Quorum Blockchain Stress Evaluation in different environments

Daniel P. Mera

CUNY John Jay College, danmera80@gmail.com

Follow this and additional works at: https://academicworks.cuny.edu/jj_etds

Part of the Computer and Systems Architecture Commons, Data Storage Systems Commons, Digital Communications and Networking Commons, Forensic Science and Technology Commons, and the Other Computer Engineering Commons

Recommended Citation

Mera, Daniel P., "Quorum Blockchain Stress Evaluation in different environments" (2019). CUNY Academic Works. https://academicworks.cuny.edu/jj_etds/118

This Thesis is brought to you for free and open access by the John Jay College of Criminal Justice at CUNY Academic Works. It has been accepted for inclusion in Student Theses by an authorized administrator of CUNY Academic Works. For more information, please contact AcademicWorks@cuny.edu.
John Jay College of Criminal Justice
Master of Science, Digital Forensics and Cybersecurity

Quorum Blockchain Stress Evaluation in different environments

Presented by:

Daniel Mera

Contact: daniel.mera@jjay.cuny.edu; danmera80@gamil.com

May 2019
I. Contents

I. Contents ......................................................................................................................... II
II. List of figures .................................................................................................................. IV
III. List of tables .................................................................................................................. V
IV. Abstract ......................................................................................................................... VI
1 Introduction ..................................................................................................................... 1
2 Overview .......................................................................................................................... 3
  2.1 Quorum ....................................................................................................................... 3
  2.2 Quorum Nodes .......................................................................................................... 4
  2.3 Transactions ............................................................................................................... 4
      2.3.1 Constellation and Tessera .................................................................................. 4
      2.3.2 Public Transactions ............................................................................................. 5
      2.3.3 Private Transactions ............................................................................................. 5
  2.4 Consensus Algorithms ............................................................................................... 7
      2.4.1 Raft Consensus ...................................................................................................... 7
      2.4.2 Istanbul BFT ........................................................................................................ 7
  2.5 Performance ................................................................................................................ 8
  2.6 Chainhammer .............................................................................................................. 8
  2.7 Quorum Maker .......................................................................................................... 8
3 Equipment and Methods ................................................................................................. 10
  3.1 Equipment .................................................................................................................. 10
      3.1.1 Local Nodes ......................................................................................................... 10
      3.1.2 Virtual Machines ................................................................................................. 10
3.1.3 Cloud Instances

3.2 Configurations

3.2.1 Operating System

3.2.2 Quorum Maker Installation

3.2.3 Chainhammer Installation

3.3 Methods

3.3.1 TPS Benchmarking

4 Results

4.1 Results Local Server Nodes

4.1.1 10 thread Results

4.1.2 30 thread Results

4.1.3 50 thread Results

4.1.4 100 thread Results

4.2 Results Virtual Machines

4.2.1 10 thread Results

4.2.2 30 thread Results

4.2.3 50 thread Results

4.2.4 100 thread Results

4.3 Results Cloud Instances

4.3.1 10 threads Results

4.3.2 30 threads Results

4.3.3 50 threads Results

4.3.4 100 threads Results

5 Conclusion

6 References

7 Annexes

7.1 Result Reports
7.1.1 Local Servers ........................................................................................................ 36
7.1.2 Virtual Machines .................................................................................................... 37
7.1.3 Cloud Instances ...................................................................................................... 38

II. List of figures

Figure 1: Quorum's Architecture and components ............................................................ 3
Figure 2: Private Transaction Process Flow ........................................................................ 5
Figure 3: Quorum Maker platform ..................................................................................... 9
Figure 4: Node 1 Setup .................................................................................................... 13
Figure 5: Accept/Reject Request ....................................................................................... 14
Figure 6: Joining node setup ........................................................................................... 15
Figure 7: Results for 10 threads and 10000 Transactions Local Servers ......................... 17
Figure 8: Local Server Nodes - 10 Threads Comparison ............................................... 18
Figure 9: Results for 30-thread 100-transaction Test Local Servers ............................... 18
Figure 10: Local Server Nodes - 30 Threads Comparison .............................................. 19
Figure 11: Results for 50-thread 1000-transaction Test Local Servers ............................ 19
Figure 12: Local Server Nodes - 50 Threads Comparison .............................................. 20
Figure 13: Results for 100-thread 100-transaction Test Local Servers ............................ 20
Figure 14: Local Server Nodes - 30 Threads Comparison .............................................. 21
Figure 15: Local Server - All Threads and all TX compared ........................................... 21
Figure 16: Results for 10-thread 100-transaction Test Virtual Machines ......................... 22
Figure 17: VM Nodes - 10 Threads Comparison ............................................................. 23
Figure 18: Results for 30-thread 1000-transaction Test Virtual Machines ....................... 23
Figure 19: VM Nodes - 30 Threads Comparison ............................................................. 24
Figure 20: Results for 50-thread 10000-transaction Test Virtual Machines ..................... 24
Figure 21: VM Nodes - 50 Threads Comparison ............................................................. 25
Figure 22: Results for 100-thread 10000-transaction Test Virtual Machines ................................25
Figure 23: VM Nodes - 100 Threads Comparison ........................................................................26
Figure 24: VM Nodes - All Threads and TXs Compared .................................................................26
Figure 25: Results for 10-thread 1000-transaction Test Cloud Instances ........................................27
Figure 26: Cloud Nodes - 10 Threads Comparison ......................................................................28
Figure 27: Results for 30-thread 100-transaction Test Cloud Instances ........................................28
Figure 28: Cloud Nodes - 30 Threads Comparison ......................................................................29
Figure 29: Results for 50-thread 10000-transaction Test Cloud Instances ......................................29
Figure 30: Cloud Nodes - 50 Threads Comparison .....................................................................30
Figure 31: Results for 100-thread 10000-transaction Test Cloud Instances ....................................30
Figure 32: Cloud Nodes - 100 Threads Comparison ...................................................................31
Figure 33: Cloud Nodes - All Threads and TXs Compared .............................................................32
Figure 34: TPS Average Graph ..................................................................................................34

III. List of tables

Table 1: Desktop Servers used ....................................................................................................10
Table 2: Virtual Machines used ..................................................................................................11
Table 3: Cloud-based Instances used ..........................................................................................11
Table 4: TPS Calculations Example ............................................................................................16
Table 5: TPS Average Table .......................................................................................................33
IV. Abstract

In today’s world, the Blockchain technology is used for different purposes has brought an increment in the development of different Blockchain platforms, services, and utilities for storing data securely and efficiently. Quorum Blockchain, an Ethereum fork created by JPMorgan Chase, has placed itself in one of the widely used, efficient and trustful Blockchain platforms available today.

Because of the importance which Quorum is contributing to the world, it is important to test and measure different aspects of the platform, not only to prove how efficient the software can be but as well as to have a clear view on what type of environment the platform could be better used.

This research has been focused on testing the efficiency and speed of the transactions been sent to the platform in different types of environments, like local server nodes, virtual machine nodes and cloud instance nodes, by focusing its tests on the average of transactions per second or TPS being made.

Once the tests were carried out, and results were obtained, there was a clear difference between the environments used. Using cloud instances for nodes improves the TPS for every single type of transactions over any other types of nodes. It might have been predicted that Cloud Instances would have done a better performance, but other factors could have caused the fallback in results for this type of nodes.
1 Introduction

Since 1991 when Stuart Haber and W. Scott Stornetta first tried to implement a cryptographical chain of blocks, blockchain has evolved and improved in these almost 30 years of scientific development. Today this technology has been used in enterprise areas such as banking. Bitcoin being the first decentralized cryptocurrency is mainly focused on saving all its transactions using this technology.

Ethereum is a new generation decentralized blockchain and smart contract platform built for almost any type of applications. It is mainly focused on lowering costs and incrementing speeds of transactions as well as decentralizing and securing the information.

Quorum, based on the Ethereum platform, is a project of JPMorgan Chase. It is a permissioned type of network; it has a distributed ledger but not a decentralized network, all nodes are known to each other. Therefore, it does not use decentralized consensus mechanisms like PoW or PoS. Quorum uses a raft-based consensus and Istanbul BFT, this makes transaction speeds faster and much higher throughput.

JPMorgan Chase’s team has Quorum available to be used and tested, but they are still developing and updating the platform and information for the ledger has been very limited. Other independent sources have published information about this project, with tests being done to it, but everything after a few months gets outdated.

There aren’t many tests being done to this platform, that is why this paper focuses on the following:

- Understand how the Quorum blockchain works,
- Setup the Quorum blockchain in different environments, and
- Test the performance of the Quorum blockchain platform in different environments.

Throughout this document, the following is presented:
Section 2 summarizes and gives an overview of what the Quorum blockchain platform and all of its components are, as well as the tools used in this research such as Chainhammer and Quorum Maker.

Section 3 explains the types of evaluation and testing that are done, including the materials used in these tests.

Section 4 shows the results from the evaluations done explaining each type of tests performed throughout the different types of networks.

Finally, the conclusion established in Section 5 explains the results obtained from the test, gives an analysis obtained from the results as well as some recommendations for future experimentation.
2 Overview

2.1 Quorum

This new platform [1] has been developed by JPMorgan Chase and is an Ethereum based distributed ledger protocol which it was mainly focused on providing the financial services industry with a public permissioned ledger of Ethereum. Quorum uses a ‘raft-based’ consensus algorithm, thus making it run faster blocktimes, transaction finality, and on-demand block creation. Additionally, it reaches Data Privacy by implementing the “private” transactions type.

It has a layer where the logic in the additional privacy functionality resides; it sits atop of the standard Ethereum protocol layer. Figure 1 shows the components in Quorum’s architecture.

As shown in Figure 1, Quorum a go-Ethereum fork has the Transaction Manager which allows access to the encrypted transaction data for private transactions, it manages local data storage and communications with other transaction managers, but it doesn’t have access to any sensitive private keys. It uses the Enclave for the cryptographic functionality. The Transaction Manager can be load balanced easily.

The Crypto Enclave is mainly responsible for private key management and encryption and decryption of private transaction data. The Enclave operates closely with the Transaction Manager to strengthen privacy by managing the encryption and decryption in an isolated manner. It acts as a virtual Hardware Security Module; it is isolated from other components.

Quorum also implements two different types of consensus, Raft-Based which is meant for faster block time, and on-demand block creation and better transaction finality, and Istanbul BFT which is a three-phase consensus, for better fault tolerance and self-verifiable blocks. Next, we also see the Network Manager, which controls the access to the network and enables the permissioned network of the nodes to be created.
Quorum's main purpose is to use cryptography to show the sensitive data only to those involved in the transaction. To solve this, the platform integrates a single shared blockchain and a combination of smart contract software architecture and modifications to Ethereum. There are some modifications done to the go-ethereum code, such as modifications to the block proposal and validation processes. Block validation process is modified in a way that all the nodes have to validate public transactions and any private transaction they are party to by executing the contract code associated with the transactions. For the rest of the private transactions, a node skips the contract code execution process. These implemented processes create a segmentation of the state database, which is split into a private state and a public state database.

2.2 Quorum Nodes

The implementation of nodes in this platform is a lightweight fork of geth, and it takes the same advantage of the regular Ethereum nodes, in which they will get updated regularly with all the future releases. Nodes in Quorum achieve consensus using Raft or Istanbul BFT, unlike Proof-of-Work, which is used in Ethereum. There is also the P2P layer which has been modified to allow connections to and from permissioned nodes. Block generation logic and block validation logic have been modified and replaced the "global state root" with the "global public state root." We also see the transaction creation modified to allow the Transaction data to be replaced by encrypted hashes; this preserves private data where it is required. Finally, the Gas price has been removed.

2.3 Transactions

Quorum introduces the idea of "Public Transactions" as well as "Private Transactions," where Quorum has extended the Ethereum Transaction Model to include an optional "privateFor" parameter, where this makes the transactions to be treated as private, and the Transaction Type method as "IsPrivate" to recognize the private transactions.

2.3.1 Constellation and Tessera

These are basically Haskell and Java implementations of a general-purpose system for submitting information securely. The messages sent are encrypted with PGP. These can be applied in many types of application where you need individual message exchange within a network. They use two submodules, the Node, and the Enclave, which were previously explained.
2.3.2 Public Transactions

These transactions [2] are the ones with the payload visible to all the participants within the same Quorum network. This type is the same as the Ethereum Transactions.

2.3.3 Private Transactions

These transactions [2] have the payload only visible to the network participants whose public keys are specified in the “privateFor” parameter of the transaction. This parameter can have multiple addresses separated by a comma. When the Quorum Node comes across a transaction whose “privateFor” values is non-null, it automatically sets the Transaction Signature to be set as 37 or 38, in other words, to ‘private,’ opposed to the 27 or 28 values which are ‘public’ within Ethereum.

In Figure 2, the image shows how private transactions are processed within the Quorum platform.

![Figure 2: Private Transaction Process Flow](image)
In this example there are three parties, A, B and C, there is a transaction generated from A to B which is now called Transaction AB, the C parties is not part of the transaction. An initial transaction is being sent to the Quorum Node by Party A, the parameter “parivateFor” includes the public keys for Party A and B. Party A’s node sends the transaction the toe Transaction Manager and it requests to store the transaction payload. Next, the Transaction Manager in the Node makes a call to the Enclave to validate the sender and encrypts the payload. Once the Enclave validates the key for Party A, it performs the transaction conversion by generating a symmetric key and a random Nonce. Later encrypting the transaction payload and Nonce with the symmetric key, following with the calculation of the SHA3-512 hash of the encrypted payload, repeating the process for the recipients of the transaction, which in this case is for Party B. Finally, it returns the encrypted payload, the hash and the encrypted keys to the Transaction Manager.

The Transaction Manager for Party A stores the encrypted payload and symmetric key using the hash as the index, securely transferring the hash, the encrypted payload and the encrypted symmetric key which have been encrypted with Party B’s public key to the Transaction Manager for Party B. The transaction Manager then responds with a Ack/Nack response (If Party A does not receive a response or a Nack from Party B then the transaction will not be propagated to the rest of the network).

If Party B’s Transaction Manager has received a successful data transmission then Party A’s Transaction Manager returns the hash to the Quorum Node which now replaces the transaction’s original payload with the hash, and changes the transaction’s V value to 37 or 38, indicating it is a private transaction with the associated encrypted payload.

Now the rest of the network receives the transaction using the standard Ethereum P2P protocol. A block containing the transaction AB is created and distributed to each party on the network. All the parties will try to process the transaction, and each node will recognize the V value as being private and required a payload decryption.

Party C, which was not part of the transaction, will receive a “NotARecipient” message and will skip the transaction. The Private State DB will not be updated.

The local transactions for each Party will search the hash within their Transaction Manager and verify if they hold the transaction. Each will call the Enclave and pass the encrypted payload and symmetric key and signature if they are part of the transaction. The Enclave will validate the signature and then decrypt the symmetric key using the Party’s private key. It will decrypt
the transaction payload using the symmetric key and return the decrypted payload to the Transaction Manager.

Finally, the Transaction Manager in each party will then send the decrypted payload to the EVM for the execution of the contracts. This will update the state of the Node’s Private StateDB only.

2.4 Consensus Algorithms

While other blockchain platforms use either Proof-of-Work (PoW) or Proof-of-Stake, Quorum implements different consensus mechanisms [3], Raft, which is meant for faster blocktimes, transaction finality, and on-demand block creation, and Istanbul BFT, which is meant for transaction finality, manageable validator set, and higher throughput.

2.4.1 Raft Consensus

It [4] is a distributed consensus algorithm designed to be easily understood; this algorithm solves the problem of getting multiple servers to agree on a shared state even when there are failures.

This consensus functions by electing a leader in a cluster, the leader’s responsibility is to accept client requests and manage the replication of the log to other servers. Data is only sent from one direction, from leader to the other servers. The leader is also responsible for generating new blocks.

2.4.2 Istanbul BFT

Istanbul Byzantine Fault Tolerance or IBFT consensus [5] is an algorithm where mutually distrusting nodes are allowed to form a network where the majority of the nodes are assumed to be correct at any moment in time.

In IBFT each block requires multiple rounds of voting by the set of validators to arrive at a mutual agreement, which is recorded as a collection of signatures on the block content.

By using IBFT, a network can continue to function correctly even if some nodes are “dishonest” or have invalid blocks. It can tolerate an “f” number of dishonest or faulty nodes within a network of 3f + 1 nodes. It is a 1/3 of faulty nodes being tolerated.

Similar to PBFT, it is based on a 3-phase commit, using Pre-Prepare, Prepare, and Commit Stages.
2.5 Performance

Quorums focuses mainly on having a strong performance by handling hundreds of transactions per second (TPS). These transaction speeds can be configured to the needs of the network and the smart contracts. This is where the RAFT and IBFT consensus algorithms ensure a much better performance [6].

2.6 Chainhammer

Dr. Andreas Kruger, a well-known blockchain and cryptocurrency expert, has prepared a set of scripts which test the efficiency of the transaction latency and transaction throughput of Ethereum type blockchains.

Chainhammer [7], as Dr. Kruger has named the scripts, now updated to version 59, in order to be used with Ethereum type blockchain networks such as Quorum and implemented in any type of machines like physical desktops or servers, Virtual Machines, or Cloud-based instances such as Google Cloud, AWS, Azure or others.

2.7 Quorum Maker

A tool created by Synechron Inc. that allows to easily create and manage Quorum networks. The software helps users create Quorum nodes by inputting basic information and configuring the backend automatically. It is possible to create any number of Quorum nodes which can be distributed on separate Linux based machines or cloud instances for a production environment.

In addition to its easy to use wizard noninteractive setup interface, it also provides a web-based interface for managing and monitoring the Quorum network. As shown in Figure 3, the interface has an individual node monitor, a block and transaction explorer, as well as a graph, an online logs viewer, it can easily deploy Smart Contracts, and it allows to configure email notifications.
Since the platform is web-based, it can be viewed and used in any device with any operating system with an internet browser and an internet connection, or an internal network connection if it is not available to the public. It is also designed to be viewed and used in any type of mobile devices.
3 Equipment and Methods

3.1 Equipment

3.1.1 Local Nodes

**Machine:** Dell PowerEdge T30

**Processor:** Intel Xeon® CPU E3-1225 v5 @ 3.30GHz x 4

**RAM:** 32 GB

**Storage:** 1 TB

**OS:** Ubuntu 16.04.2 LTS x64-bit

<table>
<thead>
<tr>
<th>Machine</th>
<th>RAM</th>
<th>Storage</th>
<th>Network Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Node 1</td>
<td>32 GB</td>
<td>1 TB</td>
<td>Moderate</td>
</tr>
<tr>
<td>Node 2</td>
<td>32 GB</td>
<td>1 TB</td>
<td>Moderate</td>
</tr>
<tr>
<td>Node 3</td>
<td>32 GB</td>
<td>1 TB</td>
<td>Moderate</td>
</tr>
</tbody>
</table>

*Table 1: Desktop Servers used*

All of the Nodes use the same type of machine as well as the operating system, Ubuntu version 18.04.2 live server x64 without a graphical interface.

3.1.2 Virtual Machines

**Host Machine:** Dell PowerEdge T30

**Processor:** Intel Xeon® CPU E3-1225 v5 @ 3.30GHz x 4

**RAM:** 32 GB

**Storage:** 1 TB

**OS:** Ubuntu 16.04.2 LTS x64-bit

**VM Software:** Oracle VM VirtualBox Version 5.1.38_Ubuntu r122592
All Virtual Machine Instances have for the operating system Ubuntu version 18.04.2 live server x64 without a graphical interface.

### 3.1.3 Cloud Instances

<table>
<thead>
<tr>
<th>Instance</th>
<th>vCPU</th>
<th>RAM</th>
<th>Storage</th>
<th>Network Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Node 1 – t2.micro</td>
<td>1</td>
<td>1</td>
<td>30 GB</td>
<td>Moderate</td>
</tr>
<tr>
<td>Node 2 – t2.micro</td>
<td>1</td>
<td>1</td>
<td>30 GB</td>
<td>Moderate</td>
</tr>
<tr>
<td>Node 3 – t2.micro</td>
<td>1</td>
<td>1</td>
<td>30 GB</td>
<td>Moderate</td>
</tr>
</tbody>
</table>

All cloud Instances have for the operating system Ubuntu version 18.04 LTS without a graphical interface.

### 3.2 Configurations

#### 3.2.1 Operating System

For the different types of machines, physical desktop servers, and virtual machines, the installation process of the OS is pretty much set to the normal installation procedures. The OS used is the Ubuntu Live Server version 18.04 LTS

The cloud-based instances are pretty much using the same Ubuntu Server 18.04 LTS version.

I began preparing the OS in all machines by installing the following libraries:

- Build-essential
- Docker-CE
• Python3
• Pip3
• Go version 1.11.4

Once those libraries and programs were installed, I continued to install and set up Quorum Maker.

3.2.2 Quorum Maker Installation

The following installations were done specifically on local desktop servers.

3.2.2.1 Node 1 (Main Node)

To install Quorum Maker the source code has to be cloned into the machine from GitHub, next installation script has to be run, then in the initial menu the first option “1) Create Network” is selected, since it is the first node to be created in the network, and the following parameters were configured for the initial node:

• **Node name**: Node1
• **Node’s IP Address**: 10.5.65.99
• **Node’s RPC Port**: 22000
• **Node’s Network Listening Port**: 22001
• **Node’s Constellation Port**: 22002
• **Node’s Raft Port**: 22003
• **Node’s Node Manager Port**: 22004
• **Node’s WS Port**: 22005

Once the parameters are configured, the program sets up and deploys the first node, and the address for the web-based UI is given. Figure 4 shows an example of how the first node was configured in Quorum Maker.

Within the same network, the Web UI can be accessed by simply writing the IP address of the node, including the Node Manager port. In this case, the UI can be viewed by entering 10.5.65.99:22004 in any web browser.
3.2.2.2 Additional nodes (Nodes 2 and 3)

For the rest of the nodes similar steps are taken with a few changes, since the nodes are going to join an existing network, in the setup menu we select the second option “2) Join Network”, this option requires a few more parameters to be configured. In the networks being tested, I configured the nodes with the following parameters:

Node 2:

- **Node Name**: Node2
- **Existing Node’s IP Address**: 10.5.65.99
- **Existing Node’s Node Manager Port**: 22004
- **New Node’s IP Address**: 10.5.65.96
- **New Node’s RPC Port**: 24000
- **New Node’s Network Listening Port**: 24001
- **New Node’s Constellation Port**: 24002
- **New Node’s Raft Port**: 24003
- **New Node’s Node Manager Port**: 24004
- **Node’s WS Port**: 24005
Node 3:

- **Node Name:** Node3
- **Existing Node’s IP Address:** 10.5.65.99
- **Existing Node’s Node Manager Port:** 22004
- **New Node’s IP Address:** 10.5.65.89
- **New Node’s RPC Port:** 24000
- **New Node’s Network Listening Port:** 24001
- **New Node’s Constellation Port:** 24002
- **New Node’s Raft Port:** 24003
- **New Node’s Node Manager Port:** 24004
- **Node’s WS Port:** 24005

Once the parameters are configured, a request is sent to the main or first node which it is necessary to accept or reject if the connection is not recognized. Figure 5 shows the Accept/Reject request.

![Figure 5: Accept/Reject Request](image)

Figure 6 demonstrates the set up for Node2, which is the same for the rest of the nodes in the network, with the only change being the IP address for each node.
Each node will have an individual web UI, which will share the same information.

