ethereum uses the concept of gas to control the size of a block. Gas measures the resources (computation time and memory) consumed by a transaction. The total amount of gas in a block is capped by the block gas limit, which indirectly determines the number of transactions fitting into a block. In our experiments, the default value for the block gas limit results in blocks with 146 transactions of the default workload (account contract). We varied the number of transactions in a single block between 74 and 1 168 (gas limit factor 0.5 to 8).

As expected, we observe that as the block size increases, throughput increases while latency decreases in the same proportion, at least when the block period is large enough. For PoW and the default node configuration, the smallest considered block period approaches the time needed for creating, signing, and executing the transactions of a block. Here, throughput and latency will not improve when further increasing the block size.

4.3 Workload

For simulating different workloads, we used one of two smart contracts as the recipient of the transactions. The contracts differ in the state changes they perform: The first one ('account') changes the balance of two addresses, while the other one ('ballot') only accesses its own state. A transaction directed towards 'account' requires 32 k gas, while a call to 'ballot' needs only 27 k gas.

The observed effect of different workloads is depicted in Fig. 2. Surprisingly, there seems to be no difference between the two workloads for PoW, while there is a difference in the case of PoA. Welch's t-test with a significance level alpha of 5% shows that there is no significant difference for throughput and latency for PoW. For PoA however, the null hypothesis (no significant difference of the mean

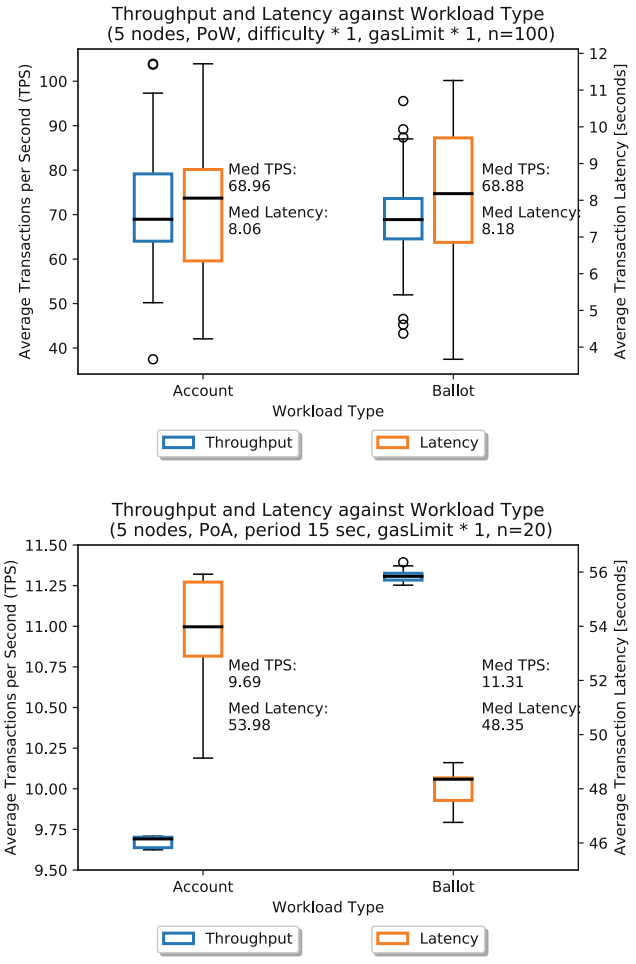

Fig. 2. Throughput and latency against workload type PoW (upper) and PoA (lower)

values of throughput and latency for the two workloads) can be strongly rejected, meaning that the difference is statistically significant. Throughput differs by 17%, which corresponds to the difference in transaction size (32 k vs. 27 k gas).

We argue that the different results obtained for PoW and PoA are due to the different block periods used. In case of PoW, the time needed for generating and processing the transactions is comparable to the block period. We noticed that the blocks were not always filled to their maximum. For PoA, on the contrary, the low block frequency allowed the nodes to generate and process the transactions within the block period. Hence, the effect of the workload on the performance was only observable when the nodes were not operating at their limits.

4.4 Node Configuration

To investigate the influence of computational power on performance, we specified four types of EC2 instances, each with a doubled amount of memory and cores (cf. Table 3). The parameters of the consensus algorithms were set to not limit performance: The gas limit was set so that the entire workload of 1 000 transactions could fit into a single block. Moreover, the parameters determining the block frequency were set to a maximum, i.e. the mining difficulty in case of PoW and the block period in case of PoA were set to their respective minimum.

