Performance Analysis of Ethereum Transactions in Private Blockchain

Sara Rouhani
Department of Computer Science
University of Saskatchewan
Saskatoon, Sk, Canada
sara.rouhani@usask.ca

Ralph Deters
Department of Computer Science
University of Saskatchewan
Saskatoon, Sk, Canada
deters@cs.usask.ca

Abstract—**Ethereum, the well-known blockchain platform, does not have any limit for block size, unlike Bitcoin. However, there are other obstacles in processing unlimited transactions per second. Ethereum blockchain code runs by different clients, and they run on different speed and present the different level of performance. This paper studies Ethereum transactions and it analyzes two most popular Ethereum clients, Geth and Parity, on a private blockchain to obtain the better understanding of the effect of different clients on Ethereum performance. The results show that the transactions are 89.8 percent on average faster in Parity client in comparison with Geth client, using the same system configuration.**

Keywords-Blockchain; Ethereum; Transaction; Performance

I. Introduction

A. Blockchain

Blockchain, the distributed system, which was firstly introduced by Bitcoin and digital currencies, promises an innovative future in collaboration and business market. It is a global ledger that can record transactions efficiently and permanently on the chain of blocks at a specific timestamp.

Blockchain consists of a set of nodes connected via peer to peer network. Nodes or parties in the network can interact directly without any trusted and third party. This trust-less infrastructure provides faster and cheaper transactions.

In blockchain, each node can create a series of transactions. Transactions pass the hash function several times then they group and record as blocks. So, basically, blockchain can be seen as a chain of blocks which include the hash of the transactions' log, and only one block can be added to the chain at a time. Each block has a timestamp and a link to the previous block, and it is identified by its hash value. Each node in the blockchain has the entire network transaction history. Therefore, information stored on the blockchain cannot be modified or deleted, and blockchain guarantees this by Cryptography.

Blockchain offers security, reliability, integrity, and durability. It reduces costs by removing third parties and submitting faster transactions. Also, it provides a variety of use cases [3] beyond just simple financial transactions such as smart contracts, decentralized application, digital identity, and a lot more.

B. Comparing Blockchain performance with centralized systems:

It is true that transactions of blockchain are slower than centralized databases because blockchain must handle more tasks in compare with centralized databases. The three additional tasks include: signature verification, consensus mechanisms, and redundancy [4]. However, it should be reminded that blockchain has a deterministic advantage when it comes to providing a reliable, robust, and secure way to store data without intermediaries interfering. In this research, we have calculated the performance of blockchain based on transactions processing speed which submitting to the blockchain directly without passing through any third party, so there is no other processing before or after submitting transactions to the blockchain which affect the processing time.

C. Public vs. Private Network

Public or permission-less blockchain consists of multiple nodes, and it is completely trust-less, so anyone can join, send transaction, and participate in the consensus process. In public blockchain, nodes should be synchronized, and if the chain is big, it gets huge time and energy. The network is completely decentralized or "fully decentralized" in this case.

Against public blockchain, we have private blockchain which nodes need permission to join a controlled blockchain and read the state of the chain. The Private blockchain is faster, safer, and more efficient but permissions are hold centralized, so it sacrifices decentralizing.

There is another type of blockchains which is hybrid and are called "Consortium blockchains", where some permission nodes control the consensus process. Based on the blockchain policies, accesses could be public or permission based. This type of blockchain is considered "patricianly decentralized".

Based on application and needs of organization public, private and hybrid blockchains can be appropriate [5].

D. Blockchain platforms

There are several blockchain platforms; Bitcoin [1], the earliest distributed ledger, Ethereum, Hyperledger, multichain, etc. Ethereum [2] has a large, global development community, completely open source and it supports variety of use cases [3] such as smart contract and decentralized applications (Dapps) which can build on top of the Ethereum blockchain because it

978-1-5386-0497-7/17/$31.00 ©2017 IEEE

has been designed to be adaptable and flexible with Turing complete scripting language built in [5].

E. Ethereum

In this part, we explain some Ethereum basically concepts: account, transactions, and clients [6, 7, 8]. Ethereum Basic unit is account. Accounts are required for everyone who wants to send any transaction to the blockchain. Ethereum includes two types of accounts: Externally Owned accounts (EOA), users directly send transactions via them, and Contract Accounts, based on the codes of the contract if it needs to call another contract it sends an internal transaction. Every account is defined by two keys, a private key, and public key. Each account's address comes from the last 20 bytes of public key, and it is the necessary part of each transaction.

