Freie Universität Berlin

Benchmarking a Blockchain-based Certification Storage System

Clemens Wickboldt

School of Business & Economics
Discussion Paper

Information Systems

2019/5

Benchmarking a Blockchain-based Certification Storage System

Clemens Wickboldt

March 22, 2019

Technical Report*

A comprehensive empirical study is performed to measure the performance of a Blockchain-based Certification Storage System in Hyperledger Fabric. This work is based on a proof of concept in the aviation industry and follows a Technical Risk & Efficacy evaluation strategy to determine the utility derived from the use of the artefact. Relevant tuning parameters for performance and scalability as well as bottlenecks are identified. The impact of configuration parameters such as blocksize, transaction arrival rate and number of concurrent users on the systems performance is investigated. Observations show that demands at throughput above system limits lead to transaction failures. Contributed are a repeatable process to performance sensitivity analysis and recommendations for configuring a Blockchain-based Certification Storage System for stable but high performance. The results can be used as a basis for optimizing the performance of similar systems.

Keywords Blockchain · Hyperledger Fabric · Benchmarking · Maintenance Events

*Current version published at https://refubium.fu-berlin.de/handle/fub188/24484?locale-attribute=en%7D

1 Introduction and Related Work

Software Performance Engineering or Benchmarking includes efforts to describe performance changes in the system. These activities are divided into two approaches, an early predictive model (C. U. Smith, 2002) and a late measurement approach (Woodside et al., 2007). Barber (2004) argues that predictive model based approaches depend on a significant amount of emprical data which is rarely available when designing a new and innovative system. That is the reason why measurement approaches are popular. Examples for the measurement approach are Arlitt et al. (2001) for a large web-based shopping system or Avritzer et al. (2002) for a large industrial system.

In the aviation industry, safety-related spare parts require sophisticated and complete documentation of workshop events. Due to trust issues and no central institution which could handle digital workshop event certficates, this process hasn't been digitized yet. A blockchain-based concept to overcome these issues has been proposed in Wickboldt and Kliewer (2018). This concept has been implemented and presented as an IT artefact called Blockchain-based Certification Storage System (BCSS) (Wickboldt and Kliewer, 2019). Workshops with domain experts show that the solution needs to be capable of handling an average of $\sim$0.16 transactions per second (tps) which equals around five million transactions a year.

A measurement approach is performed to ensure that the BCSS meets the requirements of the underlying business process. March and G. F. Smith (1995) state that the evaluation of a designed artefact requires measurement. Rudimentary benchmarks have been performed as part of the proof of concept in Wickboldt and Kliewer (2019) which showed that at least 7.99 tps are reachable. To ensure scalability, this report contributes comprehensive benchmarking results of the BCSS. An analytical evaluation in the sense of Hevner et al. (2004) is performed by examining the performance characteristics. This work extends the evaluation of Wickboldt and Kliewer (2018) and Wickboldt and Kliewer (2019) on the path of the Technical Risk & Efficacy evaluation strategy (Venable et al., 2016). The step of performance benchmarking in an artificial environment is taken to prepare the transfer to a naturalistic environment. As the proof of concept in Wickboldt and Kliewer (2019) delivered first insights into performance, summative evaluations in an artificial environment are performed to gain deep understanding about blockchain configuration parameters and their impact on performance.

In order to arrive at these contributions the remainder of this report is organized as follows. Section 2 gives an overview about the benchmarking framework and experimental setup as well as configuration parameters of the benchmark. Benchmarking results are presented and discussed in section 3. A conclusion of this report and an outlook to future research is given in section 4.

2 Performance Benchmarking with Hyperledger Caliper

This contribution is orientated at the work of Thakkar et al. (2018) who provide insights about performance measurement in Hyperledger Fabric Blockchain systems using Hy-

perledger Caliper[1] which is a benchmark tool for Hyperledger Fabric. Reports produced by Caliper include:

R1 Throughput measured in transactions per second (tps),

R2 failed transactions due to timeouts,

R3 transaction latency,

R4 resource utilization.

Four methods for performing transactions to write and read workshop event certificates are implemented in BCSS. These methods receive parameters about the status of the spare part. *writeAsset* receives information via the frontend like serial number, Cycles Since New, Hours Since New, Cycle Limit, Hour Limit, Part Number, Part Description, Part Owner and Part Status. *writeCert* receives a serial number of the certificate, certificate type, certificate owner and details corresponding to the certificate. *queryAsset* receives the serial number. Using this, a JSON string with the information of the part is returned. *queryCert* gets the serial number of a part and returns the history of all certificates for that part.

For the purpose of this benchmark, a part filled with random data is generated (*writeAsset*). Next, a certificate for a workshop event for this part is generated (*writeCert*). The third and fourth operation are reading the asset's information (*queryAsset*) and reading the certificate's information (*queryCert*). Figure 1 visualizes a sequence of a Caliper benchmark.

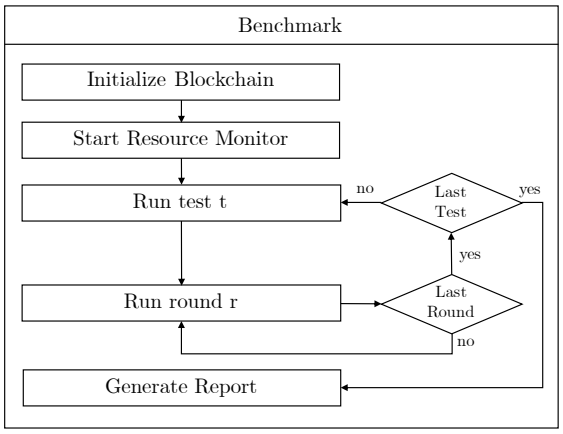

Figure 1: Sequence Diagram for Caliper Benchmark based on Caliper (2019)

The blockchain is initialized, the resource monitore ist started. The configuration file[2] is consulted for the first test t. A test varies by block size and number of users whereas a round consists of multiple benchmarks for different transaction arrival rates (TAR). All rounds r are performed. This is done for all tests. After completion of the last test, a html benchmark report is generated.

