Operating Permissioned Blockchain in Clouds: A Performance Study of Hyperledger Sawtooth

Zeshun Shi*, Huan Zhou*, Yang Hu*, Jayachander Surbiryala*†, Cees de Laat*, and Zhiming Zhao*
* Informatics Institute, Faculty of Science, University of Amsterdam, Amsterdam, Netherlands
{z.shi2, h.zhou, y.hu, delaat, z.zhao}@uva.nl
† Department of Electrical Engineering and Computer Science, University of Stavanger, Stavanger, Norway
{jayachander.surbiryala}@uis.no

Abstract—**With ever more IoT (Internet of Things) and big data applications, the emerging blockchain techniques provide fundamental supports to credibly track the transactions of digital assets. Public blockchains, e.g., bitcoin, are often energy-consuming and low efficient. Therefore, an empirical study of operating permissioned blockchains in clouds is urgently needed. In this paper, we study the performance of Sawtooth, a well-known permissioned blockchain platforms from Hyperledger, in cloud environments. Our results provide insights for blockchain operators to optimize the performance of Sawtooth through adjusting the two configuration parameters, i.e., Scheduler and Maximum Batches Per Block. Our approach can be used to test other blockchain platforms.**

Index Terms—**Permissioned Blockchain, Hyperledger Sawtooth, Cloud, Performance Study**

I. Introduction

The blockchain technologies recently gained popularity in several applications fields such as supply chains [1], governments and public sectors [2], healthcare and pharmaceutical [3], education [4] and insurance [5]. A blockchain records transactions among distributed participants as identical copies via decentralized ledgers, which are represented as a chain of blocks. A new block is generated and appended in the chain based on the consensus among distributed participants. A blockchain establishes trust among its distributed users via its immutability and security of the ledgers.

In general, blockchains can be permissionless and permissioned. In a permissionless blockchain, e.g., Bitcoin [6] or Ethereum [7], anyone can join the chain by validating transactions and/or making transactions. Since no basic trusts assumed among arbitrary anonymous participants, a consensus algorithm is usually leveraged to determine the *lucky* one who is allowed to package transactions and create a new block in a certain round. A typical example is the Proof of Work (PoW) algorithm. This mechanism is inefficient and consumes much energy in order to achieve a consensus. The wasted energy is inevitable to build trust among strangers without any prior experience. However, it is not usually the case that how the real world organizations collaborate with each other. In the scenario of most industrial applications, the participant has basic trust with most of the other partners, so the permissioned blockchain is proposed. In a permissioned blockchain, the identity of each participant is known and authenticated. It is believed that most of them would validate transactions honestly. Especially, the malicious one can be found out through Byzantine-like consensus algorithms. This mechanism empowers permissioned blockchains with great potential in various industrial applications, e.g., IoT [8], banking [9], and voting [10].

Cloud environments provide elastic and cost-effective resources for data storage, processing, and computing [11]. Nowadays, more and more enterprises are migrating their applications onto cloud for saving cost of their operations, continuously deploying and operating services, and increasing efficiency of IT infrastructure management [12] [13]. Adoption of cloud services for operating the rapid growth blockchain-based systems has encouraged researchers to study the applicability of blockchain techniques in various business environments [14]. However, the barriers of deploying a blockchain in clouds still exist, and the performance of the blockchain platform is often unstable due to too many influencing factors, especially if operating them in a dynamic cloud environment. In fact, performance is a critical factor for the enterprise to utilize a blockchain-based solution, and it directly determines whether the platform is applicable. Therefore, the evaluation of the permissioned blockchain platform in cloud environments is highly important to provide insights of operating Blockchain-as-a-Service.

This paper aims to investigate the performance of the permissioned blockchain in a cloud environment. The results can be considered as a baseline to study the operation of the permissioned blockchain based on Hyperledger Sawtooth in cloud. The rest of the paper is organized as follows: Section II introduces the basic knowledge of the blockchain, the architecture of Hyperledger Sawtooth, and the Proof of Elapsed Time (PoET) consensus algorithm. Section III delves into the problem statement and experiment methodology. Section IV presents the result on performance of our experimental study. Section V describes the related work of the blockchain performance analysis. Finally, in Section VI we conclude this paper with discussion of our discovery.

II. Background

A. Blockchain & Cloud

Permissionless blockchains allow anyone to participate and maintain transparent decentralized ledgers. The blockchain used in Bitcoin is often called the first generation, which leverages the ledger to record all the token transferring history. Then the balance of the certain account can be calculated through all its related transferring history records. The recording contents, however, are too simple to meet the requirements of real-world transactions. Therefore, there comes the second-generation permissionless blockchain, Ethereum. The most important contribution of Ethereum is to provide the ability of executing smart contract and recording the state changes in blockchain transactions. Hence, for instances, one Ethereum user, Alice, has the ability to define that only when certain condition is met, the token is then transferred to user Bob. This condition is public and no one has the ability to affect the transferring process as long as the condition is met. However, permissionless blockchains suffer the problem of low transaction throughput. Because most of them have to adopt the PoW consensus algorithm, in order to achieve trustworthiness among unauthorized participants.