It is important to understand the difference between transactions and internal transactions. The transactions signs by EOA's private key, the sender of the transaction, and after confirmation a hash value returns which we can track all the blockchain transactions by it. However, there is no signature field for internal transactions. Different sources use the terms of call or message for internal transactions.

In this study, all transactions are sent from EOAs.

F. Ethereum Transactions

The Transaction (T) is a single instruction code which sends a message from Externally Owned Account. In many aspects, blockchain transactions are like traditional transactions. Ethereum blockchain launches with a genesis block and then other transactions process and create new blocks and a new state.

Any change in the blockchain state starts with a transaction which is sent by EOA. This transaction can be a directly transferring Ether (Ethereum digital currency) to another account, or it could be a trigger for a contract. This transaction signs by sender's account private key.

Every transaction includes several fields and the miner in the network prioritize the transactions based on GasPrice field. If several transactions have been sent from the same account, the miner would calculate them by nonce value. The nonce field in the transaction is equal to the number of the operations sent by the sender, this value increment by sending every new transaction. Therefore, the transactions which are sent from the same account that have the same gas price would be executed respectively [7]. Figure 1 illustrates the diagram of transaction process in Ethereum Blockchain.

Ethereum can be viewed as a transaction-based state machine. Ethereum runs state transition function to make sure that transition from current state would lead to a new valid state. Formula 1 shows the formal explanation of a valid state transition based on Ethereum Yellow paper [7]. ϒ is the Ethereum state transition function and T represents the transaction.

$$\boldsymbol{\sigma}_{t+1} \equiv \Upsilon(\boldsymbol{\sigma}_t, T) \qquad (1)$$

The Ethereum state transition function has several tasks such as checking if the transaction is well-formed, updating account balances and nonces, refunding the reminding gas, and rewarding miners for computation. The details of implementation of this function are explained in Ethereum Yellow paper [7].

G. Transaction Safety Viewpoint

Ethereum transactions and smart contracts are serially executed by miners, and they are not concurrent. EVM (Ethereum Virtual Machine, behaves like a single processor computer and every node runs it) executes one transaction in a moment. However, the common security issues of the concurrent systems can occur similarly when a smart contract's code runs, and it sends internal transactions to call another smart contract.

The security and valid results are vital in this type of complex systems, and any attack or mistake can lead to the high cost of failure. Although the EVM designed to be deterministic, there are lots of challenge which can result in non-deterministic output. Such as the attack on the DAO contract [9], which allowed to $50 detriment. Although, it seems that the whole of the investors' funds are returned by Hard Fork [10], but this was a big alarm to think about the security aspect of smart contracts and identify potential reasons treating the deterministic results.

Even though Ethereum does not execute concurrent transactions, but it suffers from similar risks that threat concurrent systems. The potential problems are discussed in the below [11].

The main issue is external transactions, the current smart contract could be written well-formed and safe, but an untrusted external call of insecure contract can lead to unexpected results. One of the side effects of external transactions can be the race condition, the situation which the attacker can control the flow of code.

Reentrancy is the consequence of this bug, the conditions that involved functions could repeatedly be called before the first invocation of the function is finished. Ethereum doesn't have a mutex or lock to avoid this.

Cross-function Race Condition is another attack when an attacker could be able to attack using two different functions, that share the same state.

In addition to race condition from an attacker, there are some other problems because of the nature of the blockchains: this fact that order of the transactions themselves could be easily modified. This issue is known as Transaction Ordering Dependence [14].

The blockchain technology and Ethereum are still in the early stage to identify or prevent these problems automatically. Most of the works for solving such these problems is introducing frameworks to verify codes before deploying it into the blockchain [12, 13, 14].

H. Ethereum Clients

In Ethereum Blockchain first step for becoming node is running an Ethereum Client, a code to parse and verify blockchain. Ethereum clients download the whole blockchain and verify transactions. There are a variety of clients for Ethereum which have written by the different programming language. This variety has a lot of advantages; it can help bugs and issues pop-up faster. Also, it provides more reliable infrastructure. Table I Shows Ethereum clients.

Two most pioneer clients of Ethereum are Geth [15] and Parity [16]. In this study, we decided to explore Ethereum performance under these two different clients.

Geth is one of the first Ethereum clients which implemented by go programming language. Geth is easy to build, and it has mining option on its own.
Parity is Ethereum newer client with an emphasis on efficiency. Parity has been written by Rust programming language. Recently this client has been attracted many Ethereum decentralized developer's attention because of the faster sync process. Parity doesn't have mining option built into same as Geth but it supports Ethereum JSON-RPC interface so it can run with any miner which implements proof of work, such as Ethminer [17]. Table II shows the feature of Geth and Parity Clients.

