Energy trading marketplace using Ethereum private network

Dongmin Son, Sawsan Al Zahr, Gerard Memmi

Energy trading marketplace using Ethereum private network

Dongmin Son, Sawsan Al Zahr, Gerard Memmi
LTCI, Télécom Paris, Institut Polytechnique de Paris
19 Place Marguerite Perey 91120 Palaiseau, France
Email: {dongmin.son,sawsan.alzahr,gerard.memmi}@telecom-paris.fr

***Abstract*—In this paper, we evaluate and analyze the performance of a local electricity market for energy trading that we implemented on the Ethereum platform. The energy trading is based on a double auction with multiple sellers and multiple buyers, and the matched price and volume are determined by a trade reduction mechanism. We benchmark the performance of Ethereum using a systematic blockchain performance evaluation method, and based on this, we propose and analyze an efficient market operation method. In particular, we relate the limits on the scalability and real-time performance of the market to the throughput and latency of the Ethereum platform. We also identify the minimum resources necessary to operate an Ethereum client.**

***Index Terms*—Local energy market, Blockchain, Performance analysis, Ethereum, Proof-of-Work**

I. Introduction

Residential solar panels and electric vehicles are being deployed around the world to support the use of green energy. The distributed electricity supply by solar panels is having a negative impact on the centralized power supply. At the same time, it is becoming difficult to meet the demand for electric vehicles with only a centralized power source. In order to balance such a distributed supply and increased demand, the local energy market can improve this situation.

In the local energy market, there have been a lot of attempts to use the blockchain due to the characteristic of trading distributed energy between distributed individuals. 46 companies and projects around the world have tried decentralized energy trading using blockchain and 17 of which are based on Ethereum [1]. LO3 Energy may very well be one of the most representative Enterprise, they developed the Brooklyn MicroGrid project [2]. They completed a pilot of a peer-to-peer energy transaction between prosumers and consumers with residential solar panels in Brooklyn, New York. Since then, they have developed local energy markets on microgrids in Germany, Australia, Texas, and the UK. They used smart contracts based on Ethereum, but the consensus algorithm was replaced by Tendermint. Power Ledger [3] is also based on Ethereum, but uses Proof-of-Authority (PoA) consensus. They have demonstrated energy trading platforms in Australia, New Zealand, and Japan.

In this paper, we analyze the performance and cost of implementing an energy market using the Ethereum private network. We focus on the effect of input rate, network size, and block gas limit on the performance such as throughput and latency. We also show that those parameters can affect blockchain operating costs such as CPU, memory, storage usage, and network usage.

The rest of this paper is organized as follows. Section II reviews the related works for evaluating blockchain performance and the blockchain-based energy market. Section III provides an overview of the blockchain platforms. The energy market implementation is explained in Section IV. The test environment is described in Section V and we discuss the performance evaluation results in Section VI. The conclusion is summarized in Section VII.

II. Related Works

As interest in blockchain increased and several blockchain platforms appeared, the benchmark tool is designed for the multiple blockchain platforms such as Ethereum, Parity, and Hyperledger [4]. The performance was evaluated in terms of throughput, latency, and scalability by changing the workload type, input rate, and number of nodes. As the workload, database-like applications such as Yahoo Cloud Serving Benchmark (YCSB) [5], Online Transaction Processing (OLTP), and simple Ethereum applications are included, but energy market application was not considered. The peak performance of Ethereum is measured as the throughput of 284 Transaction Per Second (TPS) and the latency of 92 seconds in the YCSB workload.

The blockchain can reduce market operating costs by eliminating intermediaries and allows small consumers to participate in the energy market. The performance of the energy market depends on the scalability and speed that the blockchain platform can support. [6] uses Ethermint to implement the energy market and measure the performance by bid transactions. When the number of nodes is small, throughput increases with the input traffic and the latency is constant, around 1 second. However, as the number of nodes increases, the throughput decreases, and latency increases. When the number of nodes increases to 28, the throughput decreases to 1 TPS, and the latency increases to more than 2 minutes.

Similarly, [7] and [8] designed a local energy market on Ethereum. The former research used the Proof-of-Authority (PoA) based Ethereum private network, and later used the

Proof-of-Work (PoW) based Ropsten test network. Both pieces of research measured gas cost, not throughput and latency. [7] implements the average mechanism of the double auction as a clearing mechanism and consumes 24 million gas to clear 1,000 bids. [8] also used a slightly modified version of the average mechanism. However, since there were only 9 market participants, the gas cost can be measured only for 9 bids, and 2,792,870 gas was consumed to clearing the 9 bids. In both results, the clearing function is a fairly complex contract considering that the gas required to transfer Ether is 21,000 and the block gas limit of the Ethereum public network is 12 million gas as of 2020. The average mechanism is simple but not strategy-proof. In our study, we use a trade reduction mechanism, which is a more complex but strategy-proof clearing method to ensure truthful participants to win regardless of the bidding strategy.

