Journal of Information Technology Ampera
Vol. 2, No. 1, 2021 e-ISSN: 2774-2121
https://journal-computing.org/index.php/journal-ita/index

Measuring Throughput and Latency Distributed Ledger Technology: Hyperledger

Riko Herwanto[1], Hari Sabita[2], Fajrin Armawan

[1,2,3] Technical Information Departement, Institute Information and Business of Darmajaya, Bandar Lampung, Lampung, Indonesia
Email: [1]rikoherwanto@darmajaya.ac.id, [2]hari.sabita@darmajaya.ac.id, [3]fajrin.armawan@darmajaya.ac.id

Abstract

In this paper, we report the measuring outcomes of Hyperledger, a Distributed Ledger, which is the derivation Blockchain Technology. A technique to evaluate Hyperledger in a limited infrastructure is developed. The measured infrastructure consists of 8 nodes with a load of up to 20000 transactions/second. Hyperledger constantly runs all evaluation, namely, for 20,000 transactions, the run time 74.30s, latency 73.40ms latency, and 257 tps. This initial evaluation can provide an overview for practitioners in making choices about the adoption of blockchain technology in their IT systems.

Keywords: Blockchain, Hyperledger, Latency, MySQL, Throughput

1. INTRODUCTION

In this work, Hyperledger Fabric [1], the implementation of Distributed Ledger Technology (DLT) [2, 3] from the Linux Foundation, is used as a benchmark. DLT manages Ledger through a peer-to-peer network using consensus mechanisms and smart contracts. Hyperledger is an implementation of the Blockchain framework that is used to develop applications with a modular architecture [4], and is an open-source Blockchain project and related projects.

As such, DLT provides a new model of business confidence and opportunity. For this reason, DLT is an emerging technology in many fields, such as Financial Technology (Fintech) [5], health services [6], including government organizations [7]. Unfortunately, due to complex peer-to-peer interactions, DLT performance is more difficult to access than centralized systems [8]. This work shows that Blockchain technology is comparable to older techniques in terms of latency [10]. In some cases, performance is better, and when considering a consistency model, there will likely be a number of use cases that mean immediately where the blockchain will be a better choice than a distributed database [11].

In this paper, we have evaluated Hyperledger Fabric v1.0. Assessment shows that Hyperledger Fabric v1.0 with more than two nodes has better performance in all evaluation metrics compared to just one.

2. RESEARCH METHODS.

The complex DLT architecture is divided into four layers, e.g., Network, Node, Ledger, and Application Layers to facilitate further analysis. At each layer, several metrics and influence factors are determined. Different metrics that are influenced by various factors are measured as benchmarks.

The concepts of primary DLT workloads and simulations are introduced. Based on this analysis, a framework is designed to build a distributed foundation of the test environment and make reproducible measurements. This framework is designed so that technology is evaluated and the testing environment is easily interchangeable. As a result, the design framework is implemented with benchmarking tools. For example, performance measurement and evaluation, Hyperledger Fabric, experiments are carried out in a controlled laboratory environment.

Evaluation of measurement results provides information about the performance effects of four factors, explicit changes in transaction rates, workloads, block sizes and the impact of packet loss. Measurements show that factors from each layer can directly affect the performance of the entire network, increase transaction rates, demand workloads, configuration memory that is unfavorable, or unfavorable network conditions Evaluation of measurement results provides information about the performance effects of four factors, explicit changes in transaction rates, workloads, block sizes and the impact of packet loss. Measurements show that factors from each layer can directly affect the performance of the entire network, increase transaction rates, demand workloads, configuration memory that is unfavorable, or unfavorable network conditions.

In this work, the Hyperledger blockchain platform, Hyperledger Fabric v1.0 is evaluated. Experiments carried out on i7 Laptop, 8GB RAM, 500GB SSD, 3 Core 2 Duo CPU, 4GB RAM, 160GB HDD, and running Ubuntu 18.04 LTE, this is used to compare Hyperledger's work with Relational Database, in this case, MySQL, with data load same, this experiment is to analyze the performance of the Hyperledger Fabric v1.0 platform. Platform performance evaluation is assessed in terms of latency and throughput implementation time by varying the workload of the number of transactions and requests requested simultaneously up to 20,000

transactions, transactions are measured by submitting transactions for consensus by partners, to add transactions to the block. Execution time is the time needed for the platform to add and run transactions successfully. Throughput can be defined as "the number of successful transactions per second." Finally, latency in a blockchain can be measured because of the time taken for a particular platform to respond to each transaction.