On the other hand, a permissioned blockchain improves platform throughput performance by adopting more efficient consensus algorithms, e.g., PBFT (Practical Byzantine Fault Tolerance). This type of blockchain requires authentication first before the participant can join because Byzantine-like consensus algorithms cannot usually tolerant malicious ones more than one-third of the total participants. Anyhow, permissioned blockchains are more suitable for industrial applications, like IoT and big data, where there already exists basic authentications among processing units and collaborated organizations. Hyperledger[1] is an open source community to provide a set of solutions for permissioned-blockchain-based distributed ledgers, including Fabric[2], Iroha[3] and Sawtooth[4]. Furthermore, operating the permissioned blockchain platform in clouds and offering the ledger service is a promising solution, in order to make the blockchain platform operational and to be exploited by specific applications. Currently, several major public cloud providers, such as AWS (Amazon Web Services) [15] and Azure [16], have already attempted to provide this type of Blockchain-as-a-Service through their cloud platforms.

B. *Hyperledger Sawtooth & PoET Consensus*

Hyperledger Sawtooth is a permissioned blockchain platform for creating networks and distributed applications. The main design philosophy of Hyperledger Sawtooth is to simplify the development process of the blockchain application by separating the central system from the application layer. Enterprise users and application developers can use their own language to specify the business rules that are appropriate for their application without having to understand the underlying design of the core system [17]. The overall architecture of Hyperledger Sawtooth is shown in Fig. 1. A Sawtooth node participating in the system mainly consists of the following components: a validator, a REST API, some transaction processors, and clients. The validator is the core component of Hyperledger Sawtooth. Its main functions include receiving the transaction requests and forwarding them to the corresponding transaction processor. In addition, the validator needs to decide how to generate a new block based on the processing result of the transaction processor and how to echo the result to clients. Meanwhile, the validator also works with other validators to keep the global state of the Sawtooth network consistent. Transaction processors are used to encapsulate the application logic and business models. They work similarly to chaincodes in Hyperledger Fabric and smart contracts in Ethereum. Finally, A REST API is a bridge between the validator and clients.

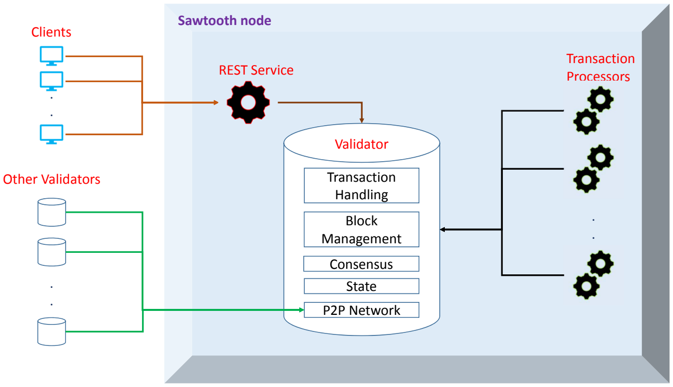

Fig. 1. Overall architecture and key components of one Sawtooth node.

PoET consensus algorithm is first developed by Intel in 2015 based on the Intel Software Guard Extensions (SGX) hardware, utilized as a Trusted Execution Environment (TEE). However, it is possible to use PoET in non-Intel-based systems using the PoET simulator with Hyperledger Sawtooth. PoET is a lottery based consensus algorithm. The use of TEE makes sure the selection process is carried out fairly. Intel merged this project with Hyperledger in 2016. After that, PoET becomes a trade mark of Hyperledger Sawtooth.

Associated with the context of Sawtooth, PoET works as follows: 1) each validator requests for a waiting time from the trusted module (enclave); 2) the enclave randomly assigns a waiting time for each validator; 3) the validator with the shortest time becomes the leader; and 4) once the waiting time has elapsed, the validator can claim the leadership with the verification of the allocated waiting time.

Apart from PoET, Hyperledger Sawtooth also supports pluggable consensus algorithms like PBFT and Raft [18]. However, it should be noted that in the current version of Sawtooth, PBFT and Raft have not been fully developed for stable use.

III. Problem Statement and Experiment Design

A. *Problem Statement*

To perform an in-depth study of the Hyperledger Sawtooth blockchain and benchmark the performance of Sawtooth with

[1] https://www.hyperledger.org/
[2] https://github.com/hyperledger/fabric
[3] https://github.com/hyperledger/iroha
[4] https://github.com/hyperledger/sawtooth-core

different cloud service providers and Virtual Machine (VM) instance configurations, we conducted several experiments with AWS [5] and ExoGENI [6] testbeds. AWS is a public cloud service platform that provides computing power, database storage, content delivery and other professional features to help businesses. On the other hand, ExoGENI is a community cloud service platform, which provides Networked Infrastructure-as-a-Service (NIaaS) for scientific experimentation. Through various experiments, we want to investigate following questions:

1) Performance Consistency Issue: with the same Sawtooth transaction workload, will the platform performance behave consistently each time with the same cloud and VM configuration?
2) Performance Stability Issue: with the same Sawtooth transaction workload, will the platform performance maintain stability varied with different clouds or VM configurations?
3) Performance Scalability Issue: with different Sawtooth transaction workloads, will the platform performance achieve scalability according to different configuration parameters?