### 3.2.3 Chainhammer Installation

Once the Quorum nodes are working and creating transactions, the benchmarking software can be configured to run. For this step, the following procedures were made:

1. Clone the project from the Chainhammer GitHub site and follow the steps for preparing the scripts.
2. Go to the cloned folder and run the following command “scripts/install.sh”. This set of scripts will install additional packages and software which makes chainhammer return graphical results.
3. The Chainhammer website suggests the following:
   a. Run a virtual environment with “virtualenv”
   b. Assign the environment “source env/bin/activate”
   c. Goto the “hammer” folder and run the simply run “tps.py” and “send.py” with the required parameters to obtain the first results.
4. For any of the machines configured for this analysis, it is necessary to configure the file “configure.py”. There is a parameter called RPCaddress which needs to be changed to the address and port for the first node. In this test case, it was changed to “http://10.5.65.99:22000”, additionally change EMPTY_BLOCKS_AT_END = 0
5. Tests are now ready to be executed and recorded with Quorum Maker and chainhammer.
3.3 Methods

3.3.1 TPS Benchmarking

Transactions per second or TPS is one of the main features the blockchain Quorum offers as an improvement from the Ethereum network. The Raft consensus algorithm is what makes the platform send hundreds of transactions per second; the algorithm helps the transactions to have fast blocktimes, which means to have faster TPS. To prevent the creation of enormous amounts of blocks, Quorum creates a new block immediately after a new transaction arrives but not before waiting for at least 50 milliseconds since the previous block arrival.

For the tests being done to the different types of machines, as mentioned in section 3.1 and 3.2, the chainhammer benchmarking scripts have been used. Different transactions were sent using blocks and non-blocks, single threads, and multithreaded transactions. For every test, the transactions were reset to 0, and the Raft consensus was used in all the networks.

Each type of network has 3 sets of transactions, the first sending 100 transactions, the second with 1000 transactions and finally the third 10000 transactions. For each amount of transactions, they are sent with 10 threads, 30 threads, 50 threads and finally 100 threads.

To obtain the TPS of a block, the number of transactions sent in the block is divided by the time the block took from start to finish; this result is multiplied by 1000.

To calculate the Average TPS, the sum of all the block’s TPS is obtained then divided by the number of blocks which were obtained.

Table 4 shows how the calculations for obtaining TPS and AVERAGE TPS are done.

<table>
<thead>
<tr>
<th>Block #</th>
<th># of TXs</th>
<th>ms</th>
<th>TPS</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>156</td>
<td>400</td>
<td>390.00</td>
</tr>
<tr>
<td>25</td>
<td>204</td>
<td>450</td>
<td>453.33</td>
</tr>
<tr>
<td>33</td>
<td>150</td>
<td>400</td>
<td>375.00</td>
</tr>
<tr>
<td>42</td>
<td>165</td>
<td>450</td>
<td>366.67</td>
</tr>
<tr>
<td>49</td>
<td>166</td>
<td>351</td>
<td>472.93</td>
</tr>
<tr>
<td>54</td>
<td>98</td>
<td>250</td>
<td>392.00</td>
</tr>
<tr>
<td><strong>AVERAGE TPS</strong></td>
<td><strong>408.32</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 4: TPS Calculations Example
4 Results

4.1 Results Local Server Nodes

4.1.1 10 thread Results

There were 3 tests done with 10 threads, each having a similar result with a small difference in the TPS. Figure 7 shows information about the test done for 10 threads and 10000 transactions, with part of the block list shown.

![Figure 7: Results for 10 threads and 10000 Transactions Local Servers](image)

The results for the following tests are also supported by a graph which shows the calculated TPS average for the test.

- When sending 100 transactions with 10 threads, the TPS average is 153.14;
- When sending 1000 transactions with 10 threads, the TPS average is 690.00;
- When sending 10000 transactions with 10 threads, the TPS average is 686.97;

Figure 8 shows a comparison between the 3 averages.
4.1.2 30 thread Results

There were 3 tests done with 30 threads, each having a similar result with small differences in the TPS values. Figure 9 shows information about the test done for 30 threads and 100 transactions, including the block list.

![Figure 8: Local Server Nodes - 10 Threads Comparison](image)

![Figure 9: Results for 30-thread 100-transaction Test Local Servers](image)

The results for the following tests are also supported by a graph which shows the calculated TPS average for the test.

Then sending 100 transactions with 30 threads the TPS average is 700.00;

When sending 1000 transactions with 30 threads, the TPS average is 690.15;

When sending 10000 transactions with 30 threads, the TPS average is 693.24;

Figure 10 shows a comparison between the 3 averages.
4.1.3 50 thread Results

There were 3 tests done with 50 threads, each having a similar result with small differences in the TPS values. Figure 11 shows information about the test done for 50 threads and 1000 transactions, including the partial block list.

![Local Server Nodes - 30 Threads - TPS Average](image)

**Figure 10:** Local Server Nodes - 30 Threads Comparison

**Figure 11:** Results for 50-thread 1000-transaction Test Local Servers

The results for the following tests are also supported by a graph which shows the calculated TPS average for the test.

Then sending 100 transactions with 50 threads the TPS average is 603.66;
When sending 1000 transactions with 50 threads, the TPS average is 684.80;
When sending 10000 transactions with 50 threads, the TPS average is 699.50;
Figure 12 shows a comparison between the 3 averages.

![Figure 12: Local Server Nodes - 50 Threads Comparison](image.png)

4.1.4 100 thread Results

There were 3 tests done with 100 threads, each having a similar result with small differences in the TPS values. Figure 13 shows information about the test done for 100 threads and 100 transactions, including the block list.

(Quorum v1.8.12 with 100 txs: 580.1 TPS)

The results for the following tests are also supported by a graph which shows the calculated TPS average for the test.

Then sending 100 transactions with 100 threads the TPS average is 149.48

When sending 1000 transactions with 100 threads, the TPS average is 705.00;

When sending 10000 transactions with 100 threads, the TPS average is 704.14;
Figure 14 shows a comparison between the 3 averages.

Figure 14: Local Server Nodes - 30 Threads Comparison

Figure 15 demonstrates the similarity of TPS averages between the different amount of transactions, this way we understand that TPS maintain speeds in all amounts of transactions. Blocks of 100 or less transactions are difficult to average since the is either one block or maximum 3.

Figure 15: Local Server - All Threads and all TX compared
4.2 Results Virtual Machines

4.2.1 10 thread Results

There were 3 tests done with 10 threads, each having a similar result with small differences in the TPS values. Figure 16 shows information about the test done for 10 threads and 100 transactions, including the block list.

(VM_Quorum_Dan_Thesis_10threads_100txs) Quorum v1.8.12 with 100 txs: 372.6 TPS

The results for the following tests are also supported by a graph which shows the calculated TPS average for the test.

Then sending 100 transactions with 10 threads the TPS average is 257.43;

When sending 1000 transactions with 10 threads, the TPS average is 483.96;

When sending 10000 transactions with 10 threads, the TPS average is 418.01;

Figure 17 shows a comparison between the 3 averages.
4.2.2 30 thread Results

There were 3 tests done with 30 threads, each having a similar result with small differences in the TPS values. Figure 18 shows information about the test done for 30 threads and 1000 transactions, including the partial block list.

(VM_Quorum_Dan_Thesis_30threads_1000txs) Quorum v1.8.12 with 1000 txs: 395.2 TPS

The results for the following tests are also supported by a graph which shows the calculated TPS average for the test.

When sending 100 transactions with 30 threads, the TPS average is 151.52;
When sending 1000 transactions with 30 threads, the TPS average is 393.33;
When sending 10000 transactions with 30 threads, the TPS average is 388.00;
Figure 19 shows a comparison between the 3 averages.

![Figure 19: VM Nodes - 30 Threads Comparison](image)

4.2.3 50 thread Results

There were 3 tests done with 50 threads, each having a similar result with small differences in the TPS values. Figure 20 shows information about the test done for 50 threads and 10000 transactions, including the partial block list.

*(VM_Quorum_Dan_Thesis_50threads_10000txs)* Quorum v1.8.12 with 10000 txs: 422.0 TPS

| TPS Average | 
|-------------|-----------|
| 60.00       |
| 47.99       |

Figure 20: Results for 50-thread 10000-transaction Test Virtual Machines

The results for the following tests are also supported by a graph which shows the calculated TPS average for the test.

When sending 100 transactions with 50 threads, the TPS average is 60.00;

When sending 1000 transactions with 50 threads, the TPS average is 479.00
When sending 10000 transactions with 50 threads, the TPS average is 421.56;

Figure 21 shows a comparison between the 3 averages.

![Figure 21: VM Nodes - 50 Threads Comparison](image)

### 4.2.4 100 thread Results

There were 3 tests done with 100 threads, each having a similar result with small differences in the TPS values. Figure 22 shows information about the test is done for 100 threads and 10000 transactions, including the partial block list.

***Quorum v1.8.12 with 10000 txs: 447.7 TPS***

![Figure 22: Results for 100-thread 10000-transaction Test Virtual Machines](image)
The results for the following tests are also supported by a graph which shows the calculated TPS average for the test.

When sending 100 transactions with 100 threads, the TPS average is 383.84;
When sending 1000 transactions with 100 threads, the TPS average is 399.04;
When sending 10000 transactions with 100 threads, the TPS average is 447.61;
Figure 23 shows a comparison between the 3 averages.

![Figure 23: VM Nodes - 10 Threads Comparison](image)

Figure 24 demonstrates the similarity of TPS averages between the different amount of transactions, this way we understand that TPS maintain speeds in all amounts of transactions. Blocks of 100 or less transactions are difficult to average since the is either one block or maximum 3.

![Figure 24: VM Nodes - All Threads and TXs Compared](image)
4.3 Results Cloud Instances

4.3.1 10 threads Results

There were 3 tests done with 10 threads, each having a similar result with small differences in the TPS values. Figure 25 shows information about the test done for 10 threads and 1000 transactions, including the partial block list.

Figure 25: Results for 10-thread 1000-transaction Test Cloud Instances

The results for the following tests are also supported by a graph which shows the calculated TPS average for the test.

When sending 100 transactions with 100 threads, the TPS average is 236.78;

When sending 1000 transactions with 100 threads, the TPS average is 262.15;

When sending 10000 transactions with 100 threads, the TPS average is 228.47;

Figure 26 shows a comparison between the 3 averages.
4.3.2 30 threads Results

There were 3 tests done with 30 threads, each having a similar result with small differences in the TPS values. Figure 27 shows information about the test done for 30 threads and 100 transactions, including the block list.

(Quorum-Dan-Thesis-30thread-100tx) Quorum v1.8.12 with 100 txs: 245.4 TPS

The results for the following tests are also supported by a graph which shows the calculated TPS average for the test.

When sending 100 transactions with 30 threads, the TPS average is 245.50;
When sending 1000 transactions with 30 threads, the TPS average is 254.79;
When sending 10000 transactions with 30 threads, the TPS average is 219.75;
Figure 28 shows a comparison between the 3 averages.
There were 3 tests done with 50 threads, each having a similar result with small differences in the TPS values. Figure 29 shows information about the test done for 50 threads and 10000 transactions, including the partial block list.

(Qorum-Dan-Thesis-50thread-10000tx) Qorum v1.8.12 with 10000 txs: 231.2 TPS

The results for the following tests are also supported by a graph which shows the calculated TPS average for the test.

The results for the following tests are also supported by a graph which shows the calculated TPS average for the test.
When sending 100 transactions with 50 threads, the TPS average is 233.33;

When sending 1000 transactions with 50 threads, the TPS average is 246.57;

When sending 10000 transactions with 50 threads, the TPS average is 231.00;

Figure 30 shows a comparison between the 3 averages.

4.3.4 100 threads Results

There were 3 tests done with 100 threads, each having a similar result with small differences in the TPS values. Figure 31 shows information about the test done for 100 threads and 10000 transactions, including the partial block list.

![Figure 30: Cloud Nodes - 50 Threads Comparison](image)

### Quorum v1.8.12 with 10000 txx: 245.3 TPS

**Information:**

**Log:**

*version: v1.8.12-stable refusal
first account of node (0x6ECD7F333597276CKS2654C660E20874F247MB)
connection established, blockhash = 2, node version string = 0x6ECD7F333597276CKS2654C660E20874F247MB
raft consensus did not return time is nanoseconds. Is it still the case?*

*Quorum v1.8.12-stable, consensus: raft, network: 50706, channel: 111, channel: -1*

![Figure 31: Results for 100-thread 10000-transaction Test Cloud Instances](image)
The results for the following tests are also supported by a graph which shows the calculated TPS average for the test.

When sending 100 transactions with 50 threads, the TPS average is 91.74;
When sending 1000 transactions with 50 threads, the TPS average is 261.98;
When sending 10000 transactions with 50 threads, the TPS average is 245.28;

Figure 32 shows a comparison between the 3 averages.

![Cloud Nodes - 10 Threads - TPS Average](image)

Figure 32: Cloud Nodes - 100 Threads Comparison

Figure 33 demonstrates the similarity of TPS averages between the different amount of transactions, this way we understand that TPS maintain speeds in all amounts of transactions. Blocks of 100 or less transactions are difficult to average since the is either one block or maximum 3.
Figure 33: Cloud Nodes - All Threads and TXs Compared
5 Conclusion

When trying to decide what technology to use specialists often do some research on the different platforms, software, or even hardware available in the market, to be used in the present and foresee the future use and relevance of the product for the company or institution. In the case of this document, it focused on analyzing the speed efficiency of the Quorum Blockchain platform in different types of environments.

After learning about the Quorum Blockchain platform, how it works, how certain parameters can be measured, installing and configuring many nodes in various types of machines and environments and finally being able to perform the stress tests and benchmarks for the 3 types of environments, Local Server Nodes, Virtual Servers Nodes, and Cloud Instance Nodes, it is possible to analyze the TPS differences with each type of environment.

<table>
<thead>
<tr>
<th>Machine Types</th>
<th>Local Server</th>
<th>Virtual Machines</th>
<th>Cloud Instances</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>10 Threads</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>100 tx</td>
<td>153.14</td>
<td>257.43</td>
<td>236.78</td>
</tr>
<tr>
<td>1000 tx</td>
<td>690.00</td>
<td>483.96</td>
<td>262.15</td>
</tr>
<tr>
<td>10000 tx</td>
<td>686.97</td>
<td>418.01</td>
<td>228.47</td>
</tr>
<tr>
<td><strong>30 Threads</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>100 tx</td>
<td>700.00</td>
<td>151.52</td>
<td>242.50</td>
</tr>
<tr>
<td>1000 tx</td>
<td>690.15</td>
<td>393.33</td>
<td>254.79</td>
</tr>
<tr>
<td>10000 tx</td>
<td>693.24</td>
<td>388.00</td>
<td>219.75</td>
</tr>
<tr>
<td><strong>50 Threads</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>100 tx</td>
<td>603.66</td>
<td>60.00</td>
<td>233.33</td>
</tr>
<tr>
<td>1000 tx</td>
<td>684.80</td>
<td>479.00</td>
<td>246.57</td>
</tr>
<tr>
<td>10000 tx</td>
<td>699.50</td>
<td>421.56</td>
<td>231.00</td>
</tr>
<tr>
<td><strong>100 Threads</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>100 tx</td>
<td>149.48</td>
<td>383.84</td>
<td>91.74</td>
</tr>
<tr>
<td>1000 tx</td>
<td>705.00</td>
<td>399.04</td>
<td>260.98</td>
</tr>
<tr>
<td>10000 tx</td>
<td>407.14</td>
<td>447.61</td>
<td>245.28</td>
</tr>
</tbody>
</table>

Table 5: TPS Average Table

Table 5 and Figure 34 are good references for obtaining a clear decision on which environment is more likely to be used or chosen because of speed and performance. The local server nodes results outperform the other types of environments in every test done. It is a possibility that it has better performance since the network used is local, unlike the cloud instances which are given by the provider. Another possibility of its better performance could be the fact that the local server is using relatively better processor speed than any other type of nodes, and this could have helped for the faster transfer times as well.
Finally, it would be of best use to continue testing the Quorum Blockchain platform with more nodes, more threads, more transactions and improving the types of equipment or technology to obtain potentially better results in all types of environments. Hopefully, future tests won’t present more difficulties caused by software updates, not only from the OS, but also from the different packages and software which Quorum, Quorum Make, and Chainhammer use. This has been a huge setback throughout the time these tests have been performed.

All the results are included in the annexes section.
6 References


7 Annexes

7.1 Result Reports

7.1.1 Local Servers
Quorum v1.8.12 with 100 txs: 592.9 TPS

information:

NODE: Geth/v1.8.12-stable/linux-amd64/go1.10.2 on http://10.5.65.99:22000
consensus=raft chain_name=??? chain_id=-1 network_id=14761
SEND: 100 transactions in Blocks 4-8 with 0 empty blocks following.
   A sample of transactions looked as if they: succeeded.
TPS: The stopclock watcher measured a final TPS of 159.1 since contract deploy,
and in between saw values as high as 0.0 TPS.
DIAG: The whole experiment was prefixed 'Local_Quorum_dan_Thesis_10threads_100TX'.
   The diagrams were saved into 'img/Local_Quorum_dan_Thesis_10threads_100TX-20190520-1546_blks4-8.png'.
   Looking only at the experiment block-timestamps, the overall TPS was ~592.9.

log:

versions: web3 4.3.0, py-solc: 2.1.0, solc 0.4.25+commit.59db8f8f1.Linux.gpp, testrpc 1.3.4, python 3.6.7 (default, Oct 22 2018, 11:32:17) [GCC 8.2.0]
web3 connection established, blockNumber = 2, node version string = Geth/v1.8.12-stable/linux-amd64/go1.10.2
first account of node is 0x0e19164781A2aECfd6B0B5EABD0CCaBb529Fb0e, balance is 1000000000 Ether
WARN: raft consensus did report timestamps in nanoseconds. Is that still the case?
Block 2 - waiting for something to happen
[filedate 1558366953] last contract address: 0x2B4F4260c2BcD86E732c1fAf11DFe99aee6e3562
[filedate 1558367183] new contract address: 0x1E0859f3F2d07E59E16a973258Cf9C8B7cD75e8d
block_number_start_here = 3
starting timer, at block 3 which has 1 transactions; at epochTime 1558367183.448804
block 8 | new #TX 100 / 653 ms = 153.1 TPS_current | total: #TX 101 / 0.6 s = 159.1 TPS_average (peak is 159.1 TPS_average)
Received signal from send.py = updated INFOFILE.
Experiment ended! Current blocknumber = 8
Updated info file: last-experiment.json THE END.

diagrams:
Local_Quorum_dan_Thesis_10threads_100TX blocks 4-8 with 2 txs ~ 0 txs/block

avg TPS 592.9 = #TX whole experiment / blocktimes diff

blocktime seconds since last block

gasUsed and gasLimit per second
Quorum v1.8.12 with 1000 txs: 690.6 TPS

information:

NODE: Geth/v1.8.12-stable/linux-amd64/go1.10.2 on http://10.5.65.99:22000
consensus=raft chain_name=??? chain_id=-1 network_id=26318
SEND: 1000 transactions in blocks 4-34 with 0 empty blocks following.
  A sample of transactions looked as if they: succeeded.
TPS: The stopclock watcher measured a final TPS of 518.5 since contract deploy,
  and in between saw values as high as 524.2 TPS.
DIAG: The whole experiment was prefixed 'Local_Quorum_dan_Thesis_10threads_1000TX'.
  The diagrams were saved into 'img/Local_Quorum_dan_Thesis_10threads_1000TX-20190520-1547_blks4-34.png'.
  Looking only at the experiment block-timestamps, the overall TPS was ~690.6.

log:

versions: web3 4.3.0, py-solc: 2.1.0, solc 0.4.25+commit.59db8ff1.Linux.gpp, testrpc 1.3.4, python 3.6.7 (default, Oct 22 2018, 11:32:17) [GCC 8.2.0]
web3 connection established, blockNumber = 2, node version string = Geth/v1.8.12-stable/linux-amd64/go1.10.2
first account of node is 0xce980Cc28905e1C2CCbB16531437Bcf9C807cD75e0d, balance is 1000000000 Ether
WARN: raft consensus did report timestamps in nanoseconds. Is that still the case?
nodeName: Quorum, nodeType: Geth, nodeVersion: v1.8.12-stable, consensus: raft, network: 26318, chainName: ???, chainId: -1

Block  2  - waiting for something to happen
[filedate 1558367183] last contract address: 0x1E0859f3F2d07E59E16a973258Cf9C807cD75e0d
[filedate 1558367271] new contract address: 0x74A242a077890c93e5cfD3fC476B41b68F326f60
blocknumber_start_here = 3
starting timer, at block 3 which has 1 transactions; at epochtime 1558367271.6160944
block 10  | new #TX 172 / 710 ms = 242.4 TPS_current | total: #TX 173 / 0.6 s = 277.3 TPS_average (peak is 277.3 TPS_average)
block 16  | new #TX 263 / 300 ms = 669.5 TPS_current | total: #TX 374 / 0.9 s = 394.9 TPS_average (peak is 394.9 TPS_average)
block 22  | new #TX 222 / 300 ms = 740.5 TPS_current | total: #TX 596 / 1.3 s = 469.4 TPS_average (peak is 469.4 TPS_average)
block 29  | new #TX 239 / 350 ms = 682.9 TPS_current | total: #TX 835 / 1.6 s = 524.2 TPS_average (peak is 524.2 TPS_average)
block 34  | new #TX 166 / 250 ms = 664.0 TPS_current | total: #TX 1001 / 1.9 s = 518.5 TPS_average (peak was 524.2 TPS_average)
Received signal from send.py = updated INFOFILE.
Experiment ended! Current blocknumber = 34
Updated info file: last-experiment.json THE END.

diagrams:
Local Quorum dan Thesis 10threads 1000TX blocks 4-34 with 1 txs ~ 0 txs/block

avg TPS 690.6 = #TX whole experiment / blocktimes diff

blocktimesTpsAv: 690.5602548314879
filename: Local_Quorum_dan_Thesis_10threads_1000TX-20190520-1547_blks4-34.png
prefix: Local_Quorum_dan_Thesis_10threads_1000TX
node: chain_id: -1
chain_name: ???
consensus: raft
name: Quorum
network_id: 26318
rpc_address: http://10.5.65.99:22000
type: Geth
version: v1.8.12-stable
web3.version.node: Geth/v1.8.12-stable/linux-amd64/go1.10.2
send: block_first: 4
block_last: 34
empty_blocks: 0
num_txs: 1000
sample_txs_successful: True
info raw

