Performance analysis of a Besu Permissioned Blockchain

Leonardo Mostarda, Roberto Tonelli, Andrea Pinna, Davide Sestili

5th Distributed Ledger Technology Workshop (DLT 2023)

Outline

- Contribution
- Blockchain Network
- Hyperledger Caliper Benchmarks
- Custom application Benchmarks
- Conclusions

Contributions

Contribution

The contributions of our work can be summarized as follows:

- **Performance analysis** of a **permissioned blockchain** in a real scenario in which validator nodes are geographically distant from each other.
- Propose a new **benchmarking tool** for Ethereum-like blockchains.

Blockchain network

The blockchain network

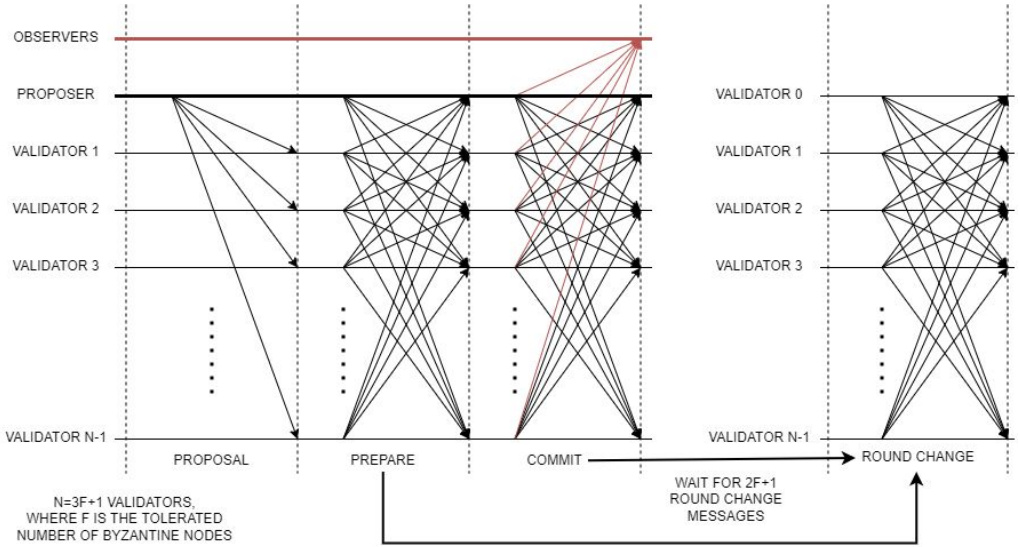
The blockchain network examined is a private blockchain with the following characteristics:

- **Besu** client;
- **Permissioned** blockchain;
- **IBFT 2.0** consensus protocol;
- **Free gas** network;
- **4 validator nodes** on docker containers;
- **6 seconds** block period;
- **Maximum** allowed blocksize;

IBFT 2.0

IBFT 2.0 consensus protocol:

- Proof-of-authority
- Inspired by PBFT
- Guarantees block finality



Hyperledger Caliper benchmarks

HYPERLEDGER CALIPER

Hyperledger caliper is a blockchain benchmarking tool for Hyperledger Besu blockchains and other blockchains. Its reports contains performance indicators including:

- **Read latency**: the time elapsed between a request is submitted and a reply is received;
- **Read throughput**: the total amount of read operations successfully submitted for which a reply has been received per second;
- **Transaction latency**: the amount of time a transaction requires to be part of the network from the moment it was submitted;
- **Transaction throughput**: the rate at which valid transactions are committed to the blockchain.

Caliper Benchmark

Each benchmark run consisted in sending loads of transactions to a validator node of the blockchain at rates of 0.2, 1, 2, 10 transactions per seconds.