[1] `https://www.hyperledger.org/projects/caliper`

[2] Sourcecode 1 in the appendix shows an example configuration file

2.1 Configuration Parameters and Experimental Setup

As this is a benchmark of a proof of concept in an artificial environment, a measurement approach is done on a development system. Tested are four operations on the BCSS. First, an asset is written onto the blockchain. Second, a certificate is issued on that asset. Next, the asset is queried. The fourth operation is querying the written certificate. The system comprises the following components: Intel(R) Xeon(R) Gold 6140 CPU @ 2.30GHz, 8 GB DDR4 RAM (ECC), 320 GB Serial Attached SCSI running Ubuntu 18.04 LTS. Hyperledger Fabric is running in version 1.4. In a preliminary analysis, the following has been observed (see figure 2):

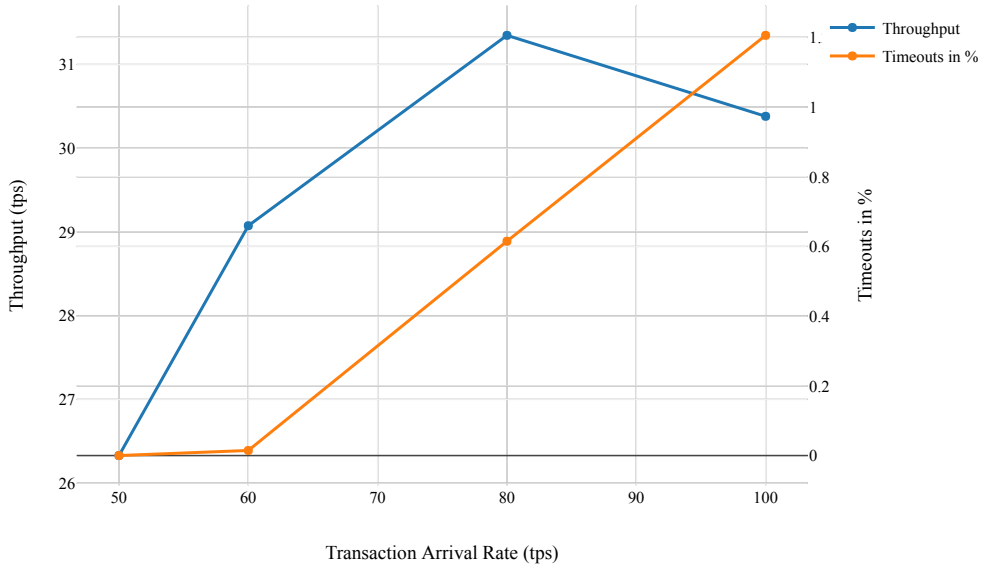

Figure 2: Transaction Arrival Rate on Throughput and Timeouts

Average throughput doesn't necessarily increase with a higher TAR measured in tps. Transactions are increasingly terminated by timeout larger arrival rates. Only at an arrival rate of more than 50 tps, parameters like block size and number of users play a role to prevent timeouts. Throughput declines with arrival rates greater than 80 tps. The following parameters are therefore set for a sensitivity analysis: The length of the queue for benchmarked transactions is set to 1000 to ensure the benchmark runs some time to produce reliable and stable results. Benchmark runs are run with TAR of 50, 60, 80 and 100 tps.

A benchmark is conducted for block sizes of $2^n, n \in \{0, 1, 2, \ \dots \ , 10\}$ over a number of users $2^n, n \in \{0, 1, 2, \ \dots \ , 6\}$. A network with two organisations with each two peers

Algorithm 1 Start Benchmark

```
1: for All Block Sizes do
2:     for All User Sizes do
3:         procedure Set Blocksize(blockSize, FabricConfigurationFile)
4:             Open FabricConfigurationFile
5:             MaxMessageCount ← blockSize
6:             Close FabricConfigurationFile
7:             Rewrite Genesis Block(FabricConfigurationFile)
8:         end procedure
9:         procedure Set Number of Clients(numberClients, CaliperConfigurationFile)
10:            Open caliper-config.json
11:            number ← numberClients
12:            Close caliper-config.json
13:        end procedure
14:        procedure Shutdown Blockchain
15:            Shut down Docker Containers
16:            Remove Local State
17:            Remove Chaincode
18:            Sleep 60 Seconds
19:        end procedure
20:        procedure Startup Blockchain
21:            Startup Docker Containers
22:            Create Channel
23:            Join Peers to Channel
24:            Install Chaincode
25:            Instantiate Chaincode
26:        end procedure
27:        Start Benchmark(CaliperConfigurationFile)
28:    end for
29: end for
```

is used. The benchmark runs are started via algorithm 1. For every permutation the block size is set in the Hyperledger Fabric blockchain configuration. Timeouts are set to standard values of Hyperledger Fabric, for validating a transaction it is set to 60 seconds, for writing a block it is set to 10 seconds.

The genesis block is written respecting the block size of the Fabric configuration file. The number of clients is written into the Caliper benchmark configuration. After setting all parameters, eventually running blockchain services are shutdown. A sleep after executing the command ensures that all docker containers are indeed stopped. The blockchain is started, buidling onto the new genesis block. The channel for communication is created. All 4 peers join the channel. The chaincode for writing and reading certificates is installed and instantiated. Finally, the benchmarking is started with the current Caliper configuration.

This algorithm runs until every permutation of block size and current users has been benchmarked. All reports R1 to R4 are then retrieved by a Python Jupyter Notebook[3] for analysis. The results of the benchmark are presented in the following section.

[3] `https://jupyter.org`

3 Benchmarking Results

As Hyperledger Caliper delivers data about throughput (R1), failure rate (R2) latency (R3) and resource utilization (R4), this section presents and discusses benchmarking results for these four performance indicators. Throughput and latency results are presented along the parameters block size and number of concurrent clients. Performance measures are presented regarding memory and utilized disk space along tested block sizes.

Figure 3 shows Pearson correlation coefficients between the benchmarking results. A correlation coefficient of 1 suggests that the respective results correlate absolutely with eachother. A correlation coefficient of -1 means that they correlate absolutely against eachother. Block size and throughput are strongly positively (0.53) correlated to eachother, suggesting a larger block size improves the throughput. Average latency and throughput have a very strong negative correlation (-0.84), suggesting that higher throughput decreases latency. Higher user count leads to a higher propability for timeouts (correlation of 0.31). On the one hand, a larger arrival rate is slightly positive correlated with higher throughput (0.13). On the other hand, it is also positively correlated (0.14) with timeouts, suggesting that a load above the performance limits of the system lead to timeouts.