<table>
<thead>
<tr>
<th>Block</th>
<th>#TX</th>
<th>TPS Current</th>
<th>TPS Average</th>
<th>Peak TPS Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>69</td>
<td>232</td>
<td>680.1</td>
<td>627.9</td>
<td>627.8</td>
</tr>
<tr>
<td>62</td>
<td>218</td>
<td>657.1</td>
<td>644.8</td>
<td>644.8</td>
</tr>
<tr>
<td>56</td>
<td>250</td>
<td>705.7</td>
<td>714.3</td>
<td>714.3</td>
</tr>
<tr>
<td>50</td>
<td>247</td>
<td>712.0</td>
<td>734.0</td>
<td>734.0</td>
</tr>
<tr>
<td>49</td>
<td>217</td>
<td>720.2</td>
<td>723.3</td>
<td>723.3</td>
</tr>
<tr>
<td>43</td>
<td>211</td>
<td>703.3</td>
<td>706.6</td>
<td>706.6</td>
</tr>
<tr>
<td>37</td>
<td>249</td>
<td>711.4</td>
<td>711.4</td>
<td>711.4</td>
</tr>
<tr>
<td>30</td>
<td>241</td>
<td>688.6</td>
<td>700.0</td>
<td>700.0</td>
</tr>
<tr>
<td>23</td>
<td>216</td>
<td>720.2</td>
<td>726.7</td>
<td>726.7</td>
</tr>
<tr>
<td>17</td>
<td>246</td>
<td>702.7</td>
<td>709.5</td>
<td>709.5</td>
</tr>
<tr>
<td>10</td>
<td>142</td>
<td>205.6</td>
<td>205.6</td>
<td>205.6</td>
</tr>
<tr>
<td>94</td>
<td>214</td>
<td>717.1</td>
<td>640.5</td>
<td>640.5</td>
</tr>
<tr>
<td>91</td>
<td>213</td>
<td>711.3</td>
<td>639.6</td>
<td>639.6</td>
</tr>
<tr>
<td>88</td>
<td>215</td>
<td>705.7</td>
<td>637.9</td>
<td>637.9</td>
</tr>
<tr>
<td>82</td>
<td>247</td>
<td>705.7</td>
<td>637.9</td>
<td>637.9</td>
</tr>
<tr>
<td>75</td>
<td>228</td>
<td>680.1</td>
<td>627.9</td>
<td>627.9</td>
</tr>
<tr>
<td>69</td>
<td>223</td>
<td>688.6</td>
<td>628.8</td>
<td>628.8</td>
</tr>
<tr>
<td>66</td>
<td>250</td>
<td>705.7</td>
<td>664.8</td>
<td>664.8</td>
</tr>
<tr>
<td>65</td>
<td>246</td>
<td>707.9</td>
<td>664.8</td>
<td>664.8</td>
</tr>
<tr>
<td>62</td>
<td>232</td>
<td>717.0</td>
<td>667.2</td>
<td>667.2</td>
</tr>
<tr>
<td>59</td>
<td>244</td>
<td>708.0</td>
<td>667.2</td>
<td>667.2</td>
</tr>
<tr>
<td>58</td>
<td>245</td>
<td>708.0</td>
<td>667.2</td>
<td>667.2</td>
</tr>
<tr>
<td>52</td>
<td>232</td>
<td>702.7</td>
<td>667.2</td>
<td>667.2</td>
</tr>
<tr>
<td>49</td>
<td>224</td>
<td>688.6</td>
<td>667.2</td>
<td>667.2</td>
</tr>
<tr>
<td>46</td>
<td>221</td>
<td>711.3</td>
<td>667.2</td>
<td>667.2</td>
</tr>
<tr>
<td>43</td>
<td>220</td>
<td>705.7</td>
<td>667.2</td>
<td>667.2</td>
</tr>
<tr>
<td>41</td>
<td>217</td>
<td>717.0</td>
<td>667.2</td>
<td>667.2</td>
</tr>
<tr>
<td>39</td>
<td>216</td>
<td>717.0</td>
<td>667.2</td>
<td>667.2</td>
</tr>
<tr>
<td>37</td>
<td>215</td>
<td>717.0</td>
<td>667.2</td>
<td>667.2</td>
</tr>
<tr>
<td>36</td>
<td>214</td>
<td>717.0</td>
<td>667.2</td>
<td>667.2</td>
</tr>
<tr>
<td>35</td>
<td>213</td>
<td>717.0</td>
<td>667.2</td>
<td>667.2</td>
</tr>
<tr>
<td>34</td>
<td>212</td>
<td>717.0</td>
<td>667.2</td>
<td>667.2</td>
</tr>
<tr>
<td>33</td>
<td>211</td>
<td>717.0</td>
<td>667.2</td>
<td>667.2</td>
</tr>
<tr>
<td>32</td>
<td>210</td>
<td>717.0</td>
<td>667.2</td>
<td>667.2</td>
</tr>
<tr>
<td>31</td>
<td>209</td>
<td>717.0</td>
<td>667.2</td>
<td>667.2</td>
</tr>
<tr>
<td>30</td>
<td>208</td>
<td>717.0</td>
<td>667.2</td>
<td>667.2</td>
</tr>
<tr>
<td>29</td>
<td>207</td>
<td>717.0</td>
<td>667.2</td>
<td>667.2</td>
</tr>
<tr>
<td>28</td>
<td>206</td>
<td>717.0</td>
<td>667.2</td>
<td>667.2</td>
</tr>
<tr>
<td>27</td>
<td>205</td>
<td>717.0</td>
<td>667.2</td>
<td>667.2</td>
</tr>
<tr>
<td>26</td>
<td>204</td>
<td>717.0</td>
<td>667.2</td>
<td>667.2</td>
</tr>
<tr>
<td>25</td>
<td>203</td>
<td>717.0</td>
<td>667.2</td>
<td>667.2</td>
</tr>
<tr>
<td>24</td>
<td>202</td>
<td>717.0</td>
<td>667.2</td>
<td>667.2</td>
</tr>
<tr>
<td>23</td>
<td>201</td>
<td>717.0</td>
<td>667.2</td>
<td>667.2</td>
</tr>
<tr>
<td>22</td>
<td>200</td>
<td>717.0</td>
<td>667.2</td>
<td>667.2</td>
</tr>
<tr>
<td>21</td>
<td>199</td>
<td>717.0</td>
<td>667.2</td>
<td>667.2</td>
</tr>
<tr>
<td>20</td>
<td>198</td>
<td>717.0</td>
<td>667.2</td>
<td>667.2</td>
</tr>
<tr>
<td>19</td>
<td>197</td>
<td>717.0</td>
<td>667.2</td>
<td>667.2</td>
</tr>
<tr>
<td>18</td>
<td>196</td>
<td>717.0</td>
<td>667.2</td>
<td>667.2</td>
</tr>
<tr>
<td>17</td>
<td>195</td>
<td>717.0</td>
<td>667.2</td>
<td>667.2</td>
</tr>
<tr>
<td>16</td>
<td>194</td>
<td>717.0</td>
<td>667.2</td>
<td>667.2</td>
</tr>
<tr>
<td>15</td>
<td>193</td>
<td>717.0</td>
<td>667.2</td>
<td>667.2</td>
</tr>
<tr>
<td>14</td>
<td>192</td>
<td>717.0</td>
<td>667.2</td>
<td>667.2</td>
</tr>
<tr>
<td>13</td>
<td>191</td>
<td>717.0</td>
<td>667.2</td>
<td>667.2</td>
</tr>
<tr>
<td>12</td>
<td>190</td>
<td>717.0</td>
<td>667.2</td>
<td>667.2</td>
</tr>
<tr>
<td>11</td>
<td>189</td>
<td>717.0</td>
<td>667.2</td>
<td>667.2</td>
</tr>
<tr>
<td>10</td>
<td>188</td>
<td>717.0</td>
<td>667.2</td>
<td>667.2</td>
</tr>
<tr>
<td>9</td>
<td>187</td>
<td>717.0</td>
<td>667.2</td>
<td>667.2</td>
</tr>
<tr>
<td>8</td>
<td>186</td>
<td>717.0</td>
<td>667.2</td>
<td>667.2</td>
</tr>
<tr>
<td>7</td>
<td>185</td>
<td>717.0</td>
<td>667.2</td>
<td>667.2</td>
</tr>
<tr>
<td>6</td>
<td>184</td>
<td>717.0</td>
<td>667.2</td>
<td>667.2</td>
</tr>
<tr>
<td>5</td>
<td>183</td>
<td>717.0</td>
<td>667.2</td>
<td>667.2</td>
</tr>
<tr>
<td>4</td>
<td>182</td>
<td>717.0</td>
<td>667.2</td>
<td>667.2</td>
</tr>
<tr>
<td>3</td>
<td>181</td>
<td>717.0</td>
<td>667.2</td>
<td>667.2</td>
</tr>
<tr>
<td>2</td>
<td>180</td>
<td>717.0</td>
<td>667.2</td>
<td>667.2</td>
</tr>
<tr>
<td>1</td>
<td>179</td>
<td>717.0</td>
<td>667.2</td>
<td>667.2</td>
</tr>
<tr>
<td>0</td>
<td>178</td>
<td>717.0</td>
<td>667.2</td>
<td>667.2</td>
</tr>
</tbody>
</table>
block 282 | new #TX 251 / 350 ms = 717.1 TPS_current | total: #TX 9486 / 14.2 s = 666.4 TPS_average (peak is 666.4 TPS_average)
block 289 | new #TX 248 / 350 ms = 708.6 TPS_current | total: #TX 9734 / 14.6 s = 668.5 TPS_average (peak is 668.5 TPS_average)
block 296 | new #TX 240 / 350 ms = 685.7 TPS_current | total: #TX 9974 / 14.9 s = 670.1 TPS_average (peak is 670.1 TPS_average)
block 297 | new #TX 27 / 50 ms = 540.1 TPS_current | total: #TX 10001 / 15.2 s = 658.2 TPS_average (peak was 670.1 TPS_average)

Received signal from send.py = updated INFOFILE.
Experiment ended! Current blocknumber = 297
Updated info file: last-experiment.json THE END.

Received signal from send.py = updated INFOFILE.
Experiment ended! Current blocknumber = 297
Updated info file: last-experiment.json THE END.

diagrams:

img/Local_Quorum_dan_Thesis_10threads_10000TX-20190520-1549_blks4-297.png

Local_Quorum_dan_Thesis_10threads_1000TX blocks 4-297 with 1 txs ~ 0 txs/block
Quorum v1.8.12 with 100 txs: 707.2 TPS

**information:**

- **NODE:** Geth/v1.8.12-stable/linux-amd64/go1.10.2 on http://10.5.65.99:22000
- **consensus=raft**
- **chain_name=???, chain_id=-1**
- **network_id=61864**
- **SEND:** 100 transactions in Blocks 4-8 with 0 empty blocks following.
- **A sample of transactions looked as if they: succeeded.**
- **TPS:** The stopwatch watcher measured a final TPS of 158.8 since contract deploy, and in between saw values as high as 0.0 TPS.
- **DIAG:** The whole experiment was prefixed 'Local_Quorum_dan_Thesis_30threads_100TX'.
  - The diagrams were saved into 'img/Local_Quorum_dan_Thesis_30threads_100TX-20190520-1558_blks4-8.png'.
  - Looking only at the experiment block-timestamps, the overall TPS was ~707.2.

**log:**

- **versions:** web3 4.3.0, py-solc: 2.1.0, solc 0.4.25+commit.59db8ff1.Linux.gpp, testrpc 1.3.4, python 3.6.7 (default, Oct 22 2018, 11:32:17) [GCC 8.2.0]
- **web3 connection established, blockNumber = 2, node version string =** Geth/v1.8.12-stable/linux-amd64/go1.10.2

  first account of node is 0xef955b60858f6eC351C0fe70936e83D66Fa9B31c, balance is 1000000000 Ether

- **WARN:** raft consensus did report timestamps in nanoseconds. Is that still the case?

- **nodeName:** Quorum, **nodeType:** Geth, **nodeVersion:** v1.8.12-stable, **consensus:** raft, **network:** 61864, **chainName:** ???, **chainId:** -1

- **Block 2 - waiting for something to happen**

  (filedate 1558367344) last contract address: 0x82d6052F59ca1CdB99cB0b5440060137717237D5

  (filedate 1558367888) new contract address: 0x2e2fC1717A91430616d5585c080fA6e5D318Fec2

- **blocknumber_start_here = 3**

  starting timer, at block 3 which has 1 transactions; at epochtime 1558367888.1424377

- **block 5 | new #TX 2 / 469 ms = 4.3 TPS_current | total: #TX 3 / 0.3 s = 9.7 TPS_average (peak is 9.7 TPS_average)**

- **block 8 | new #TX 98 / 140 ms = 702.1 TPS_current | total: #TX 101 / 0.6 s = 158.8 TPS_average (peak is 158.8 TPS_average)**

- **Received signal from send.py = updated INFOFILE.**

  Experiment ended! Current blocknumber = 8

  Updated info file: last-experiment.json THE END.

**diagrams:**
Local Quorum dan Thesis 30threads 100TX blocks 4-8 with 1 txs ~ 0 txs/block

- **avg TPS 707.2 = #TX whole experiment / blocktimes diff**
  - TPS_1blk
  - TPS_3blks
  - TPS_5blks
  - TPS_10blks

- **blocktime seconds since last block**

- **blocksizes in bytes**
  - 2849

- **gasUsed and gasLimit per second**
  - gasLimit/sec
  - gasUsed/sec

---

**info raw**

```
info raw
{"diagrams": {"blockstampsTpsAv": 707.224226114948, "filename": "img/Local_Quorum_dan_Thesis_30threads_100TX-20190520-1558_blks4-8.png", "prefix": "Local_Quorum_dan_Thesis_30threads_100TX"}, "node": {"chain_id": -1, "chain_name": "??", "consensus": "raft", "name": "Quorum", "network_id": 61864, "rpc_address": "http://10.5.65.99:22000", "type": "Geth", "version": "v1.8.12-stable", "web3.version.node": "Geth/v1.8.12-stable/linux-amd64/go1.10.2"}, "send": {"block_first": 4, "block_last": 8, "empty_blocks": 0, "num_txs": 100, "sample_txs_successful": True}, "tps": {"finalTpsAv": 158.84174837672762, "peakTpsAv": 0, "start_epochtime": 1558367888.1424377}}
```
Quorum v1.8.12 with 1000 txs: 691.5 TPS

**information:**

**NODE:** Geth/v1.8.12-stable/linux-amd64/go1.10.2 on http://10.5.65.99:22000

**consensus=raft chain_name=?? chain_id=-1 network_id=49744**

**SEND:** 1000 transactions in blocks 4-34 with 0 empty blocks following.

A sample of transactions looked as if they: succeeded.

**TPS:** The stopclock watcher measured a final TPS of 522.6 since contract deploy, and in between saw values as high as 522.6 TPS.

**DIAG:** The whole experiment was prefixed 'Local_Quorum_dan_Thesis_30threads_1000TX'.

Looking only at the experiment block-timestamps, the overall TPS was ~691.5.

**log:**

versions: web3 4.3.0, py-solc: 2.1.0, solc 0.4.25+commit.59dbf8f1.Linux.gpp, testrpc 1.3.4, python 3.6.7 (default, Oct 22 2018, 11:32:17) [GCC 8.2.0]

First account of node is 0xe7a420958CF3E7903241d8A77d99Ea1A032821, balance is 1000000000 Ether

WARN: raft consensus did report timestamps in nanoseconds. Is that still the case?

**nodeName:** Quorum, **nodeType:** Geth, **nodeVersion:** v1.8.12-stable, **consensus:** raft, **network:** 49744, **chainName:** ???, **chainId:** -1

Block 2 - waiting for something to happen

blocknumber_start_here = 3

starting timer, at block 3 which has 1 transactions; at epochtime 1558367955.7400668

received signal from send.py = updated INFOFILE.
Local Quorum dan Thesis 30threads 1000TX blocks 4-34 with 1 txs ~ 0 txs/block

avg TPS 691.5 = #TX whole experiment / blocktimes diff

blocktime seconds since last block

gasUsed and gasLimit per second
(Local_Quorum_dan_Thesis_30threads_10000TX) Quorum v1.8.12 with 10000 txs: 693.9 TPS

information:

NODE: Geth/v1.8.12-stable/ios-amd64/gol1.10.2 on http://10.5.65.99:22000
consensus=raft chain_name=??? chain_id=1 network_id=99240
SEND: 10000 transactions in blocks 4-294 with 0 empty blocks following. A sample of transactions looked as if they succeeded.

TPS: The stopwatch measured a final TPS of 672.5 since contract deploy, and in between saw values as high as 674.8 TPS.

DIAG: The whole experiment was prefixed 'Local_Quorum_dan_Thesis_30threads_10000TX'. The diagrams were saved into 'img/Local_Quorum_dan_Thesis_30threads_10000TX-20190520-1600_blks4-294.png'. Looking only at the experiment block-times, the overall TPS was -693.9.

log:

versions: web3 4.3.0, py-solc: 2.1.8, solc 0.4.25+commit.59d8f8f1/linux.gpp, testrpc 1.3.4, python 3.6.7 (default, Oct 22 2018, 11:32:37) [GCC 8.2.0]
web3 connection established, blockNumber = 2, node version string = Geth/v1.8.12-stable/ios-amd64/gol1.10.2
first account of node is 0x822b21ad35D18F9cb402673f1285147da71478d7, balance is 1000000000 Ether
WARN: raft consensus did report timestamps in nanoseconds. Is that still the case?
Block 2 - waiting for something to happen
(filedate 1558367955) last contract address: 0x0120fD229edA3bB82fEEF24397F9f0836a1476125
(filedate 1558368026) new contract address: 0xc92127Ce479999b3AB9150027cCD9805D1E15D83
blocknumber_start_here = 3
starting timer, at block 3 which has 1 transactions; at epoch time 1558368026.7513855

block 10     new #TX 156  /  698 ms = 223.4 TPS_current
block 16     new #TX 209  /  300 ms = 696.6 TPS_current
block 22     new #TX 211  /  300 ms = 783.4 TPS_current
block 29     new #TX 245  /  350 ms = 700.1 TPS_current
block 36     new #TX 249  /  352 ms = 707.2 TPS_current
block 42     new #TX 285  /  384 ms = 673.8 TPS_current
block 49     new #TX 233  /  344 ms = 677.9 TPS_current
block 55     new #TX 221  /  300 ms = 736.7 TPS_current
block 62     new #TX 253  /  350 ms = 722.9 TPS_current
block 69     new #TX 244  /  350 ms = 697.1 TPS_current
block 75     new #TX 219  /  300 ms = 730.0 TPS_current
block 81     new #TX 209  /  300 ms = 696.7 TPS_current
block 87     new #TX 222  /  300 ms = 740.0 TPS_current
block 93     new #TX 280  /  301 ms = 665.5 TPS_current
block 99     new #TX 199  /  299 ms = 664.5 TPS_current
block 107    new #TX 292  /  480 ms = 738.0 TPS_current
block 114    new #TX 239  /  350 ms = 682.9 TPS_current
block 120    new #TX 225  /  350 ms = 749.7 TPS_current
block 127    new #TX 244  /  350 ms = 697.4 TPS_current
block 133    new #TX 211  /  300 ms = 703.3 TPS_current
block 139    new #TX 215  /  300 ms = 716.7 TPS_current
block 142    new #TX 185  /  150 ms = 609.9 TPS_current
block 151    new #TX 254  /  450 ms = 564.5 TPS_current
block 159    new #TX 246  /  480 ms = 615.0 TPS_current
block 161    new #TX 69   /  100 ms = 608.0 TPS_current
block 172    new #TX 387  /  550 ms = 783.6 TPS_current
block 179    new #TX 221  /  352 ms = 628.3 TPS_current
block 185    new #TX 218  /  296 ms = 703.7 TPS_current
block 192    new #TX 241  /  350 ms = 688.9 TPS_current
block 198    new #TX 222  /  300 ms = 748.0 TPS_current
block 204    new #TX 210  /  300 ms = 780.0 TPS_current
block 212    new #TX 225  /  350 ms = 750.8 TPS_current
block 224    new #TX 415  /  682 ms = 689.2 TPS_current
block 237    new #TX 251  /  350 ms = 717.1 TPS_current
block 243    new #TX 211  /  300 ms = 701.6 TPS_current
block 249    new #TX 192  /  293 ms = 641.7 TPS_current
block 256    new #TX 244  /  350 ms = 639.9 TPS_current
block 262    new #TX 214  /  350 ms = 646.6 TPS_current
block 269    new #TX 248  /  350 ms = 685.8 TPS_current
block 276    new #TX 236  /  350 ms = 674.3 TPS_current

Local_Quorum_dan_Thesis_30threads_10000TX) Quorum v1.8.12 with 10000 txs: 693.9 TPS

Looking only at the experiment block-times, the overall TPS was -693.9.
Received signal from send.py = updated INFOFILE.
Experiment ended! Current blocknumber = 294
Updated info file: last-experiment.json THE END.

diagrams:

![Local_Quorum_dan_Thesis_30threads_10000TX-20190520-1600_blks4-294.png](img/Local_Quorum_dan_Thesis_30threads_10000TX-20190520-1600_blks4-294.png)

Local_Quorum_dan_Thesis_30threads_10000TX blocks 4-294 with 1 txs ~ 0 txs/block

avg TPS 693.9 = #TX whole experiment / blocktimes diff

blocktime seconds since last block

gasUsed and gasLimit per second
Quorum v1.8.12 with 100 txs: 603.8 TPS

information:

NODE: Geth/v1.8.12-stable/linux-amd64/go1.10.2 on http://10.5.65.99:22000
  consensus=raft chain_name=??? chain_id=-1 network_id=60878
SEND: 100 transactions in Blocks 4-9 with 0 empty blocks following.
  A sample of transactions looked as if they: succeeded.
  TPS: The stopclock watcher measured a final TPS of 157.9 since contract deploy,
  and in between saw values as high as 0.0 TPS.
  DIAG: The whole experiment was prefixed 'Local_Quorum_dan_Thesis_50threads_100TX'.
  The diagrams were saved into 'img/Local_Quorum_dan_Thesis_50threads_100TX-20190520-1602_blks4-9.png'.
  Looking only at the experiment block-timestamps, the overall TPS was ~603.8.

log:

versions: web3 4.3.0, py-solc: 2.1.0, solc 0.4.25+commit.59dbff1f1.Linux.gpp, testrpc 1.3.4, python 3.6.7 (default, Oct 22 2018, 11:32:17) [GCC 8.2.0]
web3 connection established, blockNumber = 2, node version string = Geth/v1.8.12-stable/linux-amd64/go1.10.2
first account of node is 0xFa25adcDe9Fc75b7c4e0B75A58c0Ae7656e1927e, balance is 1000000000 Ether
WARN: raft consensus did report timestamps in nanoseconds. Is that still the case?
nodeName: Quorum, nodeType: Geth, nodeVersion: v1.8.12-stable, consensus: raft, network: 60878, chainName: ???, chainId: -1

Block 2 - waiting for something to happen
(blockdate 1558368026) last contract address: 0xc92127Ce479999b3AB9150027cCD9805D1E15D83
(blockdate 1558368155) new contract address: 0xFeE19eA1F293f6D5eabF47C3854c319B66578adC
blocknumber_start_here = 3
starting timer, at block 3 which has 1 transactions; at epochtime 1558368155.4140217
block 4 | new #TX  1 /  475 ms =  2.1 TPS_current | total: #TX  2 /  0.3 s =  6.5 TPS_average (peak is 6.5 TPS_average)
block 9 | new #TX  99 /  164 ms = 603.8 TPS_current | total: #TX 101 /  0.6 s = 157.9 TPS_average (peak is 157.9 TPS_average)
Received signal from send.py = updated INFOPFILE:
Experiment ended! Current blocknumber = 9
Updated info file: last-experiment.json THE END.

diagrams:
Local Quorum dan Thesis 50threads 100TX blocks 4-9 with 1 txs ~ 0 txs/block

avg TPS 603.8 = #TX whole experiment / blocktimes diff

blocksize in bytes

gasUsed and gasLimit per second
Quorum v1.8.12 with 1000 txs: 689.9 TPS

information:

NODE: Geth/v1.8.12-stable/linux-amd64/go1.10.2 on http://10.5.65.99:22000
consensus=raft chain_name=??? chain_id=-1 network_id=79567
SEND: 1000 transactions in blocks 4-34 with 0 empty blocks following.
A sample of transactions looked as if they: succeeded.

TPS: The stopclock watcher measured a final TPS of 517.6 since contract deploy,
and in between saw values as high as 517.6 TPS.