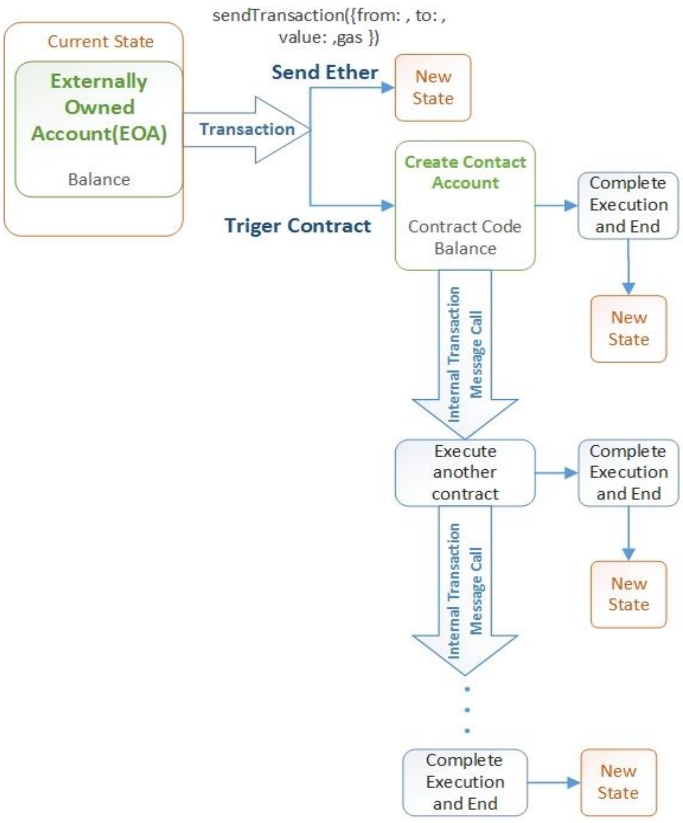

Figure 1. Ethereum Transaction Diagram

TABLE I. ETHEREUM CLIENTS[6]

Client	Language	Developers
Go-Ethereum (Geth)	Go	Ethereum Foundation
Parity	Rust	Ethcore
Cpp-Ethereum	C++	Ethereum Foundation
Pyethapp	Python	Ethereum Foundation
Ethereumjs-lib	Javascript	Ethereum Foundation
Ethereum(J)	Java	ether.camp
Ruby-Ethereum	Ruby	Jan Xie
EthereumH	Haskell	BlockApp

TABLE II. GETH AND PARITY FEATURES

	Programming	Database	State	Javascript Console	Mining
Geth	Go-lang	LevelDB	Heap	Yes	Yes
Parity	Rust	Rocksdb	Stack	Not directly but can run Geth attach or it can use Node.js CLI console	No

In Parity Technology Blog [18], there is a paper which explains an experiment to compare Parity and Geth performance for block processing time through making the clients do a full sync of frontier Chain to understand which one of them is faster in syncing. They claim that block processing speed indicates that how much mining process is fast and in private chains, how much transactions are efficient through the network. The results show that Parity is three times faster than Geth. In this study, we have done testing through sending a flow of transactions to two private Ethereum blockchains which run by two different clients; Parity and Geth and calculated the transaction processing time one by one and then all together to understand the behavior and performance of transactions which sent from these two clients in private blockchains.

II. PERFORMANCE RESULTS

For calculating the performance of Ethereum transactions, we run two Ethereum private blockchain with the same configuration. One of them executed by Parity client and the second one executed by Geth client. System configuration is 24 GiB RAM and Core i7-6700 CPU. We used Node.js program to send several bunches of transactions to the blockchain using Geth and Parity clients and captured the processing time for confirmed transactions, one by one and all together.

Figure 2 shows the time for processing 2000 transactions one by one in two clients. The average time for each transaction for Parity client is 104.609 and for Geth client is 198.9125 Millisecond which indicates that in this case Parity client has been 90 percent faster than Geth client.

Figure 3 and figure 4 demonstrate the average time and total time respectively for processing 2000 transactions for different clients by using a system with the different amount of RAM.

A Virtual Machine has been used to change the value of the RAM virtually to record the result. It was already predictable that by increasing the amount of RAM, we should expect the speed of transactions would boost. Moreover, the result explains that increasing the RAM would affect Geth client more than Parity client.