III. Blockchain Platform

Ethereum [9] is the first smart contract platform designed to overcome Bitcoin's Turing incompleteness. Ethereum supports the notion of smart contracts as well as data in blocks. A smart contract is described in the Solidity specific programming language and executed in Ethereum Virtual Machine (EVM).

Each time the smart contract is executed, a fee called gas is required, and the gas is limited to avoid infinite loops. The gas is the basic unit of work to execute a smart contract, and all transactions in Ethereum must include gas limit and gas price. The gas limit is an estimated maximum workload and the gas price is the price for the unit operation.

Miners of the Ethereum network prioritize transaction validation along with gas prices and ensure that the gas limits of the requested transaction do not exceed the block gas limit. The mining process, named Ethash, is designed to be ASIC-resistant via the memory-hardness process of writing a Directed Acyclic Graph (DAG). Creating the DAG takes about 10 minutes and requires more than 1 GB of RAM. In the meantime, Ethereum is changing the consensus algorithm from the current PoW to PoS named Casper.

Another implementation of the PoS consensus algorithm is Ethermint, which implements Ethereum on Tendermint [10]. Tendermint is a blockchain consensus algorithm of Cosmos, which was born in 2014 to solve the problems of speed, scalability, and excessive energy consumption of the PoW consensus algorithm. Tendermint is a Byzantine fault tolerance (BFT) based protocol in the PoS consensus mechanism.

Unlike a public blockchain such as Bitcoin and Ethereum, Hyperledger Fabric [11] is a permissioned blockchain that can be accessed only by authorized organizations. It is possible to construct a faster and more scalable network than the public blockchain. Hyperledger Fabric selects Kafka which is not strictly a blockchain consensus algorithm but a distributed messaging system developed by LinkedIn. Kafka utilizes the Certificate Authority (CA) and endorsement policy for Crash Fault Tolerance (CFT).

IV. Energy Market Implementation

The market is based on a discrete-time Multi-unit Double Auction (MDA). In general, the double auction has several mechanisms to determine the trading price. The average mechanism sets the trading price as the average of the lowest bid and highest ask price at the clearing point where the aggregated volumes are met. This mechanism is not strategy-proof because the last buyer and seller can have an incentive to offer a lower or higher price. In the VCG mechanism, the lowest ask price at the critical point is taken for buyers and the highest bid price is taken for sellers. This mechanism is strategy-proof because changing prices doesn't affect their own trading price. However, it causes the budget balance problem because the seller's price is higher than the buyer's price so someone has to pay a subsidiary for the gap between the matched bid price and matched ask price. To solve this problem, the trade reduction mechanism is used in [12]. In this mechanism, the matched price is determined as the lowest bid price and highest ask price at the clearing point, but bidders at those prices are excluded from the matching result for strategy-proofness. This mechanism has the disadvantage of being inefficient when there are few market participants, however, it becomes efficient as the number of participants increases.

Algorithm 1: Open

```
Input: timestamp
timeslot ← timestamp;
emit Opened(timeslot);
```

Algorithm 2: Buy

```
Input: price, volume
bids[address] ← Offer(price, volume);
if asks[address] then
    delete asks[address];
end
```

Algorithm 3: Sell

```
Input: price, volume
asks[address] ← Offer(price, volume);
if bids[address] then
    delete bids[address];
end
```

We implemented the trade reduction mechanism as a smart contract on Ethereum. The smart contract consists of open, buy, sell, and close functions. When the market is opened, the Opened event is triggered and market participants who detect the event send the desired price and volume by calling the buy and sell functions. As shown in Algorithm 2, the buy function stores the price and volume in the Offer structure and uses a mapping type that takes the participant's wallet

Algorithm 4: Trade reduction mechanism

```
   Input: bids and asks
1  heapsort bids;
2  heapsort asks;
3  last bid volume ← 0;
4  last ask volume ← 0;
5  while number of bids and asks do
6      if bid price < ask price then
7          break;
8      else
9          matched ask price ← ask price;
10         matched bid price ← bid price;
11         if bid volume >= ask volume then
12             matched ask volume ← last ask volume;
13             last ask volume ← ask volume;
14         end
15         if bid volume <= ask volume then
16             matched bid volume ← last bid volume;
17             last bid volume ← bid volume;
18         end
19     end
20 end
```

address as the key and the Offer structure as the value. To allow market participants to change their bids while the market is open, it is required to check the opposite offer (line 2-4). The mapping type has the time complexity of $O(1)$ and the space complexity of $O(n)$, while the array type has the time complexity of $O(n)$ and the space complexity of $O(n)$ [13]. Therefore there is an advantage in terms of time complexity when the market contract is deployed on Ethereum, and the trend of the gas cost is constant in this function.

