Ethereum Clique vs. HyperLedger Fabric: A Comparison of Blockchain Platforms

Alice Gong
Department of Computer and Software Engineering
Polytechnique Montreal
Montreal, Québec
alice.gong@polymtl.ca

Yuhan Li
Department of Computer and Software Engineering
Polytechnique Montreal
Montreal, Québec
yuhan.li@polymtl.ca

Yanis Toubal
Department of Computer and Software Engineering Polytechnique Montreal
Montreal, Québec
Polytechnique Montreal
yanis.toubal@polymtl.ca

***Abstract*— As Big Data keeps expanding, experts have been actively searching for the best way to store, organize and process the data. This is how Blockchain technology comes in with significant input, as they offer amazing functionalities for many fields like finance, healthcare, manufacturing, and government, just to name a few. Many platforms that use blockchain technology are available these days to respond to all these needs. However, since this technology is relatively new, it is sometimes unclear which blockchain development platform should be used for a particular context. To enlighten this uncertainty, this paper offers a comparative analysis on two major platforms: Hyperledger Fabric and Ethereum.**

Keywords— Ethereum, Hyperledger Fabric, Consensus protocol, Blockchain, Performance

MB: Megabytes.

MS: Millisecond.

NFT: Piece of data proving ownership of a digital item. Short for non-fungible token.

P2P: Platform where two entities can communicate directly which each other without going through an intermediary.

RAFT: Crash fault-tolerant algorithm used in Hyperledger Fabric.

TPS: Transaction per second.

I. Introduction

Blockchain is a technology that serves as a public record of data that ensures to be immutable and secure. The first-ever use case of Blockchain was for the creation of a decentralized cryptocurrency from which Bitcoin was created [1]. The motivation behind it was to create a way to store information in a secure and decentralized way, which in return creates a trustworthy environment for participants without the need of a third party [2].

To achieve its immutability and high security, Blockchain stores data in a decentralized and distributed network while using a sequential approach. In the financial universe of Blockchain, these communicated data are known as *transactions*. The transactions are stored in a sequence of different groups known as *blocks*, which are all unique with a pair of hash and timestamp. Whenever one block is full, it is closed and linked to the previous block that filled up. This storing method is depicted in Fig.1. However, before officially storing a transaction in a new block, there are two validation stages: transaction validation and block validation. Transaction validation can be made with a validation algorithm, for instance, public-key cryptography. Block validation can be made by using a consensus algorithm, such as the Proof-of-Work, where the majority of nodes agree on a decision [3].

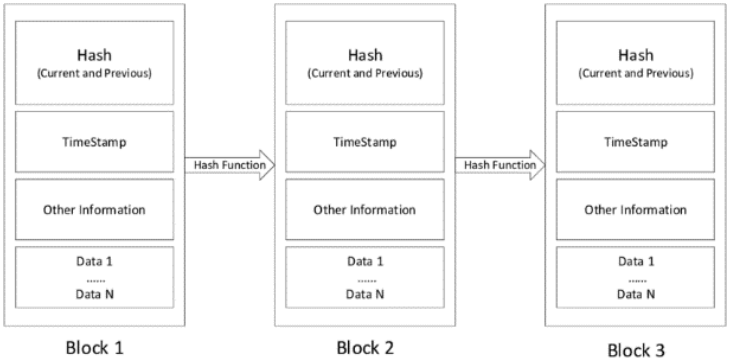

Fig 1. Blockchain General Structure [4].

Because of this structure, Blockchain represents a trusted intermediary for a transactional system. We would state its main functionalities as [5]:

- Transaction processing: To reiterate what we described above, blockchain is a linked list of immutable and chronological tamper-proof blocks. This sequence of blocks is stored at each participating node, and each block records a set of transactions with the associated metadata.
- Point-to-Point transmission: Nodes in a blockchain system can complete point-to-point transactions, or more specifically known as peer-to-peer (P2P). This makes it a decentralized network, which solves the problems of high trust, low efficiency, and insecure data storage in a centralized system [6].
- Data sharing: As aforementioned, blockchain lets nodes share information on a P2P basis. This exchange takes place from one node to another through files containing transfer information, which are generated by a source node and then broadcast to the entire network for validation [3]. It allows functionalities like automatic data records or decentralized management.