DIAG: The whole experiment was prefixed 'Local_Quorum_dan_Thesis_50threads_1000TX'.
The diagrams were saved into 'img/Local_Quorum_dan_Thesis_50threads_1000TX-20190520-1603_blks4-34.png'.
Looking only at the experiment block-timestamps, the overall TPS was ~689.9.

log:

versions: web3 4.3.0, solc: 2.1.0, solc 0.4.25+commit.59dbfb8f1.Linux.gpp, testrpc 1.3.4, python 3.6.7 (default, Oct 22 2018, 11:32:17) [GCC 8.2.0]
web3 connection established, blockNumber = 2, node version string = Geth/v1.8.12-stable/linux-amd64/go1.10.2
first account of node is 0x50169A68b4B933f1dd32C41da6108F70FD2C1edf, balance is 1000000000 Ether
WARN: raft consensus did report timestamps in nanoseconds. Is that still the case?
nodeName: Quorum, nodeType: Geth, nodeVersion: v1.8.12-stable, consensus: raft, network: 79567, chainName: ???, chainId: -1

Block 2 - waiting for something to happen
(filedate 1558368155) last contract address: 0xFeE19eA1F293f6D5eabF47C3854c319868678adC
(filedate 1558368217) new contract address: 0xD355411D2af69dBCE7EA8A7e34c88edB866382d37
blocknumber_start_here = 3
starting timer, at block 3 which has 1 transactions; at epochtime 1558368217.8737566
block 9 | new #TX 144 / 694 ms = 207.6 TPS_current | total: #TX 145 / 0.6 s = 232.6 TPS_average (peak is 232.6 TPS_average)
block 15 | new #TX 113 / 300 ms = 667.7 TPS_current | total: #TX 148 / 1.0 s = 365.6 TPS_average (peak is 365.6 TPS_average)
block 22 | new #TX 246 / 350 ms = 702.9 TPS_current | total: #TX 594 / 1.3 s = 465.9 TPS_average (peak is 465.9 TPS_average)
block 28 | new #TX 216 / 300 ms = 720.0 TPS_current | total: #TX 810 / 1.6 s = 507.4 TPS_average (peak is 507.4 TPS_average)
block 34 | new #TX 191 / 300 ms = 636.6 TPS_current | total: #TX 1001 / 1.9 s = 517.6 TPS_average (peak is 517.6 TPS_average)
Received signal from send.py = updated INFOFILE.
Experiment ended! Current blocknumber = 34
Updated info file: last-experiment.json THE END.

diagrams:
Local Quorum dan Thesis 50threads 1000TX blocks 4-34 with 1 txs ~ 0 txs/block

avg TPS 689.9 = #TX whole experiment / blocktimes diff

blocktime seconds since last block

gasUsed and gasLimit per second

info raw

{"diagrams": {"blocktimestampsTpsAv": 689.8988601819079, "filename": 'img/Local_Quorum_dan_Thesis_50threads_1000TX-20190520-1603_blks4-34.png', "prefix": 'Local_Quorum_dan_Thesis_50threads_1000TX'}, "node": {"chain_id": -1, "chain_name": '??', "consensus": 'raft', "name": 'Quorum', "network_id": 79567, "rpc_address": 'http://10.5.65.99:22000', "type": 'Geth', "version": 'v1.8.12-stable', "web3.version.node": 'Geth/v1.8.12-stable/linux-amd64/go1.10.2'}, "send": {"block_first": 4, "block_last": 34, "empty_blocks": 0, "num_txs": 1000, "sample_txs_successful": True}, "tps": {"finalTpsAv": 517.604734214383, "peakTpsAv": 517.604734214383, "start_epochtime": 1558368217.8737566}}
<table>
<thead>
<tr>
<th>Block</th>
<th>New TX</th>
<th>Total TX</th>
<th>TPS Current</th>
<th>TPS Average</th>
<th>Peak TPS Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>275</td>
<td>237.4</td>
<td>237.3</td>
<td>273.3</td>
<td>273.3</td>
<td>273.3</td>
</tr>
<tr>
<td>268</td>
<td>209.0</td>
<td>208.9</td>
<td>245.7</td>
<td>245.7</td>
<td>245.7</td>
</tr>
<tr>
<td>262</td>
<td>202.0</td>
<td>201.9</td>
<td>246.3</td>
<td>246.3</td>
<td>246.3</td>
</tr>
<tr>
<td>256</td>
<td>240.0</td>
<td>239.9</td>
<td>246.9</td>
<td>246.9</td>
<td>246.9</td>
</tr>
<tr>
<td>249</td>
<td>209.0</td>
<td>208.9</td>
<td>245.7</td>
<td>245.7</td>
<td>245.7</td>
</tr>
<tr>
<td>243</td>
<td>244.0</td>
<td>243.9</td>
<td>246.9</td>
<td>246.9</td>
<td>246.9</td>
</tr>
<tr>
<td>236</td>
<td>215.0</td>
<td>214.9</td>
<td>245.7</td>
<td>245.7</td>
<td>245.7</td>
</tr>
<tr>
<td>230</td>
<td>209.0</td>
<td>208.9</td>
<td>245.7</td>
<td>245.7</td>
<td>245.7</td>
</tr>
<tr>
<td>224</td>
<td>244.0</td>
<td>243.9</td>
<td>246.9</td>
<td>246.9</td>
<td>246.9</td>
</tr>
<tr>
<td>218</td>
<td>212.0</td>
<td>211.9</td>
<td>245.7</td>
<td>245.7</td>
<td>245.7</td>
</tr>
<tr>
<td>211</td>
<td>212.0</td>
<td>211.9</td>
<td>245.7</td>
<td>245.7</td>
<td>245.7</td>
</tr>
<tr>
<td>204</td>
<td>215.0</td>
<td>214.9</td>
<td>245.7</td>
<td>245.7</td>
<td>245.7</td>
</tr>
<tr>
<td>198</td>
<td>215.0</td>
<td>214.9</td>
<td>245.7</td>
<td>245.7</td>
<td>245.7</td>
</tr>
<tr>
<td>191</td>
<td>244.0</td>
<td>243.9</td>
<td>246.9</td>
<td>246.9</td>
<td>246.9</td>
</tr>
<tr>
<td>184</td>
<td>212.0</td>
<td>211.9</td>
<td>245.7</td>
<td>245.7</td>
<td>245.7</td>
</tr>
<tr>
<td>178</td>
<td>209.0</td>
<td>208.9</td>
<td>245.7</td>
<td>245.7</td>
<td>245.7</td>
</tr>
<tr>
<td>171</td>
<td>215.0</td>
<td>214.9</td>
<td>245.7</td>
<td>245.7</td>
<td>245.7</td>
</tr>
<tr>
<td>165</td>
<td>215.0</td>
<td>214.9</td>
<td>245.7</td>
<td>245.7</td>
<td>245.7</td>
</tr>
<tr>
<td>158</td>
<td>215.0</td>
<td>214.9</td>
<td>245.7</td>
<td>245.7</td>
<td>245.7</td>
</tr>
<tr>
<td>151</td>
<td>215.0</td>
<td>214.9</td>
<td>245.7</td>
<td>245.7</td>
<td>245.7</td>
</tr>
<tr>
<td>144</td>
<td>215.0</td>
<td>214.9</td>
<td>245.7</td>
<td>245.7</td>
<td>245.7</td>
</tr>
<tr>
<td>137</td>
<td>215.0</td>
<td>214.9</td>
<td>245.7</td>
<td>245.7</td>
<td>245.7</td>
</tr>
<tr>
<td>130</td>
<td>215.0</td>
<td>214.9</td>
<td>245.7</td>
<td>245.7</td>
<td>245.7</td>
</tr>
<tr>
<td>123</td>
<td>215.0</td>
<td>214.9</td>
<td>245.7</td>
<td>245.7</td>
<td>245.7</td>
</tr>
<tr>
<td>116</td>
<td>215.0</td>
<td>214.9</td>
<td>245.7</td>
<td>245.7</td>
<td>245.7</td>
</tr>
<tr>
<td>109</td>
<td>215.0</td>
<td>214.9</td>
<td>245.7</td>
<td>245.7</td>
<td>245.7</td>
</tr>
<tr>
<td>102</td>
<td>215.0</td>
<td>214.9</td>
<td>245.7</td>
<td>245.7</td>
<td>245.7</td>
</tr>
<tr>
<td>95</td>
<td>215.0</td>
<td>214.9</td>
<td>245.7</td>
<td>245.7</td>
<td>245.7</td>
</tr>
<tr>
<td>88</td>
<td>215.0</td>
<td>214.9</td>
<td>245.7</td>
<td>245.7</td>
<td>245.7</td>
</tr>
</tbody>
</table>

The diagrams were saved into 'img/Local_Quorum_Quakes_50threads_10000TX-20190520-1604_bliks4-291.png'.

Looking only at the experiment block-timestamps, the overall TPS was 699.9.

Block 2 - waiting for something to happen

(filedate 1558368217) last contract address: 0x035412D2bre6D8C65E29A7ee3ACbC169689E43B2d37

(filedate 1558368299) new contract address: 0x035412D2bre6D8C65E29A7ee3ACbC169689E43B2d37

information:

NODE: Geth/v1.8.12-stable/linux-amd64/go1.10.2 on http://10.5.65.99:22000
consensus=raft chain_name=??? chain_id=1 network_id=67447
SEND: 10000 transactions in blocks 4-291 with 0 empty blocks following.

TPS:
The stopclock watcher measured a final TPS of 672.9 since contract deploy,
and in between saw values as high as 683.5 TPS.

DIAG: The whole experiment was prefixed 'Local_Quorum_Quakes_50threads_10000TX'.

log:

versions: web3 4.2.0, py-solc: 2.1.0, solc 0.4.25+commit.59dbf8f1.Library.gpp, testrpc 1.3.4, python 3.6.7 (default, Oct 22 2018, 11:32:17) [GCC 8.2.0]
web3 connection established, blockNumber = 2, node version string = Geth/v1.8.12-stable/linux-amd64/go1.10.2
first account of node is 0x8d55634C97FFDA4665A0E8C1F46C25D3DF, balance is 1000000000 Ether
WARN: raft consensus did report timestamps in nanoseconds. Is that still the case?

Block 2 waiting for something to happen

(nodeDate 1558368217) last contract address: 0x035412D2bre6D8C65E29A7ee3ACbC169689E43B2d37

Information:

(Local_Quorum_dan_Thesis_50threads_10000TX) Quorum v1.8.12 with 10000 txs: 699.9 TPS
Received signal from send.py = updated INFOFILE.
Experiment ended! Current blocknumber = 291
Updated info file: last-experiment.json THE END.

Local Quorum dan Thesis 50threads 10000TX blocks 4-291 with 1 txs ~ 0 txs/block

avg TPS 699.9 = #TX whole experiment / blocktimes diff

blocktime seconds since last block

gasUsed and gasLimit per second
Quorum v1.8.12 with 100 txs: 580.1 TPS

information:

NODE: Geth/v1.8.12-stable/linux-amd64/go1.10.2 on http://10.5.65.99:22000
consensus=raft chain_name=??? chain_id=-1 network_id=65617
SEND: 100 transactions in Blocks 4-9 with 0 empty blocks following.
A sample of transactions looked as if they: succeeded.

TPS: The stopclock watcher measured a final TPS of 158.0 since contract deploy,
and in between saw values as high as 0.0 TPS.

DIAG: The whole experiment was prefixed 'Local_Quorum_dan_Thesis_100threads_100TX'.
The diagrams were saved into 'img/Local_Quorum_dan_Thesis_100threads_100TX-20190520-1613_blks4-9.png'.
Looking only at the experiment block timestamps, the overall TPS was ~580.1.

log:

versions: web3 4.3.0, py-solc: 2.1.0, solc 0.4.25+commit.59dbf8f1.Linux.gpp, testrpc 1.3.4, python 3.6.7 (default, Oct 22 2018, 11:32:17) [GCC 8.2.0]
web3 connection established, blockNumber = 2, node version string = Geth/v1.8.12-stable/linux-amd64/go1.10.2
first account of node is 0xd3aa794687788f78e9109863B15cF418CbCC2055, balance is 1000000000 Ether
WARN: raft consensus did report timestamps in nanoseconds. Is that still the case?

nodeName: Quorum, nodeType: Geth, nodeVersion: v1.8.12-stable, consensus: raft, network: 65617, chainName: ???, chainId: -1

Block  2  - waiting for something to happen
[filedate 1558368338] last contract address: 0xa7fc265cff466ef037B9B60C1fB24662834Ea7d2
[filedate 1558368797] new contract address: 0x995F3B1BE5B379138a173D0976560488016E086
blocknumber_start_here = 3
starting timer, at block 3 which has 1 transactions; at epochtime 1558368797.235757
block 9 | new #TX 100 / 669 ms = 149.5 TPS_current | total: #TX 101 / 0.6 s = 158.0 TPS_average (peak is 158.0 TPS_average)
Received signal from send.py = updated INFOFILE.
Experiment ended! Current blocknumber = 9
Updated info file: last-experiment.json THE END.

diagrams:
Local Quorum dan Thesis 100threads 100TX blocks 4-9 with 1 txs ~ 0 txs/block

avg TPS 580.1 = #TX whole experiment / blocktimes diff

blocktimes seconds since last block

gasUsed and gasLimit per second

info raw

{"diagrams": {"blockstampsTpsAv": 580.07721955942, 'filename': 'img/Local_Quorum_dan_Thesis_100threads_100TX-20190520-1613_blks4-9.png', 'prefix': 'Local_Quorum_dan_Thesis_100threads_100TX'}, 'node': {'chain_id': -1, 'chain_name': '???', 'consensus': 'raft', 'name': 'Quorum', 'network_id': 65617, 'rpc_address': 'http://10.5.65.99:22000', 'type': 'Geth', 'version': 'v1.8.12-stable', 'web3.version.node': 'Geth/v1.8.12-stable/linux-amd64/go1.10.2'}, 'send': {'block_first': 4, 'block_last': 9, 'empty_blocks': 0, 'num_txs': 100, 'sample_txs_successful': True}, 'tps': {'finalTpsAv': 157.95244142033476, 'peakTpsAv': 0, 'start_epochtime': 1558368797.235757}}
Quorum v1.8.12 with 1000 txs: 698.8 TPS

information:

NODE: Geth/v1.8.12-stable/linux-amd64/go1.10.2 on http://10.5.65.99:22000
consensus=raft chain_name=?? chain_id=-1 network_id=58204
SEND: 1000 transactions in blocks 4-34 with 0 empty blocks following.
A sample of transactions looked as if they: succeeded.

TPS: The stopclock watcher measured a final TPS of 522.4 since contract deploy,
and in between saw values as high as 522.4 TPS.

DIAG: The whole experiment was prefixed 'Local_Quorum_dan_Thesis_100threads_1000TX'.
The diagrams were saved into 'img/Local_Quorum_dan_Thesis_100threads_1000TX-20190520-1614_blks4-34.png'.

Looking only at the experiment block-timestamps, the overall TPS was ~698.8.

log:

versions: web3 4.3.0, py-solc: 2.1.0, solc 0.4.25+commit.59dbf8ff1.Linq.gpp, testrpc 1.3.4, python 3.6.7 (default, Oct 22 2018, 11:32:17) [GCC 8.2.0]
web3 connection established, blockNumber = 2, node version string = Geth/v1.8.12-stable/linux-amd64/go1.10.2
first account of node is 0xadE8B878fda42444ddDA8428c3766bBbCe552812, balance is 1000000000 Ether
WARN: raft consensus did report timestamps in nanoseconds. Is that still the case?
nodeName: Quorum, nodeType: Geth, nodeVersion: v1.8.12-stable, consensus: raft, network: 58204, chainName: ???, chainId: -1

Block  2  - waiting for something to happen
(blockdate 1558368797) last contract address: 0x995F3B1BE5B379138a173Dd97d650D48B016E086
(blockdate 1558368878) new contract address: 0xaF98F02BafCBcD3Eb465a0736eA91fD0a6c89afa
blocknumber_start_here = 3
starting timer, at block 3 which has 1 transactions; at epochtime 1558368878.7224936
block 10 | new #TX 154 / 722 ms = 213.2 TPS_current | total: #TX 155 / 0.6 s = 247.9 TPS_average (peak is 247.9 TPS_average)
block 16 | new #TX 210 / 300 ms = 700.1 TPS_current | total: #TX 365 / 0.9 s = 384.7 TPS_average (peak is 384.7 TPS_average)
block 23 | new #TX 252 / 350 ms = 720.0 TPS_current | total: #TX 617 / 1.3 s = 484.6 TPS_average (peak is 484.6 TPS_average)
block 29 | new #TX 205 / 300 ms = 683.3 TPS_current | total: #TX 822 / 1.6 s = 515.2 TPS_average (peak is 515.2 TPS_average)
block 34 | new #TX 179 / 250 ms = 716.0 TPS_current | total: #TX 1001 / 1.9 s = 522.4 TPS_average (peak is 522.4 TPS_average)
Received signal from send.py = updated INFOFILE.
Experiment ended! Current blocknumber = 34
Updated info file: last-experiment.json THE END.

diagrams:
Local Quorum dan Thesis 100threads 1000TX blocks 4-34 with 1 txs ~ 0 txs/block

![Graphs showing TPS and blocktimes]

**info raw**

```json
{"diagrams": {"blocktimestampsTpsAv": 698.84057145316, 'filename': 'img/Local_Quorum_dan_Thesis_100threads_1000TX-20190520-1614_blks4-34.png', 'prefix': 'Local_Quorum_dan_Thesis_100threads_1000TX'}, 'node': {"chain_id": -1, 'chain_name': '??', 'consensus': 'raft', 'name': 'Quorum', 'network_id': 58204, 'rpc_address': 'http://10.5.65.99:22000', 'type': 'Geth', 'version': 'v1.8.12-stable', 'web3.version.node': 'Geth/v1.8.12-stable/linux-amd64/go1.10.2'}, 'send': {"block_first": 4, 'block_last': 34, 'empty_blocks': 0, 'num_txs': 1000, 'sample_TxS_successful': True}, 'tps': {"finalTpsAv": 522.369289899102, 'peakTpsAv': 522.369289899102, 'start_epochtime': 1558368878.7224936}}
```
Block 2 - waiting for something to happen

(nodeName: Quorum, nodeType: Geth, nodeVersion: v1.8.12-stable, consensus: raft, network: 34246, chainName: ???, chainId: -1)

INFO: raft consensus did report timestamps in nanoseconds. Is that still the case?

versions: web3 4.3.0, py-solc: 0.4.25+commit.59dbff81, Linuz.gpp, testrpc 1.3.4, python 3.6.7 (default, Oct 22 2018, 11:32:17) [GCC 8.2.0]

web3 connection established, blockNumber = 2, node version string = Geth/v1.8.12-stable/linux-amd64/go1.10.2

first account of node is 0xad1d92f8a4644505e2f83d2854e0e50cfa0cfa730f, balance is 0x00000000 Ether

DIRAG: The whole experiment was prefixed 'Local_Quorum_dan_Thesis_100threads_10000TX'.

TPS: The stopclock watcher measured a final TPS of 683.7 since contract deploy, and in between saw values as high as 683.0 TPS.

The diagrams were saved to 'img/local_Quorum_dan_Thesis_10000TX_20190520-1615_bk04-289.png'.

Looking only at the experiment block-time stamps, the overall TPS was 784.3.
Received signal from send.py = updated INFOFILE.
Experiment ended! Current blocknumber = 289
Updated info file: last-experiment.json THE END.

### diagrams:

![Diagram 1: Local Quorum dan Thesis 100threads 10000TX blocks 4-289 with 1 txs ~ 0 txs/block](img/Local_Quorum_dan_Thesis_100threads_10000TX-20190520-1615_blks4-289.png)

**avg TPS 704.3 = #TX whole experiment / blocktimes diff**

**blocktime seconds since last block**

**blocksizes in bytes**

**gasUsed and gasLimit per second**

---

**info raw**
Virtual Machines
Quorum v1.8.12 with 100 txs: 372.6 TPS

information:

NODE: Geth/v1.8.12-stable/linux-amd64/go1.10.2 on http://10.5.62.60:22000
consensus=raft chain_name=?? chain_id=-1 network_id=48117
SEND: 100 transactions in Blocks 6-12 with 0 empty blocks following.
A sample of transactions looked as if they: succeeded.
TPS: The stopclock watcher measured a final TPS of 104.0 since contract deploy,
and in between saw values as high as 0.0 TPS.

DIAG: The whole experiment was prefixed 'VM_Quorum_Dan_Thesis_10threads_100txs'.
The diagrams were saved into 'img/VM_Quorum_Dan_Thesis_10threads_100txs-20190520-1642_blks6-12.png'.
Looking only at the experiment block-timestamps, the overall TPS was ~372.6.

log:

versions: web3 4.8.2, py-solc: 3.2.0, solc 0.4.25+commit.59db8ff1.Linux.gpp, testrpc 1.3.5, python 3.6.7 (default, Oct 22 2018, 11:32:17) [GCC 8.2.0]
web3 connection established, blockNumber = 4, node version string = Geth/v1.8.12-stable/linux-amd64/go1.10.2
first account of node is 0xEa88ee08ecdb15cE047Ee097f7231ac3c99af08EC, balance is 1000000000 Ether
WARN: raft consensus did report timestamps in nanoseconds. Is that still the case?
nodeName: Quorum, nodeType: Geth, nodeVersion: v1.8.12-stable, consensus: raft, network: 48117, chainName: ???, chainId: -1

Block  4  - waiting for something to happen
(filedate 1558370060) last contract address: 0x14cd6FAD3c269bCe566BE73ba0B23d280FEe8c4E
(filedate 1558370527) new contract address: 0x781D09B600dc4D36bDa9aCf9eE250dB21B1466F
blocknumber_start_here = 5
starting timer, at block 5 which has  1  transactions; at epochtime 1558370527.415408
block 10 | new #TX  74 /  601 ms = 123.1 TPS_current | total: #TX   75 /  0.7 s = 114.6 TPS_average (peak is 114.6 TPS_average)
block 12 | new #TX  26 /  101 ms = 258.1 TPS_current | total: #TX  101 /  1.0 s = 104.0 TPS_average (peak is 104.0 TPS_average)
Received signal from send.py = updated INFOFILE.
Experiment ended! Current blocknumber = 12
Updated info file: last-experiment.json THE END.

diagrams:
avg TPS 372.6 = #TX whole experiment / blocktimes diff

blocksize in bytes

gasUsed and gasLimit per second

info raw

Quorum v1.8.12 with 1000 txs: 488.1 TPS

information:

NODE: Geth/v1.8.12-stable/linux-amd64/go1.10.2 on http://10.5.62.60:22000
consensus=raft chain_name=??? chain_id=-1 network_id=35042
SEND: 1000 transactions in blocks 4-46 with 0 empty blocks following.
A sample of transactions looked as if they: succeeded.
TPS: The stopclock watcher measured a final TPS of 356.1 since contract deploy,
and in between saw values as high as 376.6 TPS.
DIAG: The whole experiment was prefixed 'VM_Quorum_Dan_Thesis_10threads_1000txs'.
The diagrams were saved into 'img/VM_Quorum_Dan_Thesis_10threads_1000txs-20190520-1644_blks4-46.png'.
Looking only at the experiment block-timestamps, the overall TPS was ~488.1.

log:

versions: web3 4.8.2, py-solc: 3.2.0, solc 0.4.25+commit.59dbf8f1_linux.go, testrpc 1.3.5, python 3.6.7 (default, Oct 22 2018, 11:32:17) [GCC 8.2.0]
web3 connection established, blockNumber = 2, node version string = Geth/v1.8.12-stable/linux-amd64/go1.10.2
first account of node is 0xdDAEa49d4988BF254EF92b1a45F2677426381b9d, balance is 1000000000 Ether
WARN: raft consensus did report timestamps in nanoseconds. Is that still the case?
nodeName: Quorum,.nodeType: Geth, nodeVersion: v1.8.12-stable, consensus: raft, network: 35042, chainName: ???, chainId: -1
Block  2  - waiting for something to happen
(filedate 1558370527) last contract address: 0x781D09B600dc4D36bDa9aCf9e9E250dB21B14e6F
(filedate 1558370647) new contract address: 0x5F4a7FD279ef1400BD3265742B8e02CC1966a5f0d

blocknumber_start_here = 3
starting timer, at block 3 which has  1  transactions; at epochtime 1558370647.470164
block 8 | new #TX  80 /  690 ms = 115.9 TPS_current | total: #TX  81 /  0.6 s = 126.9 TPS_average (peak is 126.9 TPS_average)
block 14 | new #TX 141 /  301 ms = 468.8 TPS_current | total: #TX 222 /  1.0 s = 228.0 TPS_average (peak is 228.0 TPS_average)
block 21 | new #TX 173 /  349 ms = 495.3 TPS_current | total: #TX 395 /  1.3 s = 294.0 TPS_average (peak is 294.0 TPS_average)
block 29 | new #TX 182 /  400 ms = 454.7 TPS_current | total: #TX 577 /  1.7 s = 337.7 TPS_average (peak is 337.7 TPS_average)
block 36 | new #TX 183 /  350 ms = 522.6 TPS_current | total: #TX 760 /  2.1 s = 367.0 TPS_average (peak is 367.0 TPS_average)
block 43 | new #TX 166 /  350 ms = 474.9 TPS_current | total: #TX 926 /  2.5 s = 376.6 TPS_average (peak is 376.6 TPS_average)
block 46 | new #TX  75 /  151 ms = 498.0 TPS_current | total: #TX 1001 /  2.8 s = 356.1 TPS_average (peak was 376.6 TPS_average)
Received signal from send.py = updated INFOFILE.
Experiment ended! Current blocknumber = 46
Updated info file: last-experiment.json THE END.
VM_Quorum_Dan_Thesis_10threads_1000txs blocks 4-46 with 1 txs ~ 0 txs/block

avg TPS 488.1 = #TX whole experiment / blocktimes diff

blocktime seconds since last block

gasUsed and gasLimit per second

info raw

Log:

versions: web3 4.8.2, py-solc: 3.2.0, solc 0.4.25+commit.59dbff8f Linucos.cpp, testrpc 1.3.5, python 3.6.7 (default, Oct 22 2018, 11:32:17) [GCC 8.2.0]
web3 connection established, blockNumber = 2, node version string = Geth/v1.8.12-stable/linux-amd64/go1.10.2
first account of node is 0x0eb4a242e4266c152a6959198105353e5, balance is 1000000000 Ether
WARN: Geth consensus did report timestamps in nanoseconds. Is that still the case?