At the market closing time, the market operator calls the close function. In the close function, bids with high prices and asks with lower prices are matched first until the bid price becomes smaller than the ask price or there is no more volume to match. As shown in Algorithm 4, the close function uses the heap to sort the prices (line 1-2) and determines the matched price and volume according to the trade reduction mechanism (line 3-20). Heapsort has a time complexity of $O(n \log n)$ and a space complexity of $O(n)$. The trade reduction mechanism compares bid and ask prices as the number of participants to find the clearing point, so the time complexity is $O(n)$. Therefore, the gas cost of the close function follows the log-linear trend.

V. Test Environment

Among many blockchain platforms, Ethereum does not offer the best performance, but it is the most mature. In the case of the public blockchain, the parameters such as the number of mining nodes or block gas limit cannot be adjusted. Therefore we have established a separate private network. In this section, we introduce the test environment to evaluate the fundamental performance of the Ethereum private network and the performance and cost of implementing the energy market on top of it.

TABLE I
Test Environment

CPU	32 vCPUs @ 2.60GHz
Memory	110 GB RAM
Storage	650 GB SSD
Network	linked via 1 Virtual Linux Bridge connected in 36.4 Gbits/s links

To set up the private network, we used the docker image of Ethereum client, geth v1.9.5 on the Openstack virtual machine having the resources described in TABLE I. The module that injects transactions to the Ethereum client was implemented using node.js v10.17.0 and web3 v1.2.1.

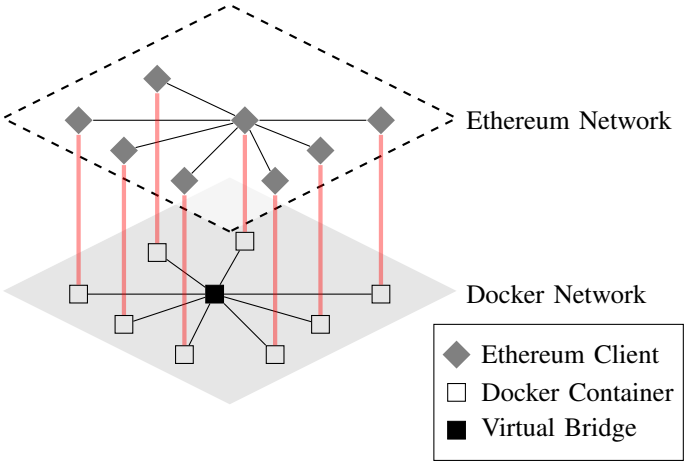

Fig. 1. Network Topology in Virtual Machine

As shown in Fig. 1, all nodes in the Docker network are connected via a virtual bridge that supports 36.4 Gbit/s. However, Ethereum clients are logically connected to 1 node to form a star topology and inject all transactions to that central node. The centralized traffic causes the overflow issue in txpool and the limitation of the test duration. Therefore, in the future, we plan to apply the mesh topology to compare the performance according to the network topology and test the distributed traffic model for a longer test period.

The test workflow is shown in Fig. 2. The shell script is used to control the number of nodes and collect the state of computing resources such as CPU, RAM, storage, and network. The number of nodes is tested up to 30 nodes. Docker provides a virtual environment to operate multiple Ethereum clients independently. The block gas limit can be changed while the Ethereum client is running, but it is not applied immediately and is gradually changed. Therefore, when creating a Docker container, the block gas limit must be set in advance. The Node.js module controls the input rate from 1 to 100 TPS and generates transactions. The latency is measured as the difference in timestamp between send and receipt. Transactions can be injected up to 5 minutes or up