- Data protection: As a result of blockchain's decentralized methodology, the network is secured by a consensus protocol and cryptographic key, which abolish the involvement of a third party. Users of blockchain technology avoid dependence on third parties for the security of all transactions and assets [3].
- Data ownership: As transaction verification is done cryptographically without an explicit need of a third party, data ownership can be put back into the user's hands. This allows users to store or directly exchange the assets they own (ex.: Bitcoin digital currency). Moreover, blockchain transactions are not limited to the financial sector, they can also represent physical or digital properties, smart contracts between different parties, or any other data and document (ex.: NFTs) [7].

With these advanced features, blockchain technology grew in demand these past few years. Many blockchain platforms are now available such as Hyperledger Fabric, Ethereum, Corda, etc. In this study, we focus on analyzing the architecture and configurations of two blockchain implementations, namely Ethereum Clique and Hyperledger Fabric. The ultimate objective is to evaluate their performances based on the comparison of their properties and behavior. To analyze their behavior, we collected multiple data for both platforms using a benchmarking tool called BlockCompass. Therefore, this research paper also offers a thorough understanding of BlockCompass, a rather new technology that deserves to be explored further.

The remainder of this paper is organized as follows: in Section II, we give a general background on Hyperledger Fabric and Ethereum by presenting their main characteristics and features relevant to our experience. As for the experiment itself, the methodology and results are presented in Section III. Following this, we give a more detailed analysis of the results and experiment in Section IV. Finally, the paper is summarized in Section V, where we draw our conclusions and extend our reasoning.

II. CONTEXT – BACKGROUND

A. *Hyperledger Fabric*

Hyperledger Fabric is an open-source Blockchain platform with the main objective of satisfying business use cases [1]. This technology is under the Linux Foundation which is very reputable in the open-source community [8]. This Blockchain platform has important characteristics that are: the usage of a crash-fault tolerant consensus protocol, high modularity and configurability, support for smart contracts in general-purpose programming languages, and a permissioned platform.

Hyperledger Fabric uses a crash-fault tolerant consensus protocol which is based on RAFT. BFT can also be implemented. This type of consensus protocol builds resilience in the system by making sure the algorithm is capable of reaching a consensus even when the system isn't 100% operational for example when a Node ceases to work. This protocol can reach a consensus as long as half of nodes + 1 which is the majority of nodes can agree with one another. However, this type of protocol isn't effective against malicious activities which makes the blockchain more vulnerable to threats. This consensus protocol is only viable in a permissioned setting since the users of the blockchain are usually known so a perpetrator can be identified if he does malicious activities. The great advantage of this protocol is the performance benefits since a higher number of throughput and lower latency can be achieved [9].

Hyperledger Fabric is highly modular and configurable. The architecture's goal was to be very modular. Here are the different modules that compose the blockchain. First, we have a pluggable ordering service that orders the transactions and broadcast blocks. Secondly, we have a pluggable membership service provider which associates the network entities with cryptographic identities. Thirdly, we have smart contracts that are run within a container. Fourthly, we have a ledger that can be configured to support many DBMSs. Finally, we have an endorsement and validation policy enforcement that can be specified for each application. All those characteristics allow Hyperledger Fabric to be very versatile and adaptable to the use case of businesses [8].

Hyperledger Fabric supports smart contracts, which are called chaincode in Fabric, written in general-purpose programming languages such as Java, Go. It contains the logic of a blockchain application. There are 3 important characteristics to note in general:

1. They run in concurrency;
2. They can be deployed dynamically in most cases by anyone in the network;
3. Application code shouldn't be trusted and it could be malicious.

Hyperledger Fabric uses a new approach called the execute-order-validate which works in 3 steps:

1. Execute a transaction and verify it;
2. Order the transactions via a pluggable consensus protocol;
3. Validate the transactions using the endorsement policy of the application.

This approach increases the scale and the performance of the system. The support of general-purpose programming languages makes it easier for developers to focus on the logic of the smart contracts rather than learning a new language [8].