Figure 5 shows the graph of sending different amount of transactions to the system with 24 GB RAM with two different clients. As is indicated in table 3, Parity Client is at least 87 Percent faster than Geth with minimum 1000 transactions up to 10000 transactions.

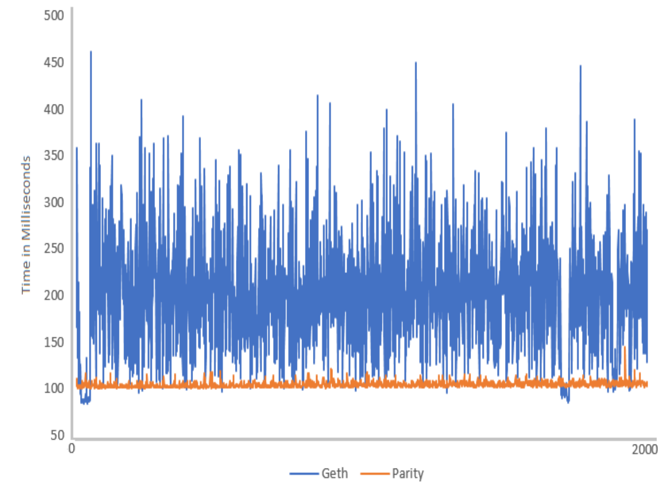

Figure 2. Time in Milliseconds for 2000 Transactions of Parity and Geth clients

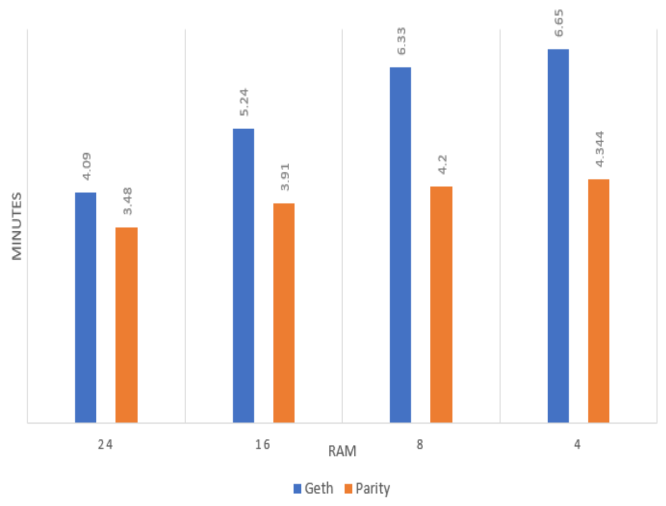

Figure 4. Total time for processing 2000 transactions on a client with different amount of RAM

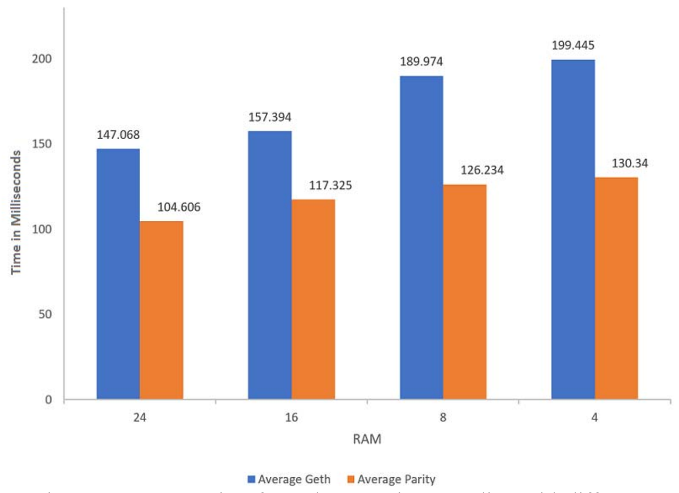

Figure 3. Average time for each transaction on a client with different amount of RAM

TABLE III. THE PERCENTAGE OF INCREASING SPEED IN PARITY CLIENT IN COMPARISON WITH GETH CLIENT.