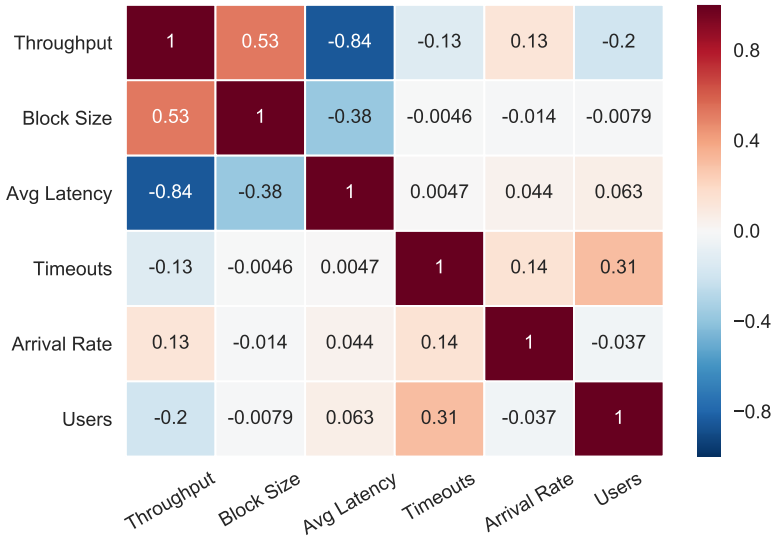

Figure 3: Correlation Heatmap for Benchmarking Results

Next, numerical results of the benchmarks are presented and discussed in section 2 using line plots. The data is grouped by the desired transaction arrival rate. The raw numbers are shown in a table next to the figure.

3.1 Transaction Throughput and Failed Transactions

Transaction output is measured in transactions per second. Measured is the mean throughput along all blocksizes and calculated for any tested user count. Timeouts are calculated in % of all transactions. These two measured variables are considered together, since for a BCSS in production a high throughput only makes sense without transaction failures.

Concurrent Users

As shown in figure 4, the systems performance is decreasing with an increasing number of clients which are active at the same time. Read operations are generally quicker (⌀ 32.17tps) than write operations (⌀ 26.34 tps). Maximum throughput of 37.57 tps for reading at a TAR of 100 and 31.22 tps for writing also at a TAR of 100 is reached with only one concurrently active client. If 32 users are active the minimum throughput is reached for reading at 25.58 tps (TAR 50) and 19.79 for writing (TAR 100).

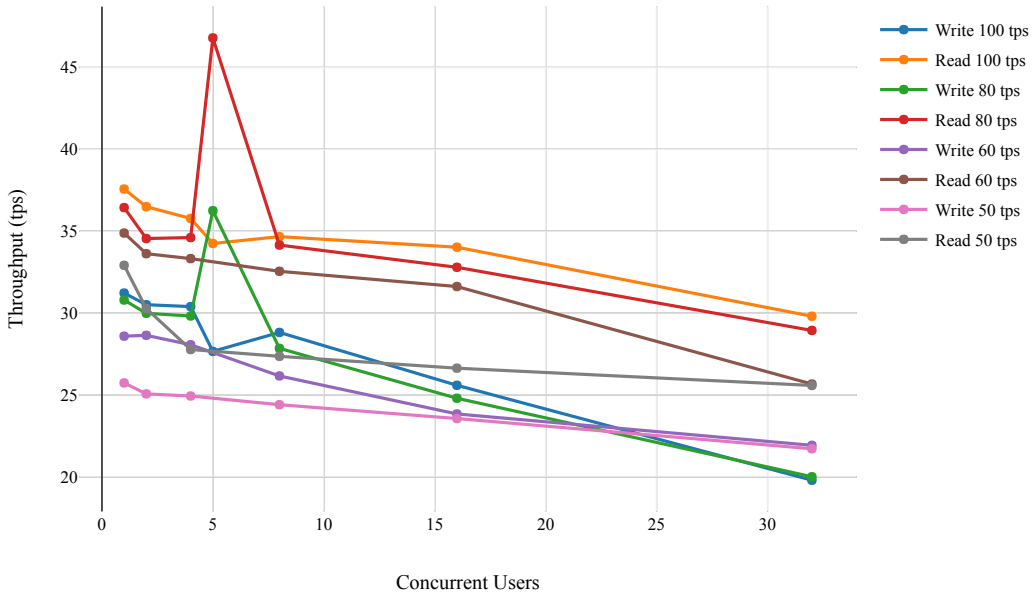

Figure 4: Mean Throughput per Number of Users

Figure 5 shows the number of transactions which are not handled because of a timeout. Until 4 concurrent users, there are no timeouts at all for reading and writing operations.

Concurrent Users	Throughput 100 tps write	Throughput 100 tps read	Throughput 80 tps write	Throughput 80 tps read	Throughput 60 tps write	Throughput 60 tps read	Throughput 50 tps write	Throughput 50 tps read
1	31.22	37.57	30.79	36.44	28.59	34.88	25.74	32.92
2	30.51	36.49	29.99	34.55	28.64	33.62	25.06	30.28
4	30.39	35.78	29.82	34.61	28.06	33.32	24.94	27.78
8	28.82	34.66	27.84	34.15	26.16	32.55	24.4	27.36
16	25.59	34.02	24.8	32.8	23.84	31.62	23.56	26.64
32	19.79	29.81	20.01	28.94	21.92	25.66	21.72	25.58

Table 1: Mean Throughput per Number of Users

Writing operations tend to timeout more significantly, especially at a higher transaction arrival rate. The highest timeout rate for reading is at 100 tps desired throughput and 32 users (0.43%). The highest timeout rate for writing is at also 100 tps desired throughput and also 32 users (15.21%).