2.1 Throughput

Throughput is defined as the amount of data that is successfully transferred between nodes per unit of time, usually measured in seconds, as shown in Figure 1. Throughput is often defined for individual connections or sessions, but in some cases, total network throughput is determined. Ideally, throughput must equal capacity. Capacity depends on the physical layer technology used. Network capacity must be sufficient to handle the burden that occurs, even when the maximum is busy in network traffic. Theoretically, throughput will increase when the load offered increases, up to the maximum network capacity. However, network throughput depends on access methods (for example, relaying tokens or operator sensing), network load, and error rates. In the figure below, this shows an ideal situation, where the throughput increases linearly with the load offered, and in fact, where the actual throughput decreases because the load offered reaches a certain maximum point.

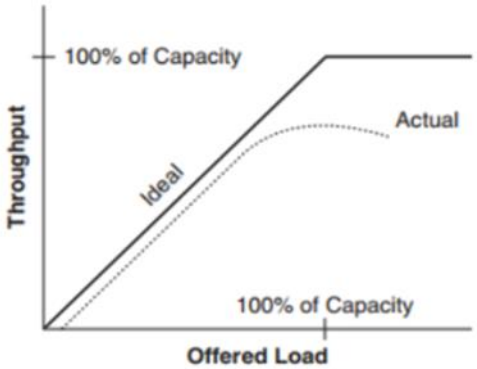

Figure 1. Throughput

2.2 Latency

Latency is related to the time taken to send messages from one end of the network to the other. Latency may also be the time lag required in sending data packets from sender to recipient. The higher the lag time or latency, the higher

the risk of access failure. Network latency is also often interpreted as the level of late delivery to voice and data communication networks. Latency is strictly measured in terms of time. For example, the network to send messages requires 24 milliseconds (milliseconds) from one end to the other. In general, there are three components of latency, namely, the delay of propagation, transit, and queuing.

2.3 Experiment Platform

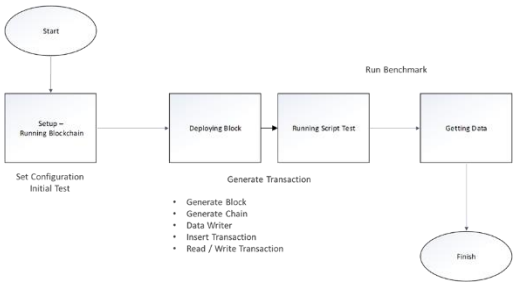

Figure 2. Experimental Design

In this paper, the evaluation framework for Ledger that is personally distributed is designed and developed. For this purpose, various layers have been determined. This is the network layer, above the node layer and the Ledger layer to the application layer. For each of these layers, metrics and factors have been identified, which make it possible to measure / influence the performance of the DLT network. Also, workloads have been determined, which emphasize the individual aspects of DLT or represent realistic use cases. The four phases of the experiment are:

a. Design Phase

The aim is to determine the framework that runs experiments on DLT. This will support technology configuration and actual measurement and evaluation of results. The design phase is to determine various objectives, namely, Throughput and Latency, discussed, divided into three phases, application, measurement, and evaluation, which will be discussed below.

b. Tested Phase

Testbed is prepared before the benchmarking process. This includes the installation and configuration of software, e.g., OS, Docker machines, Networks, which are needed to facilitate the following steps. After that, Ledger Hosts, equipped with GO language and Docker CE, are needed to perform DLT benchmarks. Testbed includes many tools, such as, Chaincode-Payload-Size,

Chaincode-Scalability, Channel Scalability, which are used during the measurement phase to record experimental hosts.