Block 2 - waiting for something to happen

Looking only at the experiment block-timestamps, the overall TPS was ~418.3.

Diag: The whole experiment was prefixed 'VM_Quorum_Dan_Thesis_10threads_10000txs'.

TPS: The stopclock watcher measured a final TPS of 405.5 since contract deploy,
A sample of transactions looked as if they: succeeded.
SEND: 10000 transactions in blocks 4-483 with 0 empty blocks following.

The diagrams were saved into 'img/VM_Quorum_Dan_Thesis_10threads_10000txs-20190520-1647_blks4-483.png'.

Looking only at the experiment block-timestamps, the overall TPS was ~418.3.
block 326 new #TX 158 / 350 ms = 451.2 TPS_current total: #TX 6967 / 16.6 s = 420.6 TPS_average (peak was 431.8 TPS_average)
block 333 new #TX 149 / 354 ms = 421.3 TPS_current total: #TX 7116 / 16.9 s = 420.0 TPS_average (peak was 431.8 TPS_average)
block 339 new #TX 129 / 297 ms = 434.3 TPS_current total: #TX 7245 / 17.8 s = 419.5 TPS_average (peak was 431.8 TPS_average)
block 348 new #TX 202 / 453 ms = 445.9 TPS_current total: #TX 7447 / 18.1 s = 420.2 TPS_average (peak was 431.8 TPS_average)
block 356 new #TX 170 / 407 ms = 434.9 TPS_current total: #TX 7622 / 18.3 s = 421.1 TPS_average (peak was 431.8 TPS_average)
block 364 new #TX 175 / 397 ms = 417.8 TPS_current total: #TX 7792 / 18.6 s = 419.6 TPS_average (peak was 431.8 TPS_average)
block 373 new #TX 192 / 449 ms = 427.5 TPS_current total: #TX 7984 / 19.0 s = 420.2 TPS_average (peak was 431.8 TPS_average)
block 381 new #TX 192 / 394 ms = 436.4 TPS_current total: #TX 8156 / 19.3 s = 421.1 TPS_average (peak was 431.8 TPS_average)
block 387 new #TX 82 / 300 ms = 273.1 TPS_current total: #TX 8238 / 19.7 s = 418.8 TPS_average (peak was 431.8 TPS_average)
block 395 new #TX 136 / 413 ms = 329.5 TPS_current total: #TX 8374 / 20.1 s = 415.9 TPS_average (peak was 431.8 TPS_average)
block 405 new #TX 127 / 492 ms = 268.2 TPS_current total: #TX 8502 / 20.6 s = 413.2 TPS_average (peak was 431.8 TPS_average)
block 413 new #TX 127 / 395 ms = 321.7 TPS_current total: #TX 8629 / 21.1 s = 409.1 TPS_average (peak was 431.8 TPS_average)
block 423 new #TX 206 / 509 ms = 484.7 TPS_current total: #TX 8835 / 21.6 s = 408.2 TPS_average (peak was 431.8 TPS_average)
block 433 new #TX 221 / 544 ms = 486.3 TPS_current total: #TX 9056 / 22.1 s = 410.3 TPS_average (peak was 431.8 TPS_average)
block 443 new #TX 197 / 450 ms = 347.4 TPS_current total: #TX 9253 / 22.6 s = 409.7 TPS_average (peak was 431.8 TPS_average)
block 453 new #TX 194 / 511 ms = 379.6 TPS_current total: #TX 9447 / 23.1 s = 409.2 TPS_average (peak was 431.8 TPS_average)
block 463 new #TX 127 / 490 ms = 259.4 TPS_current total: #TX 9574 / 23.6 s = 406.5 TPS_average (peak was 431.8 TPS_average)
block 473 new #TX 217 / 580 ms = 434.3 TPS_current total: #TX 9791 / 24.0 s = 408.8 TPS_average (peak was 431.8 TPS_average)
block 480 new #TX 154 / 347 ms = 443.6 TPS_current total: #TX 9945 / 24.3 s = 408.6 TPS_average (peak was 431.8 TPS_average)
block 483 new #TX 56 / 151 ms = 369.8 TPS_current total: #TX 10001 / 24.7 s = 409.5 TPS_average (peak was 431.8 TPS_average)
Received signal from send.py = updated INFOFILE.
Experiment ended! Current blocknumber = 483
Updated info file: last-experiment.json THE END.

diagrams:
VM_Quorum_Dan_Thesis_10threads_10000txs blocks 4-483 with 1 txs ~ 0 txs/block

avg TPS 418.3 = #TX whole experiment / blocktimes diff

blocktime seconds since last block

gasUsed and gasLimit per second

info raw

{"diagrams": {"blocktimesTpsAv": 418.33332409995535, "filename": 'img/VM_Quorum_Dan_Thesis_10threads_10000txs-20190520-1647_blks4-483.png', "prefix": 'VM_Quorum_Dan_Thesis_10threads_10000txs'}, "node": {"chain_id": -1, "chain_name": '???', "consensus": 'raft', "name": 'Quorum', "network_id": 91247, "rpc_address": 'http://10.5.62.60:22000', "type": 'Geth', "version": 'v1.8.12-stable', "web3.version.node": 'Geth/v1.8.12-stable/linux-amd64/go1.10.2'}, "send": {"block_first": 4, "block_last": 483, "empty_blocks": 0, "num_txs": 10000, "sample_txs_successful": True}, "tps": {"finalTpsAv": 405.479942967927, "peakTpsAv": 431.83254652410164, "start_epochtime": 1558370847.982318}}
Quorum v1.8.12 with 100 txs: 424.4 TPS

information:

NODE: Geth/v1.8.12-stable/linux-amd64/go1.10.2 on http://10.5.62.60:22000
consensus=raft chain_name=??? chain_id=-1 network_id=50462

SEND: 100 transactions in Blocks 4-9 with 0 empty blocks following.
A sample of transactions looked as if they: succeeded.

TPS: The stopclock watcher measured a final TPS of 159.9 since contract deploy,
and in between saw values as high as 0.0 TPS.

DIAG: The whole experiment was prefixed 'VM_Quorum_Dan_Thesis_30threads_100txs'.
The diagrams were saved into 'img/VM_Quorum_Dan_Thesis_30threads_100txs-20190520-1649_blks4-9.png'.
Looking only at the experiment block-timestamps, the overall TPS was ~424.4.

log:

versions: web3 4.8.2, py-solc: 3.2.0, solc 0.4.25+commit.59dbf8f1.Linux.gpp, testrpc 1.3.5, python 3.6.7 (default, Oct 22 2018, 11:32:17) [GCC 8.2.0]
web3 connection established, blockNumber = 2, node version string = Geth/v1.8.12-stable/linux-amd64/go1.10.2
first account of node is 0xc719f9bF66cB9f77312b2416DC75eaf1994df379, balance is 1000000000 Ether
WARN: raft consensus did report timestamps in nanoseconds. Is that still the case?

nodeName: Quorum, nodeType: Geth, nodeVersion: v1.8.12-stable, consensus: raft, network: 50462, chainName: ???, chainId: -1

Block 2 - waiting for something to happen
(filedate 1558370847) last contract address: 0xD7763Cab9Cdf7BB92271D51Fc55eeaf1445f0f3d
(filedate 1558370970) new contract address: 0xc2258126Ff4AE3Ed5e141F138a16b2b30ca08716
blocknumber_start_here = 3

block 9 | new 100 /  660 ms = 151.6 TPS_current | total: 101 / 0.6 s = 159.9 TPS_average (peak is 159.9 TPS_average)
Received signal from send.py = updated INFOFILE.
Experiment ended! Current blocknumber = 9
Updated info file: last-experiment.json THE END.

diagrams:
VM_Quorum_Dan_Thesis_30threads_100txs blocks 4-9 with 1 txs ~ 0 txs/block

avg TPS 424.4 = #TX whole experiment / blocktimes diff

blocktime seconds since last block

gasUsed and gasLimit per second

info raw

{"diagrams": {"blocktimestampsTpsAv": 424.40103638063994, "filename": "img/VM_Quorum_Dan_Thesis_30threads_100txs-20190520-1649_blks4-9.png", "prefix": "VM_Quorum_Dan_Thesis_30threads_100txs"}, "node": {"chain_id": -1, "chain_name": "???", "consensus": "raft", "name": "Quorum", "network_id": 50462, "rpc_address": "http://10.5.62.60:22000", "type": "Geth", "version": "v1.8.12-stable", "web3.version.node": "Geth/v1.8.12-stable/linux-amd64/go1.10.2"}, "send": {"block_first": 4, "block_last": 9, "empty_blocks": 0, "num_txs": 100, "sample_txs_successful": true, "tps": {"finalTpsAv": 159.8806870902407, "peakTpsAv": 0, "start_epochtime": 1558370970.8073413}}}
Quorum v1.8.12 with 1000 txs: 395.2 TPS

information:

NODE: Geth/v1.8.12-stable/linux-amd64/go1.10.2 on http://10.5.62.60:22000
consensus=raft chain_name=??? chain_id=-1 network_id=57312
SEND: 1000 transactions in blocks 4-55 with 0 empty blocks following.
A sample of transactions looked as if they: succeeded.

TPS: The stopclock watcher measured a final TPS of 314.9 since contract deploy,
and in between saw values as high as 332.4 TPS.

DIAG: The whole experiment was prefixed 'VM_Quorum_Dan_Thesis_30threads_1000txs'.
The diagrams were saved into 'img/VM_Quorum_Dan_Thesis_30threads_1000txs-20190520-1650_blks4-55.png'.
Looking only at the experiment block-timestamps, the overall TPS was ~395.2.

log:

versions: web3 4.8.2, py-solc: 3.2.0, solc 0.4.25+commit.59db8ff1.Linux.gpp, testrpc 1.3.5, python 3.6.7 (default, Oct 22 2018, 11:32:17) [GCC 8.2.0]
web3 connection established, blockNumber = 2, node version string = Geth/v1.8.12-stable/linux-amd64/go1.10.2
first account of node is 0xD3d3784a8A97668182D37a9c450C4541BcfEc207, balance is 1000000000 Ether
WARN: raft consensus did report timestamps in nanoseconds. Is that still the case?

Block  2  - waiting for something to happen
[filedate 1558370970] last contract address: 0xc2258126Ff4AE3Ed5e141F138a16b2b30ca08716
[filedate 1558371034] new contract address: 0x2ef34Cad340351Aa6fae1dB05F6c8c4a95eE939f

starting timer, at block 3 which has  1  transactions; at epochtime 1558371031.2742448

block 7  | new #TX  56 /  685 ms =  81.8 TPS_current | total: #TX  57 /  0.6 s =  88.0 TPS_average (peak is  88.0 TPS_average)
block 14 | new #TX 157 /  366 ms = 428.6 TPS_current | total: #TX 214 /  1.0 s = 205.6 TPS_average (peak is 205.6 TPS_average)
block 22 | new #TX 159 /  388 ms = 410.1 TPS_current | total: #TX 373 /  1.4 s = 261.6 TPS_average (peak is 261.6 TPS_average)
block 30 | new #TX 160 /  396 ms = 404.1 TPS_current | total: #TX 533 /  1.8 s = 299.3 TPS_average (peak is 299.3 TPS_average)
block 37 | new #TX 154 /  356 ms = 432.0 TPS_current | total: #TX 687 /  2.2 s = 319.5 TPS_average (peak is 319.5 TPS_average)
block 44 | new #TX 129 /  344 ms = 375.3 TPS_current | total: #TX 816 /  2.5 s = 325.4 TPS_average (peak is 325.4 TPS_average)
block 52 | new #TX 134 /  400 ms = 334.8 TPS_current | total: #TX 950 /  2.9 s = 332.4 TPS_average (peak is 332.4 TPS_average)
block 55 | new #TX 51 /  150 ms = 339.9 TPS_current | total: #TX 1001 /  3.2 s = 314.9 TPS_average (peak was 332.4 TPS_average)

Received signal from send.py = updated INFOFILE.
Experiment ended! Current blocknumber = 55
Updated info file: last-experiment.json THE END.

diagrams:
VM_Quorum_Dan_Thesis_30threads_1000txs blocks 4-55 with 1 txs ~ 0 txs/block

![Graphs showing TPS, blocktimes, blocksizes, and gas usage over time.](img/VM_Quorum_Dan_Thesis_30threads_1000txs-20190520-1650_blks4-55.png)

info raw

```json
```
(VM_Quorum_Dan_Thesis_30threads_10000txs) Quorum v1.8.12 with 10000 txs: 402.0 TPS

Information:

- **Node**: Geth/v1.8.12-stable/linux-amd64/g01.10.2 on http://10.5.62.60:22800
- **Consensus**: raft
- **NetworkID**: 92545
- **ChainID**: -1
- **Chain**: ???
- **Node**: Quorum, **NodeType**: Geth, **NodeVersion**: v1.8.12-stable, **Consensus**: raft, **Network**: 92545, **ChainID**: -1
- **Web3 Connection Established**: blockNumber = 2, node version string = Geth/v1.8.12-stable/linux-amd64/g01.10.2
- **First Account of Node**: 0x675850f0b906f3b9c04511a477e9e5e5a212f0, balance is 10000000000 Ether
- **Warn**: raft consensus did report timestamps in nanoseconds. Is that still the case?

Logs:

- **Block 2 - Waiting for something to happen**
- **Filedate**: 1558371034
- **Last Contract Address**: 0x2ef34Cad3a4915a1a6feae1d0b058ecbc8495e9f39f
- **Filedate**: 1558371118
- **New Contract Address**: 0xA93f1b15E7a7601C930dAee60Fe7AFA2BEBbAC10
- **Blocknumber start here**: 3
- **Starting Timer, at block 3 which has 1 transactions: at epochtime 1555371118.789346**
- **Block 10**
- **New Transactions**: 0x63b8f8fd6/Linux.gpp, testrpc 1.3.5, python 3.6.7 (default, Oct 22 2018, 11:32:17) [GCC 8.2.0]
- **Web3 Connection Established**: blockNumber = 2, node version string = Geth/v1.8.12-stable/linux-amd64/g01.10.2
- **First Account of Node**: 0x675850f0b906f3b9c04511a477e9e5e5a212f0, balance is 10000000000 Ether
- **Warn**: raft consensus did report timestamps in nanoseconds. Is that still the case?
- **Node Name**: Quorum, **Node Type**: Geth, **Node Version**: v1.8.12-stable, **Consensus**: raft, **Network**: 92545, **ChainID**: -1

- **Version Information**:
  - web3 4.8.2, py-solc: 3.2.0, solc 0.4.25+commit.59dbbf8f1, Linux.gpp, testrpc 1.3.5, python 3.6.7 (default, Oct 22 2018, 11:32:17) [GCC 8.2.0]
  - web3 connection established, blockNumber = 2, node version string = Geth/v1.8.12-stable/linux-amd64/g01.10.2
  - first account of node is 0x675850f0b906f3b9c04511a477e9e5e5a212f0, balance is 10000000000 Ether

Looking only at the experiment block timestamps, the overall TPS was ~402.0.
Received signal from send.py = updated INFOFILE.

Experiment ended! Current blocknumber = 503
Updated info file: last-experiment.json THE END.
VM_Quorum_Dan_Thesis_30threads_10000txs blocks 4-503 with 1 txs ~ 0 txs/block

avg TPS 402.0 = #TX whole experiment / blocktimes diff

blocktime seconds since last block

gasUsed and gasLimit per second

Quorum v1.8.12 with 100 txs: 414.0 TPS

**information:**

NODE: Geth/v1.8.12-stable/linux-amd64/go1.10.2 on http://10.5.62.60:22000

consensus=raft chain_name=??? chain_id=-1 network_id=80285
SEND: 100 transactions in Blocks 4-9 with 0 empty blocks following.
A sample of transactions looked as if they: succeeded.

TPS: The stopclock watcher measured a final TPS of 108.1 since contract deploy,
and in between saw values as high as 0.0 TPS.

DIAG: The whole experiment was prefixed 'VM_Quorum_Dan_Thesis_50threads_100txs'.
The diagrams were saved into 'img/VM_Quorum_Dan_Thesis_50threads_100txs-20190520-1653_blks4-9.png'.
Looking only at the experiment block-timestamps, the overall TPS was ~414.0.

**log:**

versions: web3 4.8.2, py-solc: 3.2.0, solc 0.4.25+commit.59db8ff1.Linux.gpp, testrpc 1.3.5, python 3.6.7 (default, Oct 22 2018, 11:32:17) [GCC 8.2.0]

web3 connection established, blockNumber = 2, node version string = Geth/v1.8.12-stable/linux-amd64/go1.10.2

first account of node is 0xF946aa12a3432536Fba6F27d5000fbE84F485f6d, balance is 1000000000 Ether

WARN: raft consensus did report timestamps in nanoseconds. Is that still the case?

nodeName: Quorum, nodeType: Geth, nodeVersion: v1.8.12-stable, consensus: raft, network: 80285, chainName: ???, chainId: -1

Block  2  - waiting for something to happen
[filedate 1558371110] last contract address: 0xA93f1b15E7a760c590cdaee60fe7a7a28e8bdc10
[filedate 1558371232] new contract address: 0x19F66cdB4A7Fcd3F43BB87568B5c0094031eCC7

blocknumber_start_here = 3

block 8 | new #TX  97 / 746 ms = 130.1 TPS_current | total: #TX  98 / 0.6 s = 156.8 TPS_average (peak is 156.8 TPS_average)
block 9 | new #TX  3 / 50 ms = 60.4 TPS_current | total: #TX 101 / 0.9 s = 108.1 TPS_average (peak is 108.1 TPS_average)

Received signal from send.py = updated INFOFILE.

Experiment ended! Current blocknumber = 9
Updated info file: last-experiment.json THE END.

diagrams:
VM_Quorum_Dan_Thesis_50threads_100txs blocks 4-9 with 1 txs ~ 0 txs/block

avg TPS 414.0 = #TX whole experiment / blocktimes diff

blocktime seconds since last block

gasUsed and gasLimit per second

info raw

```json
{ 'diagrams': { 'blocktimestampsTpsAv': 414.0469531505516, 'filename': 'img/VM_Quorum_Dan_Thesis_50threads_100txs-20190520-1653_blks4-9.png', 'prefix': 'VM_Quorum_Dan_Thesis_50threads_100txs'}, 'node': { 'chain_id': -1, 'chain_name': '???', 'consensus': 'raft', 'name': 'Quorum', 'network_id': 80285, 'rpc_address': 'http://10.5.62.60:22000', 'type': 'Geth', 'version': 'v1.8.12-stable', 'web3.version.node': 'Geth/v1.8.12-stable/linux-amd64/go1.10.2'}, 'send': { 'block_first': 4, 'block_last': 9, 'empty_blocks': 0, 'num_txs': 100, 'sample_txs_successful': True}, 'tps': { 'finalTpsAv': 108.14452998196055, 'peakTpsAv': 0, 'start_epochtime': 1558371232.3077273}}
```
Quorum v1.8.12 with 1000 txs: 476.5 TPS

information:

NODE: Geth/v1.8.12-stable/linux-amd64/go1.10.2 on http://10.5.62.60:22000  
consensus=raft chain_name=??? chain_id=-1 network_id=37357  
SEND: 1000 transactions in blocks 4-46 with 0 empty blocks following.  
A sample of transactions looked as if they: succeeded.  
TPS: The stopclock watcher measured a final TPS of 351.4 since contract deploy,  
and in between saw values as high as 386.8 TPS.  
DIAG: The whole experiment was prefixed 'VM_Quorum_Dan_Thesis_50threads_1000txs'.  
The diagrams were saved into 'img/VM_Quorum_Dan_Thesis_50threads_1000txs-20190520-1655_blks4-46.png'.  
Looking only at the experiment block-timestamps, the overall TPS was ~476.5.

log:

versions: web3 4.8.2, py-solc: 3.2.0, solc 0.4.25+commit.59db8f1f-linux.gpp, testrpc 1.3.5, python 3.6.7 (default, Oct 22 2018, 11:32:17) [GCC 8.2.0]  
web3 connection established, blockNumber = 2, node version string = Geth/v1.8.12-stable/linux-amd64/go1.10.2  
first account of node is 0x4566171DD41B444A048db46f1e69c4A815c7afC2, balance is 1000000000 Ether  
WARN: raft consensus did report timestamps in nanoseconds. Is that still the case?  