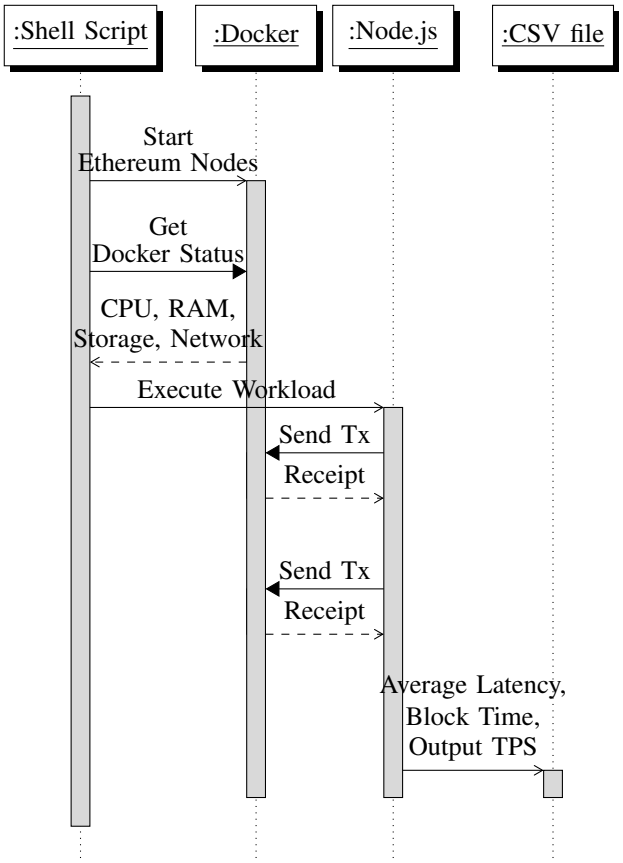

Fig. 2. Test Workflow for default contract.

to 1,000 transactions. If more than 1,000 transactions are injected, the Ethereum client is overloaded due to the size limit of the txpool. The output TPS is calculated based on the duration and number of transactions. The results such as output TPS, average latency, and block time are stored in CSV format. The test was repeated 10 times and the result is the average of 10 iterations.

VI. Performance Evaluation

A. Ethereum private network

In this section, the fundamental performance of the private network is evaluated using the default contract, Ether transfer. We transfer 1 Wei, the smallest unit from one account to another. The complexity of this contract is O(1) and 21,000 gas is consumed per transaction. In Ethereum, the factors that can affect the throughput and latency are block gas limits, number of nodes, and input rate. The input rate is the number of transactions sent to the Ethereum network per second, and the throughput is the number of transactions mined on the Ethereum network per second.

First, to analyze the effect of the block gas limit on throughput, we fixed the number of nodes at 1, changed the block gas limit from 1 million to 100 million gas, and measured the throughput by varying the block gas limit. The transactions were injected up to 1,000 with an input rate of 100 TPS.

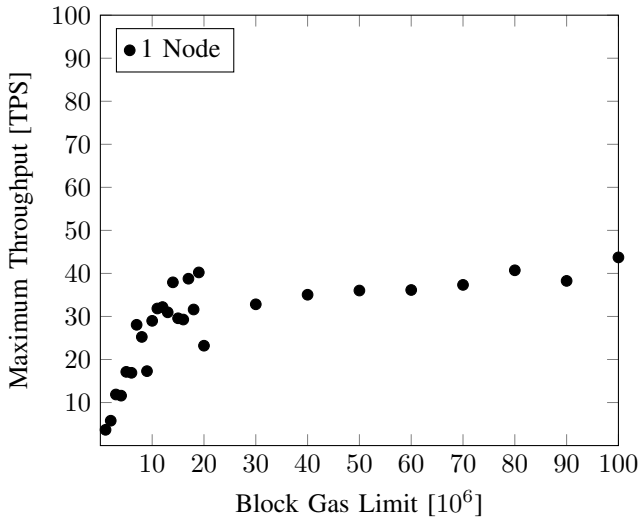

Fig. 3. Maximum throughput against block gas limit with 1 node, for the input rate of 100 TPS.

Fig. 3 shows how the maximum throughput changes with different block gas limits. We observe that the throughput increases as the block gas limit increases when the block gas limit is small, but after 20 million gas the throughput is not increased more than 50. The bottleneck can be explained by the limit of the transaction pool of Ethereum clients. The mining process in Ethereum can be modeled as a dual-queue model, with a memory pool using $M/M/1$ and a mining pool using the $M/M/c$ queue [14]. The theoretical maximum throughput $E(T)$ can be calculated using:

$$E(T) = \frac{\min(N_B, N_P)}{t_B} \tag{1}$$

where N_B is the number of transactions per block, N_P is the size of the memory pool, and t_B is block time. Ethereum has a block gas limit, which is the limit of the gas cost that can be contained in a block. The miners select a limited number of transactions from the memory pool so that the accumulated gas cost is below the block gas limit.

In the public network, the block gas limit is about 12 million gas and the default complexity is 21,000 gas. In the private network, we established, the block gas limit is set from 1 million to 100 million gas, which determines the number of transactions that can fit in a block. Therefore, the number of transactions per block is determined by the block gas limit $\lceil G_B \rceil$ and gas cost per transaction G_x as below:

$$N_B = \frac{\lceil G_B \rceil}{G_x} \tag{2}$$

In Geth v1.9.5, N_P is 1,024 for non-executable transactions and the target t_B is 10-20 seconds and G_x is 21,000. Therefore, the theoretical maximum throughput is 50 TPS when $\lceil G_B \rceil$ is increased enough and the block time is 20 seconds.