B. Experimental Design (Setup and Workloads)

All the experiments of this paper are conducted in two clouds. In AWS, we select data centers of Virginia, California, Frankfurt, Sydney, Sao Paulo, and Singapore. In ExoGENI testbeds, we use following data centers: Pittsburgh Supercomputing Center (PIS), Oakland Scientific Facility (OSF), RCI in Chapel Hill (RCI), West Virginia Net (WVN), University of Alaska (UAF), and UMass Amherst (UMASS). We use three different instance types from both cloud providers, i.e., "XOSmal", "XOMedium", "XOLarge" from ExoGENI and "t2.Small", "t2.Medium", "t2.Large" from AWS. Table I shows the detailed configurations of instance type from both clouds, all the VMs are installed with "Ubuntu 16.04" operation system.

TABLE I
RESOURCE TYPE OFFERED BY EXOGENI AND AWS.

Cloud Provider	Resource Name	CPU Cores	Memory	DISK Size
ExoGENI	XOSmall	1	1G	10G
ExoGENI	XOMedium	1	3G	25G
ExoGENI	XOLarge	2	6G	50G
Amazon	t2.Small	1	2G	8G
Amazon	t2.Medium	2	4G	8G
Amazon	t2.Large	2	8G	8G

For the blockchain setup, we adopt Hyperledger Sawtooth v1.1 as the permissioned blockchain platform, and PoET is used as the consensus algorithm. During our experiments, Intkey [7] was used as the benchmark application. Intkey allows to set, increase, and decrease the value of entries stored in a state dictionary. Hence, it can be used to generate comprehensive and stable transaction workloads. Finally, when deploying the Sawtooth blockchain in the cloud, we chose to deploy only one Sawtooth node to one cloud VM. Because if we deploy multiple Sawtooth nodes on a single VM, there will be congestion among these nodes. In such scenario, we cannot clarify that the variation of performance is caused by conflict within the nodes or by some other parameters. It is also the reason that we choose VMs instead of containers. Because VMs can provide much better performance isolation than containers. In addition to VMs that have deployed the Sawtooth blockchain (by default in this paper, the Sawtooth platform consists of 5 nodes), we also deployed a monitor node to collect the real-time performance log data using InfluxDB[8]. Moreover, the entire process of provisioning, deploying, and executing is automated by CloudsStorm[9] [19], from which we can prototype an experiment by assembling available infrastructures and services, and execute them via a united engine [20]. Finally, the performance of the blockchain is measured by the commonly adopted metric "throughput", defined in equation 1, which is the rate at which transactions are committed to the blockchain platform [21].

$$Throughput = \frac{total\ committed\ transactions}{total\ time\ taken\ in\ seconds} \quad (1)$$

IV. EVALUATION AND EXPERIMENTAL RESULTS

A. Performance Consistency

In this section, we investigated the performance consistency of the Sawtooth blockchain with a single cloud service provider. All of the experiments were executed on AWS. By repeatedly executing the same Sawtooth benchmark workload multiple times, we can see the variation of the performance. Fig. 2 shows the results of benchmarking different input transaction rate with the same workload for 20 times. The x-axis is the number of times for testing, and the y-axis represents workload execution duration. Our results show that when the input transaction rate is low, the Sawtooth platform performs more consistently. But at the same time, it should be noted that the workload completion time is longer at lower rate, e.g., 3 tps (transactions per second), which means the input rate of the workload has not reached the performance bottleneck of the platform. When we increased the input transaction rate from 3 tps to 15 tps, the average throughput and variance of duration time increased a lot, as shown in Table. II.

Another observation is that as the transaction input rate increases, the overall duration time of the Sawtooth workload decreases. When the transaction input rate is around 12 tps, the current Sawtooth platform processing bottleneck is reached. As the rate continues to increase from 12 tps to 15 tps, the

[5] https://aws.amazon.com/

[6] http://www.ExoGENI.net/

[7] https://sawtooth.hyperledger.org/docs/core/releases/latest/cli/intkey.html

[8] https://www.influxdata.com/

[9] https://cloudsstorm.github.io/

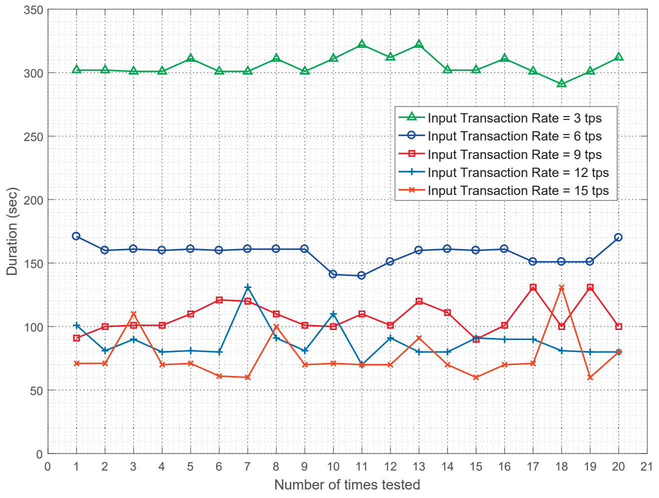

Fig. 2. Variation in performance of different input transaction rates.

TABLE II
PERFORMANCE VARIATION OF DIFFERENT INPUT TRANSACTION RATES.

Input Tx Rate	Average Throughput	Average Duration	Variance
3 tps	2.93 tps	305.90 sec	57.39
6 tps	5.67 tps	157.65 sec	60.13
9 tps	8.36 tps	107.50 sec	132.25
12 tps	10.24 tps	87.95 sec	172.25
15 tps	12.03 tps	76.40 sec	316.44

results of the two transaction rates show a random pattern with a few overlaps.