Hyperledger Fabric is permissioned and doesn't require a cryptocurrency, as opposed to public blockchain which is permissionless and that usually requires a cryptocurrency as an economic incentive for the anonymous participant. Hyperledger Fabric is permissioned which means that the participants are known in the blockchain and that they obey a governance model which results in a level of trust in the platform. However, this isn't a full degree of trust, and, taking this into account, the system is designed in such a way that the transactions are secured to satisfy parties with a common goal. Since the participants in the blockchain aren't anonymous, a cryptocurrency isn't necessarily needed as an incentive and it's possible to use a crash-fault tolerant protocol or even a

byzantine-fault tolerant protocol without expensive mining. The risk of participants taking part in malicious activity is also reduced since it would be easy to identify the actor of the attack in such a blockchain and the governance model would dictate what would be the next course of action against the attacker [8].

B. *Ethereum*

Ethereum is an open-sourced permissionless blockchain platform whose purpose is to facilitate the development and deployment of decentralized applications, otherwise known as dApps. These applications consist of a graphical interface and of smart contracts written in Solidity that renders its logic. Essentially, smart contracts are immutable programs that are executed autonomously by a series of pre-established conditions, thus allowing decentralization.

Ethereum possesses its native cryptocurrency called Ether, which acts as a store of value, and is used as a means of payment within the platform [10]. However, Ether also serves as fuel to maintain the Ethereum network, as it is used to incentivize miners to keep it growing. Typically, Ethereum's network is configured with the Proof-of-Work consensus protocol (PoW), during which miners compete to solve a difficult puzzle. New blocks are validated and secured in that way, and miners are rewarded in Ether. Since this mechanism requires notable computation power, it also offers security and integrity; an individual wanting to alter with data would be faced with a daunting and impractical task because of the amount of processing power involved.

For our work, we used Ethereum Clique, which is a testnet of Ethereum. Contrarily to the main network, Ethereum Clique uses a Proof-of-Authority consensus algorithm. This mechanism requires much less computational power as it consists in trusting a small number of block producers to achieve consensus, rather than requiring nodes to compete amongst themselves to solve complex problems. PoA is a new family of Byzantine fault-tolerant (BFT) consensus algorithm. This type of consensus protocol builds resilience in the system by making sure the algorithm can reach a consensus even when the system isn't 100% operational and when malicious activities are happening. This protocol provides high availability and security but is very slow causing it to create a low throughput and a high latency [9].

In brief, Hyperledger Fabric is a permissioned blockchain platform, whereas Ethereum is permissionless. However, Ethereum Clique is a private and permissioned blockchain platform. Ethereum is mainly used to develop and host decentralized apps and Hyperledger Fabric, for business use cases. Furthermore, as explained previously, Ethereum possesses its own cryptocurrency called Ether, as opposed to Hyperledger Fabric, which does not need a mandatory currency to work. Finally, Hyperledger uses a CFT consensus protocol, such as Raft, and Ethereum uses PoW. Ethereum Clique, on the other hand, Ethereum Clique uses PoA.

III. EXPERIMENTS

The goal of the experiments is to compare Hyperledger Fabric and Ethereum Clique's while handling multiple varying workloads. In this section, we present the benchmark tool, as well as the setup of our experiments, more precisely the deployment infrastructure, and finally, the evaluated workloads and metrics.

A. *Benchmark Tool Architecture*

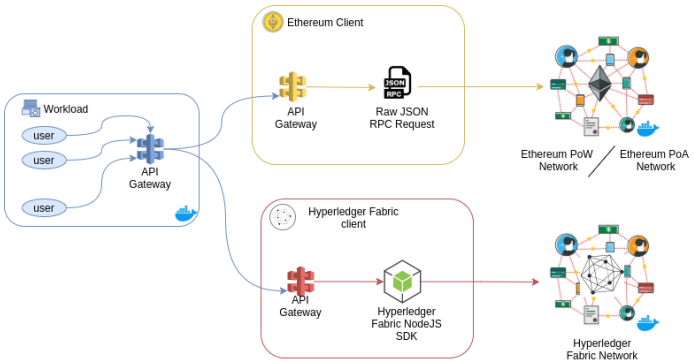

Fig 2. BlockCompass Architecture [12].