Number of Transaction	**Geth**	**Parity**	**Percentage of increasing speed in Parity**
1000	3.27	1.74	87.9%
2000	6.63	3.48	90.51%
3000	9.8	5.13	87.71%
4000	13.33	6.91	90.90%
5000	16.86	8.71	93.57%
10000	34.882	18.522	88.32%

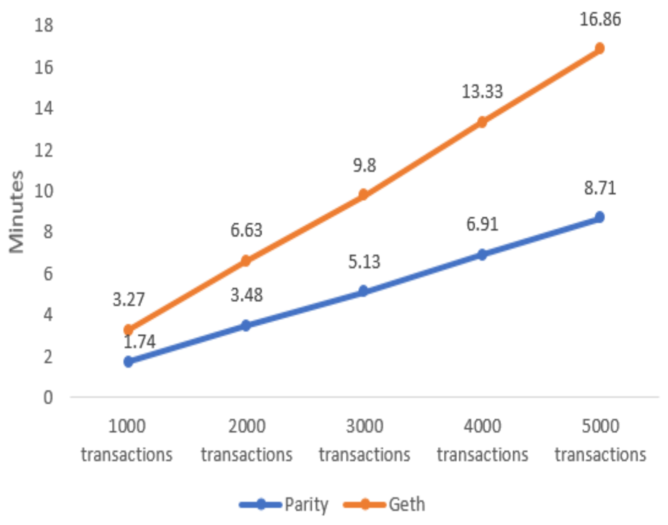

Figure 5. Total time in Minutes for processing different amount of transactions by different clients

III. SUMMARY & FUTURE WORK

This paper discusses blockchain transactions from Ethereum blockchain viewpoint. We investigate the different types of transactions, the problems threat the deterministic result, and we explore Ethereum transactions performance using Ethereum private blockchain which runs by two clients, Parity and Geth, separately.

This study presents the results indicate that Parity processes transactions significantly faster than Geth, 89.82 % by the average for a different set of transactions from 1000 to 10000 transactions. There are several reasons which make Parity faster than Geth, using a more rapid database or using different implementation approaches such as stack. Studying on the grounds and details of performance would lead to the new generation of Ethereum clients which can reduce the limitations on scalability and faster blockchains.

Besides performance, the security and safety of transactions are another challenge for blockchain to prevent the probable attack similar to concurrent systems. People need to trust this new technology to be encouraged and deposit their money into the blockchain.

REFERENCES

[1] S Nakamoto, "Bitcoin: a peer-to-peer electronic cash system", online, http://bitcoin.org/bitcoin., 2008

[2] Ethereum. https://github.com/ethereum/

[3] "Smart contract: 12 use cases for business & beyond" Prepared by: smart contracts Alliance, In collaboration with Deloitte An industry initiative of the Chamber of Digital Commerce, https://www.cryptocoinsnews.com/smart-contracts-12-use-cases-for-business-and-beyond/, 2016

[4] "Blockchain vs centralized databases", http://www.multichain.com/blog/2016/03/blockchains-vs-centralized-databases/, 2016

[5] V. Buterin, "On public and private blockchain", https://blog.ethereum.org/2015/08/07/on-public-and-private-blockchains/, 2015

[6] Ethereum white paper, "A next-generation smart contract and decentralized application platform", https://github.com/ethereum/wiki/wiki/White-Paper

[7] G. Wood, Ethereum yellow paper: "A secure decentralised generalised transaction ledger", https://ethereum.github.io/yellowpaper/paper.pdf

[8] Ethereum Homestead Documentation, http://www.ethdocs.org/en/latest/

[9] V. Buterin. Critical update re: DAO vulnerability. https://blog.ethereum.org/2016/06/17/critical-update-re-dao-vulnerability, 2016

[10] https://blog.ethereum.org/2016/07/20/hard-fork-completed, 2016

[11] https://github.com/ConsenSys/smart-contract-best-practices#solidity-tips

[12] K. Bhargavan, A. Delignat-Lavaud, C. Fournet, A. Gollamudi, G. Gonthier, N. Kobeissi, N. Kulatova, A. Rastogi, T. Sibut-Pinote, N. Swamy, and S. ZanellaB´eguelin. Formal verification of smart contracts: Short paper. In PLAS, pages 91–96. ACM, http://dl.acm.org/citation.cfm?id=2993611, 2016.

[13] J. Pettersson and R. Edstr¨om. Safer Smart Contracts through Type-Driven Development. Master's thesis, Chalmers University of Technology, Department of Computer Science and Engineering, Sweden, https://publications.lib.chalmers.se/records/fulltext/234939/234939.pdf, 2016

[14] L. Luu, D.-H. Chu, H. Olickel, P. Saxena, and A. Hobor, "Making Smart Contracts Smarter," *In Processing of the 23rd ACM Conference on Computer and Communications Security*, Hofburg Palace, Vienna, Austria, 2016.

[15] https://github.com/ethereum/go-ethereum/

[16] https://parity.io/

[17] https://ethermine.org/

[18] https://blog.parity.io/performance-analysis/