2021 IEEE International Conference on Omni-Layer Intelligent Systems (COINS) | 978-1-6654-3156-9/21/$31.00 ©2021 IEEE | DOI: 10.1109/COINS51742.2021.9524264

A Blockchain cloud architecture deployment for an industrial IoT use case

Luc Gerrits[1], Edouard Kilimou[1], Roland Kromes[1], Lionel Faure[2], and François Verdier[1]

[1]University Cote d'Azur, LEAT / CNRS UMR 7248, Sophia Antipolis - France
{luc.gerrits, tetouhe.kilimou, roland.kromes, francois.verdier}@univ-cotedazur.fr

[2]TAS Group, 15 traverse des Brucs, 06560 Valbonne Sophia-Antipolis - France
lionel.faure@tasgroup.fr

***Abstract*—This paper presents a blockchain cloud deployment for an industrial vehicle use case. Hyperledger Sawtooth is used as distributed ledger with Rancher and Kubernetes as container-orchestration. This paper aims to explore the feasibility and scalability of the use case in a real-world scenario cloud deployment. Also, the blockchain performances is analyzed by stressing the implementation with virtual IoT clients, changing the blockchain settings, number of peer members and editing the core software default parameters. Benchmarks will reveal that Hyperledger Sawtooth performance can be as high as 25 transaction per seconds for a network of 4 nodes with PBFT consensus. Results lead us to discuss the possible components reducing throughput and blockchain commit rate.**

***Index Terms*—Blockchain, Cloud, Hyperledger Sawtooth, PBFT**

I. Introduction

Blockchain technology has emerged with Bitcoin in 2008 [1] with a novel protocol aimed to distribute a ledger among multiple peers using a common consensus — Proof of Work (PoW). The blockchain ledger is composed of blocks linked to each other using the previous block's hash (i.e., a chain of blocks). Each block contains transactions that are equivalent to the data that the blockchain contains. The data transparency, traceability and immutability enable the existence of cryptocurrencies (bitcoin in this case). More recently, blockchain technology has evolved to allow the execution of custom business logic in the form of programs inside the blockchain, called Smart Contracts [2]. Ethereum Virtual Machine (EVM) can execute Solidity compiled code directly in each network's peer. After the Ethereum release with its smart contracts, groundbreaking new applications and use cases emerged from this technology (supply-chain [3], smart grid [4], healthcare [5]).

This paper focuses on industrial blockchain platforms capable of answering an industrial use case. The vehicular use case requires sending approximately 25 transactions per hour (calculated from ONISR 2019 accident report [6]). Performances of a blockchain depend on multiple parameters such as finality, transaction execution rate and smart contracts execution rate. The smart contracts execution rate is an important element to build an industrial blockchain.

The selected use case is represented in Fig. 1 and could represent a vehicle infrastructure. In this use case, Renault's cars are connected to blockchains deployed on several clouds and are able to connect each time an accident occurs. The ideal scenario is that Renault's cloud is connected with other types of organizations like insurance companies, expertise, police and car mechanics.

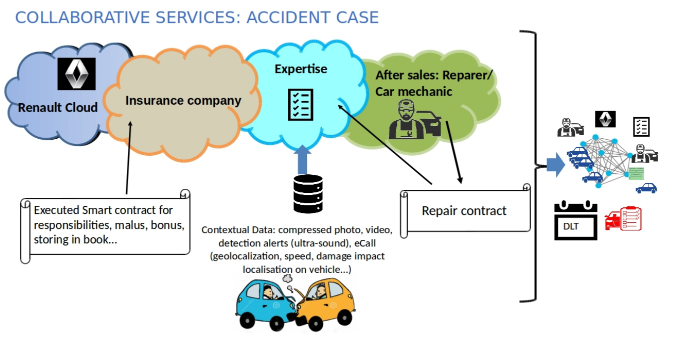
COLLABORATIVE SERVICES: ACCIDENT CASE
Renault Cloud
Insurance company
Expertise
After sales: Reparer/ Car mechanic
Executed Smart contract for responsibilities, malus, bonus, storing in book...
Contextual Data: compressed photo, video, detection alerts (ultra-sound), eCall (geolocalization, speed, damage impact localisation on vehicle...)
Repair contract
DLT