We also noticed that when the input transaction rate or workload is set to a high value, the transactions from different nodes can be easily forked, which leads to the Sawtooth blockchain fail to reach a consensus. This threshold value is determined by the consensus algorithms, blockchain configuration settings, and network conditions of cloud providers.

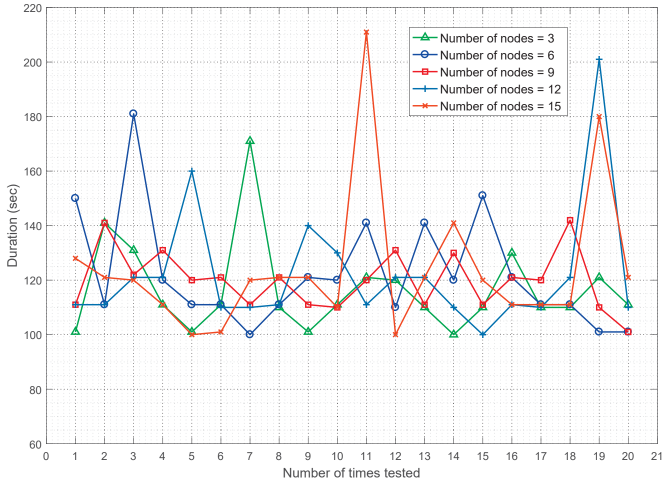

Fig. 3. Variation in performance of different number of VMs.

TABLE III
PERFORMANCE VARIATION OF DIFFERENT NUMBER OF VMS.

Number of VMs	Average Throughput	Average Duration	Variance
3	7.75 tps	116.60 sec	16.34
6	7.43 tps	122.20 sec	20.10
9	7.47 tps	119.80 sec	10.58
12	7.46 tps	122.05 sec	22.23
15	7.40 tps	124.00 sec	26.13

Fig. 3 and Table III shows no obvious impact on performance consistency of the Sawtooth blockchain with different number of VM nodes. We used the same workload and input transaction rate (9 tps) to benchmark the Sawtooth blockchain platform performance. When the number of VM instances changed from 3 to 15, the variance of the platform performance did not change significantly. Actually, the PoET consensus mechanism is especially designed for large networks [22] and it may not be relatively efficient for small networks, comparing to other algorithms, such as Raft or PBFT. Those algorithms may achieve better performance in small networks, but cannot maintain the performance consistency when scaling out the participant nodes. On the contrary, Table III indicates a trend that PoET is able to keep the performance consistency to fit a large-scale network.

B. *Performance Stability*

In this experiment, we studied the stability of the Sawtooth blockchain with the same transaction workload in different cloud providers, data centers, and network bandwidth between the VMs. In order to show the performance variation in different clouds, we deployed Sawtooth blockchains in both AWS and ExoGENI testbeds. For each cloud provider, three types of VM specifications and six data centers that are located geographically differently were used. Besides, the Linux Traffic Control (TC) tool was used to configure the bandwidth between the VMs.

Fig. 4 shows the throughput variation of Sawtooth with different bandwidths, here we use the same workload and input transaction rate (15 tps). In general, when the network bandwidth is greater than 100 MB, the median of platform throughput is around 12 tps. This indicates that platform performance is stable irrespective of bandwidth, and there are only a few cases where 1 or 2 outliers across different bandwidths. When the bandwidth is 100 MB, platform performance begins to show a dropping trend. As the bandwidth drops extremely to 1 MB, the platform performance drops to around one-third (4 tps). From the above results, we can conclude that the bandwidth has a certain impact on the performance of the Sawtooth blockchain, but the Sawtooth blockchain is not sensitive to bandwidth if the bandwidth is beyond a certain threshold. In this case, the platform performance will be influenced, if the bandwidth is below 100 MB.

Fig. 5 shows the performance analysis result of Sawtooth with different VM instance configurations. Actually, when the Sawtooth workload is small, the performance variance

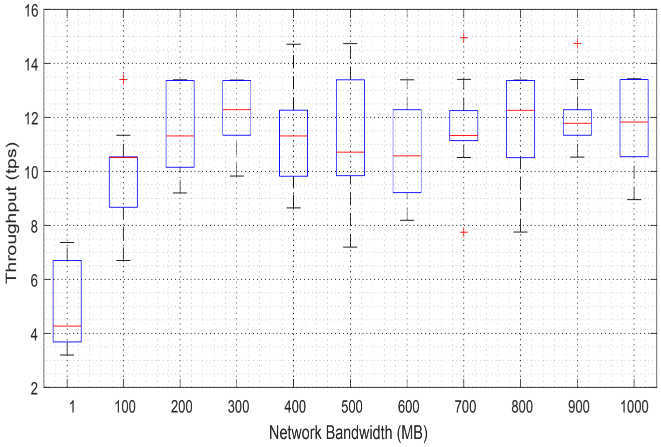

Fig. 4. Variation in performance of different network bandwidth.