Table 5. Scalability: increase of performance between successive node types

		large to xlarge	xlarge to 2xlarge	2xlarge to 4xlarge
Throughput	PoW	60.3%	48.9%	31.2%
	PoA	49.9%	35.6%	17.2%
Latency	PoW	37.8%	24.1%	18.9%
	PoA	29.0%	1.1%	22.1%

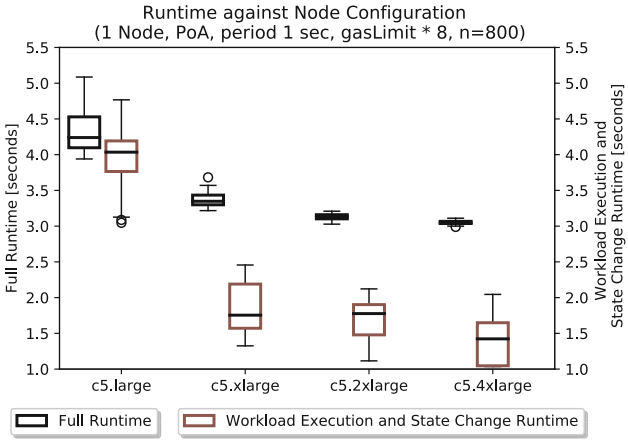

Fig. 3. Runtime analysis of different node configurations

As expected, the time required to complete the workload decreases as computing power increases. Table 5 shows the performance changes between successive node types. While the effect of doubling memory and CPUs is still noticeable, it becomes less apparent with more powerful node types. For PoA, this is illustrated in Fig. 3. The changes are more significant for PoW than for PoA.

Next to the reduced runtime for workload execution and state changes, transactions are also generated and signed faster when the computational power of a node is increased. By querying the number of transactions per block, we found that with less powerful node configurations, nodes are unable to pack the entire workload into a single block, even though the block size (the gas limit) is large enough. This is presumably due to a combination of slow workload generation, slow workload execution, and high block frequency. In such a case, at least one more complete block period is required for the rest of the workload, which aggravates the effect of slow generation and execution of the workload.

4.5 Network Size

Sometimes, increasing the number of machines is a reasonable approach to increase performance. However, factors such as network communication and consensus costs also play a crucial role in the context of blockchains. The time needed to propagate blocks to the majority of the nodes simply cannot be reduced by adding nodes to the network. In fact, communication and consensus efforts rather increase. On the other hand, information propagation is faster in a private setting than in the public Ethereum network due to the much smaller number of nodes in the network. For PoA, our measurements indicate that the performance does not change significantly with different network sizes. We assume that this is due to our experimental setup.

For PoW, however, the picture is different, and the effect of the network size also depends heavily on other parameters. The two major factors are block frequency and the computational strength of a node. Generally speaking, if the nodes are unable to propagate blocks and transactions within a short period of time, other parameters such as the number of nodes or a high block frequency cannot unfold their impact on performance. A high block frequency does not have the desired effect if the transactions and blocks cannot be propagated within the network during one block period. As the computational power of the t2.xlarge nodes could not compensate for network communication and consensus costs, measurements were performed with computationally stronger nodes (c5.4xlarge) to analyze these overheads more accurately.

In Table 6, one can observe that expected (calculated) and measured throughput drift apart with increasing network size. While there is a match of around 90–100% with smaller network sizes, larger networks only show a match of around 60–75%. Networks with a larger number of nodes do not fully use their available resources. This points to information propagation or computational power as a limiting factor. Broadly speaking, a negative correlation between the network size and the performance gain for an additional node could be identified.

Table 6. Calculated vs. Measured Throughput (PoW, $10\times difficulty$, c5.4xlarge)

Network size [nodes]	Block period [s]	Calculated runtime [s]	Throughput [tps]	Measured throughput [tps]
2	3.8	26.9	37.2	33.3 (89%)
4	2.3	15.8	63.5	63.2 (99%)
6	1.6	11.3	88.7	65.9 (74%)
8	1.3	9.2	108.8	75.6 (70%)
10	1.2	8.1	124.1	76.6 (62%)

5 Bottlenecks

The effect of a parameter may also depend on the settings of other parameters. When altering one parameter of the system, another one may become the bottleneck. In contrast to related work, we therefore argue that beyond discussing single parameters, information on the order of the bottlenecks is needed.