Fig. 1. Blockchains are used for declaring automatically the accidents in the Renault's vehicular infrastructure

The cars will contain IoT devices, and will be able to establish a connection with Renault's cloud technology based on a blockchain with smart contracts. The cars will contain various sensors that can determine the state of the car (odometers, radars, camera, etc.). The data recorded before the accident would be sent to Renault's cloud using a smart contract. After transaction execution, other companies connected to Renault's cloud could get access to the blockchain ledger and would allow adding new transaction processing and verification. With the help of police and predefined juridically legal smart contracts, the insurance company could, for example, determine who was responsible for the accident situation. This infrastructure contains a lot of different entities. The obtained structure is rather an ecosystem than a typical vehicle infrastructure.

This paper focuses on implementing a blockchain in a cloud. A fixed cloud architecture using a reasonable amount of processing power simulates a real-world cloud environment. Thanks to TAS Group, it is possible to create such cloud environment. TAS Group is a leading technology company, listed on the stock exchange, providing advanced solutions for cards, payment systems and capital markets. The group operates data centers in France (Sophia Antipolis) and Italy from which it offers hosting and cloud computing services.

The article continues as follows: In Sect. II is a brief description of the blockchain Hyperledger Sawtooth, the smart contracts, and the cloud technology, including related works about vehicle infrastructure systems. Sect. III describes the deployment of the use case to a real world scenario using a cloud. Sect. IV investigates the feasibility, scalability and performances of the proposed cloud model. Sect. V discusses about future improvements and conclude about the cloud deployment.

II. State of the Art

A. Hyperledger Sawtooth blockchain

Hyperledger Sawtooth is part of Hyperledger blockchain framework family, initially contributed by Intel [7]. Sawtooth supports both permissioned and permissionless deployments. Sawtooth software architecture is highly modular (depicted in Fig. 2), which suits enterprise use case.

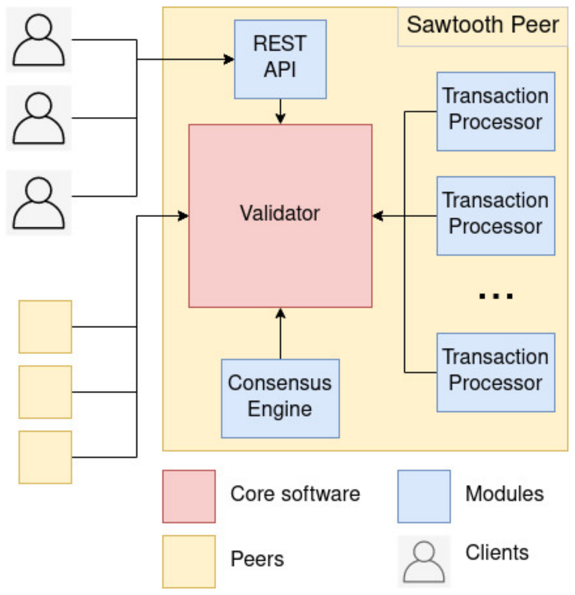

Fig. 2. Hyperledger Sawtooth architecture

Sawtooth modular structure is constituted basically of a validator which is the core of the peer, one or multiple transaction processors handling transaction business logic and a REST API providing convenient HTTP communication with the peers. Moreover, Sawtooth supports two main consensus algorithms (PoET and PBFT) and other algorithms for development purposes or under development.

- **PBFT**: Practical Byzantine Fault Tolerance is introduced in 1999 by Barbara Liskov and Miguel Castro [8]. It is a voting based consensus algorithm used for small P2P networks that does not require open membership. PBFT is derived from Byzantine problems. IBM researchers studied PBFT performances and showed that the number of nodes is limited to 20 (throughput follows a $O(1/n)$ function) [9]. PBFT allows blocks to be validated, based on a minimum percentage of agreement of the nodes in the network (usually 2/3), so even if there are a few faulty or malicious nodes in the network, the overall system can continue to operate in a stable manner.
- **PoET**: Based on Software Guard Extension (SGX) technology, PoET was implemented in Intel's Sawtooth Lake, before Intel joined Linux Foundation's Hyperledger consortium [10]. In PoET consensus algorithm the network peers have to wait a randomly distributed amount of time. This amount is distributed after every new block committing. The peer that had the smallest amount of wait time will have the right to publish a new block.