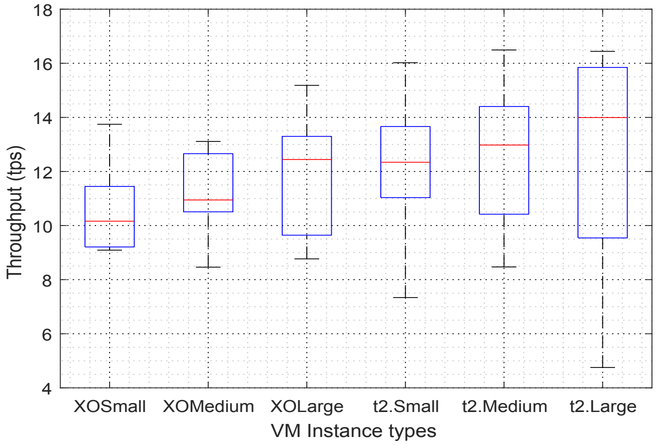

Fig. 5. Variation in performance of different VM instance types.

between different VM instance is not obvious because the number of workload does not reach the limit of the platform. So here we tested a large workload to see the difference between instance types. Overall, AWS outperforms ExoGENI because it has a better VM specification (CPU, Memory, and DISK) in a similar instance type. One observation is that as the VM instance type changes from small to medium and from medium to large, the average performance (throughput median) of Sawtooth has a significant improvement, which can be observed on both ExoGENI and EC2. Further, as the specifications of the VM increase, the performance of the platform is less concentrated. However, it should be noted that this does not mean that the platform is more unstable. In fact, when we input a huge workload, the blockchain deployed in the cluster with small or medium VM configurations often have forks, causing the platform fails to reach a consensus. In this case, we can only restart the blockchain platform and re-run the workload. But in a cluster with large VM configurations, although the throughput is sometimes low (small and medium configurations may have failed in this situation), it still can finally achieve consensuses and work normally, which proves that the configuration with larger VMs improves the platform's resilience.

Finally, Fig. 6 shows the performance of Sawtooth in different data centers. Here, we used the same instance type ("XOSmall" for ExoGENI and "t2.Small" for AWS). The six ExoGENI data centers are located in the United States. Among them, the performance of UAF and PIS is more stable than others, and WVN has the highest average throughput. We checked the resource utilization of different data centers and found this could be an important reason for the above phenomenon. When many customers use the same cloud resource at the same time, the data center becomes busy and the network condition gets worse (e.g., UMASS). On the contrary, when the data center is relatively idle, the performance of Sawtooth platform is better (e.g., WVN). In the AWS section, the California rack has the highest and most stable performance with an average throughput of around 12 tps, followed by the data centers of Frankfurt and Virginia. In contrast, the throughput of Australia and Singapore are lower. An interesting observation is that although Singapore has the lowest point of all test performance, its throughput is quite stable and maintained between 8 and 10. We also noticed that when Sawtooth platform was first built, the performance was extremely high and it went steady with several runs. This is the reason why some high-performance outliers occur.

C. *Performance Scalability*

In this section, we investigated the impact of different workloads on platform performance with the same cloud virtual infrastructure configuration. We also tested the platform performance variation when some parameter settings of Sawtooth changed. Here, we changed two parameters provided by Sawtooth: Maximum Batches Per Block (MBPB) and the scheduler type. Batch is the atomic unit of the state change of Sawtooth. In Sawtooth, transactions are carried out in batches. A batch contains a number of transactions, and when a particular transaction fails in the batch, all subsequent transactions fail. We can customize the MBPB parameter in Sawtooth blockchains to meet application requirements. In addition, transactions can be scheduled in the model of serial or parallel, which both produce deterministic results and are completely interchangeable. The running scheduler can schedule the next transaction based on the dependency graph of current transactions. If there is no dependency among the transactions, the parallel scheduler can deliver transactions to multiple transaction processors. On the contrary, the scheduler in serial model always delivers the transactions one by one to the transaction processors. In this experiment, 2 transaction processors are leveraged to perform the comparison.

Fig. 7 shows the impact on the platform performance with different workload input transaction rates and scheduler models. In this experiment, we used the same instance type and fixed execution time to observe changes in platform performance for different input transaction rates. As shown from the figure, parallel scheduling model has overall better throughput and platform performance than serial scheduling