As aforementioned, the benchmarking tool selected for this study is BlockCompass. An advantage of using BlockCompass is that it provides real-time charts illustrating the different performance indicators that it supports as of now, such as latency, transmission rate, throughput, as well as resource consumption [11].
This tool is composed of many modules, such as a workload component, an Ethereum client, a Hyperledger Fabric client, a resource monitor, a MongoDB replace set, a back-end server, and a front-end client.

1) *Workload component*

Its responsibility is to simulate concurrent user requests. The users are independent of one another, and a new request is only sent after receiving a result from the previous request. The workload simulates four types of users [12]:

- A temperature sensor which updates every second;
- A device sensor which reports on its state (ON/OFF) every five seconds;
- A GPS sensor sharing its location every two seconds;
- A sound sensor reporting sound data every 24 seconds.

Another responsibility of this module is to monitor the latency, transmission rate, throughput, and error rate which are stored in the database [11].

2) *Ethereum Client*

Its responsibility is to adapt the communications between the user requests (sent by the workload) and the Ethereum network [11].

3) *Hyperledger Fabric Client*

Similarly, this client's responsibility is to adapt the communications between the user requests (sent by the workload) and the Hyperledger Fabric network [11].

4) Resource Monitor

As the name suggests, the resource monitor monitors the Blockchain network to collect the CPU usage, the memory usage, and the network input/output consumption [11].

5) MongoDB Replica Set

This replica set contains different collections, storing all the data relative to the workload and the resource monitor. It also offers a feature allowing access to real-time data updates, which is used to create real-time charts [11].

6) Back-end Server

This module consists of an Express server that provides many endpoints, allowing communication between the front-end client with itself, and thus the database as well [11].

7) Front-end Client

The front-end client is built with Angular 10 and displays the previously mentioned real-time charts [11]. Besides that, this interface also provides information on each metrics' average, standard deviation, variance, etc. in the form of tables.

B. Setup of BlockCompass Benchmark Tool

To set up this BlockCompass, the network configuration file, which defines multiple factors allowing the proper functioning of the tool, was configured to set the target Blockchain platform and the number of nodes present in the selected network. For our study, we specified a number of five nodes. Moreover, for each experiment, we set different data sizes, which represent the payload size carried by every transaction. The initial data size varies between 100 and 150. We simply multiplied this data size by 2, 5, and 8 to create three different-sized workloads [11].

C. Deployment Infrastructure

For this study, we conducted our experiments on an Amazon EC2 instance running Ubuntu. We configured it to be a t2.large virtual machine with 2 virtual CPUs and 8GB RAM. Moreover, we initially configured it to have 25GB of disk space, however, while we ran our experiments, we encountered errors regarding insufficient space, so in the end, we increased the storage to 50.

For Hyperledger Fabric, we used the release version 2.3.2. As for Ethereum, we used Go Ethereum (Geth) version 1.10.3, a Golang implementation of Ethereum blockchain in the form of standalone client software. As mentioned previously, each network was configured to be composed of five nodes.

As for the deployment, both Hyperledger Fabric and Ethereum's deployments were installed on an Ubuntu operating system, with each network running inside a docker container.

D. Evaluation Metrics

For this study, the impact of the workloads on the blockchain platforms was of interest. To compare them, we used two types of metrics, such as resource consumption and performance metrics. Resource consumption metrics quantify resource usage. Below are the following resource consumption metrics observed in this study:

- CPU: It measures the CPU usage;
- Memory: It measures the memory usage;
- Input/Output: It represents the total transferred input and output data;
- Users: It represents the number of users present in the workload.

As for performance metrics, these are metrics that evaluates the performance of a blockchain:

- Emit rate: This metric represents the total number of transactions per second communicated on average to the Ethereum or Hyperledger Fabric network.
- Throughput: It measures the number of transactions that successfully completed their action.
- Latency: It represents the time, in milliseconds, it takes to get a response from one transaction
- Error rate: This rate represents the failed transaction, or more specifically the ones for which no confirmation has been received within a certain time. This failure could be caused by the loss of transactions due to network congestion, node synchronization problem, or overflow in the memory pool of the node [13].