This work only studies the use case with a PBFT consensus. Shi et al. [11] have studied Hyperledger Sawtooth's performance as a private blockchain, using the PoET consensus. The author used AWS and ExoGENI cloud providers, which provided them multiple resources configuration directly from the service. Their study results in an average throughput of 7 transaction per seconds and that Sawtooth input transaction rate is limited (Fork and node fail rise for higher input transaction rate).

B. Smart contracts

Smart contracts' general idea is to deploy and execute a program inside the blockchain (executed in a smart contracts virtual machine). In industrial use cases, the smart contracts implementation will determine how the data will be processed. Thus deciding the type of smart contracts is essential when selecting a blockchain platform.

Hyperledger Sawtooth has a transaction processor that is analogous to a smart contract. A transaction processor is a program deployed by the node owner and runs next to a validator. Each transaction processor describes one specific application business logic (i.e., transaction family), and can be written in Python, Rust, C++, and Java. In a Hyperledger Sawtooth network, each node validator has one or multiple transaction processors, which allow parallel transaction execution. Hyperledger Sawtooth also developed a transaction processor that enables full compatibility with Solidity smart contracts using an Ethereum Virtual Machine (EVM). Solidity is an object-oriented statically-typed programming language initially used for the development of Smart Contracts on Ethereum Virtual Machines.

Other blockchains, such as the Substrate blockchain framework [12], provides two possibilities by implementing a traditional smart contracts EVM or application-specific runtimes modules. Runtime modules in Substrate are business logic deployed directly inside the blockchain storage. Thus the modules are part of the blockchain state. In this paper a Hyperledger Sawtooth transaction processor was developed in Rust [13]. Rust is a language created by Mozilla Research that takes full advantage of modern hardware (parallelism, concurrency, memory protection). Rust language performance has often been compared to C++.

C. Blockchain and cloud computing

In the proposed ecosystem the blockchain is deployed by a cloud and the end devices are IoT. IoT associated with blockchain requires additional computing resources because IoT devices cannot embed the blockchain's size and blockchain processing power requirements, thus edge computing is best suited for IoTs. For these reasons, end-devices perform only some cryptography processing, transaction formatting, and send data to the blockchain hosted in the cloud.

Cloud computing is the on-demand availability of computer resources through services available on the Internet. Cloud computing has a centralized infrastructure (of data-centers) in multiple locations worldwide. Each data-centers are hosting the cloud, and different services allow the deployment of applications, data, networks, and other infrastructures.

In public blockchain networks, hardware resources have diverse nature due to the network's open-source nature; for example, in Bitcoin networks, miners have evolved from workstations with CPUs (first generation miners) to application-specific integrated circuits (ASICs miners) [14].

In consortium blockchains networks, nodes are owned by consortium organizations. Availability constraints often force organizations to pay for Infrastructure-as-a-Service (IaaS) from various cloud providers (or can host their own nodes). This leads to multi-cloud-based networks. However, in this paper, the solution is based on a single cloud provided by TAS Group.

Other cloud providers offer Blockchain as a Service based on Software as a Service model: Microsoft and ConsenSys introduce Ethereum Blockchain-as-a-Service (BaaS) on Microsoft Azure in 2015 [15], Amazon introduced Ethereum and Hyperledger Fabric BaaS in 2019 [16]. The only BaaS service based on Hyperledger Sawtooth found is *Sextant*, a one click blockchain deployment solution offered by Blockchain Technology Partner (BTP), in Amazon Web Services Marketplace [17]. It is noticeable that all these offers are based on a specific blockchain platform and less configurable by users. In the context of this paper, for performance analyzes purpose, blockchains settings need to be changed to make observations according to these modifications.

Hyperledger Sawtooth's core is based on a modular architecture whose components are suitable with application containers. In this paper, Rancher [18] is used which is an open source software platform that use Kubernetes for containers orchestration.