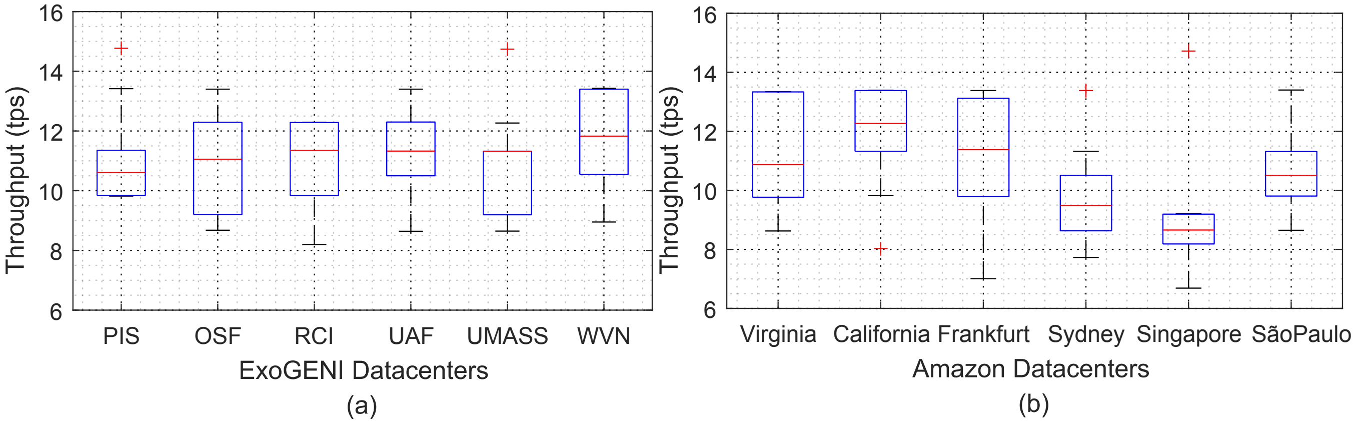

Fig. 6. Variation in performance of different cloud providers across various data centers.

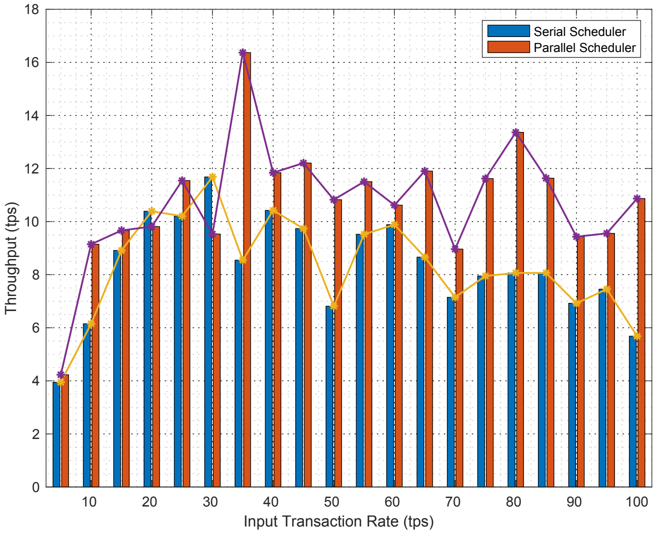

Fig. 7. Impact of different workloads and scheduler types on performance.

model. In fact, when changing from serial scheduler to parallel scheduler, the maximum value of throughput has increased from 11.68 tps to 16.37 tps, and overall throughput increased by almost 30%. It can be observed that when the input rate is increased from 10 tps to 100 tps, the platform throughput first reaches the highest point and then falls. Afterward, as the rate continues to increase, platform performance begins to stabilize gradually. We also notice that the performance of the serial scheduler shows a significant downward trend when dealing with the larger transaction input rate. The reason here is that the increased input rate has reached the upper limit of the platform performance, and extra transactions have to be pended or directly rejected. However, the parallel model can achieve better performance at the same input rate, so it is more stable to handle the workload with the large input rate. Therefore, when some transactions are in a non-uniform duration (e.g., realistic complex workloads), the performance advantage in parallel model is greater than that in serial model.

Fig. 8 shows the impact of the MBPB on platform performance. Here, the latency means the average execution duration for each transaction. As the MBPB value increases from 10 to 200, the latency first decreases rapidly and then reaches its threshold and tends to be stable. The result also shows that when the MBPB value is less than a certain threshold, the parallel model is significantly better than the serial model. After that, the two models show a random interlaced state because the performance bottleneck has been reached.

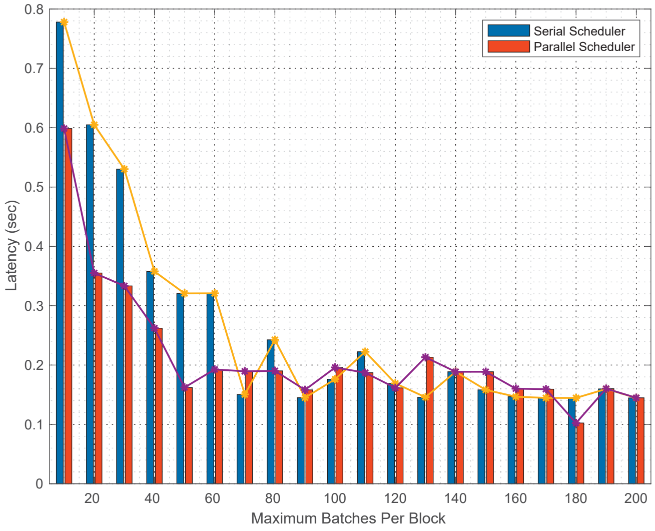

Fig. 8. Impact of different MBPB values and scheduler types on performance.

V. Related work

When operating a permissioned blockchain for a distributed application, performance (i.e., transactions per second) is an important factor to consider according to the application requirements. There already some studies that focused on the performance analysis of permissioned blockchains, either from on the comparison between different platforms or just from the consensus algorithms viewpoint. Among these studies, the most used platforms are Hyperledger Fabric and Ethereum. Nasir et al., [23] compared the performance of Hyperledger Fabric, v0.6 and v1.0, and their result demonstrated that Hyperledger Fabric v1.0 is more stable than Hyperledger Fabric v0.6 with many metrics. Rouhani et al., [24] did a performance analysis of Ethereum and compared two popular clients, i.e., Geth and Parity. It turned out that in the same configuration, Parity will perform almost 90% faster than Geth. In contrast, Pongnumkul et al., [25] compared the performance of Hyperledger Fabric and Ethereum private deployments. Their result shows that the Hyperledger Fabric consistently outperformed Ethereum in all evaluation metrics such as execution time, latency and throughput. Hao et al., [26] also used Hyperledger Fabric and Ethereum as the underlying blockchain platform, but they were more concerned with the consensus algorithm and the result showed that PBFT is generally better than PoW. Furthermore, Sukhwani et al., [27] designed a model to analysis the performance of PBFT consensus in the case of large peers, but their work still needs more experiments.