The transactions sent are smart contract calls of the *“simple.sol”* smart contract provided with Caliper.

```
pragma solidity >=0.4.22 <0.6.0;

contract simple {
    mapping(string => int) private accounts;

    function open(string memory acc_id, int amount) public {
        accounts[acc_id] = amount;
    }

    function query(string memory acc_id) public view returns (int amount) {
        amount = accounts[acc_id];
    }

    function transfer(string memory acc_from, string memory acc_to, int amount)
        public {
            accounts[acc_from] -= amount;
            accounts[acc_to] += amount;
        }
}
```

Caliper Benchmark results

Tx Send Rate (TPS)	0.2	1	2	10
Maximum latency (s)	5.20	6.66s	12.39	13.42
Minimum latency (s)	0.23	0.64	0.69	0.97
Average latency (s)	2.70	3.58	6.41	7.23
Throughput (TPS)	0.2	0.9	1.9	9.5

Caliper Benchmark results

Sudden increase in Maximum Latency and Average latency

Tx Send Rate (TPS)	0.2	1	2	10
Maximum latency (s)	5.20	6.66s	12.39	13.42
Minimum latency (s)	0.23	0.64	0.69	0.97
Average latency (s)	2.70	3.58	6.41	7.23
Throughput (TPS)	0.2	0.9	1.9	9.5

Caliper Benchmark results

Sudden increase in Maximum Latency and Average latency

Tx Send Rate (TPS)	0.2	1	2	10
Maximum latency (s)	5.20	6.66s	12.39	13.42
Minimum latency (s)	0.23	0.64	0.69	0.97
Average latency (s)	2.70	3.58	6.41	7.23
Throughput (TPS)	0.2	0.9	1.9	9.5

Further investigation can be done with the custom benchmark tool

Custom application benchmarks

Custom application

The custom application devised is a benchmark tool for BESU and other EVM compatible blockchains that allows to set up a workload parametrized with the following parameters:

- **Transaction send delay:** milliseconds waited before sending a subsequent transaction
- **Transaction number:** number of transactions in the workload
- **Timeout:** time after which a transaction is considered timedout if not included in the chain
- **Communication protocol** (either HTTP or WebSocket)
- **Node address:** the ip and port number of the contacted node
- **Sender key**: private key used for signing the transactions
- **Mode:** either *smart contract method call mode* or *regular transaction mode*

Custom Java application

The Information contained in the reports generated by the application are:

- **Latency** of each transaction identified with its nonce
- **Send timestamp**
- **Block number** where the transaction was included
- **Block timestamp**
- **Block proposer’s ethereum address**
- **Number of blocks** minted during the benchmark run
- **Average latency**

custom Java application

The benchmark runs involved sending loads of transactions to a validator node of the network that called the *“transfer”* method of an ERC20 smart contract at different send rates (0.2 , 1 , 2 , 10 and 100 tps).

Each benchmark run lasted for 2 minutes

```
function transfer(address to, uint256 amount) public virtual override returns (bool) {
    address owner = _msgSender();
    _transfer(owner, to, amount);
    return true;
}
```

Java custom tool reports

Average latency increases significantly at 2 tps send rates and above.

What's going on?

The reports have data helping us shed light on the reason for this.

Tx Send Rate (TPS)	0.2	1	2	10	100
Maximum latency (s)	5.99	6.289	12.235	13.001	22.112
Minimum latency (s)	0.897	0.190	0.402	0.337	4.720
Average latency (s)	3.405	3.206	5.246	6.759	12.638
Throughput (TPS)	0.208	0.997	1.779	8.927	69.136

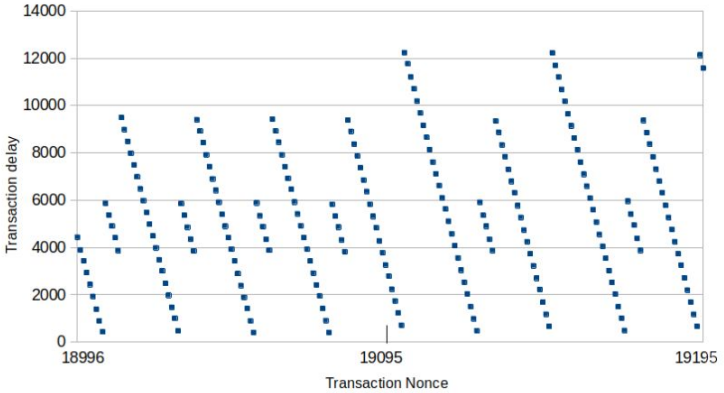
Tx Send Rate (TPS)	0.2	1	2	10	100
Maximum latency (s)	6.917	6.664	12.196	12.774	18.528
Minimum latency (s)	0.833	0.567	0.289	0.377	2.835
Average latency (s)	3.781	3.596	5.092	6.580	10.712
Throughput (TPS)	0.199	0.960	1.855	8.879	66.454