Edge computing brings the computer resources closer to the end-devices and pre-processes information to save bandwidth and storage. This paper use case requires a decentralized network of blockchain nodes to be deployed on a cloud because it can i) simulate the decentralization using Kubernetes, ii) handle the process requirements of an entire blockchain network and iii) do an easy configuration variation.

Other paper such as Ampel *et al.* [19] evaluate the throughput of Sawtooth using the now deprecated Calliper benchmark tool. They found that the blockchain can achieve a 2300tps throughput, but do not show detailed results about the reject rate and commit rate (only input tps and transaction latency). Moreover, Benahmed *et al.* [20] has a more in-deph analysis of Sawtooth (CPU, RAM, tps, and node count) and concluded a 3 tps after 1000 transactions with the PoET consensus.

III. Cloudification of vehicle use case

A. Problem statement

Hyperledger Sawtooth has been tested numerous times to study its strengths and weaknesses in research environments. This paper describes and study the Hyperledger Sawtooth blockchain closer to industrial conditions. This study will answer:

- The first step on how to deploy a local private blockchain project to a cloud. What is the cloud architecture of an industrial blockchain?
- What are the performances of the deployed private blockchain? Does the realized deployment perform differently than a local execution? Are the results consistent in time, and what is the impact of the number of peers in the blockchain network?
- What issues are emerging from a cloud-based blockchain?
- Verify PBFT consensus network limitation.

The cloud deployment aims to get as close as possible to a real case situation. There are multiple ways to configure the Infrastructure-as-a-Service (IaaS), and the provided deployment is the compilation of authors research and the TAS Group engineer's guidance.

B. Cloud architecture

It is a Kubernetes cluster of 6 cloud instances, including three masters and three workers. The cluster is managed by *Rancher*. The blockchain network runs on workers with a total capacity of 282 GB of RAM (94.2 GB x 3 workers) and 192 CPU cores (64 CPU cores x 3 workers). To reduce latency due to cluster management tools running, a significant amount of resources is provided to Kubernetes masters; masters have a total capacity of 48 GB RAM (16 GB x 3 masters) and 24 CPU cores (8 CPU cores x 3 masters). By default, each instance has 50 GB of storage. For validators transactions storage, 500 GB HDD storage is added to workers, so each worker has a total of 550 GB.

C. Cloud deployment model and benchmarking

Sawtooth network is deployed using Docker and Kubernetes resources. Kubernetes pods run the blockchain Sawtooth nodes. Each pod includes Sawtooth modules that run in different docker containers. As shown on Fig. 2, the pods run a validator container, a REST API container, a consensus engine container, and 10 *cartp* containers. *cartp* is the use case transaction processor written in Rust. 10 transaction processors are deployed to take advantage of parallel transaction execution. For each node, the validator is exposed to other nodes through a Kubernetes clusterIP service. Each test hardware resources configuration is described in Table I. The entire blockchain network, as described above, is deployed in a single Kubernetes namespace called *sim-sawtooth-net*. To expose the network to external client benchmark, only nodes' REST APIs are exposed through a single ClusterIP service. A Kubernetes Ingress is defined on this service. Thus client benchmark can access to the network using a domain name define on it.

Influxdb is used with *Grafana* to extract and display blockchain related metrics, this functionality is called under the namespace: *monitoring*. *Grafana* is exposed to external access through a Kubernetes clusterIP service, on which is defined an ingress.

The following figure (Fig. 3) summarizes the Sawtooth network deployment; the clients (i.e., simulating the IoT devices) are located in our laboratory on a local computing machine.