Block 2 - waiting for something to happen
[filedate 1558371232] last contract address: 0x19F66cdB4A7Fcd3F438B837568b5c0094931eCC7  
[filedate 1558371306] new contract address: 0xC067133c988557ad12f6e429F4538EB884F73988

Received signal from send.py = updated INFOFILE.  
Experiment ended! Current blocknumber = 46  
Updated info file: last-experiment.json THE END.

diagrams:
VM_Quorum_Dan_Thesis_50threads_1000txs blocks 4-46 with 2 txs ~ 0 txs/block

avg TPS 476.5 = #TX whole experiment / blocktimes diff

blocktime seconds since last block

gasUsed and gasLimit per second
(VM_Qorum_Dan_Thesis_50threads_10000txs) Quorum v1.8.12 with 10000 txs: 422.0 TPS

information:

NODE: Geth/v1.8.12-stable/linux-amd64/goi.10.2 on http://10.5.62.60:22000
consensus=raft chain_name=??? chain_id=1 network_id=72590
SEND: 10000 transactions in blocks 4-477 with 0 empty blocks following.
A sample of transactions looked as if they succeeded.

TPS: The stopclock watched measured a final TPS of 412.9 since contract deploy, and in between saw values as high as 416.2 TPS.

DIAG: The whole experiment was prefixed 'VM_Qorum_Dan_Thesis_50threads_10000txs'. The diagrams were saved into 'img/VM_Qorum_Dan_Thesis_50threads_10000txs-20190520-1656_blks4-477.png'.

Looking only at the experiment block-timestamps, the overall TPS was ~422.0.

log:

versions: web3 4.8.2, py-solc: 3.4.2, solc 0.4.25+commit.59dbff87f1 Linus.gpp, testrpc 1.3.5, python 3.6.7 (default, Oct 22 2018, 11:32:17) [GCC 8.2.0]
web3 connection established, blockNumber = 2, node version string = Geth/v1.8.12-stable/linux-amd64/goi.10.2
first account of node is 0x52ed8554ab8a6b23c7fa10945167fe85800000 balance is 1000000000 Ether

WARN: raft consensus did report timestamps in nanoseconds. Is that still the case?

nodeName: Qorum, nodeType: Geth, nodeVersion: v1.8.12-stable, consensus: raft, network: 72590, chainName: ??, chainId: -1

Block 2 - waiting for something to happen

information:

(VM_Qorum_Dan_Thesis_50threads_10000txs) Quorum v1.8.12 with 10000 txs: 422.0 TPS
block 341 | new #TX 140 / 350 ms = 399.7 TPS_current
block 349 | new #TX 160 / 400 ms = 400.1 TPS_current
block 357 | new #TX 139 / 400 ms = 347.6 TPS_current
block 365 | new #TX 163 / 400 ms = 407.7 TPS_current
block 374 | new #TX 150 / 450 ms = 333.3 TPS_current
block 383 | new #TX 174 / 450 ms = 386.8 TPS_current
block 390 | new #TX 142 / 350 ms = 405.7 TPS_current
block 398 | new #TX 179 / 400 ms = 447.0 TPS_current
block 406 | new #TX 202 / 400 ms = 505.2 TPS_current
block 414 | new #TX 181 / 400 ms = 452.7 TPS_current
block 422 | new #TX 166 / 400 ms = 489.5 TPS_current
block 430 | new #TX 208 / 495 ms = 428.3 TPS_current
block 440 | new #TX 217 / 500 ms = 433.7 TPS_current
block 448 | new #TX 217 / 407 ms = 418.3 TPS_current
block 457 | new #TX 213 / 450 ms = 473.4 TPS_current
block 466 | new #TX 198 / 450 ms = 373.2 TPS_current
block 477 | new #TX 173 / 547 ms = 316.5 TPS_current

Received signal from send.py = updated INFOFILE.
Experiment ended! Current blocknumber = 477
Updated info file: last-experiment.json THE END.

diagrams:
avg TPS 422.0 = #TX whole experiment / blocktimes diff

blocktime seconds since last block

gasUsed and gasLimit per second
Quorum v1.8.12 with 100 txs: 381.0 TPS

information:

NODE: Geth/v1.8.12-stable/linux-amd64/go1.10.2 on http://10.5.62.60:22000
  consensus=raft chain_name=??? chain_id=-1 network_id=69744
SEND: 100 transactions in Blocks 4-11 with 0 empty blocks following.
  A sample of transactions looked as if they: succeeded.
TPS: The stopwatch watcher measured a final TPS of 104.1 since contract deploy,
and in between saw values as high as 0.0 TPS.
DIAG: The whole experiment was prefixed 'VM_Quorum_Dan_Thesis_100threads_100txs'.
  The diagrams were saved into 'img/VM_Quorum_Dan_Thesis_100threads_100txs-20190520-1658_blks4-11.png'.
Looking only at the experiment block-timestamps, the overall TPS was ~381.0.

log:

versions: web3 4.8.2, py-solc: 3.2.0, solc 0.4.25+commit.59dbf8f1.Linux.gpp, testrpc 1.3.5, python 3.6.7 (default, Oct 22 2018, 11:32:17) [GCC 8.2.0]
web3 connection established, blockNumber = 2, node version string = Geth/v1.8.12-stable/linux-amd64/go1.10.2
  first account of node is 0x8b790bBB6B90670CBAaE98E5e2FFA17484479260, balance is 1000000000 Ether
WARN: raft consensus did report timestamps in nanoseconds. Is that still the case?
nodeName: Quorum, nodeType: Geth, nodeVersion: v1.8.12-stable, consensus: raft, network: 69744, chainName: ???, chainId: -1

Block  2  - waiting for something to happen
(filedate 1558371394) last contract address: 0xA261023bE3B577bb1c95B3bCEde66C4f19A0C650
(filedate 1558371505) new contract address: 0xc2a22d0ADE32A6E14FDba47748054a58ae62
blocknumber_start_here = 3
starting timer, at block 3 which has  1  transactions; at epochtime 1558371505.422505
block 7 | new #TX  24 /  640 ms =  37.5 TPS_current | total: #TX  25 /  0.7 s =  38.5 TPS_average (peak is  38.5 TPS_average)
block 11 | new #TX  76 /  198 ms =  384.0 TPS_current | total: #TX  101 /  1.0 s = 104.1 TPS_average (peak is 104.1 TPS_average)
Received signal from send.py = updated INFOFILE.
Experiment ended! Current blocknumber = 11
Updated info file: last-experiment.json THE END.

diagrams:
VM_Quorum_Dan_Thesis_100threads_100txs blocks 4-11 with 1 txs ~ 0 txs/block

avg TPS 381.0 = #TX whole experiment / blocktimes diff

blocksize in bytes

gasUsed and gasLimit per second

info raw

{"diagrams": {"blocktimestampsTpsAv": 380.97571215713333, "filename": "img/VM_Quorum_Dan_Thesis_100threads_100txs-20190520-1658_blks4-11.png", "prefix": "VM_Quorum_Dan_Thesis_100threads_100txs"}, "node": {"chain_id": -1, "chain_name": "???, "consensus": "raft", "name": "Quorum", "network_id": 69744, "rpc_address": "http://10.5.62.60:22000", "type": "Geth", "version": "v1.8.12-stable", "web3.version.node": "Geth/v1.8.12-stable/linux-amd64/go1.10.2"}, "send": {"block_first": 4, "block_last": 11, "empty_blocks": 0, "num_txs": 100, "sample_txs_successful": True}, "tps": {"finalTpsAv": 104.09588296927876, "peakTpsAv": 0, "start_epochtime": 1558371505.422505}}
Quorum v1.8.12 with 1000 txs: 409.9 TPS

information:

NODE: Geth/v1.8.12-stable/linux-amd64/go1.10.2 on http://10.5.62.60:22000
consensus=raft chain_name=??? chain_id=-1 network_id=43502
SEND: 1000 transactions in blocks 4-54 with 0 empty blocks following.
    A sample of transactions looked as if they: succeeded.
TPS:  The stopwatch watched measured a final TPS of 320.5 since contract deploy,
    and in between saw values as high as 323.6 TPS.
DIAG: The whole experiment was prefixed 'VM_Quorum_Dan_Thesis_100threads_1000txs'.
    The diagrams were saved into 'img/VM_Quorum_Dan_Thesis_100threads_1000txs-20190520-1659_blks4-54.png'.
    Looking only at the experiment block-timestamps, the overall TPS was ~409.9.

log:

versions: web3 4.8.2, py-solc: 3.2.0, solc 0.4.25+commit.59db8ff1.Linux.gpp, testrpc 1.3.5, python 3.6.7 (default, Oct 22 2018, 11:32:17) [GCC 8.2.0]
web3 connection established, blockNumber = 2, node version string = Geth/v1.8.12-stable/linux-amd64/go1.10.2
first account of node is 0x8B3786454BaF3Dc4Ae11539f9E51716D72BebD21, balance is 1000000000 Ether
WARN: raft consensus did report timestamps in nanoseconds. Is that still the case?
nodeName: Quorum, nodeType: Geth, nodeVersion: v1.8.12-stable, consensus: raft, network: 43502, chainName: ???, chainId: -1

Block 2 - waiting for something to happen
(filedate 1558371505) last contract address: 0xc2a22d0ADE32A6E14FDba47752874054a58ae62
(filedate 1558371560) new contract address: 0xbFd85A6985D9Dce043C86f04AC568aa1167110785
blocknumber_start_here = 3
starting timer, at block 3 which has 1 transactions; at epochtime 1558371560.513208
block 8 | new #TX 61 / 699 ms = 87.2 TPS_current | total: #TX 62 / 0.7 s = 91.1 TPS_average (peak is 91.1 TPS_average)
block 16 | new #TX 156 / 400 ms = 389.7 TPS_current | total: #TX 218 / 3.1 s = 194.6 TPS_average (peak is 194.6 TPS_average)
block 25 | new #TX 284 / 450 ms = 653.6 TPS_current | total: #TX 422 / 3.5 s = 273.7 TPS_average (peak is 273.7 TPS_average)
block 33 | new #TX 150 / 400 ms = 375.0 TPS_current | total: #TX 572 / 2.0 s = 291.5 TPS_average (peak is 291.5 TPS_average)
block 42 | new #TX 165 / 450 ms = 366.6 TPS_current | total: #TX 737 / 2.3 s = 315.3 TPS_average (peak is 315.3 TPS_average)
block 49 | new #TX 166 / 351 ms = 471.4 TPS_current | total: #TX 903 / 2.8 s = 323.6 TPS_average (peak is 323.6 TPS_average)
block 54 | new #TX 98 / 250 ms = 392.4 TPS_current | total: #TX 1001 / 3.1 s = 320.5 TPS_average (peak was 323.6 TPS_average)
Received signal from send.py = updated INFOFILE.
Experiment ended! Current blocknumber = 54
Updated info file: last-experiment.json THE END.

diagrams:
VM_Qorum_Dan_Thesis_100threads_1000txs blocks 4-54 with 1 txs ~ 0 txs/block

avg TPS 409.9 = #TX whole experiment / blocktimes diff

blocksize in bytes

gasUsed and gasLimit per second

info raw

 information:

NODE: Geth/v1.8.12-stable/linux-amd64/gol.10.2 on http://10.5.62.60:22008
consensus=raft, chain name=???, chain id=1 network id=33807
SEND: 10000 transactions in blocks 4-450 with 0 empty blocks following.
A sample of transactions looked as if they succeeded.

TPS: The stopclock watcher measured a final TPS of 433.3 since contract deploy,
and in between saw values as high as 436.3 TPS.

DIAG: The whole experiment was prefixed 'VM_Qorum_Dan_Thesis_100threads_10000txs'.
The diagrams were saved into 'img/VM_Qorum_Dan_Thesis_100threads_10000txs-20190520-1700_blks4-450.png'.
Looking only at the experiment block-timestamps, the overall TPS was 447.7.

log:

versions: web3 4.8.2, py-solc: 3.2.0, solc 0.4.25+commit.59db8ff1, Linux, geth-testrpc 1.3.5, python 3.6.7 (default, Oct 22 2018, 11:32:17) [GCC 8.2.0]
web3 connection established, blockNumber = 2, node version string = Geth/v1.8.12-stable/linux-amd64/gol.10.2
first account of node is 0x73f6eb65b2df0a46a59f7b742b36d4ae1a3e9a0f2, balance is 100000000 Ether
WARNING: raft consensus did report timestamps in nanoseconds. Is that still the case?

nodeName: Quorum, nodeType: Geth, nodeVersion: v1.8.12-stable, consensus: raft, network: 33807, chainName: ???, chainId: 1

Block 2 - waiting for something to happen

(filedate 1558371568) last contract address: 0xb7a2a1f96702460157e46c8643b7ef4a49df526617
(filedate 1558371621) new contract address: 0x53779a1193628b40a4b5596db797c523f526617

Blocknumber_start_here = 3
starting timer, at block 3 which has 1 transactions; at epochtime 1558375621.4790025

The diagrams were saved into 'img/VM_Quorum_Dan_Thesis_100threads_10000txs-20190520-1700_blks4-450.png'.

DiAG: The whole experiment was prefixed 'VM_Quorum_Dan_Thesis_100threads_10000txs'.
and in between saw values as high as 436.3 TPS.

TPS: The stopclock watcher measured a final TPS of 433.3 since contract deploy,
A sample of transactions looked as if they: succeeded.

information:

NODE: Geth/v1.8.12-stable/linux-amd64/gol.10.2 on http://10.5.62.60:22008
consensus=raft, chain name=???, chain id=1 network id=33807
SEND: 10000 transactions in blocks 4-450 with 0 empty blocks following.
A sample of transactions looked as if they succeeded.

TPS: The stopclock watcher measured a final TPS of 433.3 since contract deploy,
and in between saw values as high as 436.3 TPS.

DIAG: The whole experiment was prefixed 'VM_Quorum_Dan_Thesis_100threads_10000txs'.
The diagrams were saved into 'img/VM_Quorum_Dan_Thesis_100threads_10000txs-20190520-1700_blks4-450.png'.
Looking only at the experiment block-timestamps, the overall TPS was 447.7.
block 341 | new #TX 174 / 401 ms = 433.8 TPS_current | total: #TX 7438 / 17.5 s = 426.0 TPS_average (peak was 430.6 TPS_average)
block 350 | new #TX 206 / 448 ms = 460.1 TPS_current | total: #TX 7644 / 17.9 s = 426.2 TPS_average (peak was 430.6 TPS_average)
block 360 | new #TX 190 / 498 ms = 381.8 TPS_current | total: #TX 7834 / 18.4 s = 425.4 TPS_average (peak was 430.6 TPS_average)
block 370 | new #TX 192 / 501 ms = 383.2 TPS_current | total: #TX 8026 / 18.8 s = 426.5 TPS_average (peak was 430.6 TPS_average)
block 377 | new #TX 236 / 451 ms = 523.1 TPS_current | total: #TX 8433 / 19.7 s = 428.0 TPS_average (peak was 430.6 TPS_average)
block 386 | new #TX 196 / 446 ms = 439.3 TPS_current | total: #TX 8629 / 20.1 s = 428.3 TPS_average (peak was 430.6 TPS_average)
block 395 | new #TX 196 / 446 ms = 439.3 TPS_current | total: #TX 8629 / 20.1 s = 428.3 TPS_average (peak was 430.6 TPS_average)
block 399 | new #TX 196 / 446 ms = 439.3 TPS_current | total: #TX 8629 / 20.1 s = 428.3 TPS_average (peak was 430.6 TPS_average)
block 400 | new #TX 236 / 451 ms = 523.1 TPS_current | total: #TX 8433 / 19.7 s = 428.0 TPS_average (peak was 430.6 TPS_average)
block 408 | new #TX 19 / 46 ms = 415.1 TPS_current | total: #TX 8753 / 20.9 s = 419.5 TPS_average (peak was 430.6 TPS_average)
block 417 | new #TX 431 / 902 ms = 477.9 TPS_current | total: #TX 9184 / 21.4 s = 438.0 TPS_average (peak was 430.6 TPS_average)
block 427 | new #TX 265 / 498 ms = 531.8 TPS_current | total: #TX 9449 / 21.8 s = 433.2 TPS_average (peak is 433.2 TPS_average)
block 437 | new #TX 244 / 502 ms = 486.0 TPS_current | total: #TX 9693 / 22.3 s = 435.2 TPS_average (peak is 435.2 TPS_average)
block 446 | new #TX 223 / 450 ms = 495.6 TPS_current | total: #TX 9916 / 22.7 s = 436.3 TPS_average (peak is 436.3 TPS_average)
block 450 | new #TX 85 / 200 ms = 425.1 TPS_current | total: #TX 10001 / 23.1 s = 433.3 TPS_average (peak was 436.3 TPS_average)

Received signal from send.py = updated INFOFILE.
Experiment ended! Current blocknumber = 450
Updated info file: last-experiment.json THE END.

diagrams:
VM_Quorum_Dan_Thesis_100threads_10000txs blocks 4-450 with 1 txs ~ 0 txs/block

avg TPS 447.7 = #TX whole experiment / blocktimes diff

blocktimes seconds since last block

gasUsed and gasLimit per second
7.1.3 Cloud Instances
information:

NODE: Geth/v1.8.12-stable/linux-amd64/go1.10.2 on http://52.170.94.181:22000
consensus=raft chain_name=??? chain_id=-1 network_id=60733
SEND: 100 transactions in blocks 4-14 with 0 empty blocks following.
    A sample of transactions looked as if they: succeeded.
TPS: The stopclock watched measured a final TPS of 89.9 since contract deploy,
    and in between saw values as high as 0.0 TPS.
DIAG: The whole experiment was prefixed 'Quorum-Dan-Thesis-singlethread-10tx'.
    The diagrams were saved into 'img/Quorum-Dan-Thesis-singlethread-10tx-20190519-2353_blks4-14.png'.
    Looking only at the experiment block-timestamps, the overall TPS was ~235.9.

log:

versions: web3 4.8.2, py-solc: 3.2.0, solc 0.4.25+commit.59db8f8f1.Linux.gpp, testrpc 1.3.5, python 3.6.7 (default, Oct 22 2018, 11:32:17) [GCC 8.2.0]
web3 connection established, blockNumber = 2, node version string = Geth/v1.8.12-stable/linux-amd64/go1.10.2
first account of node is 0x6DC13b33b4245B8849ABA4974De9a6aA30CC5584, balance is 1000000000 Ether
WARN: raft consensus did report timestamps in nanoseconds. Is that still the case?
nodeName: Quorum, nodeType: Geth, nodeVersion: v1.8.12-stable, consensus: raft, network: 60733, chainName: ???, chainId: -1

Block 2 - waiting for something to happen
(filedate 1558309777) last contract address: 0x603a9A46cc4De92e60dDD3955a7eA5D384D6a40A
(filedate 1558310022) new contract address: 0xcb40D73e16507593B909875e3F8083B4EA02b6C9
blocknumber_start_here = 3
starting timer, at block 3 which has 1 transactions; at epochtime 1558310022.1387742
block 6 | new #TX 6 / 956 ms = 6.3 TPS_current | total: #TX 7 / 0.7 s = 9.4 TPS_average (peak is 9.4 TPS_average)
block 14 | new #TX 94 / 397 ms = 236.6 TPS_current | total: #TX 101 / 1.1 s = 89.9 TPS_average (peak is 89.9 TPS_average)
Received signal from send.py = updated INFOFILE.
Experiment ended! Current blocknumber = 14
Updated info file: last-experiment.json THE END.

diagrams:
Quorum-Dan-Thesis-singlethread-10tx blocks 4-14 with 1 txs ~ 0 txs/block
Quorum v1.8.12 with 1000 txs: 262.3 TPS

**information:**

NODE: Geth/v1.8.12-stable/linux-amd64/go1.10.2 on http://52.170.94.181:22000

consensus=raft chain_name=?? chain_id=-1 network_id=10705

SEND: 1000 transactions in blocks 4-82 with 0 empty blocks following.

A sample of transactions looked as if they: succeeded.

TPS: The stopclock watcher measured a final TPS of 212.7 since contract deploy, and in between saw values as high as 213.8 TPS.

DIAG: The whole experiment was prefixed 'Quorum-Dan-Thesis-10thread-1000tx'.

The diagrams were saved into 'img/Quorum-Dan-Thesis-10thread-1000tx-20190520-0005_blks4-82.png'.

Looking only at the experiment block-timestamps, the overall TPS was ~262.3.

**log:**

versions: web3 4.8.2, py-solc: 3.2.0, solc 0.4.25+commit.59db88f1.Linux.gpp, testrpc 1.3.5, python 3.6.7 (default, Oct 22 2018, 11:32:17) [GCC 8.2.0]

web3 connection established, blockNumber = 2, node version string = Geth/v1.8.12-stable/linux-amd64/go1.10.2

first account of node is 0xF99bE820105085Ffe0d1E886bf137Eb066b4D7f4, balance is 1000000000 Ether

WARN: raft consensus did report timestamps in nanoseconds. Is that still the case?

default account: Quorum, nodeType: Geth, nodeVersion: v1.8.12-stable, consensus: raft, network: 10705, chainName: ???, chainId: -1

Block 2 - waiting for something to happen

Experiment ended! Current blocknumber = 82

Updated info file: last-experiment.json THE END.

**diagrams:**
Quorum-Dan-Thesis-10thread-1000tx blocks 4-82 with 1 txs ~ 0 txs/block

avg TPS 262.3 = #TX whole experiment / blocktimes diff

blocktime seconds since last block

gasUsed and gasLimit per second

info raw

(Quorum-Dan-Thesis-10thread-10000tx) Quorum v1.8.12 with 10000 txs: 228.5 TPS

information:

NODE: Geth/v1.8.12-stable/linux-amd64/go1.10.2 on http://52.178.94.181:22800 consensus=raft chain_name=?? chain_id=1 network_id=65784
SEND: 10000 transactions in blocks 4-868 with 0 empty blocks following.
A sample of transactions looked as if they succeeded.
TPS: The stopclock watcher measured a final TPS of 224.0 since contract deploy, and in between saw values as high as 243.2 TPS.
DIAG: The whole experiment was prefixed 'Quorum-Dan-Thesis-10thread-10000tx'.
The diagrams were saved into 'img/Quorum-Dan-Thesis-18thread-10000tx-20190520-0013_blks4-868.png'.
Looking only at the experiment block-timestamps, the overall TPS was ~228.5.

log:

versions: web3 4.8.2, py-solc: 3.2.0, solc 0.4.25+commit.59dbf8f1.LinearLayout gp, testrpc 1.3.5, python 3.6.7 (default, Oct 22 2018, 11:32:17) [GCC 8.2.0]
web3 connection established, blockNumber = 2, node version string = Geth/v1.8.12-stable/linux-amd64/go1.10.2
first account of node is 0x243ff3049f5ed36a8e893d93759bdc9728d8b0d, balance is 100000000 Ether
WARN: raft consensus did report timestamps in nanoseconds. Is that still the case?