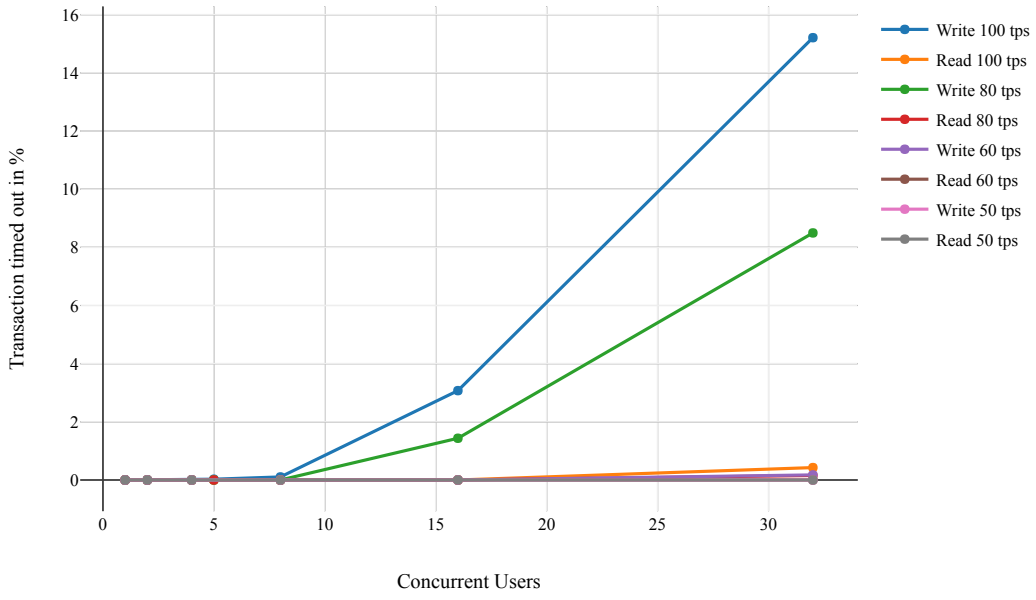

Figure 5: Mean Timeouts per Number of Users

The presented performance does not necessarily depend on the number of users but on the number of transactions in the transaction queue. This relationship between performance and fulfillment rate is related to the overall performance of the host system. If too many transactions are sent to the orderer with a high TAR or high number of users, transactions which are not handled after 60 seconds get discarded to ensure a

steady performance overall.

Concurrent Users	Timeout % 100 tps write	Timeout % 100 tps read	Timeout % 80 tps write	Timeout % 80 tps read	Timeout % 60 tps write	Timeout % 60 tps read	Timeout % 50 tps write	Timeout % 50 tps read
1	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0
8	0.11	0	0	0	0	0	0	0
16	3.07	0	1.44	0	0	0	0	0
32	15.21	0.43	8.49	0.15	0.18	0	0	0

Table 2: Mean Timeouts per Number of Users

To summarize, high TAR and low number of clients lead to a high throughput. Timeouts can be prevented until 16 users and a TAR of 60 tps. This leads to a performance of 23.84 tps for writing and 31.62 tps for reading.

Block Size

Figure 6 shows the relation between transaction throughput and block size. Up to a block size of 128 throughout increases.

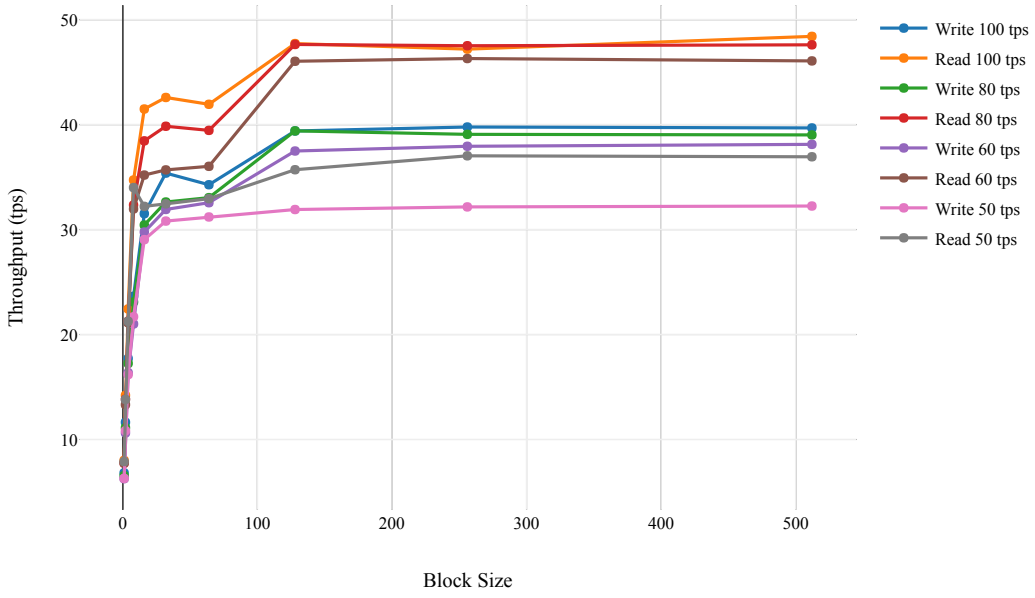

Figure 6: Mean Throughput and Block Size

Maximum reading throughput of 48.4 tps is reached at TAR of 100 tps and a block size of 512. Maximum writing throughput of 39.77 tps is reached at TAR of 100 tps and a block size of 256. Minimum reading throughput of 7.75 tps is reached at TAR of 80 tps and a block size of 1. Minimum writing throughput of 6.27 tps is reached at TAR of 60 tps and a block size of 1. Also here, a higher transaction arrival rate results in higher throughput. Reading is faster than writing.