c. Measurement Phase

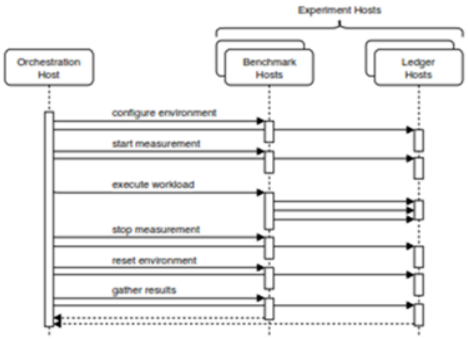

Figure 3. Workflow Design

Figure 3. explain the measurement phase, starting with the initial configuration steps. In running the experiment, every configuration change is applied to the trial host, such as network interference. After this, initialize all monitors on the experimental host where the experiment takes place. This may include recording network traffic or using resources, for example, Execution of workloads initiated by Host Orchestration, which will not interfere with the process and wait until it is finished. The benchmark host takes over and operates the experiment based on the workload definition on the Ledger host. All previous configuration and initialization steps allow the system to run the experiment without external intervention. All changes during the experiment are timed or directly induced by execution of the workload on the Ledger host. After workload execution is complete, the experiment is shut down. The monitor is stopped on the Ledger host, and any interruptions that occur during the measurement configuration step are canceled. For example, deleting all lost artificial network connections that have been placed on the network, to allow continuous uninterrupted work on the testbed.

Finally, any information collected is taken from the Ledger host and collected at the Orchestration Host. It was possible to immediately reset the Ledger host except for the Host Orchestration, for further measurements

d. Evaluation Phase

The final phase is the evaluation phase, where the data collected must be evaluated, starting with the preprocessing phase. The results that come from several hosts must go through the following steps:

- Simplify further processing, such as converting to standard file formats.

- Cleans up duplicate traffic recorded on many hosts.
- Normalization.
- Integration and connect various data sources

Processed data can then be evaluated based on relevant metrics. Several evaluation methods are defined in this work, and the format of the previously processed data makes it possible to add further evaluation methods so that they are easily obtained:
• Transaction & Read Latency: Measure the time the transaction was issued to be completed and the response available for the application that issued the transaction. The maximum, minimum, and latency for the test cycle is provided.
• Transaction & Read Throughput: Measures the flow rate of all transactions through the system, in transactions per second, during cycles.

In the experiment, the transaction is first executed to pre-fill the chain / Ledger with Block. This is the starting point for estimating how the system behaves when there is enough data in the system.
Then, a read-write transaction is run where each transaction reads and modifies the Block randomly. In each experiment, one (or two) parameters vary (marked in a dashed line), while keeping the other parameters fixed. The OS file system cache is not deleted between the Insert transaction and the read-write transaction assuming that, in practical settings, most of the data will immediately come from the file system cache. The experiment uses the following parameters:

- Total number of chains (default - 10)
- Number of transactions simulated in parallel on each chain (default - 10)
- Total Blocks throughout the chain (keys are distributed evenly throughout the chain) (default - 20,000)
- Number of keys that are read and modified randomly by each transaction (default - 4)
- Size values for each block (default - 200 bytes)
- Number of transactions in each block (default - 50)

Chaincode scripts were developed to set default experiments. The script is written using JavaScript Object Notation (JSON), which is a concise format for exchanging computer data. The format is text based, human readable, and is used to represent simple data structures and associative arrays (called objects). JSON is used to send structured data through a network connection. In this work, the scripts to set the configuration are as follows:

```
type
type configuration struct {
    chainMgrConf *chainmgmt.ChainMgrConf
    batchConf    *chainmgmt.BatchConf
    dataConf     *dataConf
    txConf       *txConf
}

func defaultConf() *configuration {
    conf := &configuration{}
    conf.chainMgrConf = &chainmgmt.ChainMgrConf{DataDir:
"/tmp/fabric/ledgerPerfTests", NumChains: 1}
    conf.batchConf = &chainmgmt.BatchConf{BatchSize: 10, SignBlock:
false}
    conf.txConf = &txConf{numTotalTxs: 20000,
numParallelTxsPerChain: 2, numWritesPerTx: 4, numReadsPerTx: 4}
    conf.dataConf = &dataConf{numKVs: 20000, kvSize: 200}
    return conf
}
```

3. RESULTS AND DISCUSSION

Figure 4. shows DLT increases transaction rates to around 257 transactions per second (tps). We simulate a network with eight (8) nodes. The distance between each node is constant for one (1) second. We produce various block levels that vary from 0 to 10 blocks. The size of the block is set to 50 transactions per block. The transaction arrival rate is 65 transactions per second.

Figure 4. also shows the execution time of each transaction, Blockchain spends more time on larger data volumes, reaching 2027 ms with 6KB of data, the average read and write time is 1.22 ms. However, we found that the time spent reading and writing data spent by Blockchain was 1660 times higher than MySQL.