Block 2 - waiting for something to happen
(filedate 1558310748) last contract address: 0xaee8f620531D0cBeF87b88bC3e2A0809928feDE2
(filedate 1558311221) new contract address: 0xaee8f620531D0cBeF87b88bC3e2A0809928feDE2
blocknumber_start_here = 3
starting timer, at block 3 which has 1 transactions; at epochtime 1558311221.5993257
block 107  new #TX 123 / 458 ms = 273.2 TPS_current | total: #TX 127 / 5.9 s = 207.9 TPS_average (peak is 207.9 TPS_average)
block 117  new #TX 120 / 508 ms = 239.8 TPS_current | total: #TX 1346 / 6.5 s = 206.4 TPS_average (peak is 207.9 TPS_average)
block 130  new #TX 144 / 659 ms = 218.4 TPS_current | total: #TX 1498 / 7.1 s = 208.4 TPS_average (peak is 208.4 TPS_average)
block 143  new #TX 155 / 646 ms = 248.0 TPS_current | total: #TX 1645 / 7.8 s = 211.0 TPS_average (peak is 211.0 TPS_average)
block 156  new #TX 150 / 656 ms = 228.7 TPS_current | total: #TX 1795 / 8.4 s = 213.5 TPS_average (peak is 213.5 TPS_average)
block 168  new #TX 141 / 593 ms = 237.9 TPS_current | total: #TX 1936 / 9.1 s = 213.7 TPS_average (peak is 213.7 TPS_average)
block 181  new #TX 158 / 648 ms = 243.9 TPS_current | total: #TX 2094 / 9.7 s = 216.2 TPS_average (peak is 216.2 TPS_average)
block 193  new #TX 145 / 608 ms = 241.0 TPS_current | total: #TX 2272 / 10.3 s = 217.2 TPS_average (peak is 217.2 TPS_average)
block 206  new #TX 157 / 648 ms = 242.1 TPS_current | total: #TX 2396 / 10.8 s = 221.1 TPS_average (peak is 221.1 TPS_average)
block 218  new #TX 134 / 563 ms = 256.5 TPS_current | total: #TX 2545 / 11.3 s = 225.5 TPS_average (peak is 225.5 TPS_average)
block 226  new #TX 112 / 447 ms = 258.5 TPS_current | total: #TX 2771 / 11.7 s = 226.4 TPS_average (peak is 226.4 TPS_average)
block 235  new #TX 114 / 458 ms = 253.3 TPS_current | total: #TX 2895 / 12.6 s = 229.4 TPS_average (peak is 229.4 TPS_average)
block 252  new #TX 126 / 459 ms = 289.0 TPS_current | total: #TX 3136 / 13.6 s = 238.9 TPS_average (peak is 238.9 TPS_average)
block 262  new #TX 115 / 453 ms = 254.1 TPS_current | total: #TX 3136 / 13.6 s = 238.9 TPS_average (peak is 238.9 TPS_average)
block 272  new #TX 134 / 497 ms = 269.5 TPS_current | total: #TX 3278 / 14.0 s = 233.0 TPS_average (peak is 233.0 TPS_average)
block 281  new #TX 117 / 456 ms = 268.0 TPS_current | total: #TX 3387 / 14.5 s = 233.6 TPS_average (peak is 233.6 TPS_average)
block 289  new #TX 112 / 450 ms = 268.9 TPS_current | total: #TX 3392 / 14.5 s = 233.6 TPS_average (peak is 233.6 TPS_average)
block 299  new #TX 119 / 450 ms = 264.5 TPS_current | total: #TX 3627 / 15.4 s = 235.4 TPS_average (peak is 235.4 TPS_average)
block 308  new #TX 125 / 453 ms = 276.2 TPS_current | total: #TX 3752 / 15.9 s = 236.7 TPS_average (peak is 236.7 TPS_average)
block 317  new #TX 123 / 447 ms = 274.9 TPS_current | total: #TX 3875 / 16.3 s = 237.7 TPS_average (peak is 237.7 TPS_average)
block 326  new #TX 116 / 450 ms = 257.7 TPS_current | total: #TX 3991 / 16.7 s = 238.3 TPS_average (peak is 238.3 TPS_average)
block 335  new #TX 112 / 450 ms = 268.9 TPS_current | total: #TX 4112 / 17.2 s = 239.0 TPS_average (peak is 239.0 TPS_average)
block 344  new #TX 114 / 458 ms = 253.3 TPS_current | total: #TX 4226 / 17.7 s = 239.8 TPS_average (peak is 239.8 TPS_average)
block 354  new #TX 144 / 508 ms = 288.0 TPS_current | total: #TX 4378 / 18.2 s = 240.4 TPS_average (peak is 240.4 TPS_average)
block 363  new #TX 119 / 452 ms = 263.3 TPS_current | total: #TX 4639 / 18.6 s = 240.9 TPS_average (peak is 240.9 TPS_average)
block 373  new #TX 136 / 496 ms = 227.8 TPS_current | total: #TX 4625 / 19.1 s = 242.2 TPS_average (peak is 242.2 TPS_average)
block 382  new #TX 122 / 450 ms = 271.2 TPS_current | total: #TX 4747 / 19.5 s = 242.9 TPS_average (peak is 242.9 TPS_average)
block 391  new #TX 118 / 452 ms = 261.3 TPS_current | total: #TX 4865 / 20.0 s = 243.2 TPS_average (peak is 243.2 TPS_average)
block 400  new #TX 115 / 468 ms = 249.9 TPS_current | total: #TX 4900 / 20.5 s = 246.2 TPS_average (peak is 246.2 TPS_average)
block 411  new #TX 114 / 543 ms = 209.9 TPS_current | total: #TX 5094 / 21.3 s = 239.1 TPS_average (peak is 239.2 TPS_average)

Looking only at the experiment block-timestamps, the overall TPS was ~228.5.
block 422 | new #TX 144 / 701 ms = 205.5 TPS_current | total: #TX 5238 / 21.9 s = 239.7 TPS_average (peak was 243.2 TPS_average)
block 434 | new #TX 119 / 598 ms = 199.1 TPS_current | total: #TX 5357 / 22.5 s = 238.6 TPS_average (peak was 243.2 TPS_average)
block 446 | new #TX 129 / 640 ms = 201.7 TPS_current | total: #TX 5486 / 23.2 s = 236.3 TPS_average (peak was 243.2 TPS_average)
block 460 | new #TX 164 / 773 ms = 212.2 TPS_current | total: #TX 5650 / 23.9 s = 236.2 TPS_average (peak was 243.2 TPS_average)
block 474 | new #TX 139 / 683 ms = 203.4 TPS_current | total: #TX 5789 / 24.6 s = 235.2 TPS_average (peak was 243.2 TPS_average)
block 486 | new #TX 134 / 654 ms = 205.0 TPS_current | total: #TX 5923 / 25.1 s = 235.9 TPS_average (peak was 243.2 TPS_average)
block 505 | new #TX 83 / 449 ms = 184.7 TPS_current | total: #TX 6103 / 26.2 s = 233.2 TPS_average (peak was 243.2 TPS_average)
block 516 | new #TX 139 / 669 ms = 217.7 TPS_current | total: #TX 6246 / 26.7 s = 234.0 TPS_average (peak was 243.2 TPS_average)
block 539 | new #TX 125 / 635 ms = 205.7 TPS_current | total: #TX 6491 / 27.4 s = 232.5 TPS_average (peak was 243.2 TPS_average)
block 550 | new #TX 109 / 551 ms = 197.8 TPS_current | total: #TX 6600 / 27.9 s = 232.4 TPS_average (peak was 243.2 TPS_average)
block 570 | new #TX 143 / 616 ms = 221.1 TPS_current | total: #TX 6800 / 28.4 s = 234.0 TPS_average (peak was 243.2 TPS_average)
block 594 | new #TX 118 / 608 ms = 194.2 TPS_current | total: #TX 7010 / 29.1 s = 233.7 TPS_average (peak was 243.2 TPS_average)
block 614 | new #TX 138 / 603 ms = 212.8 TPS_current | total: #TX 7238 / 29.8 s = 232.9 TPS_average (peak was 243.2 TPS_average)
block 634 | new #TX 125 / 635 ms = 203.3 TPS_current | total: #TX 7462 / 30.5 s = 233.2 TPS_average (peak was 243.2 TPS_average)
block 656 | new #TX 118 / 608 ms = 192.6 TPS_current | total: #TX 7692 / 31.1 s = 231.7 TPS_average (peak was 243.2 TPS_average)
block 676 | new #TX 125 / 635 ms = 203.3 TPS_current | total: #TX 7922 / 31.7 s = 232.4 TPS_average (peak was 243.2 TPS_average)
block 698 | new #TX 143 / 616 ms = 221.1 TPS_current | total: #TX 8150 / 32.3 s = 234.0 TPS_average (peak was 243.2 TPS_average)
block 721 | new #TX 118 / 608 ms = 194.2 TPS_current | total: #TX 8380 / 32.9 s = 233.7 TPS_average (peak was 243.2 TPS_average)
block 747 | new #TX 143 / 616 ms = 221.1 TPS_current | total: #TX 8610 / 33.5 s = 234.0 TPS_average (peak was 243.2 TPS_average)
block 767 | new #TX 118 / 608 ms = 192.6 TPS_current | total: #TX 8840 / 34.1 s = 233.2 TPS_average (peak was 243.2 TPS_average)
block 787 | new #TX 125 / 635 ms = 203.3 TPS_current | total: #TX 9070 / 34.7 s = 233.7 TPS_average (peak was 243.2 TPS_average)
block 808 | new #TX 143 / 616 ms = 221.1 TPS_current | total: #TX 9300 / 35.3 s = 234.0 TPS_average (peak was 243.2 TPS_average)
block 828 | new #TX 125 / 635 ms = 203.3 TPS_current | total: #TX 9530 / 35.9 s = 233.7 TPS_average (peak was 243.2 TPS_average)
block 848 | new #TX 143 / 616 ms = 221.1 TPS_current | total: #TX 9760 / 36.5 s = 234.0 TPS_average (peak was 243.2 TPS_average)
block 872 | new #TX 118 / 608 ms = 194.2 TPS_current | total: #TX 9990 / 37.1 s = 233.7 TPS_average (peak was 243.2 TPS_average)
block 892 | new #TX 143 / 616 ms = 221.1 TPS_current | total: #TX 10220 / 37.7 s = 234.0 TPS_average (peak was 243.2 TPS_average)

Received signal from send.py = updated INFOFILE.
Experiment ended! Current blocknumber = 868
Updated info file: last-experiment.json THE END.
Quorum-Dan-Thesis-10thread-10000tx blocks 4-868 with 1 txs ~ 0 txs/block

avg TPS 228.5 = #TX whole experiment / blocktimes diff

blocktime seconds since last block

gasUsed and gasLimit per second

info raw

Quorum v1.8.12 with 100 txs: 245.4 TPS

Information:

Node: Geth/v1.8.12-stable/linux-amd64/go1.10.2 on http://52.170.94.181:22000
Consensus: raft, chain_name: ???, chain_id: -1, network_id: 32160
Send: 100 transactions in blocks 4-14 with 0 empty blocks following.
A sample of transactions looked as if they succeeded.
TPS: The stopclock watcher measured a final TPS of 96.2 since contract deploy,
and in between saw values as high as 0.0 TPS.
Diag: The whole experiment was prefixed 'Quorum-Dan-Thesis-30thread-100tx'.
The diagrams were saved into 'img/Quorum-Dan-Thesis-30thread-100tx-20190520-0026_blk14.png'.
Looking only at the experiment block-timestamps, the overall TPS was ~245.4.

Log:

Versions: web3 4.8.2, py-solc: 3.2.0, solc 0.4.25+commit.59dbf8ff.Linux.gpp, testrpc 1.3.5, python 3.6.7 (default, Oct 22 2018, 11:32:17) [GCC 8.2.0]
Web3 connection established, blockNumber = 2, node version string = Geth/v1.8.12-stable/linux-amd64/go1.10.2
First account of node is 0x125463c2452cC60284cB1E17dad33d49a706db96, balance is 1000000000 Ether
warn: raft consensus did report timestamps in nanoseconds. Is that still the case?
nodeName: Quorum, nodeType: Geth, nodeVersion: v1.8.12-stable, consensus: raft, network: 32160, chainName: ???, chainId: -1

Block 2 - waiting for something to happen

Block number start here = 3
Starting timer, at block 3 which has 1 transactions; at epochtime 1558311971.1954312
Block 6 | new #TX 3 / 726 ms = 4.1 TPS_current | total: #TX 4 / 0.7 s = 5.9 TPS_average (peak is 5.9 TPS_average)
Block 14 | new #TX 97 / 400 ms = 242.8 TPS_current | total: #TX 101 / 1.0 s = 96.2 TPS_average (peak is 96.2 TPS_average)

Received signal from send.py = updated INFOFILE.
Experiment ended! Current blocknumber = 14
Updated info file: last-experiment.json THE END.

Diagrams:
Quorum-Dan-Thesis-30thread-100tx blocks 4-14 with 1 txs ~ 0 txs/block

avg TPS 245.4 = #TX whole experiment / blocktimes diff

blocktime seconds since last block

gasUsed and gasLimit per second

info raw

Quorum v1.8.12 with 1000 txs: 254.3 TPS

NODE: Geth/v1.8.12-stable/linux-amd64/go1.10.2 on http://52.170.94.181:22000
consensus=raft chain_name=??? chain_id=-1 network_id=17788
SEND: 1000 transactions in blocks 4-84 with 0 empty blocks following.
A sample of transactions looked as if they: succeeded.
TPS: The stopclock watcher measured a final TPS of 203.6 since contract deploy,
and in between saw values as high as 211.3 TPS.
DIAG: The whole experiment was prefixed 'Quorum-Dan-Thesis-30thread-1000tx'.
The diagrams were saved into 'img/Quorum-Dan-Thesis-30thread-1000tx-20190520-0037_blks4-84.png'.
Looking only at the experiment block-timestamps, the overall TPS was ~254.3.

log:

versions: web3 4.8.2, py-solc: 3.2.0, solc 0.4.25+commit.59df8ff1.Linux.gpp, testrpc 1.3.5, python 3.6.7 (default, Oct 22 2018, 11:32:17) [GCC 8.2.0]
web3 connection established, blockNumber = 2, node version string = Geth/v1.8.12-stable/linux-amd64/go1.10.2
first account of node is 0x2652E60132482289e39E11f66911F3591f8bd84e, balance is 1000000000 Ether
WARN: raft consensus did report timestamps in nanoseconds. Is that still the case?
Block 2 - waiting for something to happen
(filedate 1558311971) last contract address: 0xcdB0314540e4020aca29B9D1f9Af3E6dbc6dc67e
(filedate 1558312665) new contract address: 0x2D376682f1B6cfb7C1FA5bdF79F07951c886D04a
blocknumber_start_here = 3
starting timer, at block 3 which has 1 transactions; at epochtime 1558312665.114366
block 11 | new #TX 70 / 1017 ms = 68.8 TPS_current | total: #TX 71 / 1.0 s = 67.7 TPS_average (peak is 67.7 TPS_average)
block 19 | new #TX 109 / 482 ms = 271.0 TPS_current | total: #TX 180 / 1.5 s = 124.0 TPS_average (peak is 124.0 TPS_average)
block 26 | new #TX 90 / 348 ms = 258.3 TPS_current | total: #TX 270 / 2.1 s = 131.3 TPS_average (peak is 131.3 TPS_average)
block 38 | new #TX 127 / 616 ms = 206.0 TPS_current | total: #TX 397 / 2.6 s = 151.5 TPS_average (peak is 151.5 TPS_average)
block 51 | new #TX 175 / 633 ms = 276.5 TPS_current | total: #TX 572 / 3.1 s = 181.7 TPS_average (peak is 181.7 TPS_average)
block 61 | new #TX 131 / 500 ms = 262.0 TPS_current | total: #TX 703 / 3.7 s = 190.9 TPS_average (peak is 190.9 TPS_average)
block 72 | new #TX 146 / 552 ms = 264.3 TPS_current | total: #TX 849 / 4.1 s = 205.9 TPS_average (peak is 205.9 TPS_average)
block 81 | new #TX 117 / 448 ms = 261.4 TPS_current | total: #TX 966 / 4.6 s = 211.3 TPS_average (peak is 211.3 TPS_average)
block 84 | new #TX 35 / 151 ms = 232.4 TPS_current | total: #TX 1001 / 4.9 s = 203.6 TPS_average (peak was 211.3 TPS_average)
Received signal from send.py = updated INFOFILE.
Experiment ended! Current blocknumber = 84
Updated info file: last-experiment.json THE END.
diagrams:
Quorum-Dan-Thesis-30thread-1000tx blocks 4-84 with 1 txs ~ 0 txs/block

avg TPS 254.3 = #TX whole experiment / blocktimes diff

blocktime seconds since last block

gasUsed and gasLimit per second
(Quorum-Dan-Thesis-30thread-10000tx) Quorum v1.8.12 with 10000 txs: 219.7 TPS

information:

NODE: Geth/v1.8.12-stable/linux-amd64/gol1.10.2 on http://52.178.94.181:22800
consensus=raft, chain_name=??, chain_id=1, network_id=52538
SEND: 10000 transactions in blocks 4-881 with 0 empty blocks following.

A sample of transactions looked as if they succeeded.

TPS: The stopclock watcher measured a final TPS of 214.8 since contract deploy,
and in between saw values as high as 241.4 TPS.

DIAG: The whole experiment was prefixed 'Quorum-Dan-Thesis-30thread-10000tx'.

The diagrams were saved into `img/Quorum-Dan-Thesis-10000tx-20180529-0020_bls4-881.png'.
Looking only at the experiment block-timestamps, the overall TPS was 219.7.

log:

versions: web3 4.8.2, py-solc: 3.2.0, solc 0.4.25+commit.59dbf8f1.Linux.gpp, testrpc 1.3.5, python 3.6.7 (default, Oct 22 2018, 11:32:17) [GCC 8.2.0]
web3 connection established, blockNumber = 2, node version string = Geth/v1.8.12-stable/linux-amd64/gol1.10.2
first account of node is 0x99141d2f1431F61359Fc3fE5b4D2799329615f8B, balance is 1000000000 Ether
WARN: raft consensus did report timestamps in nanoseconds. Is that still the case?

nodeName: Queth, nodeType: Geth, nodeversion: v1.8.12-stable, consensus: raft, network: 52538, chainName: ???, chainId: -1

Block 2 - waiting for something to happen

(nodeDate 1558311612) new contract address: 0xEE50faf32B79f2e63355C895509b9Db270654f68
(nodeDate 1558311221) last contract address: 0xaee8f620531D0cBeF87b88BC3e2A0809928FeDE2

blocknumber_start_here = 3
starting timer, at block 3 which has 1 transactions; at epochtime 1558311612.0192466

block 5 | new #TX | 2 / 849 ms = 2.4 TPS_current | total: #TX | 3 / 0.7 s = 4.6 TPS_average | peak is 4.6 TPS_average
block 10 | new #TX | 6 / 346 ms = 1.7 TPS_current | total: #TX | 6 / 0.9 s = 2.1 TPS_average | peak is 2.1 TPS_average
block 100 | new #TX | 123 / 280 ms = 1.1 TPS_current | total: #TX | 126 / 2.7 s = 2.1 TPS_average | peak is 2.1 TPS_average
block 257 | new #TX | 74 / 257 ms = 0.3 TPS_current | total: #TX | 74 / 0.3 s = 0.3 TPS_average | peak is 0.3 TPS_average
block 266 | new #TX | 127 / 233 ms = 0.5 TPS_current | total: #TX | 127 / 0.5 s = 0.5 TPS_average | peak is 0.5 TPS_average
block 276 | new #TX | 126 / 232 ms = 0.5 TPS_current | total: #TX | 126 / 0.5 s = 0.5 TPS_average | peak is 0.5 TPS_average
block 284 | new #TX | 103 / 257 ms = 0.4 TPS_current | total: #TX | 103 / 0.4 s = 0.4 TPS_average | peak is 0.4 TPS_average
block 297 | new #TX | 153 / 232 ms = 0.5 TPS_current | total: #TX | 153 / 0.5 s = 0.5 TPS_average | peak is 0.5 TPS_average
block 309 | new #TX | 111 / 282 ms = 0.4 TPS_current | total: #TX | 111 / 0.4 s = 0.4 TPS_average | peak is 0.4 TPS_average
block 318 | new #TX | 83 / 171 ms = 0.5 TPS_current | total: #TX | 83 / 0.2 s = 0.3 TPS_average | peak is 0.3 TPS_average
block 327 | new #TX | 85 / 180 ms = 0.5 TPS_current | total: #TX | 85 / 0.2 s = 0.3 TPS_average | peak is 0.3 TPS_average
block 338 | new #TX | 106 / 155 ms = 0.7 TPS_current | total: #TX | 106 / 0.2 s = 0.3 TPS_average | peak is 0.3 TPS_average
block 348 | new #TX | 101 / 208 ms = 0.5 TPS_current | total: #TX | 101 / 0.2 s = 0.3 TPS_average | peak is 0.3 TPS_average
block 357 | new #TX | 87 / 194 ms = 0.5 TPS_current | total: #TX | 87 / 0.2 s = 0.3 TPS_average | peak is 0.3 TPS_average
block 365 | new #TX | 80 / 193 ms = 0.5 TPS_current | total: #TX | 80 / 0.2 s = 0.3 TPS_average | peak is 0.3 TPS_average
block 374 | new #TX | 78 / 175 ms = 0.5 TPS_current | total: #TX | 78 / 0.2 s = 0.3 TPS_average | peak is 0.3 TPS_average
block 382 | new #TX | 70 / 172 ms = 0.5 TPS_current | total: #TX | 70 / 0.2 s = 0.3 TPS_average | peak is 0.3 TPS_average
block 390 | new #TX | 73 / 176 ms = 0.5 TPS_current | total: #TX | 73 / 0.2 s = 0.3 TPS_average | peak is 0.3 TPS_average
block 402 | new #TX | 144 / 223 ms = 0.6 TPS_current | total: #TX | 144 / 0.3 s = 0.4 TPS_average | peak is 0.4 TPS_average
block 410 | new #TX | 82 / 185 ms = 0.5 TPS_current | total: #TX | 82 / 0.2 s = 0.3 TPS_average | peak is 0.3 TPS_average

starting timer, at block 3 which has 1 transactions; at epochtime 1558311612.0192466

Block 2 - waiting for something to happen

(nodeDate 1558311612) new contract address: 0xEE50faf32B79f2e63355C895509b9Db270654f68
(nodeDate 1558311221) last contract address: 0xaee8f620531D0cBeF87b88BC3e2A0809928FeDE2

blocknumber_start_here = 3
starting timer, at block 3 which has 1 transactions; at epochtime 1558311612.0192466

Block 2 - waiting for something to happen

(nodeDate 1558311612) new contract address: 0xEE50faf32B79f2e63355C895509b9Db270654f68
(nodeDate 1558311221) last contract address: 0xaee8f620531D0cBeF87b88BC3e2A0809928FeDE2

blocknumber_start_here = 3
starting timer, at block 3 which has 1 transactions; at epochtime 1558311612.0192466