Block Size	Throughput 100 tps write	Throughput 100 tps read	Throughput 80 tps write	Throughput 80 tps read	Throughput 60 tps write	Throughput 60 tps read	Throughput 50 tps write	Throughput 50 tps read
1	6.82	8.01	6.56	7.75	6.27	7.76	6.29	7.9
2	11.64	14.21	11.12	13.84	10.62	13.32	10.83	13.82
4	17.72	22.44	17.25	21.28	16.36	21.12	16.18	21.26
8	23.65	34.72	23.1	32.33	21.01	31.97	21.69	34
16	31.52	41.48	30.43	38.45	29.77	35.19	29.04	32.21
32	35.38	42.58	32.62	39.84	31.92	35.68	30.81	32.45
64	34.27	41.93	33.04	39.45	32.56	36.02	31.2	32.94
128	39.39	47.72	39.4	47.63	37.48	46.03	31.91	35.7
256	39.77	47.19	39.07	47.51	37.93	46.29	32.16	37.03
512	39.68	48.4	39.02	47.6	38.12	46.07	32.24	36.93

Table 3: Mean Throughput and Block Size

Figure 7 shows the relation between block size and failed transactions.

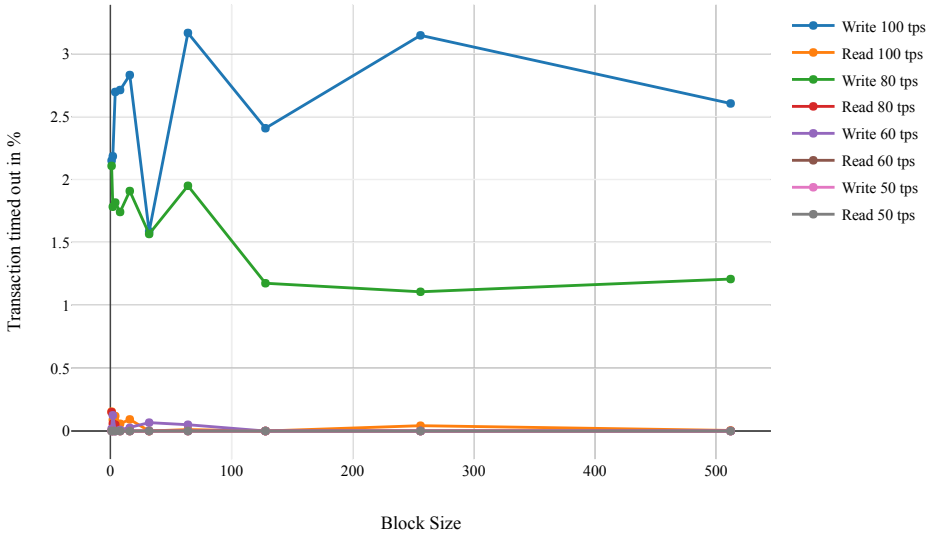

Figure 7: Mean Timeouts and Block Size

Maximum timeouts for reading of 0.15% are reached at a TAR of 100 and block size of 1. The most timeouts for writing (3.17%) are experienced with a TAR of 100 and a block size of 64. Block sizes equal to or greater than 128 deliver stable performance at a TAR of 60 for reading and writing. Below that block sizes, timeouts occur. It can also be seen that runs with a transaction arrival rate greater 80 have significant timeouts, independent from block size.

Block Size	Timeout % 100 tps write	Timeout % 100 tps read	Timeout % 80 tps write	Timeout % 80 tps read	Timeout % 60 tps write	Timeout % 60 tps read	Timeout % 50 tps write	Timeout % 50 tps read
1	2.15	0.15	2.11	0.15	0.02	0	0	0
2	2.18	0.09	1.78	0.06	0.12	0	0	0
4	2.7	0.12	1.82	0.05	0	0	0	0
8	2.71	0.06	1.74	0	0.01	0	0	0
16	2.83	0.09	1.91	0	0.02	0	0	0
32	1.59	0	1.57	0	0.07	0	0	0
64	3.17	0.01	1.95	0	0.05	0	0	0
128	2.41	0	1.18	0	0	0	0	0
256	3.15	0.04	1.11	0	0	0	0	0
512	2.6	0	1.21	0	0	0	0	0

Table 4: Mean Timeouts and Block Size

For a high throughput a high TAR of 100 and large blocks of at least 256 transactions are recommended. Additionally, to prevent timeouts, block sizes smaller than 128 and TAR larger than 60 should be avoided. A block size of 128 and an expected arrival rate of transactions of 60 still leads to 37.48 tps in writing and 46.03 of reading which is close to the maximum.

3.2 Transaction Latency

Transaction latency is an indicator for the time an issued transaction is completed and a response is available to the application that issued the transaction. It is measured in miliseconds (ms). Latency depends on the amount of transactions in a block but also on concurrent users.

Block Size

As figure 8 shows, average transaction latency decreases rapidly until 32 transactions per block. This is true for all transaction arrival rates. Maximum average latency of 64.15 ms is reached at a TAR of 80 for reading operations with one transaction per block. Writing with a TAR of 60 and a block size of 1 leads to the maximum average latency of 81.46 ms. Minimum average latency of 2.9 ms is reached with a TAR of 50 and block size of 32 for reading and with a TAR of 50 and block size of 512 for writing (11.41 ms).

If the recommended parameters from the throughput and failure tests of a TAR of 60 and a block size of 128 are considered, an average latency of 13.45 ms for writing and 5.11 ms for reading is observerd which is close to the minimum observations.