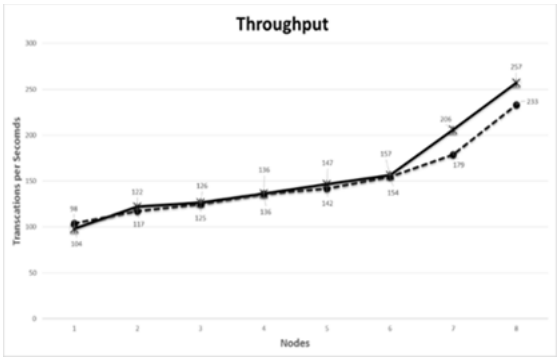

Figure 4. Throughput

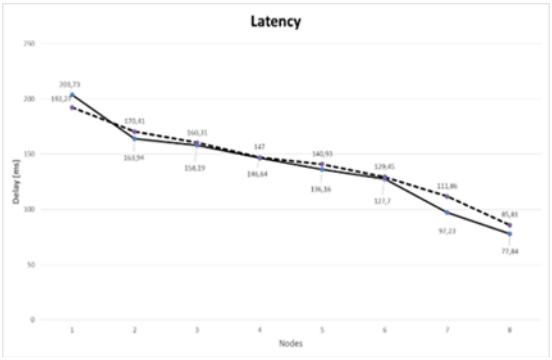

Figure 5. Latency

Throughput is directly dependent on two (2) parameters: the size of the block, which is the number of bytes that can contain transactions in each block, and the time interval between blocks, that is, the average time needed by the system to access a new block. To increase Hyperledger throughput, one can increase Block size and place more transactions, or to reduce inter-block time intervals, so that blocks are processed at a higher level. In other words, transactions per second (tps) are affected by an increase in the number of nodes. As shown in Figure 4, with an increase in transactions, throughput increases linearly to around 257 tps. At that time, as shown in Figure 5, latency decreased to 100 ms with a decrease of about 62 percent latency per node. Latency is the time lag required in sending data packets from sender to recipient. The higher the delay or latency, the higher the chance of access failure. Fortunately, latency decreases when more nodes join transactions.

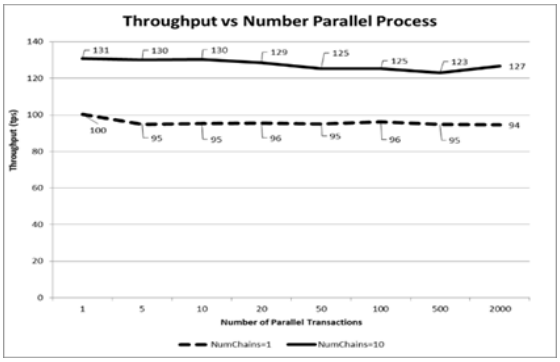

Figure 6. Throughput vs. Number of Parallel Processes

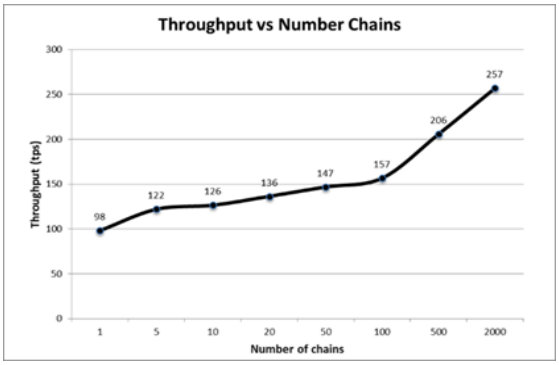

Figure 7. Throughput vs. Total Chains

Figure 6. shows that the number of parallel transactions on the chain does not affect overall throughput. We simulate the effect of the number of chains on output by running tests on several different chains. In this experiment, we used four parallel transactions that were simulated on each chain, 20,000 blocks in the entire chain with keys distributed evenly throughout the chain, four keys randomly read and modified by each transaction, a block size of 200 bytes, and 50 transactions in each Block. Because the processes in each block can be done simultaneously, processes with ten (10) chains have a 25 percent higher throughput compared to one (1) chain process. Thus, the more chains used in the process, the higher the throughput. Obviously, given Chain codes, which include scripts for processing inter-block transactions, can be installed on nodes that connect code between nodes and allow execution of parallel chain code, and will affect execution time.