IV. EXPERIMENTAL RESULTS

After running the BlockCompass workload for each platform and size, we obtained many metric results in the form of tables and graphs. After reading into the tables, we separated our data into the types of evaluated metrics, such as resource consumption metrics and performance metrics. Moreover, for easy understanding and analysis, we grouped the tables into four main ones, as presented right below.

TABLE I. AVERAGE VALUES FOR ETHEREUM CLIQUE'S RESOURCE CONSUMPTION METRICS

Metrics	**Data Size**		
	2	***5***	***8***
CPU (%)	12.83	12.596	7.448
Memory (%)	4.062	4.292	4.193
Input/Output (MB)	993.472	759.209	533.12
Users (number)	64.393	61.158	60.667

TABLE II. AVERAGE VALUES FOR HYPERLEDGER FABRIC'S RESOURCE CONSUMPTION METRICS

Metrics	Data Size		
	2	***5***	***8***
CPU (%)	5.777	5.407	3.657
Memory (%)	0.67	0.675	0.663
Input/Output (MB)	1243.795	951.608	420.211
Users (number)	63.627	62.727	61.959

TABLE III. AVERAGE VALUES FOR ETHEREUM CLIQUE'S PERFORMANCE METRICS

Metrics	Data Size		
	2	***5***	***8***
Emit rate (tps)	404.682	337.368	162.3
Throughput (tps)	404.682	337.368	148.35
Latency (ms)	1361.287	1730.587	4035.823
Error rate	0	0	0.064

TABLE IV. AVERAGE VALUES FOR HYPERLEDGER FABRIC'S PERFORMANCE METRICS

Metrics	Data Size		
	2	***5***	***8***
Emit rate (tps)	738.657	582.707	276.588
Throughput (tps)	738.657	582.707	276.588
Latency (ms)	186.349	318.834	1517.271
Error rate	0	0	0

Table 1 and Table 2 represent the resource consumption metrics for, respectively, Ethereum Clique and Hyperledger Fabric. We notice a pattern for the CPU, Input/Output, and users metrics, as they all seem to decrease when the data size grows.

From Table 3 and Table 4, it can be confirmed that the error rate has an impact on the throughput values. Indeed, all the throughout values are identical to the emit rate values, except for the largest data size in Table 3 for Ethereum Clique. This can be explained by the fact that this workload is the only one without a null error rate, which means that it's successful transactions (throughput) is lower than the emitted ones (emit rate). Another important observation to mention would be that the throughput and latency metrics are inversely proportional. As the amount of emitted and successful transactions per second are decreasing following the workload's growth, we notice that the latency is increasing. This trend is coherent since we know that when the workload becomes heavier, it consumes fewer resources and performs less efficiently.

V. DISCUSSION

A. *Performance Analysis*

Hyperledger Fabric uses a smaller amount of **memory** and **CPU** compared to Ethereum, which is the case for all the workloads as we can see from Figures 3 and 4. When looking at the CPU comparison, Hyperledger Fabric only slightly decreases when the workloads get bigger. For Ethereum Clique, It has a bigger variation as represented by the steep slope. As for the memory, both platforms are fairly constant, which is represented in Figure 4 by the almost horizontal lines, across the workloads.

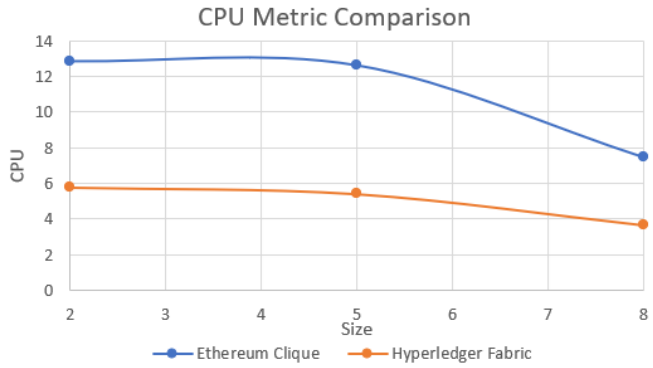

Fig 3. CPU Metric Comparison between Ethereum Clique and Hyperledger Fabric.