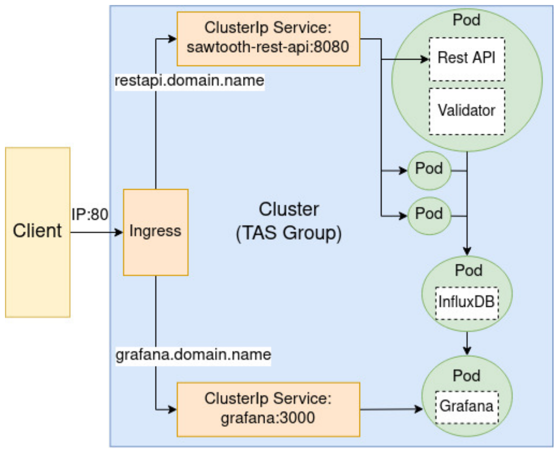

Fig. 3. Hyperledger Sawtooth network cloud deployment

500 vehicles are simulated and will send crash transactions in a round-robin way for the 500 car-owner identities. A crash transaction consists of sending an accident ID, a signature, and a data public key:

- Command: Action to execute in the transaction processor
- Accident ID: Data location ID. It is a 46 characters content addressing identifier.
- Signature: Signature (64 bytes) of the data sent (using the car private key).
- Data public key: Public key (32 bytes) associated to the private key that signed the data.

A benchmark was built using a mix of Bash, Python, Docker, and JavaScript. The cluster is controlled using the Rancher CLI that helps us to call Kubernetes commands (i.e. *kubectl* to delete all pods and start from fresh setup). Bash scripts encapsulate Rancher CLI calls and the Python script orchestrates the benchmark using subprocesses that call bash scripts and docker-compose (Fig. 4).

Each tests consists of initializing the 500 vehicles and then sending 10000 crash transactions at an input transaction rate of 5, 10, 15, 20, 30, 40 and 50 tps. The input transaction rate (*tps*) is defined by the number of transaction sent to the blockchain network per seconds. It is important to note that *tps* is **not equal** to the *committed transaction rate* (i.e. the number of transactions added and finalized in the blockchain). Tests are executed 3 times to minimize the cloud resource variance. Resources allocated to each test is described in Table I.

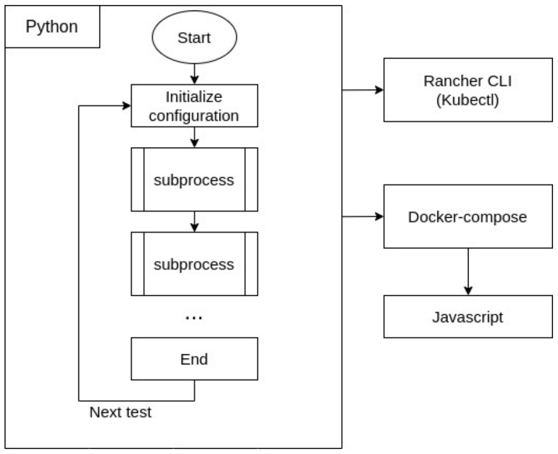

Fig. 4. Benchmark structure

TABLE I
RESOURCES ALLOCATED FOR EACH TEST

Env #	Blockchain Nodes	CPU/node (vCPU)	RAM/node (GB)	Disk/node (GB)
1	4	48	10	412.5
2	6	32	10	275
3	12	16	10	137.5
4	18	10	10	91.6
5	24	8	10	68.7

IV. Cloudification Results

A. *Feasibility and scalability*

The paper studies an industrial use case where multiple private companies own the blockchain. In a consortium scenario, Sawtooth allows all batch signers and transaction signers to submit batches and transactions. In this blockchain, the transactions are encapsulated into batches. It should be noted that in practice different private keys are used to sign the batch and the transaction. Also, it means that only specific nodes can join the validator network, and only specific nodes can participate in consensus. In the paper use case, all validators participate in the PBFT consensus.

A cloud architecture fits well for a consortium scenario. Any company that desires to start a consortium blockchain can easily add and remove validator nodes thanks to container orchestration. Moreover, if a partnership requires adding new validators, the network can change the Sawtooth settings (using *Settings transaction processor*) by adding new PBFT members.

During all the tests, the cloud resources limits (CPU and RAM) where never been reached. This means that increasing the cloud resources is useless when using Sawtooth blockchain.

In the following, we will observe that Sawtooth do not scale linearly. With the same number of nodes, increasing resources does not affect throughput performances. Also, PBFT consensus has already been studied to not scaling well due to exponential number of network communication. The following experimentation confirm the previous statement.