Figure 7. shows that increasing the number of chains increases overall throughput, the main reason seems to be parallel validation and blocking the commit process. In this work, we use ten (10) chains, four (4) parallel transactions that are simulated in each chain, 20,000 blocks in all chains with keys distributed evenly throughout the chain, four (4) keys are read and modified randomly by each transaction, a block size of 200 bytes, and 50 transactions in each block. As shown in Figure 4, increasing the number of chains will increase throughput. As explained in Figure 4, transactions per second (tps) increase with an increase in nodes. Because each node has a copy of the ledger, this will allow increased throughput. However, as shown in Figure 7, when the chain is below 100, throughput does not seem to be significantly affected, around 38 percent of transactions per second. However, if someone adds the number of chains up to 5 times, the percentage of throughput increases to around 62 percent.

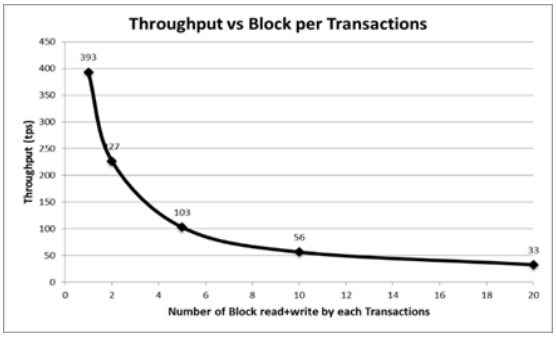

Figure 8. Throughput vs. Number of Transactions per Block

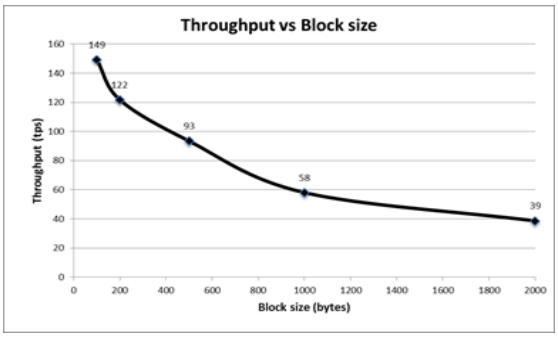

Figure 9. Throughput vs Block Size

Figure 8 shows that increasing the number of blocks, operated in each transaction, reduces overall throughput. In this work, we use ten chains, ten transactions are simulated in parallel on each chain, 20,000 blocks in all chains with keys distributed evenly throughout the chain, four keys are read and modified randomly by each transaction, each Block size is 200 bytes and 50 transactions in each Block consisting of the volume of transactions to be processed; each additional Block will reduce overall throughput because transactions are ordered and grouped and sent as blocks for review. Each colleague processes one block at a time.

Figure 9 shows that increasing Block size reduces overall throughput. In this work, we use ten chains, four parallel transactions are simulated on each chain, 20,000 blocks in all chains with keys distributed evenly throughout the chain, four (4) keys are read and modified randomly by each transaction, block size is 200 bytes, and 50 transactions in each block. As the block size increases, latency increases because the arrival rate increases with the block size. Thus, as the Block size increases, the number of transactions processed per second will increase which increases the throughput of all transactions.

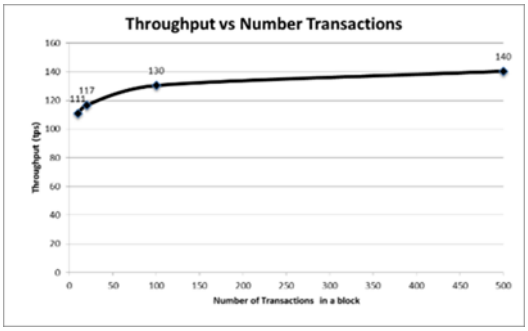

Figure 10. Throughput vs Number Transaction

To produce Figure 10, we used ten (10) chains, four (4) simulated parallel transactions simulated in each chain, 20,000 blocks in all chains with keys distributed evenly throughout the chain, four (4) keys read and modified randomly by each transaction, the Block size is 200 bytes, with 50 transactions in each Block. Increasing the number of transaction sizes will increase overall throughput. This increase has the potential for writing simultaneous mass transactions for each Block. The increase seems to occur in more than 100 transactions.