With respect to the research that uses Hyperledger Sawtooth as a blockchain platform, Hulea et al., [28] presented a reliable pharmaceutical cold chain management platform based on Hyperledger Sawtooth. Similarly, Caro et al., [29] proposed a blockchain-based agricultural food IoT framework AgriBlockIoT, and deployed the framework on two blockchain platforms, i.e., Hyperledger Sawtooth and Ethereum. They evaluated the performance of the AgriBlockIoT framework with latency, CPU, and network usage. But it should be noticed that their performance analysis only stayed at a preliminary stage. And, there is currently no papers on operating Hyperledger Sawtooth in a dynamic cloud environment.

VI. Discussion & Conclusion

In this paper, we presented a comprehensive performance analysis of Hyperledger Sawtooth with PoET consensus algorithm to demonstrate its performance variation in a dynamic cloud environment. Since operating permissioned Blockchain-as-a-Service in clouds is more complex and dynamic than local clusters, there are more factors that affect the platform performance. Therefore, we generally conducted our experiments from the perspective of performance consistency, performance stability, and performance scalability. Based on our analysis, we have obtained the following observations according to the above proposed questions.

Firstly, with respect to question 1, the platform performance of Hyperledger Sawtooth in the same data center is relatively consistent. When running the same workload for several times, the platform throughput has some changes and fluctuates within a small range. However, it should be noted that the input transaction rate has a significant impact on platform performance consistency with the same workload. We found that the larger the input transaction rate, the smaller the consistency of the platform performance, which means that performance will vary significantly when running a huge workload in a short period of time. In addition, when adjusting the number of VM instances from 3 to 15, the blockchain performance does not have any obvious trend. This means the Sawtooth blockchain platform is scalable because the performance is able to keep relatively consistent even after increasing the number of VM instances.

Secondly, for question 2, performance stability is influenced by the network condition of cloud providers, resource utilization of data centers, and other factors. In our experiments, the AWS California data center has the highest throughput. By contrast, although Singapore has the lowest average performance, it has the most stable performance among all tested data centers from AWS and ExoGENI. When we adjusted the different cluster configurations, we found that the performance stability of Hyperledger Sawtooth is not sensitive to network bandwidth beyond a certain threshold. By improving the VM specifications (e.g., CPU & Memory) for a large workload, platform performance improved significantly. However, an interesting finding is that with an increase in specifications of cloud VM instances, the stability of the performance decreases. Here, we believe that this happens because of survivor bias. In fact, small and medium nodes often have forks and crash with a large workload. In this case, we can only restart and evaluate the Sawtooth blockchain again. Whereas in the case of large VM instances, they are more stable, has higher system resistance, and self-recovery ability.

Finally, for question 3, we found that users can optimize the performance of Hyperledger Sawtooth by adjusting the configuration parameters (Scheduler and MBPB). The parallel scheduler can increase performance by around 30% when running a series of incremental workloads in our experiments.

To the best of our knowledge, this is the first study which focuses on the comprehensive performance analysis of Hyperledger Sawtooth in a dynamic cloud environment. However, it should be noted that since Hyperledger Sawtooth is a relatively new permissioned blockchain platform, many of its functionalities and modules are still under development and debugging. In future work, we will analyze the performance of different permissioned blockchain platforms in the cloud. It is also important to compare more consensus algorithms, more cloud platforms, and cross-cloud operations.

Acknowledgment

This research is funded by the EU Horizon 2020 research and innovation program under grant agreements 825134 (ARTICONF project), 654182 (ENVRIPLUS project) and 824068 (ENVRIfair project).

References

[1] F. Tian, “An agri-food supply chain traceability system for china based on rfid & blockchain technology,” in *2016 13th international conference on service systems and service management (ICSSSM)*, pp. 1–6, IEEE, 2016.

[2] S. Ølnes, “Beyond bitcoin enabling smart government using blockchain technology,” in *International Conference on Electronic Government*, pp. 253–264, Springer, 2016.

[3] Q. Xia, E. B. Sifah, K. O. Asamoah, J. Gao, X. Du, and M. Guizani, “Medshare: Trust-less medical data sharing

among cloud service providers via blockchain," *IEEE Access*, vol. 5, pp. 14757–14767, 2017.

[4] G. Chen, B. Xu, M. Lu, and N.-S. Chen, "Exploring blockchain technology and its potential applications for education," *Smart Learning Environments*, vol. 5, no. 1, p. 1, 2018.

[5] V. Gatteschi, F. Lamberti, C. Demartini, C. Pranteda, and V. Santamaría, "Blockchain and smart contracts for insurance: Is the technology mature enough?," *Future Internet*, vol. 10, no. 2, p. 20, 2018.