B. Performances

The paper analyzes Sawtooth performance following the test model in Sect. III-C. By varying the number of validator nodes and the input transaction rate, we evaluate Sawtooth's performance using the total committed transactions, the number of transactions committed per second, and the number of rejected transactions per second[1].

Hyperledger Sawtooth blockchain prevents DDoS attacks using the maximum batches-per-block as a limit. It helps to rate-limit the network as desired, but the paper's aims to achieve a high throughput by increasing this limit as high as possible (e.g. 1000 batches-per-block). In preliminary results, increasing the batches-per-block parameter shows a second limitation: the pending transaction queue size. In combination with a QUEUE_MULTIPLIER, the maximum batches-per-block also indirectly determine the maximum pending transaction possible in a node. Sawtooth version 1.2 defines the pending queue size limit as QUEUE_MULTIPLIER $\times$ Moving Avg batch rate. By default, QUEUE_MULTIPLIER is equal to 10, which results in a queue size of about 500-1k batches. Increasing QUEUE_MULTIPLIER to 200 leads to an average of 8-9k pending queue size. Expanding the pending queue size using the QUEUE_MULTIPLIER can affect the stability of the Sawtooth node. Indeed, the blockchain nodes totally fail by setting the maximum batch-per-block to 1000 and a QUEUE_MULTIPLIER higher than 100. This study takes this limitation into account. Thus, the maximum batches-per-block is fixed to 100 to prevent *sawtooth-core* from freezing because of process resource starvation.

It is important to note that this study aims to explore the maximum capabilities of the blockchain. Thus, all tests have a fixed a QUEUE_MULTIPLIER of 200 and a batch-per-block of 100.

1) Performance observation on the number of nodes: Increasing the number of nodes reduces performance. Starting from an input tps of 20, and scaling the number of nodes up to 24, reduces the total committed transactions, thus reducing commits per second and increasing rejects.

As depicted in Fig. 5, the total commits decreases for a network configured with 12, 18, and 24 nodes when input tps is higher than 15-20. For a network configured with 4 and 6 nodes, there are not rejects. Thus 100% of input transaction throughput is committed successfully.

[1]More graphical results are available on: https://github.com/projet-SIM/sawtooth-benchmark-coins-2021

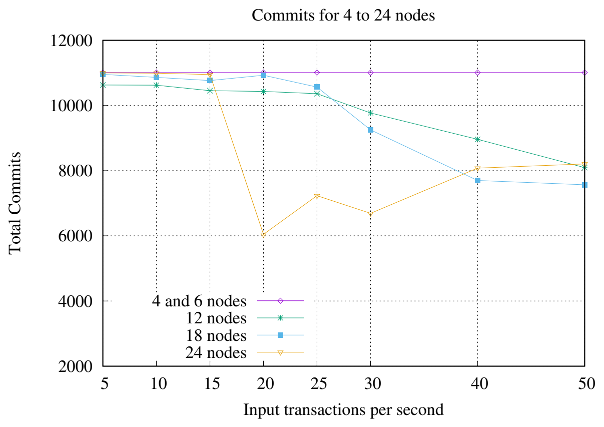

Fig. 5. Hyperledger Sawtooth for 4 to 24 nodes network performance

In each test, the pending transaction queue is being filled then emptied. It can take up to 15-20min to empty the pending transactions queue.

2) Performance observation on the transaction validation rate: Depicted in Fig. 6, we can observe the committed transactions per second. In the best situation we should have a linear variation of commit rate from input tps. However, the peak committed transaction rate is 25 when sending an input throughput of 50 tps on 4 nodes. We also notice that the general trend of the commit rate in each node configuration is stabilizing after an input transaction rate of 25. As previously explained, the reject rate is increasing for 12, 18 and 24 nodes configurations.