4. CONCLUSION

In this paper, we report a structured, experimental approach to characterizing the performance of the Hyperledger Blockchain platform. The framework developed in this paper was built to be expanded. This includes adding factors, metrics, workloads, and more factors that can be expanded with various network limitations such as limited network speed or network delays. Metrics must enter information about the load of each node, to allow making appropriate statements about distributed Ledger bottlenecks. In this work, this feature has been partially implemented. Thus, the amount of workload can be easily increased. We find that the read system throughput is found to be linear while the Write process is almost linear at low transaction rates under 1000 transactions per second. Read and write latency is affected by the number of nodes.

The number of participating nodes can increase and can result in better throughput and latency. Increasing the participating nodes may require infrastructure scaling to achieve the desired performance. More infrastructure scaling trials should be conducted on a testbed. Evaluating the limits of a distributed Ledger setting might also attract adjustment parameters to optimize its performance in the running environment.

Performance, that is, throughput, execution time, and latency, shows that in general, Hyperledger produces the same performance for a number of nodes, regardless of the load. However, Hyperledger's performance is affected because the number of nodes changes, and, thus, the number of Blocks, Block size, and Number of Transactions. To achieve higher throughput, greater efficiency, block intervals must be made as little as possible. We have found that the block interval for the Blocked-based Hyperledger protocol cannot be less than 12 seconds. This will ensure faster spread and low latency. These results imply that blockchain might be more suitable for data intensive applications / systems.

REFERENCES

[1] Androulaki E, Barger A, Bortnikov V, Cachin C, Christidis K, De Caro A, Enyeart D, Ferris C, Laventman G, Manevich Y, Muralidharan S. *Hyperledger fabric: a distributed operating system for permissioned blockchains.* Proceedings of the Thirteenth EuroSys Conference. Porto, Portugal. 2018; p. 30.

[2] H. Sukhwani, N. Wang, K. S. Trivedi and A. Rindos. *Performance Modeling of Hyperledger Fabric (Permissioned Blockchain Network)*. *2018 IEEE 17th International Symposium on Network Computing and Applications (NCA)*. Cambridge, MA. 2018: pp. 1-8.

[3] Walport M. *Distributed ledger technology: Beyond blockchain*. UK Government Office for Science. Tech. Rep; 2016.
[4] Davidson S, De Filippi P, Potts J. *Disrupting governance: The new institutional economics of distributed ledger technology*. 2016.
[5] Chishti S, Barberis J. *The FinTech book: the financial technology handbook for investors, entrepreneurs and visionaries*. New York. John Wiley & Sons; 2016.
[6] Cunningham J, Ainsworth J. *Enabling patient control of personal electronic health records through distributed ledger technology*. Stud Health Technol Inform. 2018; 245: 45-48.
[7] Genkin D, Papadopoulos D, Papamanthou C. *Privacy in decentralized cryptocurrencies*. Communications of the ACM. 2018; 61(6):78-88.
[8] Kocsis I, Pataricza A, Telek M, Klenik A, Deé F, Cseh D. *Towards Performance Modeling of Hyperledger Fabric*. InInternational IBM Cloud Academy Conference (ICACON). 2017.
[9] D. Chays and Yuetang Deng, *Demonstration of AGENDA tool set for testing relational database applications.* Proceedings of *25th International Conference on Software Engineering, 2003.* Portland, OR, USA. 2003: pp. 802-803.
[10] R. Yasaweerasinghelage, M. Staples and I. Weber. *Predicting Latency of Blockchain-Based Systems Using Architectural Modelling and Simulation*. *2017 IEEE International Conference on Software Architecture (ICSA)*, Gothenburg. 2017: pp. 253-256.
[11] Rauchs M, Glidden A, Gordon B, Pieters GC, Recanatini M, Rostand F, Vagneur K, Zhang BZ. *Distributed ledger technology systems: a conceptual framework*. Cambridge, UK. 2018.
[12] Qassim Nasir, Ilham A. Qasse, Manar Abu Talib, and Ali Bou Nassif. *Performance Analysis of Hyperledger Fabric Platforms*. Security and Communication Networks, vol. 2018, Article ID 3976093, 14 pages, 2018
[13] Gaur N, Desrosiers L, Ramakrishna V, Novotny P, Baset SA, O'Dowd A. *Hands-On Blockchain with Hyperledger: Building decentralized applications with Hyperledger Fabric and Composer*. Packt Publishing Ltd; 2018.
[14] A. Baliga, N. Solanki, S. Verekar, A. Pednekar, P. Kamat and S. Chatterjee, Performance Characterization of Hyperledger Fabric. *2018 Crypto Valley Conference on Blockchain Technology (CVCBT)*, Zug. 2018; pp. 65-74.
[15] White B, Lepreau J, Stoller L, Ricci R, Guruprasad S, Newbold M, Hibler M, Barb C, Joglekar A. *An integrated experimental environment for distributed systems and networks*. Proceedings of the 5th symposium on Operating systems design and implementation. Boston, Massachusetts. 2002;36(SI):255-270.