[6] S. Nakamoto *et al.*, "Bitcoin: A peer-to-peer electronic cash system," 2008.

[7] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum project yellow paper*, vol. 151, pp. 1–32, 2014.

[8] M. Samaniego and R. Deters, "Blockchain as a service for iot," in *2016 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, pp. 433–436, IEEE, 2016.

[9] Y. Guo and C. Liang, "Blockchain application and outlook in the banking industry," *Financial Innovation*, vol. 2, no. 1, p. 24, 2016.

[10] A. B. Ayed, "A conceptual secure blockchain-based electronic voting system," *International Journal of Network Security & Its Applications*, vol. 9, no. 3, pp. 01–09, 2017.

[11] H. Zhou, X. Ouyang, Z. Ren, J. Su, C. de Laat, and Z. Zhao, "A blockchain based witness model for trustworthy cloud service level agreement enforcement," in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*, IEEE, 2019.

[12] K. Jeferry, G. Kousiouris, D. Kyriazis, J. Altmann, A. Ciuffoletti, I. Maglogiannis, P. Nesi, B. Suzic, and Z. Zhao, "Challenges emerging from future cloud application scenarios," *Procedia Computer Science*, vol. 68, pp. 227–237, 2015.

[13] S. Koulouzis, P. Martin, H. Zhou, Y. Hu, J. Wang, T. Carval, B. Grenier, J. Heikkinen, C. de Laat, and Z. Zhao, "Time-critical data management in clouds: Challenges and a dynamic real-time infrastructure planner (drip) solution," *Concurrency and Computation: Practice and Experience*, p. e5269, 2019.

[14] H. Zhou, C. de Laat, and Z. Zhao, "Trustworthy cloud service level agreement enforcement with blockchain based smart contract," in *2018 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, pp. 255–260, IEEE, 2018.

[15] Amazon Web Services (AWS), "Blockchain on aws." https://aws.amazon.com/blockchain/. [Online; accessed 26-Apr-2019].

[16] Microsoft Azure, "Blockchain workbench." https://azure.microsoft.com/en-us/features/blockchain-workbench/. [Online; accessed 25-Apr-2019].

[17] Hyperledger Sawtooth, "Introduction." https://sawtooth.hyperledger.org/docs/core/releases/latest/introduction.html. [Online; accessed 17-Mar-2019].

[18] Hyperledger Sawtooth, "Raft documentation." https://sawtooth.hyperledger.org/docs/raft/nightly/master/. [Online; accessed 22-Mar-2019].

[19] H. Zhou, Y. Hu, J. Su, C. de Laat, and Z. Zhao, "Cloudsstorm: An application-driven framework to enhance the programmability and controllability of cloud virtual infrastructures," in *International Conference on Cloud Computing*, pp. 265–280, Springer, 2018.

[20] Z. Zhao, A. Belloum, C. de Laat, P. Adriaans, and B. Hertzberger, "Distributed execution of aggregated multi domain workflows using an agent framework," in *2007 IEEE Congress on Services (Services 2007)*, pp. 183–190, IEEE, 2007.

[21] Hyperledger Performance and Scale Working Group, "Hyperledger blockchain performance metrics white paper." https://www.hyperledger.org/resources/publications/blockchain-performance-metrics. [Online; accessed 28-Mar-2019].

[22] Hyperledger Sawtooth, "Frequently-asked questions." https://sawtooth.hyperledger.org/faq/. [Online; accessed 22-Mar-2019].

[23] Q. Nasir, I. A. Qasse, M. Abu Talib, and A. B. Nassif, "Performance analysis of hyperledger fabric platforms," *Security and Communication Networks*, vol. 2018, 2018.

[24] S. Rouhani and R. Deters, "Performance analysis of ethereum transactions in private blockchain," in *2017 8th IEEE International Conference on Software Engineering and Service Science (ICSESS)*, pp. 70–74, IEEE, 2017.

[25] S. Pongnumkul, C. Siripanpornchana, and S. Thajchayapong, "Performance analysis of private blockchain platforms in varying workloads," in *2017 26th International Conference on Computer Communication and Networks (ICCCN)*, pp. 1–6, IEEE, 2017.

[26] Y. Hao, Y. Li, X. Dong, L. Fang, and P. Chen, "Performance analysis of consensus algorithm in private blockchain," in *2018 IEEE Intelligent Vehicles Symposium (IV)*, pp. 280–285, IEEE, 2018.

[27] H. Sukhwani, J. M. Martínez, X. Chang, K. S. Trivedi, and A. Rindos, "Performance modeling of pbft consensus process for permissioned blockchain network (hyperledger fabric)," in *2017 IEEE 36th Symposium on Reliable Distributed Systems (SRDS)*, pp. 253–255, IEEE, 2017.

[28] M. Hulea, O. Rosu, R. Miron, and A. Aştilean, "Pharmaceutical cold chain management: Platform based on a distributed ledger," in *2018 IEEE International Conference on Automation, Quality and Testing, Robotics (AQTR)*, pp. 1–6, IEEE, 2018.

[29] M. P. Caro, M. S. Ali, M. Vecchio, and R. Giaffreda, "Blockchain-based traceability in agri-food supply chain management: A practical implementation," in *2018 IoT Vertical and Topical Summit on Agriculture-Tuscany (IOT Tuscany)*, pp. 1–4, IEEE, 2018.