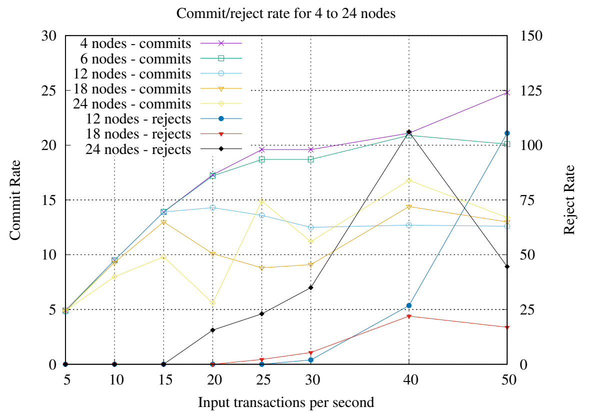

Fig. 6. Hyperledger Sawtooth commit rate for 4 to 24 nodes network performance

3) Performance observation on the transaction validation rate variance: Depicted in Fig. 7, the variance of the commit rate represents the transaction's squared deviation (for three tests) from its mean. In a 4 nodes configuration, the mean commit rate (in Fig. 6), using 50 input tps, is 25 commits per second with a variance (in Fig. 7) of 39 on 3 tests. We can observe that over all tests the variance of commit rate increase significantly for all input transaction rate higher than 20.

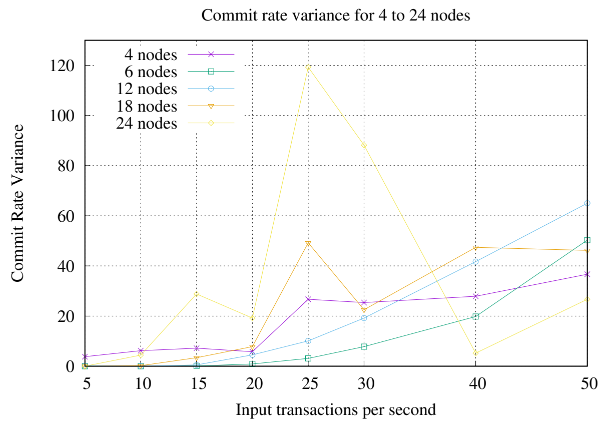

Fig. 7. Hyperledger Sawtooth commit rate variance for 4 to 24 nodes network performance

We can conclude from the performance study the minimum configuration to reach 25 validated transactions per second for Hyperledger Sawtooth using PBFT consensus. Using the 4 nodes configuration the resources defined are: 48 CPU, 10GB RAM, and 412.5 GB disk space.

V. Discussion and Conclusion

When increasing input throughput, we observed a transaction commit per second limitation. First of all, hardware monitoring shows that Hyperledger Sawtooth peers are not using all available resources. Secondly, transaction processor execution latency has been optimized in Rust language which provide one of the fastest execution latency. Also, the consensus algorithm can be modified, but after discussing it with the Hyperledger developers, the consensus configuration can not improve the transaction rate significantly. Finally, *sawtooth-core* can threshold the block production due to software limitations. In Sawtooth version 1.2 multiple core libraries are coded using only a single thread (Python). Thus, we can conclude that most of Sawtooth blockchain throughput limitation is caused by *sawtooth-core*. Version 2.0 of Sawtooth will be fully coded in Rust and thus, with multi-processing enabled, can provide faster transaction and block commit rate.

In this paper we have successfully deployed a Hyperledger Sawtooth blockchain network on cloud infrastructure. We achieved to set up a benchmark model with multiple hardware configurations. The number of input transactions is 9.3 times better than our previous results obtained in a local setup by Gerrits *et al.* [21]. We expose detailed data concerning the transaction threshold of 25 commits per second – without rejects - of Hyperledger Sawtooth on a 4 nodes network configuration. In addition, we confirm that using PBFT consensus Sawtooth version 1.2 can not have more than 12 nodes.

As a perspective, in future works we aim to compare Hyperledger Sawtooth performances with the Substrate blockchain framework and Ethereum blockchain using the same cloud infrastructure.

Acknowledgment

This work has been supported by the French government, through the UCA^{JEDI} and EUR DS4H Investments in the Future projects managed by the National Research Agency (ANR) with the reference number ANR-15-IDEX-0001 and ANR-17-EURE-0004.

References

[1] S. Nakamoto *et al.*, “Bitcoin: A peer-to-peer electronic cash system,” 2008.