[16] Buchman E. *Tendermint: byzantine fault tolerance in the age of blockchains*. MA thesis. Department Engineering Systems and Computing. University of Guelph. Ontario, Canada. 2016.
[17] Dhillon V, Metcalf D, Hooper M. *The hyperledger project. In Blockchain enabled applications*. Apress, Berkeley, CA. 2017: 139-149
[18] Nakamoto S. Bitcoin: *A peer-to-peer electronic cash system*. 2008
[19] Nasir Q, Qasse IA, Abu Talib M, Nassif AB. *Performance analysis of hyperledger fabric platforms.* Security and Communication Networks. Vol. 2018; 14.
[20] Cachin C. *Architecture of the hyperledger blockchain fabric*. In Workshop on distributed cryptocurrencies and consensus ledgers. 2016:. 310: 4-11.
[21] C. Decker and R. Wattenhofer. *Information propagation in the Bitcoin network*. *IEEE P2P 2013 Proceedings*, Trento, 2013, pp. 1-10.
[22] Cunningham J, Ainsworth J. *Enabling patient control of personal electronic health records through distributed ledger technology*. Stud Health Technol Inform. 2018; 245:45-8.
[23] Genkin D, Papadopoulos D, Papamanthou C. *Privacy in decentralized cryptocurrencies*. Communications of the ACM. 2018; 61(6):78-88.
[24] Bolze R, Cappello F, Caron E, Daydé M, Desprez F, Jeannot E, Jégou Y, Lanteri S, Leduc J, Melab N, Mornet G. *Grid'5000: A large scale and highly reconfigurable experimental grid testbed*. The International Journal of High Performance Computing Applications. 2006; (4):481-494.
[25] Croman K, Decker C, Eyal I, Gencer AE, Juels A, Kosba A, Miller A, Saxena P, Shi E, Sirer EG, Song D. *On scaling decentralized blockchains*. In International Conference on Financial Cryptography and Data Security 2016). Springer, Berlin, Heidelberg. 2016; 106-125.
[26] Maull R, Godsiff P, Mulligan C, Brown A, Kewell B. *Distributed ledger technology: Applications and implications*. Strategic Change. 2017; 26(5): 481-489.
[27] N. Papadis, S. Borst, A. Walid, M. Grissa and L. Tassiulas. *Stochastic Models and Wide-Area Network Measurements for Blockchain Design and Analysis.* IEEE INFOCOM 2018 - IEEE Conference on Computer Communications. Honolulu, HI. 2018; pp. 2546-2554.
[28] Sajana P, Sindhu M, Sethumadhavan M. *On Blockchain Application: Hyperledger Fabric and Ethereum*. International Journal of Pure and Applied Mathematics. 2018;118(18): 2965-2970.
[29] S. Pongnumkul, C. Siripanpornchana and S. Thajchayapong. *Performance Analysis of Private Blockchain Platforms in Varying Workloads*. 2017 26th International Conference on Computer Communication and Networks (ICCCN). Vancouver, BC. 2017; pp. 1-6.

[30] P. Thakkar, S. Nathan and B. Viswanathan, *Performance Benchmarking and Optimizing Hyperledger Fabric Blockchain Platform*, IEEE 26th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS). Milwaukee, WI. 2018; pp. 264-276.

[31] Dinh TT, Wang J, Chen G, Liu R, Ooi BC, Tan KL. *Blockbench: A framework for analyzing private blockchains*. Proceedings of the 2017 ACM International Conference on Management of Data. 2017; pp. 1085-1100