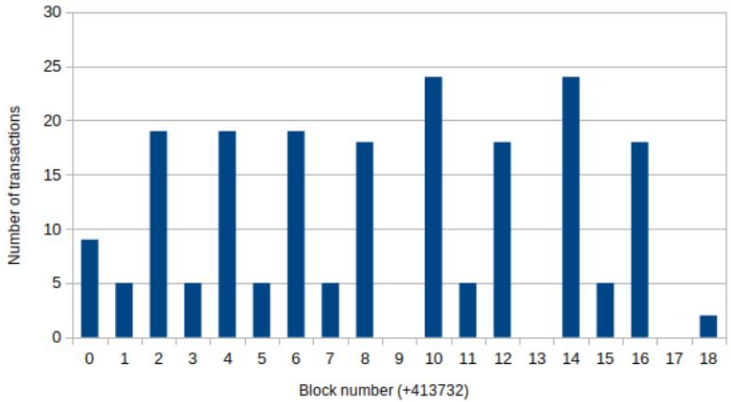
Investigation at 2 tps

The reports tell us in which block each transaction is included.

Observation:

Blocks containing many transactions are alternated with blocks containing fewer transaction.





Investigation at 2 tps

The reports also inform us that block proposers change in a round robin fashion, with certain proposers committing blocks with fewer transactions than other.

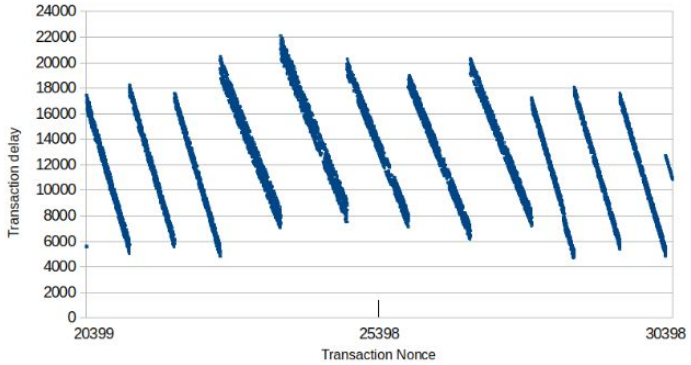
Block number	0	1	2	3	4	5	6	7
Validator	0x5a	0x7c	0xb3	0x2d	0x5a	0x7c	0xb3	0x2d

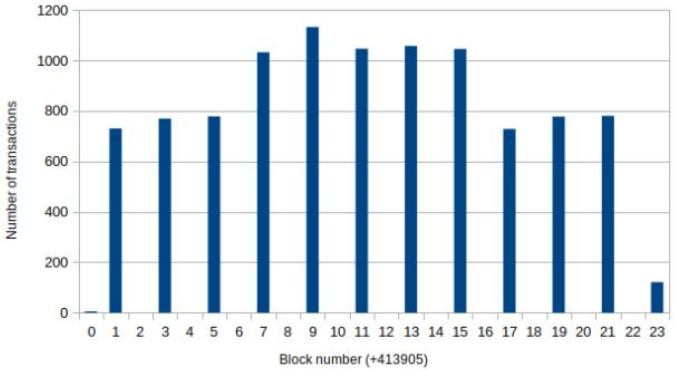
Investigation at 100 tps

It is possible to observe that at higher sends rates such as 100 tps, some validators propose empty blocks.

2 out of 4 validators propose blocks containing transactions.

2 validators out of 4 validators propose empty blocks.





Conclusions

Conclusions

This work presented a tool for the analysis of blockchain performances under various conditions that allows us to retrieve information that is not provided with other benchmark tools, such as Caliper.

The new information allowed us to detect certain validators anomalous behaviour that is the source of the latency anomalies detected both with Caliper and our tool.

It is a first step towards building a tool for detecting anomalies in a specific blockchain.

E-mail: davide.sestili@unicam.it