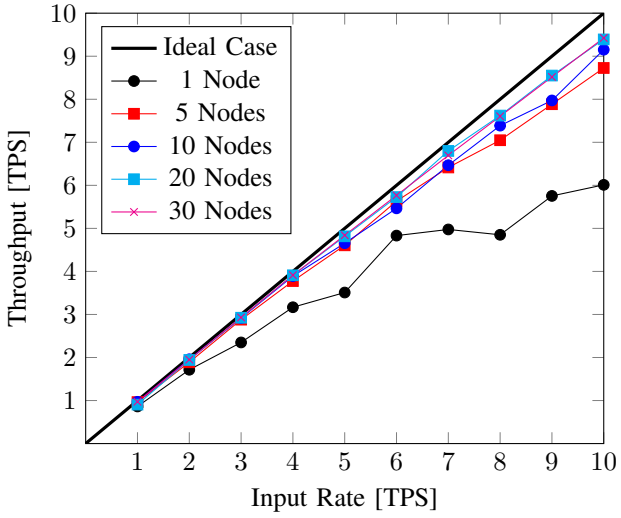

Fig. 4. Average throughput according to the input rate and the number of nodes for the block gas limit of 60 million gas.

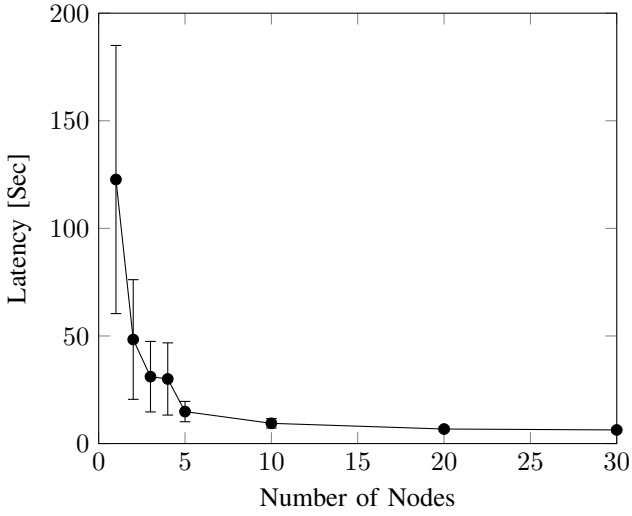

Fig. 6. Average latency according to the number of nodes for the input rate of 10 TPS. Range bars show the standard deviation of the latency.

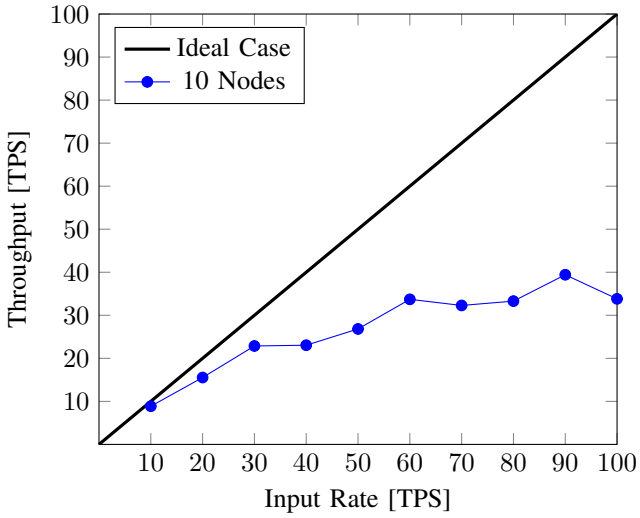

Fig. 5. Average throughput trend with 10 nodes at input rates higher than 10 TPS.

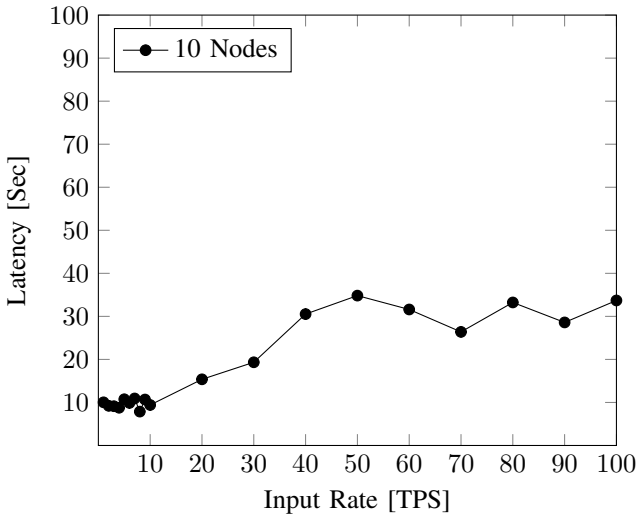

Fig. 7. Average latency according to input rates at 10 nodes.