Block 2 - waiting for something to happen
<table>
<thead>
<tr>
<th>Block</th>
<th>New TX</th>
<th>Time (ms)</th>
<th>TPS Current</th>
<th>Total TX</th>
<th>Time (s)</th>
<th>TPS Average</th>
<th>Peak was</th>
</tr>
</thead>
<tbody>
<tr>
<td>410</td>
<td>519</td>
<td>244.7</td>
<td>223.7</td>
<td>1115</td>
<td>22.1</td>
<td>241.4</td>
<td>241.4</td>
</tr>
<tr>
<td>342</td>
<td>793</td>
<td>214.5</td>
<td>229.6</td>
<td>881</td>
<td>22.9</td>
<td>241.4</td>
<td>241.4</td>
</tr>
<tr>
<td>447</td>
<td>792</td>
<td>228.9</td>
<td>225.5</td>
<td>881</td>
<td>23.4</td>
<td>241.4</td>
<td>241.4</td>
</tr>
<tr>
<td>459</td>
<td>680</td>
<td>208.4</td>
<td>225.8</td>
<td>881</td>
<td>24.5</td>
<td>241.4</td>
<td>241.4</td>
</tr>
<tr>
<td>469</td>
<td>496</td>
<td>191.5</td>
<td>224.3</td>
<td>881</td>
<td>24.5</td>
<td>241.4</td>
<td>241.4</td>
</tr>
<tr>
<td>481</td>
<td>616</td>
<td>258.0</td>
<td>224.4</td>
<td>881</td>
<td>25.2</td>
<td>241.4</td>
<td>241.4</td>
</tr>
<tr>
<td>490</td>
<td>433</td>
<td>246.3</td>
<td>224.4</td>
<td>881</td>
<td>24.9</td>
<td>241.4</td>
<td>241.4</td>
</tr>
<tr>
<td>516</td>
<td>800</td>
<td>218.8</td>
<td>224.4</td>
<td>881</td>
<td>26.4</td>
<td>241.4</td>
<td>241.4</td>
</tr>
<tr>
<td>519</td>
<td>670</td>
<td>207.6</td>
<td>223.1</td>
<td>881</td>
<td>27.2</td>
<td>241.4</td>
<td>241.4</td>
</tr>
<tr>
<td>529</td>
<td>742</td>
<td>229.2</td>
<td>224.5</td>
<td>881</td>
<td>27.8</td>
<td>241.4</td>
<td>241.4</td>
</tr>
<tr>
<td>541</td>
<td>593</td>
<td>198.9</td>
<td>224.3</td>
<td>881</td>
<td>28.3</td>
<td>241.4</td>
<td>241.4</td>
</tr>
<tr>
<td>553</td>
<td>595</td>
<td>184.9</td>
<td>223.7</td>
<td>881</td>
<td>28.9</td>
<td>241.4</td>
<td>241.4</td>
</tr>
<tr>
<td>563</td>
<td>583</td>
<td>181.0</td>
<td>221.4</td>
<td>881</td>
<td>29.6</td>
<td>241.4</td>
<td>241.4</td>
</tr>
<tr>
<td>575</td>
<td>789</td>
<td>228.2</td>
<td>221.4</td>
<td>881</td>
<td>30.3</td>
<td>241.4</td>
<td>241.4</td>
</tr>
<tr>
<td>586</td>
<td>541</td>
<td>218.2</td>
<td>221.5</td>
<td>881</td>
<td>30.8</td>
<td>241.4</td>
<td>241.4</td>
</tr>
<tr>
<td>600</td>
<td>698</td>
<td>199.1</td>
<td>222.3</td>
<td>881</td>
<td>31.4</td>
<td>241.4</td>
<td>241.4</td>
</tr>
<tr>
<td>607</td>
<td>467</td>
<td>205.6</td>
<td>220.1</td>
<td>881</td>
<td>31.1</td>
<td>241.4</td>
<td>241.4</td>
</tr>
<tr>
<td>621</td>
<td>784</td>
<td>216.8</td>
<td>221.7</td>
<td>881</td>
<td>32.6</td>
<td>241.4</td>
<td>241.4</td>
</tr>
<tr>
<td>629</td>
<td>595</td>
<td>225.6</td>
<td>221.5</td>
<td>881</td>
<td>33.2</td>
<td>241.4</td>
<td>241.4</td>
</tr>
<tr>
<td>649</td>
<td>451</td>
<td>202.0</td>
<td>221.3</td>
<td>881</td>
<td>33.1</td>
<td>241.4</td>
<td>241.4</td>
</tr>
<tr>
<td>658</td>
<td>457</td>
<td>188.1</td>
<td>220.7</td>
<td>881</td>
<td>34.6</td>
<td>241.4</td>
<td>241.4</td>
</tr>
<tr>
<td>668</td>
<td>492</td>
<td>184.8</td>
<td>220.6</td>
<td>881</td>
<td>35.6</td>
<td>241.4</td>
<td>241.4</td>
</tr>
<tr>
<td>677</td>
<td>455</td>
<td>219.8</td>
<td>219.8</td>
<td>881</td>
<td>35.6</td>
<td>241.4</td>
<td>241.4</td>
</tr>
<tr>
<td>687</td>
<td>598</td>
<td>205.6</td>
<td>220.4</td>
<td>881</td>
<td>36.1</td>
<td>241.4</td>
<td>241.4</td>
</tr>
<tr>
<td>695</td>
<td>447</td>
<td>194.6</td>
<td>220.1</td>
<td>881</td>
<td>36.5</td>
<td>241.4</td>
<td>241.4</td>
</tr>
<tr>
<td>704</td>
<td>467</td>
<td>191.5</td>
<td>220.2</td>
<td>881</td>
<td>37.1</td>
<td>241.4</td>
<td>241.4</td>
</tr>
<tr>
<td>715</td>
<td>605</td>
<td>191.6</td>
<td>219.3</td>
<td>881</td>
<td>37.6</td>
<td>241.4</td>
<td>241.4</td>
</tr>
<tr>
<td>725</td>
<td>497</td>
<td>193.3</td>
<td>219.1</td>
<td>881</td>
<td>38.1</td>
<td>241.4</td>
<td>241.4</td>
</tr>
<tr>
<td>734</td>
<td>459</td>
<td>189.5</td>
<td>218.5</td>
<td>881</td>
<td>38.6</td>
<td>241.4</td>
<td>241.4</td>
</tr>
<tr>
<td>743</td>
<td>516</td>
<td>201.6</td>
<td>216.9</td>
<td>881</td>
<td>39.4</td>
<td>241.4</td>
<td>241.4</td>
</tr>
<tr>
<td>755</td>
<td>680</td>
<td>223.7</td>
<td>217.7</td>
<td>881</td>
<td>39.9</td>
<td>241.4</td>
<td>241.4</td>
</tr>
<tr>
<td>768</td>
<td>643</td>
<td>209.9</td>
<td>217.4</td>
<td>881</td>
<td>40.6</td>
<td>241.4</td>
<td>241.4</td>
</tr>
<tr>
<td>781</td>
<td>658</td>
<td>213.7</td>
<td>217.8</td>
<td>881</td>
<td>41.2</td>
<td>241.4</td>
<td>241.4</td>
</tr>
<tr>
<td>793</td>
<td>599</td>
<td>211.8</td>
<td>216.7</td>
<td>881</td>
<td>42.0</td>
<td>241.4</td>
<td>241.4</td>
</tr>
<tr>
<td>805</td>
<td>750</td>
<td>213.3</td>
<td>218.0</td>
<td>881</td>
<td>42.4</td>
<td>241.4</td>
<td>241.4</td>
</tr>
<tr>
<td>814</td>
<td>453</td>
<td>201.0</td>
<td>216.8</td>
<td>881</td>
<td>43.1</td>
<td>241.4</td>
<td>241.4</td>
</tr>
<tr>
<td>826</td>
<td>698</td>
<td>203.3</td>
<td>217.7</td>
<td>881</td>
<td>43.6</td>
<td>241.4</td>
<td>241.4</td>
</tr>
<tr>
<td>835</td>
<td>440</td>
<td>188.3</td>
<td>217.2</td>
<td>881</td>
<td>44.0</td>
<td>241.4</td>
<td>241.4</td>
</tr>
<tr>
<td>846</td>
<td>597</td>
<td>196.6</td>
<td>216.9</td>
<td>881</td>
<td>44.6</td>
<td>241.4</td>
<td>241.4</td>
</tr>
<tr>
<td>856</td>
<td>483</td>
<td>192.4</td>
<td>216.3</td>
<td>881</td>
<td>45.2</td>
<td>241.4</td>
<td>241.4</td>
</tr>
<tr>
<td>867</td>
<td>549</td>
<td>206.2</td>
<td>216.5</td>
<td>881</td>
<td>45.6</td>
<td>241.4</td>
<td>241.4</td>
</tr>
<tr>
<td>876</td>
<td>454</td>
<td>178.6</td>
<td>216.2</td>
<td>881</td>
<td>45.4</td>
<td>241.4</td>
<td>241.4</td>
</tr>
<tr>
<td>881</td>
<td>247</td>
<td>169.8</td>
<td>214.8</td>
<td>881</td>
<td>46.6</td>
<td>241.4</td>
<td>241.4</td>
</tr>
</tbody>
</table>

Received signal from send.py = updated INFOFILE.
Experiment ended! Current blocknumber = 881
Updated info file: last-experiment.json THE END.
Quorum-Dan-Thesis-30thread-100tx blocks 4-881 with 1 txs ~ 0 txs/block

avg TPS 219.7 = #TX whole experiment / blocktimes diff

blocktime seconds since last block

gasUsed and gasLimit per second

blocksizes in bytes
Quorum v1.8.12 with 100 txs: 233.0 TPS

**information:**

NODE: Geth/v1.8.12-stable/linux-amd64/go1.10.2 on http://52.170.94.181:22000

consensus=raft chain_name=??? chain_id=1 network_id=25232

SEND: 100 transactions in Blocks 4-14 with 0 empty blocks following.

A sample of transactions looked as if they: succeeded.

TPS: The stopclock watcher measured a final TPS of 74.7 since contract deploy, and in between saw values as high as 0.0 TPS.

DIAG: The whole experiment was prefixed 'Quorum-Dan-Thesis-50thread-100tx'.

The diagrams were saved into 'img/Quorum-Dan-Thesis-50thread-100tx-20190520-0055_blks4-14.png'.

Looking only at the experiment block/timestamps, the overall TPS was ~233.0.

**log:**

versions: web3 4.8.2, py-solc: 3.2.0, solc 0.4.25+commit.59db0ff1.Linux.gpp, testrpc 1.3.5, python 3.6.7 (default, Oct 22 2018, 11:32:17) [GCC 8.2.0]

web3 connection established, blockNumber = 2, node version string = Geth/v1.8.12-stable/linux-amd64/go1.10.2

first account of node is 0x8ed3D18f3191969127F9B4784b19c259956f8fcf, balance is 1000000000 Ether

WARN: raft consensus did report timestamps in nanoseconds. Is that still the case?

nodeName: Quorum, nodeType: Geth, nodeVersion: v1.8.12-stable, consensus: raft, network: 25232, chainName: ???, chainId: -1

Block 2 - waiting for something to happen

[filedate 1558312665] last contract address: 0x2D376682f1B6cfb7C1FA5bdF79F87951c8B6D24a

[filedate 1558313705] new contract address: 0xCFC94aD4Ed28CC346Dd2C05837d0BBDF623938E814a

blocknumber_start_here = 3

starting timer, at block 3 which has 1 transactions; at epochtime 1558313705.4622135

block 5 | new #TX 2 / 751 ms = 2.7 TPS_current | total: #TX 3 / 0.6 s = 4.6 TPS_average (peak is 4.6 TPS_average)

block 12 | new #TX 80 / 320 ms = 250.2 TPS_current | total: #TX 83 / 1.0 s = 81.2 TPS_average (peak is 81.2 TPS_average)

block 14 | new #TX 18 / 100 ms = 180.3 TPS_current | total: #TX 101 / 1.4 s = 74.7 TPS_average (peak is 74.7 TPS_average)

Received signal from send.py = updated INFOFILE.

Experiment ended! Current blocknumber = 14

Updated info file: last-experiment.json THE END.

**diagrams:**
avg TPS 233.0 = #TX whole experiment / blocktimes diff

blocktime seconds since last block

gasUsed and gasLimit per second
(Quorum-Dan-Thesis-50thread-1000tx) Quorum v1.8.12 with 1000 txs: 246.6 TPS

information:
NODE: Geth/v1.8.12-stable/linux-amd64/go1.10.2 on http://52.170.94.181:22000
consensus=raft chain_name=??? chain_id=-1 network_id=48799
SEND: 1000 transactions in blocks 4-86 with 0 empty blocks following.
A sample of transactions looked as if they: succeeded.
TPS: The stopclock watcher measured a final TPS of 204.9 since contract deploy,
and in between saw values as high as 204.9 TPS.
DIAG: The whole experiment was prefixed 'Quorum-Dan-Thesis-50thread-1000tx'.
The diagrams were saved into 'img/Quorum-Dan-Thesis-50thread-1000tx-20190520-0057_blks4-86.png'.
Looking only at the experiment block-timestamps, the overall TPS was ~246.6.

log:
versions: web3 4.8.2, py-solc: 3.2.0, solc 0.4.25+commit.59dbff8f1.Linux.gpp, testrpc 1.3.5, python 3.6.7 (default, Oct 22 2018, 11:32:17) [GCC 8.2.0]
web3 connection established, blockNumber = 2, node version string = Geth/v1.8.12-stable/linux-amd64/go1.10.2
first account of node is 0xC53AF67fCff4e5e197613B7904cC0A2B6C4dD9A0, balance is 1000000000 Ether
WARN: raft consensus did report timestamps in nanoseconds. Is that still the case?
nodeName: Quorum, nodeType: Geth, nodeVersion: v1.8.12-stable, consensus: raft, network: 48799, chainName: ???, chainId: -1
Block 2 - waiting for something to happen
[filedate 1558313705] last contract address: 0xCFC94aD4Ed28CC346Dd2C05837d0BF62393E814a
[filedate 1558313878] new contract address: 0xC6A2c18ac047d1d056C1e91863E728528E1EB8e5
blocknumber_start_here = 3
starting timer, at block 3 which has 1 transactions; at epochtime 1558313878.2095642
block 6  | new #TX 13 / 772 ms = 16.8 TPS_current | total: #TX 14 / 0.7 s = 20.9 TPS_average (peak is 20.9 TPS_average)
block 13 | new #TX 87 / 350 ms = 248.6 TPS_current | total: #TX 101 / 1.1 s = 89.1 TPS_average (peak is 89.1 TPS_average)
block 23 | new #TX 129 / 500 ms = 257.9 TPS_current | total: #TX 230 / 1.6 s = 143.6 TPS_average (peak is 143.6 TPS_average)
block 32 | new #TX 118 / 450 ms = 262.3 TPS_current | total: #TX 348 / 2.0 s = 170.7 TPS_average (peak is 170.7 TPS_average)
block 41 | new #TX 121 / 450 ms = 268.9 TPS_current | total: #TX 469 / 2.7 s = 176.5 TPS_average (peak is 176.5 TPS_average)
block 53 | new #TX 143 / 602 ms = 237.4 TPS_current | total: #TX 612 / 3.5 s = 173.0 TPS_average (peak was 176.5 TPS_average)
block 71 | new #TX 219 / 898 ms = 244.0 TPS_current | total: #TX 831 / 4.4 s = 187.4 TPS_average (peak is 187.4 TPS_average)
block 86 | new #TX 170 / 753 ms = 225.8 TPS_current | total: #TX 1001 / 4.9 s = 204.9 TPS_average (peak is 204.9 TPS_average)
Received signal from send.py = updated INFOFILE.
Experiment ended! Current blocknumber = 86
Updated info file: last-experiment.json THE END.

diagrams:
info raw

information:

NODE: Geth/v1.8.12-stable/linux-amd64/go1.10.2 on http://52.170.94.181:22000
consensus=raft chain_name=??? chain_id=1 network_id=26842
SEND: 10000 transactions in blocks 4-887 with 0 empty blocks following.
A sample of transactions looked as if they succeeded.

TPS: The stopclock watcher measured a final TPS of 225.1 since contract deploy,
and in between saw values as high as 235.0 TPS.

DIAG: The whole experiment was prefixed 'Quorum-Dan-Thesis-50thread-10000tx'.
The diagrams were saved into 'img/Quorum-Dan-Thesis-50thread-10000tx-20190520-0100_blks4-807.png'.
Looking only at the experiment block-timestamps, the overall TPS was -231.2.

log:

versions: web3 4.8.2, py-solc: 3.2.0, solc 0.4.25+commit.59db8ff1, linux-gpp, testrpc 1.3.5, python 3.6.7 (default, Oct 22 2018, 11:32:37) [GCC 8.2.0]
web3 connection established, blockNumber = 3, node version string = Geth/v1.8.12-stable/linux-amd64/go1.10.2
first account of node is 0x6f767896f6B52EB3c276e9A8770692753812e448, balance is 1000000000 Ether
WARN: raft consensus did report timestamps in nanoseconds. Is that still the case?
nodeName: Quorum, nodelistype: Geth, nodenodeversion: v1.8.12-stable, consensus: raft, network: 26842, chainName: ???, chainId: -1

Block 2 - waiting for something to happen
(filedate 1558313878) last contract address: 0xC6A2c18ac047d1d056C1e91863E728528E1E9B8
(filedate 1558314854) new contract address: 0x6f767896f6B52EB3c276e9A8770692753812e448

blocknumber_start_here = 3
starting_timer, at block 3 which has 1 transactions; at epochtime 1558314054.4711573

log:

Looking only at the experiment block-timestamps, the overall TPS was ~231.2.
Received signal from send.py = updated INFOFILE.

Experiment ended! Current blocknumber = 807
Updated info file: last-experiment.json THE END.
Quorum-Dan-Thesis-50thread-10000tx blocks 4-807 with 1 txs ~ 0 txs/block

avg TPS 231.2 = #TX whole experiment / blocktimes diff

blocksizes in bytes

gasUsed and gasLimit per second
Quorum v1.8.12 with 100 txs: 278.0 TPS

**information:**

NODE: Geth/v1.8.12-stable/linux-amd64/go1.10.2 on http://52.170.94.181:22000

consensus=raft chain_name=?? chain_id=-1 network_id=82907

SEND: 100 transactions in Blocks 4-12 with 0 empty blocks following.

A sample of transactions looked as if they: succeeded.

TPS: The stopclock watcher measured a final TPS of 75.4 since contract deploy, and in between saw values as high as 0.0 TPS.

DIAG: The whole experiment was prefixed 'Quorum-Dan-Thesis-100thread-100tx'.

The diagrams were saved into 'img/Quorum-Dan-Thesis-100thread-100tx-20190520-0104_blks4-12.png'.

Looking only at the experiment block-timestamps, the overall TPS was ~278.0.

**log:**

versions: web3 4.8.2, py-solc: 3.2.0, solc 0.4.25+commit.59db8ff1.Linux.gpp, testrpc 1.3.5, python 3.6.7 (default, Oct 22 2018, 11:32:17) [GCC 8.2.0]

web3 connection established, blockNumber = 2, node version string = Geth/v1.8.12-stable/linux-amd64/go1.10.2

first account of node is 0xa37F1a09d48df5a865dCE298D1d0347f3DFcF38c, balance is 1000000000 Ether

WARN: raft consensus did report timestamps in nanoseconds. Is that still the case?

nodeName: Quorum, nodeType: Geth, nodeVersion: v1.8.12-stable, consensus: raft, network: 82907, chainName: ???, chainId: -1

Block  2  - waiting for something to happen

(filedate 1558314054) last contract address: 0x6f767896f6B52EB3c276e9A8770692753812e448

(filedate 1558314259) new contract address: 0x88b560Cc1951933DfdD66D071c1682248b610b3

blocknumber_start_here = 3

starting timer, at block 3 which has 1 transactions; at epochtime 1558314259.3257244

block 6 | new #TX 4 / 741 ms = 5.4 TPS_current | total: #TX 5 / 0.7 s = 7.6 TPS_average (peak is 7.6 TPS_average)

block 11 | new #TX 63 / 299 ms = 210.4 TPS_current | total: #TX 68 / 1.0 s = 67.8 TPS_average (peak is 67.8 TPS_average)

block 12 | new #TX 33 / 50 ms = 664.1 TPS_current | total: #TX 101 / 1.3 s = 75.4 TPS_average (peak is 75.4 TPS_average)

Received signal from send.py = updated INFOFILE.

Experiment ended! Current blocknumber = 12

Updated info file: last-experiment.json THE END.

**diagrams:**
Quorum-Dan-Thesis-100thread-100tx blocks 4-12 with 1 txs ~ 0 txs/block

avg TPS 278.0 = #TX whole experiment / blocktimes diff

blocktime seconds since last block

gasUsed and gasLimit per second

info raw

**Quorum v1.8.12 with 1000 txs: 261.0 TPS**

**Information:**

NODE: Geth/v1.8.12-stable/linux-amd64/go1.10.2 on http://52.170.94.181:22000

consensus=raft chain_name=??? chain_id=-1 network_id=89475

SEND: 1000 transactions in blocks 4-81 with 0 empty blocks following.

A sample of transactions looked as if they: succeeded.

TPS: The stopclock watcher measured a final TPS of 198.9 since contract deploy, and in between saw values as high as 198.0 TPS.

DIAG: The whole experiment was prefixed 'Quorum-Dan-Thesis-100thread-1000tx'.

The diagrams were saved into 'img/Quorum-Dan-Thesis-100thread-1000tx-20190520-0106_blks4-81.png'.

Looking only at the experiment block-timestamps, the overall TPS was ~261.0.

**Log:**

versions: web3 4.8.2, py-solc: 3.2.0, solc 0.4.25+commit.59dbf8f1_LINUX, testrpc 1.3.5, python 3.6.7 (default, Oct 22 2018, 11:32:17) [GCC 8.2.0]

web3 connection established, blockNumber = 2, node version string = Geth/v1.8.12-stable/linux-amd64/go1.10.2

first account of node is 0xc3142c20BD6f6e6D64F46a875552130d2bD7, balance is 1000000000 Ether

WARN: raft consensus did report timestamps in nanoseconds. Is that still the case?

nodeName: Quorum, nodeType: Geth, nodeVersion: v1.8.12-stable, consensus: raft, network: 89475, chainName: ???, chainId: -1

Block 2 - waiting for something to happen

(filedate 1558314259) last contract address: 0x88b560Cc1951933DfdD66D071c16B2248b610Eb3

(filedate 1558314395) new contract address: 0x41f8F5b7D65D37dd83C163c08d608B7aD9D774C0

blocknumber_start_here = 3

starting timer, at block 3 which has 1 transactions; at epochtime 1558314395.6819928

block 6 | new #TX 8 / 1078 ms = 7.4 TPS_current | total: #TX 9 / 0.7 s = 13.6 TPS_average (peak is 13.6 TPS_average)

block 13 | new #TX 97 / 349 ms = 277.6 TPS_current | total: #TX 106 / 1.1 s = 98.1 TPS_average (peak is 98.1 TPS_average)

block 21 | new #TX 88 / 403 ms = 218.4 TPS_current | total: #TX 194 / 1.6 s = 120.6 TPS_average (peak is 120.6 TPS_average)

block 32 | new #TX 149 / 547 ms = 272.4 TPS_current | total: #TX 343 / 2.1 s = 164.4 TPS_average (peak is 164.4 TPS_average)

block 41 | new #TX 113 / 450 ms = 251.1 TPS_current | total: #TX 456 / 2.6 s = 177.7 TPS_average (peak is 177.7 TPS_average)

block 51 | new #TX 129 / 500 ms = 258.0 TPS_current | total: #TX 585 / 3.0 s = 195.0 TPS_average (peak is 195.0 TPS_average)

block 59 | new #TX 84 / 403 ms = 208.5 TPS_current | total: #TX 669 / 4.3 s = 156.4 TPS_average (peak was 195.0 TPS_average)

block 81 | new #TX 332 / 1149 ms = 289.0 TPS_current | total: #TX 1001 / 5.0 s = 198.9 TPS_average (peak is 198.9 TPS_average)

Received signal from send.py = updated INFOFILE.

Experiment ended! Current blocknumber = 81

Updated info file: last-experiment.json THE END.

**Diagrams:**
Quorum-Dan-Thesis-100thread-1000tx blocks 4-81 with 1 txs ~ 0 txs/block

avg TPS 261.0 = #TX whole experiment / blocktimes diff

blocktime seconds since last block

gasUsed and gasLimit per second

info raw


Received signal from send.py = updated INFOFILE.
Experiment ended! Current blocknumber = 803
Updated info file: last-experiment.json THE END.

diagrams:
Quorum-Dan-Thesis-100thread-10000tx blocks 4-803 with 1 txs ~ 0 txs/block

avg TPS 245.3 = #TX whole experiment / blocktimes diff

blocktime seconds since last block

gasUsed and gasLimit per second