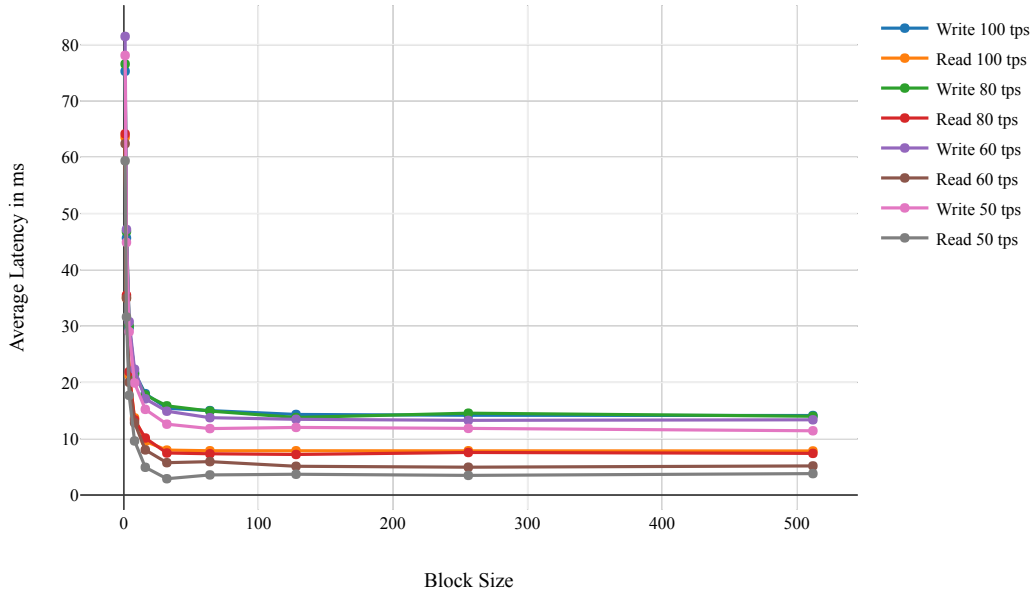

Figure 8: Average Latency and Blocksize

Block Size	Avg Latency 100 tps write	Avg Latency 100 tps read	Avg Latency 80 tps write	Avg Latency 80 tps read	Avg Latency 60 tps write	Avg Latency 60 tps read	Avg Latency 50 tps write	Avg Latency 50 tps read
1	75.27	63.55	76.55	64.15	81.46	62.4	78.09	59.36
2	45.64	35.01	46.8	35.48	47.17	34.99	44.87	31.64
4	29.74	21.25	30.08	21.9	30.78	20.02	29.02	17.7
8	21.77	13.7	21.47	13.36	22.35	12.84	19.91	9.61
16	18.01	9.53	17.78	10.13	17.09	8.02	15.21	4.93
32	15.41	7.98	15.84	7.47	14.89	5.75	12.59	2.9
64	15	7.87	14.91	7.33	13.74	5.93	11.8	3.57
128	14.31	7.86	13.78	7.2	13.45	5.11	12.01	3.69
256	14.17	7.87	14.54	7.56	13.26	4.94	11.84	3.49
512	14.12	7.83	14	7.4	13.36	5.17	11.41	3.81

Table 5: Average Latency and Block Size

Concurrent Users

Figure 9 shows the relation between the number of concurrent users and the application's latency. Transaction latency increases with an increase of concurrent users in the network. This is especially true for writing operations across all transaction arrival rates.

Maximum average latency of 20.21 ms is reached at a TAR of 80 and 32 concurrent clients for reading operations. Maximum average latency for writing is reached at a TAR of 60 and 32 clients. Minimum average latency for reading of 11.6 ms is reached at 50

tps and one user. With the same parameters the minimum average latency for writing of 21.72 ms is reached.

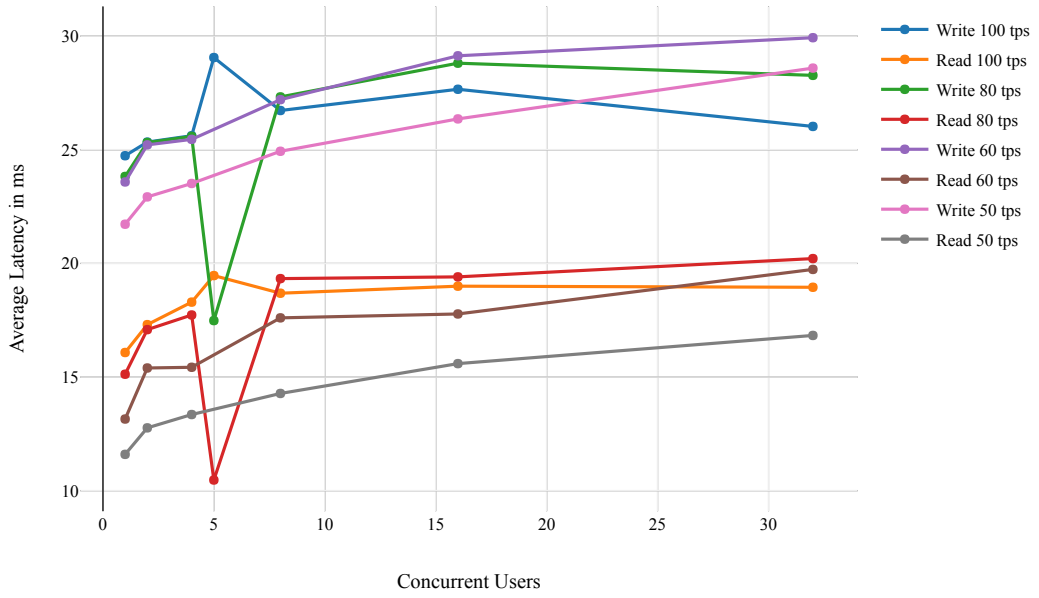

Figure 9: Average Latency and Number of Users

Concurrent Users	Avg Latency 100 tps write	Avg Latency 100 tps read	Avg Latency 80 tps write	Avg Latency 80 tps read	Avg Latency 60 tps write	Avg Latency 60 tps read	Avg Latency 50 tps write	Avg Latency 50 tps read
1	24.74	16.08	23.83	15.12	23.58	13.15	21.72	11.6
2	25.34	17.31	25.32	17.09	25.22	15.4	22.93	12.76
4	25.62	18.3	25.59	17.73	25.46	15.43	23.52	13.35
8	26.73	18.69	27.33	19.33	27.21	17.6	24.93	14.28
16	27.66	19	28.81	19.41	29.14	17.78	26.36	15.59
32	26.03	18.95	28.27	20.21	29.93	19.74	28.59	16.83

Table 6: Average Latency and Number of Users

The recommended user limit for preventing timeouts is 16 at 60 tps. If these parameters are taken into account a latency of 29.14 ms for writing and 17.78 ms for reading can be achieved.

3.3 Resource Utilization

Figure 10 shows the relation between block size and the highest number of RAM used within a benchmark run. It is observerd that a higher block size leads to a larger allocation of RAM. But this is only true until a block size of 128. Larger block sizes don't necessarily require more RAM.