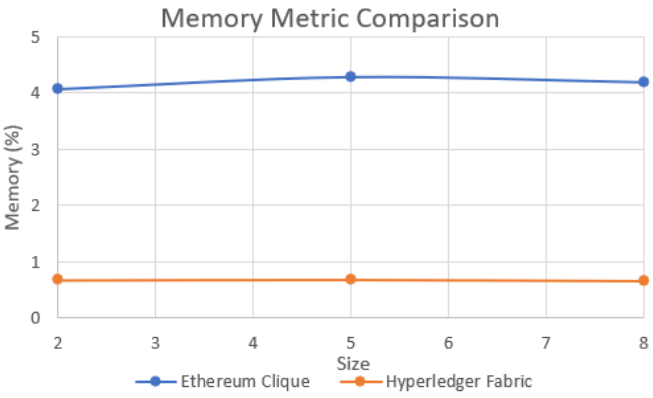

Fig 4. Memory Metric Comparison between Ethereum Clique and Hyperledger Fabric.

This trend can be explained by the different consensus protocols used by each platform. As previously mentioned, Hyperledger Fabric uses Raft as a crash-fault-tolerant consensus protocol. This protocol can be considered very performant since it does not take into account the security aspect. In the case of Ethereum Clique, it uses the Proof-of-Authority consensus protocol, which is faster than the Proof-of-Work protocol on Ethereum's main network, being that it consists in trusting a small number of block producers to achieve consensus, rather than requiring nodes to compete amongst themselves to solve complex problems.

As for the network **input/output**, we can observe, in Figure 5, that Hyperledger Fabric has a higher network usage than

Ethereum Clique on average, meaning that the Hyperledger Fabric network receives and sends more data than the network of Ethereum Clique. It can also be observed that these trendlines seem to be reversing when the workload gets large.

The first observation can be explained by the fact that the Fabric architecture necessitates more network communication between its nodes to handle transactions. As for the second observation, many reasons could explain this result. One reason would be the non-null error rate as observed in Table 3 which caused more network communication. Another reason would be that the Ethereum network has more trouble when dealing with a higher workload as we are about to see with the throughput and latency.

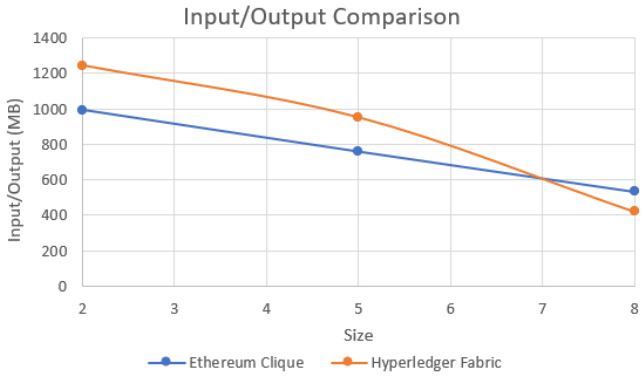

Fig 5. Network Input/Output Metric Comparison between Ethereum Clique and Hyperledger Fabric.

We can observe that Hyperledger Fabric has a higher value in terms of throughput and a smaller value in terms of latency than Ethereum across the same size workloads. As for different size workloads, we can note that as the size of the workload increases, the transactions take a longer time to be processed, which results in lower throughput and higher latency.

Once again, the consensus algorithm can also explain why Hyperledger Fabric transaction requests are faster than Ethereum Clique, as observed in the results of the throughput and latency. Overall, Hyperledger Fabric has better results for write latency and throughput.

Another interesting thing to mention is that the latency, throughput, and error rate metrics also represent the scalability of the system. As stated earlier, the latency increases as the number of transactions grows for both platforms. However, as observed in Figures 6 and 7, the latency growth is much faster for Ethereum Clique than for Hyperledger Fabric, meaning that Fabric offers better scalability. On the other hand, the error rate for Clique seems to be higher since it had an average rate of 0.064 for the large dataset, which makes Fabric more fault-tolerant.