In order to analyze the impact of the number of nodes on the throughput, we adjusted the number of nodes from 1 to 30 and measured the throughput according to the input rate. In the ideal case, the transaction is mined as soon as the transaction occurs, so the input rate and throughput are the same. The block gas limit is set to 60 million gas and the difficulty is set to 0x20000 in all tests.

As shown in the Fig. 4, the throughput gets closer to the ideal case as the number of nodes increases. This is due to the higher the number of nodes, the higher the hash rate, and the higher the mining probability in the same difficulty. If the hash rate is not sufficient for the difficulty, the block time will increase. Longer block time cause more pending transactions and consequently reduce throughput.

In Ethereum, the difficulty is automatically adjusted to keep the block time within 10-20 seconds by the Ethash algorithm. However, if the block time is outside the target time of 10-20 seconds, only 0.05% of the current difficulty can be changed, so it takes time to reach the appropriate difficulty level. In the future, we plan to study how the change of difficulty according to the change in the hash rate and number of nodes affects the performance in the long term.

No matter how large the number of nodes, if the input rate increases excessively, Ethereum cannot handle all the traffic. To measure the peak throughput, the number of nodes is fixed at 10 and the input rate is increased up to 100 TPS. Other parameters such as block gas limit, number of transactions, test time, number of tests are the same as in the previous test.

Fig. 5 shows that Ethereum can handle up to 40 TPS. This result is much lower throughput than the [4], which can handle up to 284 TPS. One of the reasons is the limit on the number

of input transactions. Since the txpool has a size of 1,024, only 1,000 transactions are injected to avoid missing transactions. If the input rate is 100 TPS, the 1,000 transactions can be injected within 10 seconds and the block time is typically 10 to 20 seconds. If the block time takes 20 seconds, the throughput drops to 50 TPS. In the worst case, the first mining took 20 seconds, and if 999 transactions are included in the first block but 1 transaction was not included, then the total mining time can be delayed to 40 seconds and the throughput would be further reduced to 25 TPS.

To test the effect of the number of nodes on latency, we fixed the input rate as 10 TPS and increased the number of nodes from 1 to 30. We observe that the latency decreases and stabilizes as the number of nodes increases as shown in Fig. 6. For example, with 1 node, the average latency is about 2 minutes, but the latency is decreased to 6 seconds when the number of nodes is 30. In particular, the latency becomes stable after 5 nodes and, in Fig. 4, the throughput is also stable when the number of nodes is more than 5. These results can be explained by the increasing probability of mining as the number of nodes increases. Conversely, in Ethermint which is based on PoS [6], the latency is less than 1 second with 1 node but it is increased to 2 minutes when the number of nodes is 20. The main cause is due to the communication overhead for consensus.

Fig. 7 shows that the latency increases with the input rate. The latency is 10 seconds at a low input rate, however increases to 30 seconds as the input rate increases. This means that the Ethereum network is overloaded when the input rate exceeds the maximum throughput which is 40 TPS. In Ethermint, latency is not affected by the input rate, but in the case of Hyperledger Fabric [11], the more input rate, the shorter the block time, and the latency was reduced.

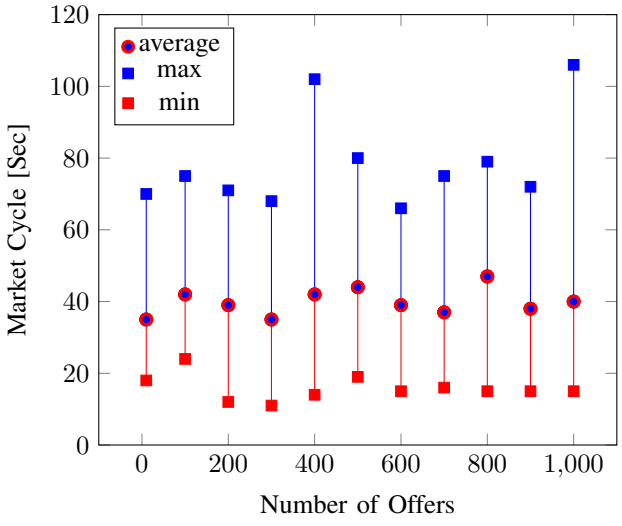

Fig. 8. Market cycle according to the number of bids. Average, maximum and minimum values are measured by testing 20 iterations for each number of offers.