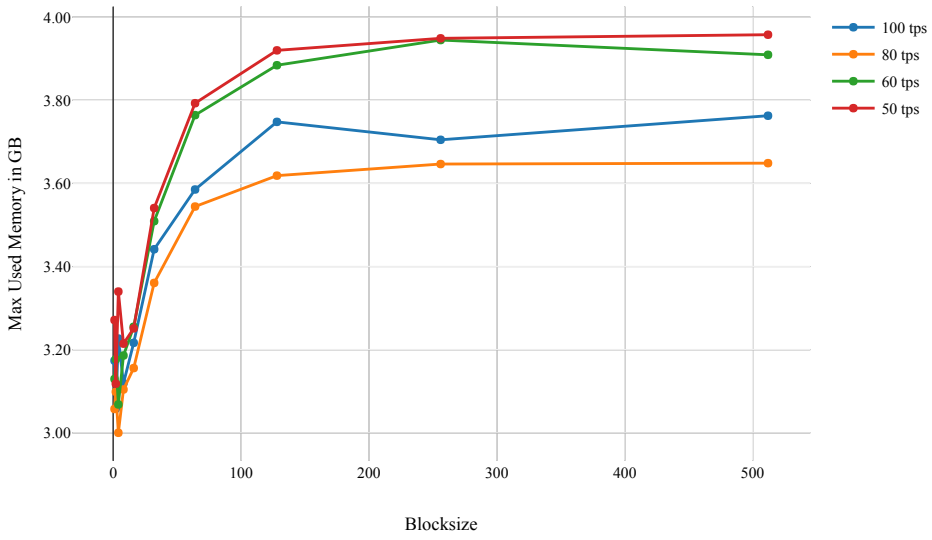

Figure 10: Maximum Used Memory per Block Size

Block Size	Memory in GB 100 tps	Memory in GB 80 tps	Memory in GB 60 tps	Memory in GB 50 tps
1	3.17	3.06	3.13	3.27
2	3.06	3.1	3.18	3.12
4	3.23	3	3.07	3.34
8	3.12	3.1	3.19	3.21
16	3.22	3.16	3.26	3.25
32	3.44	3.36	3.51	3.54
64	3.59	3.54	3.76	3.79
128	3.75	3.62	3.88	3.92
256	3.7	3.65	3.94	3.95
512	3.76	3.65	3.91	3.96

Table 7: Maximum Used Memory per Block Size

Used memory is at maximum of 3.96 GB in benchmarking rounds using a blocksize of 512 at 50 tps and lowest with 3.0 GB using a block size of 4 at 80 tps. This is probably due to the high throughput and low time outs which leads to the highest number of successfull transactions in the memory. The highest memory usage is seen with target arrrival rates of 50 and 60 tps. This are rates at which there are no timeouts. Higher arrival rates of 80 and 100 tps use less memory but don't finish all transactions within time.

Figure 11 shows the relation between used disk space and block size. Larger block sizes lead to a lower need for diskspace per transaction.

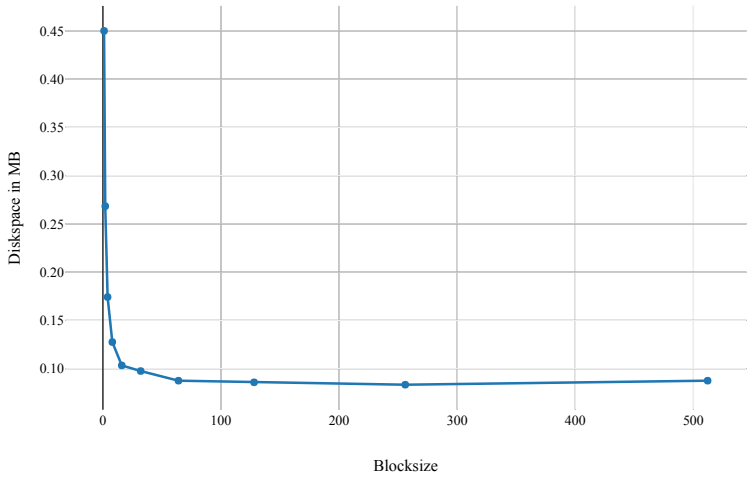

Figure 11: Used Disc Space per Blocksize

Block Size	Used Disc Space per Transaction in MB
1	0.45
2	0.27
4	0.17
8	0.13
16	0.1
32	0.1
64	0.09
128	0.09
256	0.08
512	0.09

Table 8: Used Disc Space per Blocksize

The least amount of disk space of 0.08 MB is needed when configuring the system with 256 transactions per block. Block sizes equal to or larger then 16 don't consume

more then 0.1 MB of disk space.

Dependent on the performance of the hard drive but true in any case: Transactions with less need for disk space are obviously faster written and also faster read than transactions which require more disk space. This information is viable to understand why operations (especially writing) take longer when the block sizes are smaller. Regarding throughput and failure rate, a block size of 128 has been recommended. This would take 0.09 MB per transaction.

4 Conclusion and Outlook

A measurement approach of a Blockchain-based Certification Storage System has been performed to test the system for performance in an artificial environment. The artefact based on Hyperledger Fabric has been benchmarked with Hyperledger Caliper. Tested was a single orderer environment with four peers on one host. Results were given regarding throughput, failure rate, latency and resoure utilization.

It has been shown that the systems performance is decisively dependent on configuration parameters. Reading operations are generally faster then writing operations. A higher block size leads to a higher throughput and to less latency until the performance limits of the host system is reached. The most important factor for systems stability is transaction arrival rate, meaning the rate at which transactions reach the BCSS but not necessarily the rate the system can process them. If this rate is higher than what the system is capable of, timeouts increase and throughput decreases. It is therefore of great importance to scale the host system to the business network's requirement. This report has shown that the tested host is able to process 16 concurrrent users with an arrival rate up to 60 tps. The recommended block size is 128 as there is no significant increase in throughput for larger block sizes without sacrificing system stability. With this configuration 37.48 tps when writing and 46.03 when reading are achievable on this system.

The benchmarking results are coherent with similar benchmarks. Timeouts could be decreased using 4 cores instead of 2 (Thakkar et al., 2018). Also throughput is greatly affected by the number of CPU cores. Thakkar et al. (2018) oberserve, considering their own configuration, 32 tps at 2 CPU cores and 848 tps 16 cores. This is especially true when dividing communication into separate channels. However the main trends, that the number of concurrent users is correlates negatively with the performance are also true in their work.