[2] G. Wood *et al.*, “Ethereum: A secure decentralised generalised transaction ledger,” *Ethereum project yellow paper*, vol. 151, 2014.

[3] F. Longo, L. Nicoletti, A. Padovano, G. d’Atri, and M. Forte, “Blockchain-enabled supply chain: An experimental study,” *Computers and Industrial Engineering*, vol. 136, pp. 57–69, 2019.

[4] A. S. Musleh, G. Yao, and S. M. Muyeen, “Blockchain applications in smart grid–review and frameworks,” vol. 7, pp. 86 746–86 757, conference Name: IEEE Access.

[5] V. Sagar and P. Kaushik, “Ethereum 2.0 blockchain in healthcare and healthcare based internet-of-things devices,” in *International Conference on Paradigms of Computing, Communication and Data Sciences*, ser. Algorithms for Intelligent Systems. Springer, pp. 225–233.

[6] ONISR. Bilan 2019 de la sécurité routière. [Online]. Available: https://www.onisr.securite-routiere.gouv.fr/

[7] Intel Corporation. Hyperledger sawtooth. [Online]. Available: https://sawtooth.hyperledger.org/docs/core/releases/latest/

[8] M. Castro and B. Liskov, “Pratical byzantine fault tolence,” *Processing of the Third Symposium on Operating Systems Design and Implementation*, February 1999.

[9] C. Stathakopoulou, T. David, M. Pavlovic, and M. Vukolić, “Mir-BFT: High-Throughput Robust BFT for Decentralized Networks,” in *arXiv*, 2021.

[10] L. Chen, L. Xu, N. Shah, Z. Gao, Y. Lu, and W. Shi, “On Security Analysis of Proof-of-Elapsed-Time (PoET),” in *Stabilization, Safety, and Security of Distributed Systems*, P. Spirakis and P. Tsigas, Eds. Springer International Publishing, 2017, pp. 282–297.

[11] Z. Shi, H. Zhou, Y. Hu, S. Jayachander, C. de Laat, and Z. Zhao, “Operating permissioned blockchain in clouds: A performance study of hyperledger sawtooth,” in *2019 18th International Symposium on Parallel and Distributed Computing (ISPDC)*, 2019, pp. 50–57.

[12] Parity Technologies. Substrate. [Online]. Available: https://www.parity.io/substrate/

[13] N. D. Matsakis and F. S. Klock, “The rust language,” in *Proceedings of the 2014 ACM SIGAda Annual Conference on High Integrity Language Technology*. Association for Computing Machinery, 2014, p. 103–104.

[14] M. Bedford Taylor, “The evolution of bitcoin hardware,” *Computer*, vol. 50, no. 9, pp. 58–66, 2017.

[15] M. Gray. Ethereum blockchain as a service now on azure. [Online]. Available: https://azure.microsoft.com/en-us/blog/ethereum-blockchain-as-a-service-now-on-azure/

[16] AWS. Announcing general availability of amazon managed blockchain. [Online]. Available: https://aws.amazon.com/fr/about-aws/whats-new/2019/04/introducing-amazon-managed-blockchain/

[17] Blockchain Technology Partners. Product details. [Online]. Available: https://blockchaintp.com/sextant/sawtooth/

[18] Rancher Labs. Rancher documentation. [Online]. Available: https://rancher.com/docs/

[19] B. Ampel, M. Patton, and H. Chen, “Performance modeling of hyperledger sawtooth blockchain,” in *2019 IEEE International Conference on Intelligence and Security Informatics (ISI)*. IEEE, 2019, pp. 59–61.

[20] S. Benahmed, I. Pidikseev, R. Hussain, J. Lee, S. A. Kazmi, A. Oracevic, and F. Hussain, “A comparative analysis of distributed ledger technologies for smart contract development,” in *2019 IEEE 30th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*. IEEE, 2019, pp. 1–6.

[21] L. Gerrits, R. Kromes, and F. Verdier, “A true decentralized implementation based on iot and blockchain: a vehicle accident use case,” in *2020 International Conference on Omni-layer Intelligent Systems (COINS)*, 2020, pp. 1–6.