Bottleneck Structure. For private Ethereum networks using Geth, we derived the hierarchy of bottlenecks as depicted in Fig. 4. Parameters higher up in the hierarchy become a bottleneck as soon as the underlying ones no longer represent a bottleneck.

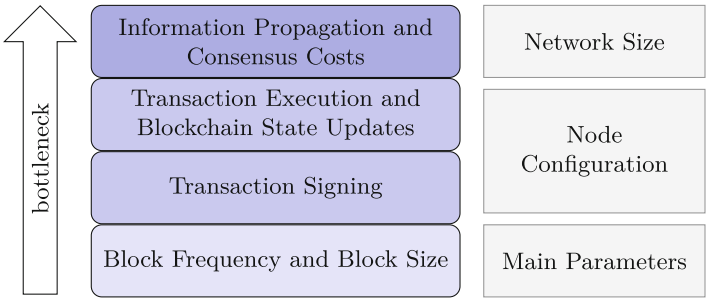

Fig. 4. Hierarchy of bottlenecks

Block Frequency and Block Size. The performance of a blockchain is, by definition, mainly a function of these two parameters, which according to our experiments are at the bottom of the hierarchy. With block frequencies at the limit of one block per second and large-sized blocks, processes such as transaction signing, transaction execution, changing the blockchain state, and propagating information become bottlenecks.

Node Configuration. The computational power of the nodes becomes the bottleneck as soon as the nodes in the network can no longer generate, sign, propagate and execute the transactions during the block period.

Network Size. Due to the overhead inflicted on the network when adding a node, the network size is at the top of the bottleneck hierarchy. When operating a network at its limits, parameters such as block frequency, block size, and node configuration restrict performance before performance is further reduced due to a larger network.

6 Discussion

The parameter settings of a private Ethereum network have a tremendous impact on performance. The values for throughput reported in literature range from a maximum of 284 tps in [5] to a minimum of 0.5 tps in [2]. Our experiments show that with a block period of 1 s, a block size large enough to fit 1 000 transactions into the block, an AWS EC2 instance of type c5.4xlarge, and a network of a single node, the throughput can be as high as 328 tps on average.

Block Frequency. When increasing the mining difficulty, the block frequency decreases, throughput decreases as well and latency rises. This is to be expected and consistent with more general findings already reported elsewhere. Our results confirm the results of [3], who show that an adjustment of the block interval may have a positive effect on performance. Furthermore, the results are in line with the move of various blockchain communities to increase the block frequency for better performance, as well as with Buterin who stated in a blog post [1] that the block propagation time in a network puts a constraint on the maximum block frequency.

Block Size. An appropriate choice of block frequency and block size is crucial. As to be expected with PoA, we observed a performance boost proportional to the increase in block size. Surprisingly, similar experiments with PoW show a saturation point. We understand this behaviour as a consequence of the work load being generated too slowly and the transactions being processed not fast enough. The most important finding is that an increment in block size substantially affects performance only if the block period is larger than the time needed for creating, signing, propagating, and executing the transactions as well as reaching consensus.

Unlike [5], we argue that an increase in block size has the potential to boost the performance of a private Ethereum network if the above-mentioned prerequisites are fulfilled. Our results agree with [22] who advocate the optimization of block size as a strategy for improving the scalability of blockchains to a certain extent. However, we were unable to confirm their observation that as the block size increases, the performance first increases, but decreases when it reaches a certain level. It may be that the block size was not chosen large enough in our

experiments to trigger this effect. Finally, we want to emphasize that similar to block frequency, the block size may change over time if the value of targetGasLimit in the Geth client and the value of gasLimit in the genesis block do not match. It is not clear whether other authors have taken this issue into account.

Workload. The experiments regarding workload show differences between PoW and PoA that we attribute to the different block periods used (1 s for PoW vs. 15 s for PoA). Our observations are in line with previous findings [5,24] that report a dependency of performance on the type of smart contract used. In our experiments a throughput difference of around 17% was measured.

Node Configuration. When operating a private Ethereum blockchain at its limits by choosing a high block frequency and large block size, node configuration becomes more important. In a network with five nodes, throughput almost tripled and latency halved when increasing the computational power of the nodes by a factor of four. In our opinion, the marked difference between the our node configurations can be attributed to the fact that stronger machines feature faster transaction generation and signing, better network communication and consensus handling, quicker execution of the transactions, and faster state changes.