Although the throughput meets the requirements, the stable performance is only given to 16 simultaneous users. This is not enough for a realistic environment. As the research follows a Technical Risk & Efficacy evaluation strategy, future research should move the evalation to a more naturalstic environment. Technically, this means using multiple servers to expand performance limits and increasing the number of peers. In this case, ordering transactions can't be done on one host but can be harmonized by a service called *kafka*[4]. Considering that the current system can deliver a competitive throughput, it

[4] `https://hyperledger-fabric.readthedocs.io/en/release-1.4/kafka.html`

can be assumed that the use of a BCSS to manage certificates of workshop events is scalable through further decentralization and thus provides a solid basis for real-world use cases.

References

Arlitt, Martin, Diwakar Krishnamurthy, and Jerry Rolia (2001). "Characterizing the scalability of a large web-based shopping system". In: *ACM Transactions on Internet Technology* 1.1, pp. 44–69. ISSN: 1533-5399.

Avritzer, Alberto et al. (2002). "Software performance testing based on workload characterization". In: *Proceedings of the 3rd international workshop on Software and performance*. ACM, pp. 17–24.

Barber, Scott (2004). "Creating effective load models for performance testing with incomplete empirical data". In: *Web Site Evolution, Sixth IEEE International Workshop on (WSE'04)*. IEEE, pp. 51–59.

Caliper, Hyperledger (2019). *Hyperledger Caliper Architecture*. Electronic Article. URL: `https://hyperledger.github.io/caliper/docs/2_Architecture.html` (visited on 03/10/2019).

Hevner, Alan et al. (2004). "Design Science in Information Systems Research". In: *MIS quarterly* 28.1, pp. 75–105. DOI: 10.2307/25148625.

March, Salvatore T and Gerald F Smith (1995). "Design and natural science research on information technology". In: *Decision support systems* 15.4, pp. 251–266. ISSN: 0167-9236.

Smith, Connie U (2002). "Software performance engineering". In: *Encyclopedia of Software Engineering*.

Thakkar, Parth, Senthil Nathan, and Balaji Vishwanathan (2018). "Performance Benchmarking and Optimizing Hyperledger Fabric Blockchain Platform". In: *arXiv preprint arXiv:1805.11390*.

Venable, John, Jan Pries-Heje, and Richard Baskerville (2016). "FEDS: a framework for evaluation in design science research". In: *European Journal of Information Systems* 25.1, pp. 77–89. ISSN: 0960-085X.

Wickboldt, Clemens and Natalia Kliewer (2018). "Blockchain zur dezentralen Dokumentation von Werkstattereignissen in der Luftfahrtindustrie". In: *HMD Praxis der Wirtschaftsinformatik*, pp. 1–14. ISSN: 1436-3011.

— (2019). "BLOCKCHAIN FOR WORKSHOP EVENT CERTIFICATES – A PROOF OF CONCEPT IN THE AVIATION INDUSTRY (to appear)". In: *European Conference on Information Systems*.

Woodside, Murray, Greg Franks, and Dorina C Petriu (2007). "The future of software performance engineering". In: *2007 Future of Software Engineering*. IEEE Computer Society, pp. 171–187.

Appendix

```
{"blockchain": {
"type": "fabric",
"config": "./fabric.json"
},
"test": {
"name": "simple",
"description" : "Caliper Benchmark that writes Assets, certs, retrieves Assets and Certs.",
"clients": {
"type": "local",
"number": 1
},
"rounds": [
{
"label" : "open",
"txNumber" : [1000, 1000],
"rateControl" : [{"type": "fixed-rate", "opts": {"tps" : 100}}, {"type": "fixed-rate", "opts":
 ↪   {"tps" : 100}}],
"callback" : "benchmark/simple/writeAsset.js"
},
{
"label" : "open",
"txNumber" : [1000, 1000],
"rateControl" : [{"type": "fixed-rate", "opts": {"tps" : 100}}, {"type": "fixed-rate", "opts":
 ↪   {"tps" : 100}}],
"callback" : "benchmark/simple/writeCert.js"
},
{
"label" : "query",
"txNumber" : [1000, 1000],
"rateControl" : [{"type": "fixed-rate", "opts": {"tps" : 100}}, {"type": "fixed-rate", "opts":
 ↪   {"tps" : 100}}],
"arguments": {  "partSerialNumber": "1"},
"callback" : "benchmark/simple/queryAsset.js"
},
{
"label" : "query",
"txNumber" : [1000, 1000],
"rateControl" : [{"type": "fixed-rate", "opts": {"tps" : 100}}, {"type": "fixed-rate", "opts":
 ↪   {"tps" : 100}}],
"arguments": {  "partSerialNumber": "1"},
"callback" : "benchmark/simple/queryCert.js"
}]
},
"monitor": {
"type": ["docker", "process"],
"docker":{
"name": ["peer0.org1.example.com", "peer1.org1.example.com", "peer0.org2.example.com",
 ↪   "peer1.org2.example.com", "orderer.example.com"]
},
"process": [
{
"command" : "node",
"arguments" : "local-client.js",
"multiOutput" : "avg"
}],"interval": 1}}
```

Source Code 1: Example Caliper Benchmark Configuration File

Diskussionsbeiträge - Fachbereich Wirtschaftswissenschaft - Freie Universität Berlin
Discussion Paper - School of Business and Economics - Freie Universität Berlin

2019 erschienen:

2019/1 WANGENHEIM, Jonas von
English versus Vickrey Auctions with Loss Averse Bidders
Economics

2019/2 GÖRLITZ, Katja; Merlin PENNY und Marcus TAMM
The long-term effect of age at school entry on competencies in adulthood
Economics

2019/3 BEZNOSKA, Martin
Do Couples Pool Their Income? Evidence from Demand System Estimation for Germany
Economics

2019/4 BÖNKE, Timm; Astrid HARNACK und Miriam WETTER
Wer gewinnt? Wer verliert? Die Entwicklung auf dem deutschen Arbeitsmarkt seit den frühen Jahren der Bundesrepublik bis heute
Economics