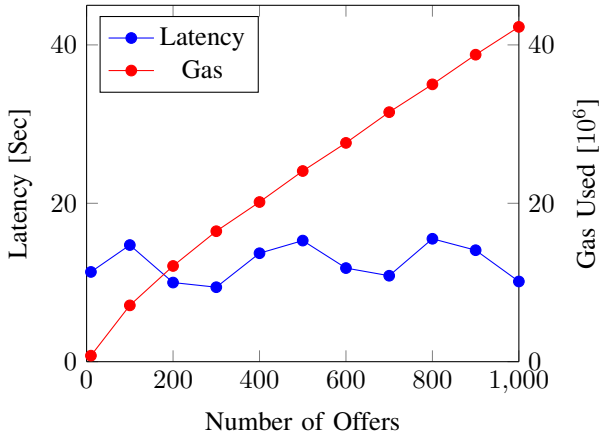

Fig. 9. Latency and gas consumption of close function according to the number of bids

B. Energy Market Performance

In this section, we evaluated the minimum market cycle, latency, and gas consumption to find out how fast the market can operate and how many participants the market can accommodate. The energy market we implemented was tested by increasing the number of bids from 10 to 1,000 in 100 units as the complexity increases with the number of market participants.

For testing the minimum market cycle, all bids are sent as soon as the market is opened and the market is closed as soon as all the bids are successfully recorded on Ethereum. The market cycle is the difference between the time when the market is requested to open and the time the market close is recorded on Ethereum. The result is shown in Fig. 8 and the average value of 20 repeated experiments. As a result, the market cycle could not be reduced to less than 10 seconds, the lower bound of the block time. On average, it was possible to perform a market cycle of 1 minute, but considering exceptional cases, it is safe to have a market cycle of 2 minutes or more.

Fig. 9 shows the latency and gas consumption for the close function, which accounts for most of the complexity of the market contract. Gas consumption is shown to increase linearly with the number of bids. This means that the complexity of the closing function follows a logarithmic linear trend with the number of bids as mentioned in section IV. On the other hand, latency is affected by the block time rather than the number of bids and is measured randomly between 10 and 15 seconds which is the range of the target block time in Ethereum.

C. Cost

To measure the minimum requirements for Ethereum client, we captured CPU, memory, disk, and network utilization when injecting 10 TPS of traffic across 10 nodes in the test environment described in Section V. For the first 3 minutes, the Ethereum client starts up and initializes, and unlocks the accounts for preparing transaction transfer. After the initialization is complete, the Ethereum client is limited to allocating

TABLE II
RESOURCE UTILIZATION.

	Average CPU (%)	Average Memory (GB)	Total Receive (MB)	Total Send (MB)	Total Write (MB)	Total Read (MB)
Node 0	136.5	8.4	10.0	30.0	0.0	0.0
Node 1	106.2	1.7	1.0	0.4	0.0	1.0
Node 2	111.8	1.8	1.0	0.3	0.0	1.0
Node 3	111.6	1.9	1.0	0.3	0.0	0.0
Node 4	106.9	1.9	2.0	0.3	0.0	1.0
Node 5	109.9	1.5	1.0	0.3	0.0	0.0
Node 6	107.8	2.0	1.0	0.4	0.0	0.0
Node 7	109.2	2.0	1.2	0.3	0.0	0.0
Node 8	117.5	2.0	1.1	0.3	0.0	0.0
Node 9	108.8	2.0	1.2	0.3	0.0	0.0

1 CPU through the docker environment setting, and 100% of the allocated CPU is used during the transaction.

Looking at the results in TABLE II, it can be seen that Node 0 uses significantly more CPU and memory compared to other nodes using 100% CPU and 2 GB of memory. This is because all transactions are being injected through Node 0. We tested 1,000 transactions of transferring Ether from 1,000 accounts to each other. The accounts need to be unlocked to trigger the transactions, and the account unlocking uses a lot of memory because the private key is decrypted and stored in memory. It takes about 1 second and several MB of RAM to decrypt one private key. In our tests, 1,000 accounts are unlocked, thus several GB of RAM is consumed. The 2 GB memory, which is common across all nodes, was used for the Ethash algorithm. The Ethash is a memory-hard mechanism, and stores several GB of Directed Acyclic Graph (DAG) in memory. The part of the DAG is used for mining.

On the other hand, the Ethereum client consumes very little network communication and disk resources. Node 0 used 30 MB to share the injected transactions to other nodes, and most nodes used 1-2 MB to share mining-related information. For disk resources, the test time is as short as 5 minutes, so it utilizes memory rather than storing most of the information on the disk.