An analysis of the time needed solely for executing the transactions and changing a node's blockchain state confirmed previous work [18] that reported a decrease by 25% for processing transactions when increasing the memory of a node from 4 GB to 24 GB. Indeed, when changing the node configuration from c5.4xlarge (8 GB RAM) to c5.4xlarge (32 GB RAM), we observed an improvement by roughly 26%. Another finding is that computationally weak nodes may have troubles propagating transactions to the network.

Network Size. With the standard PoA settings, no significant performance difference could be observed when changing the network size. This is due to our experimental setup where all nodes seal blocks in a pre-assigned and static time interval. On the other hand, when adding nodes in the PoW setting, our results show that the performance rises at first but starts to decrease once a peak has been reached. Although it may seem puzzling that more power can actually reduce the performance of a network, these results are in line with previous work [5]. We found that additional network communication and consensus costs are the main limiting factor for the scalability of the network. Moreover, the effect of adding more nodes to a network also depends on block frequency, node configuration, and the current size of the network. Nodes may get out of sync and uncle blocks may be created if the time needed for propagating information in the network is larger than the block period.

Bottlenecks. Our results confirm the findings in [24]. We also observed that information propagation and consensus costs are the main factors limiting the scalability of private Ethereum networks. However, the bottleneck may actually shift from one parameter to another. The combined effect of block frequency and block size as well as of the node configuration may limit the scalability before information propagation and consensus costs become relevant.

7 Conclusions

We investigated the effects of various parameters on the performance and scalability of private Ethereum blockchains. To this end, we conducted 4 000 measurements and visualized the impact of parameter changes in several charts and tables. More details can be found in [19].

Summarizing, we conclude that the effects of different parameters are intertwined such that the optimal setting of one parameter often depends on the setting of the others as well. Our results indicate that scaling is only possible to a limited extent due to the current design of Ethereum. As a specific contribution, we identified a hierarchy of bottlenecks.

Limitations. Our experimental setup contains several sources for potential errors. First, the process of measuring itself might have influenced the system under observation. We ran a Node.js instance on each node that shared the resources with the Ethereum client. Likewise, the additional services needed for peer discovery (Bootnode) and live monitoring (ETH-Netstats) might have influenced the system. It is also unclear if and to which extent the chosen virtualization on the level of the operating system (Docker) produces results that differ from nodes running on physically separate machines. Finally, the way we generated and distributed the transactions may have affected the results.

It is unclear to which extent our observations can be generalized. Our work was exploratory and descriptive by nature. Inferential approaches may be needed to confirm, explain, and extend the findings. For some results, it may turn out that the experiments have been carried out on too small a scale. Although the volume of generated transactions equals eight days of transactions on the Ethereum main chain, the test data may not fully reflect the variability of real data.

Future Work. This study provides insights into the effects of different parameters on performance and scalability in private Ethereum networks. Still, further research is needed. Security considerations were beyond the scope of the present paper. Hence, future work should address for instance trade-offs between security and performance.

Our experimental setting can readily be used to collect data on further combinations of parameters. In particular, it may be worthwhile adding multivariate to our bivariate analyses to gain a better understanding of the interaction of parameters.

The current setup is tailored to the Geth client and the web3.js API. These technologies, the blockchain client and the used API, represent further parameters to be analyzed. It may well be that a client like Parity exhibits different performance characteristics.

Finally, a layer could be introduced that facilitates the addition of new workload types to the framework.