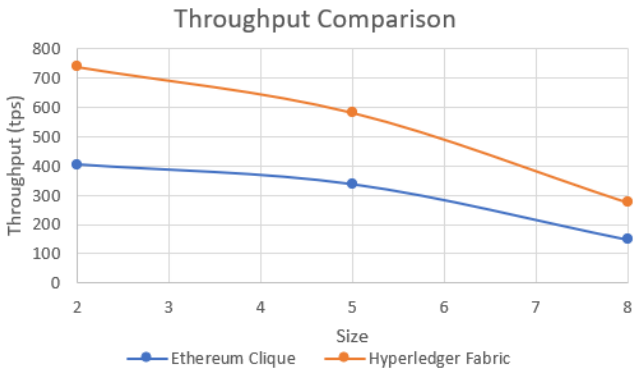

Fig 6. Throughput Metric Comparison between Ethereum Clique and Hyperledger Fabric

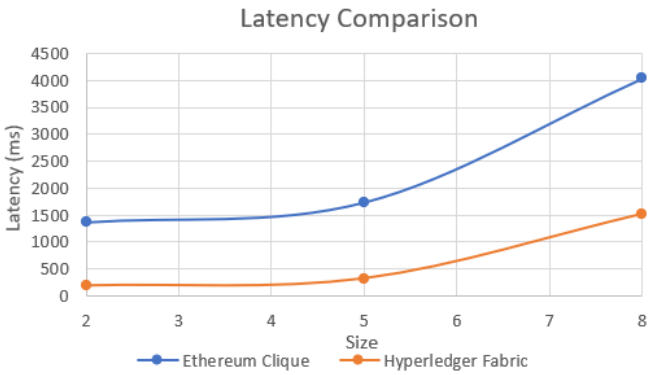

Fig 7. Latency Metric Comparison between Ethereum Clique and Hyperledger Fabric.

Based on the metrics seen above, one might think that Hyperledger Fabric would always be the default choice when choosing a blockchain platform. However, one should also consider the characteristics of each platform before making a choice. In that sense, we would recommend Hyperledger Fabric in the case of a highly configurable blockchain especially for a business need, when there is an emphasis for performance namely a high throughput and low latency, when there is a low risk of blockchains attack, and when there is no need to use a cryptocurrency. As for Ethereum, we would recommend it when there is a need to host a decentralized app, where security against malicious activities is very important, when there is a need for decentralization and a permissionless platform and when there is a need to use a cryptocurrency (Ether). Another thing to note is that, as mentioned in the description of both platforms, Smart contracts are written in Solidity in Ethereum, and they are written in general-purpose programming languages for Hyperledger Fabric. Even though this fact alone isn't major when choosing a platform, It still is a great advantage since, in theory, It would be easier to write smart contracts on Fabric.

B. Threats Validity

Since the tool we used for this experiment, namely BlockCompass, was relatively new, the accuracy of the results

might be a subject of discussion. However, to make sure that the data we got was as precise as possible, we repeated the experiment several times, at different points in time, to make sure that the results we reported were as accurate as possible. In the end, since we got relatively the same results for every experiment we did and by using our theoretical knowledge of these blockchains, we concluded that the data we got from the benchmarking tool was good.

Moreover, the setup of the experiment wasn't an accurate representation of what we would do in real-world usage. Indeed, as explained earlier in the experiment setup, the experiment was done on a single EC2 machine in a controlled environment. Such a system would be in a distributed environment which would imply many additional factors that could influence the results. The origin of the workload requests, the location of the servers, and the state of the network are examples of such factors.

As for the workloads we used a static setup. Indeed, as described earlier, we used the same number of nodes (5) and the same number of users. As for the workloads, we considered that a small workload would be a multiple of 2, a medium a multiple of 5, and a high multiple of 8. Although these numbers make a lot of sense and we got some great results using them, we didn't try to go further and see how the platforms would behave under heavier loads. By only testing with these 3 specific workloads, we might be missing a greater picture and only "zooming" on a particular zone. Following the same line of thought, since we only changed the size of the workloads (by changing the multiple), the data we collected is fairly limited since we only touched one parameter.