VII. CONCLUSION

We implemented and evaluated the energy trading marketplace on Ethereum private network. The take-away of this study is to present the reachable performance and minimum cost when operating the energy market based on the blockchain and to help building a cost-effective energy market.

The evaluation provides insight into design tradeoffs and performance bottlenecks in the energy market. For example, in this paper, we found that on the Ethereum private network consisting of 10 nodes, the native application shows a throughput of 40 TPS with a latency of 10 seconds.

In case of the energy marketplace application based on the double auction using trade reduction mechanism, the longest market cycle is measured up to 2 minutes and 40 million gas is required to process 1,000 bids. In general, the intraday wholesale energy market in the EU and US operates on a 5- and 15-minute cycle market and the average number of households supplied by a distribution transformer in residential areas range from 40 to hundreds. Therefore, the performance is sufficient to meet those key requirements of the local energy market in terms of market cycle and number of participants.

The cost such as CPU and memory usage measurements confirms that the minimum requirement to run an Ethereum client is 1 core of 64-bit CPU and 2 GB of memory. In general, it is known to have as many nodes as possible in Ethereum because the number of nodes and CPUs represent the security level of PoW. However, even if more nodes, CPUs, and memory are used, it is difficult to increase throughput or reduce latency due to Ethereum's difficulty adjustment mechanism, so it is important to have an appropriate number of nodes. We argue that such insights help establishing cost-effective energy markets developed on a blockchain.

REFERENCES

[1] M. Andoni, V. Robu, D. Flynn, S. Abram, D. Geach, D. Jenkins, P. McCallum, and A. Peacock, "Blockchain technology in the energy sector: A systematic review of challenges and opportunities," *Renewable and Sustainable Energy Reviews*, vol. 100, pp. 143–174, 2019.

[2] E. Mengelkamp, J. Gärttner, K. Rock, S. Kessler, L. Orsini, and C. Weinhardt, "Designing microgrid energy markets: A case study: The brooklyn microgrid," *Applied Energy*, vol. 210, pp. 870–880, 2018.

[3] P. Ledger, "Power ledger white paper," *Power Ledger Pty Ltd, Australia, White paper*, pp. 16–21, 2017.

[4] T. T. A. Dinh, J. Wang, G. Chen, R. Liu, B. C. Ooi, and K.-L. Tan, "Blockbench: A framework for analyzing private blockchains," in *Proceedings of the 2017 ACM International Conference on Management of Data*, 2017, pp. 1085–1100.

[5] B. F. Cooper, A. Silberstein, E. Tam, R. Ramakrishnan, and R. Sears, "Benchmarking cloud serving systems with ycsb," in *Proceedings of the 1st ACM symposium on Cloud computing*, 2010, pp. 143–154.

[6] K.-L. Brousmichc, A. Anoaica, O. Dib, T. Abdellatif, and G. Deleuze, "Blockchain energy market place evaluation: an agent-based approach," in *2018 IEEE 9th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*. IEEE, 2018, pp. 321–327.

[7] M. Foti and M. Vavalis, "Blockchain based uniform price double auctions for energy markets," *Applied Energy*, vol. 254, p. 113604, 2019.

[8] D. Han, C. Zhang, J. Ping, and Z. Yan, "Smart contract architecture for decentralized energy trading and management based on blockchains," *Energy*, p. 117417, 2020.

[9] V. Buterin *et al.*, "A next-generation smart contract and decentralized application platform," *white paper*, vol. 3, no. 37, 2014.

[10] J. Kwon, "Tendermint: Consensus without mining," *Draft v. 0.6, fall*, vol. 1, no. 11, 2014.

[11] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. De Caro, D. Enyeart, C. Ferris, G. Laventman, Y. Manevich *et al.*, "Hyperledger fabric: a distributed operating system for permissioned blockchains," in *Proceedings of the Thirteenth EuroSys Conference*, 2018, pp. 1–15.

[12] J. Horta, D. Kofman, D. Menga, and A. Silva, "Novel market approach for locally balancing renewable energy production and flexible demand," in *2017 IEEE International Conference on Smart Grid Communications (SmartGridComm)*. IEEE, 2017, pp. 533–539.

[13] A. Jha, R. K. Bhattacharjee, M. Nandi, and F. A. Barbhuiya, "A framework for maintaining citizenship record on blockchain," in *Proceedings of the 2019 ACM International Symposium on Blockchain and Secure Critical Infrastructure*, 2019, pp. 29–38.

[14] R. A. Memon, J. P. Li, and J. Ahmed, "Simulation model for blockchain systems using queuing theory," *Electronics*, vol. 8, no. 2, p. 234, 2019.