References

1. Buterin, V.: Toward a 12-second Block Time. https://blog.ethereum.org/2014/07/11/toward-a-12-second-block-time/. Accessed 19 Nov 2018
2. Chen, S., Zhang, J., Shi, R., Yan, J., Ke, Q.: A comparative testing on performance of blockchain and relational database: foundation for applying smart technology into current business systems. In: Streitz, N., Konomi, S. (eds.) DAPI 2018. LNCS, vol. 10921, pp. 21–34. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-91125-0_2
3. Croman, K., et al.: On scaling decentralized blockchains. In: Clark, J., Meiklejohn, S., Ryan, P.Y.A., Wallach, D., Brenner, M., Rohloff, K. (eds.) FC 2016. LNCS, vol. 9604, pp. 106–125. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53357-4_8
4. Dinh, T.T.A., Liu, R., Zhang, M., Chen, G., Ooi, B.C., Wang, J.: Untangling blockchain: a data processing view of blockchain systems (2017). http://arxiv.org/pdf/1708.05665.pdf
5. Dinh, T.T.A., Wang, J., Chen, G., Liu, R., Ooi, B.C., Tan, K.L.: BLOCKBENCH: a framework for analyzing private blockchains. In: International Conference on Management of Data, SIGMOD 2017, pp. 1085–1100. ACM (2017)
6. Ethereum: web3.js. https://web3js.readthedocs.io/. Accessed 21 Dec 2018
7. Ethereum 2.0. https://github.com/ethereum/eth2.0-specs. Accessed 20 Feb 2019
8. Ethereum Repository. https://github.com/ethereum. Accessed 19 Nov 2018
9. Ethereum Sharding. https://github.com/ethereum/wiki/wiki/Sharding-FAQs. Accessed 20 Feb 2019
10. Ethernodes.org. https://www.ethernodes.org/network/1. Accessed 19 Nov 2018
11. Eyal, I., Gencer, A.E., Sirer, E.G., Renesse, R.V.: Bitcoin-NG: a scalable blockchain protocol. In: 13th Conference on Networked Systems Design and Implementation, NSDI 2016, pp. 45–59. USENIX Association (2016)
12. Gervais, A., Karame, G.O., Wüst, K., Glykantzis, V., Ritzdorf, H., Capkun, S.: On the security and performance of proof of work blockchains. In: SIGSAC Conference on Computer and Communications Security, CCS 2016, pp. 3–16. ACM (2016)
13. Hyperledger. https://www.hyperledger.org. Accessed 14 Oct 2018
14. Kan, J., Chen, S., Huang, X.: Improve blockchain performance using graph data structure and parallel mining (2018). http://arxiv.org/abs/1808.10810
15. Li, W., Sforzin, A., Fedorov, S., Karame, G.O.: Towards scalable and private industrial blockchains. In: Workshop on Blockchain, Cryptocurrencies and Contracts, BCC 2017, pp. 9–14. ACM (2017)
16. Nasir, Q., Qasse, I.A., Abu Talib, M., Nassif, A.B.: Performance analysis of hyperledger fabric platforms. Secur. Commun. Netw. **2018**, 1–14 (2018)
17. Pongnumkul, S., Siripanpornchana, C., Thajchayapong, S.: Performance analysis of private blockchain platforms in varying workloads. In: 26th International Conference on Computer Communication and Networks, ICCCN 2017 (2017)
18. Rouhani, S., Deters, R.: Performance analysis of ethereum transactions in private blockchain. In: 8th International Conference on Software Engineering and Service Science, ICSESS 2017, pp. 70–74. IEEE (2017)
19. Schäffer, M.: Performance and scalability of smart contracts in private ethereum blockchains. Master's thesis, TU Wien (2019)
20. Stackoverflow: Developer Survey Results (2018). https://insights.stackoverflow.com/survey/2018#technology. Accessed 19 Nov 2018

21. Thakkar, P., Nathan, S., Viswanathan, B.: Performance benchmarking and optimizing hyperledger fabric blockchain platform. In: 26th International Symposium on Modeling. Analysis, and Simulation of Computer and Telecommunication Systems, MASCOTS, pp. 264–276. IEEE (2018)
22. Xin, W., Zhang, T., Hu, C., Tang, C., Liu, C., Chen, Z.: On scaling and accelerating decentralized private blockchains. In: 2017 IEEE 3rd International Conference on Big Data Security on Cloud (bigdatasecurity), IEEE International Conference on High Performance and Smart Computing (HPSC), and IEEE International Conference on Intelligent Data and Security (IDS), pp. 267–271 (2017)
23. Zhang, H., Jin, C., Cui, H.: A method to predict the performance and storage of executing contract for ethereum consortium-blockchain. In: Chen, S., Wang, H., Zhang, L.-J. (eds.) ICBC 2018. LNCS, vol. 10974, pp. 63–74. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-94478-4_5
24. Zheng, P., Zheng, Z., Luo, X., Chen, X., Liu, X.: A detailed and real-time performance monitoring framework for blockchain systems. In: 40th International Conference on Software Engineering: Software Engineering in Practice, ICSE-SEIP 2018, pp. 134–143. ACM (2018)