VI. Conclusion

By analyzing and testing Ethereum Clique and Hyperledger Fabric, this study showed that, overall, the performance of Hyperledger Fabric was greater except for the network input/output where Ethereum Clique did better. We also concluded that those observations could be explained by the intrinsic differences between the two blockchains especially the consensus algorithm. Moreover, we mentioned the situations where Hyperledger Fabric would be more advantageous to use as well as the situations where Ethereum would be more advantageous. As for future studies, it would be interesting to try out the same experiment with different benchmark tools such as BlockBench and Caliper and compare the results. Also, another interesting experiment would be to try out different configurations of workloads by changing many parameters.

References

[1] M. Fokaefs, Class Lecture, Topic: “Blockchain.” LOG8430, Department of Computer and Software Engineering, Polytechnique Montreal, Montreal, Nov. 2021.

[2] A. Hayes, “Blockchain Explained,” *Investopedia*, Nov 4th, 2021. Accessed on: Dec. 12th, 2021. [Online]. Available: https://www.investopedia.com/terms/b/blockchain.asp

[3] D. Puthal, N. Malik, S. P. Mohanty, E. Kougianos and G. Das, “Everything You Wanted to Know About the Blockchain: Its Promise, Components, Processes, and Problems” in IEEE Consumer Electronics Magazine, vol. 7, no. 4, pp. 6-14, July 2018, doi: 10.1109/MCE.2018.2816299.

[4] “A survey on security and privacy issues of blockchain technology” - Scientific Figure on ResearchGate. Accessed on: Dec 12th, 2021. Available from: https://www.researchgate.net/figure/BlockchainStructure_fig1_325173502

[5] Ali, A. Jaradat, A. Kulakli and A. Abuhalimeh, “A Comparative Study: Blockchain Technology Utilization Benefits, Challenges and Functionalities,” in IEEE Access, vol. 9, pp. 12730-12749, 2021, doi: 10.1109/ACCESS.2021.3050241.

[6] X. Li, Y. Mei, J. Gong, F. Xiang and Z. Sun, “A Blockchain Privacy Protection Scheme Based on Ring Signature” in IEEE Access, vol. 8, pp. 76765-76772, 2020, doi: 10.1109/ACCESS.2020.2987831.

[7] E. Karafiloski and A. Mishev, “Blockchain solutions for big data challenges: A literature review” IEEE EUROCON 2017 -17th International Conference on Smart Technologies, 2017, pp. 763-768, doi: 10.1109/EUROCON.2017.8011213.

[8] Hyperledger Fabric, “Introduction”, 2020. Accessed on: Dec 14th, 2021. [Online]. Available: https://hyperledgerfabric.readthedocs.io/en/latest/whatis.html

[9] “How the consensus protocol impacts blockchain throughput,” *NEC*, 22n.d.. Accesed on Dec 12th, 2021. [Online]. Available: https://www.nec.com/en/global/insights/article/2020022520/index.html#:~:text=Crash%20fault%20tolerance%20(CFT)%20builds,component%20of%20the%20system%20fails

[10] Ethereum, “What is Ether (ETH)?”, 2021. Accesed on: Dec 14th, 2021 [Online]. Available: https://ethereum.org/en/eth/

[11] M. Fokaefs, M. Rasolroveicy, and W. Haouari. (2021). “Public or Private? A Techno-Economic Analysis of Blockchain,” in 31st Annual International CASCON, Toronto, Ontario, Canada, 2021, pp. 3

[12] W. Haouri. 2021. BlockCompass User Manual. Montreal, Canada. Accesed on: Dec 14th, 2021 [PDF]. Available: https://drive.google.com/file/d/1F-loFQBfcEHoDu5qUpOw9YMRCGCrvXr/view

[13] J. Abdella, Z. Tari, A. Anwar, A. Mahmood and F. Han, "An Architecture and Performance Evaluation of Blockchain-Based Peer-to-Peer Energy Trading," in IEEE Transactions on Smart Grid, vol. 12, no. 4, pp. 3364-3378, July 2021, doi: 10.1109/TSG.2021.3056147.