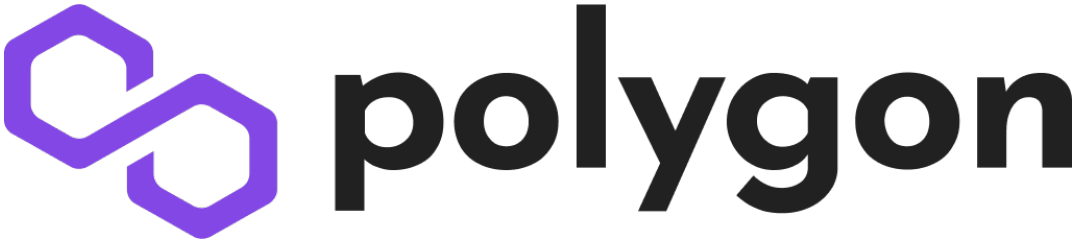
Simulation of CloudSLA contract functions over three different blockchains

Davide Pruscini, Giosuè Cotugno

Master degree in Computer Science, curriculum A
A.Y. 2021/2022

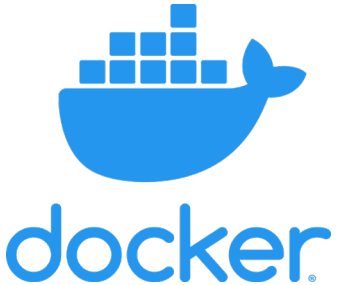
Blockchains tested

GoQuorum



ConsenSys Quorum

- Consensus algorithms
  - GoQuorum → IBFT, QBFT, Raft
  - Hyperledger Besu → IBFT, QBFT, Clique
- 5 nodes
  - 4 validators
  - 1 RPC node
- Provider
  - HTTP + WebSocket
- Key management
  - GoQuorum → Quorum Key Manager
  - Hyperledger Besu → Not supported inside the client



How to setup

- Add validator nodes
  1. Generate validator folders with *quorum-genesis-tool*
  2. Copy these folder inside *./config/nodes/*
  3. Update **prometheus.yml** in the *./config/prometheus/* directory to configure metrics to display in Grafana
  4. Update **docker-compose.yml** in the root directory
  5. Add the new node's enode address to the static nodes file and permissions file

- Change consensus mechanism
  1. Open **.env** file in the root directory
  2. Change XYZ_CONS_ALGO variable

Polygon

- Consensus algorithms
  - Polygon → IBFT, PoS
- 5 nodes
  - 4 validators
  - 1 RPC node
- Provider
  - HTTP
- Key management
  - Natively supported by SDK

How to setup

- Execute **run_polygon.sh** from *cloud-chain-simulation/polygon/*

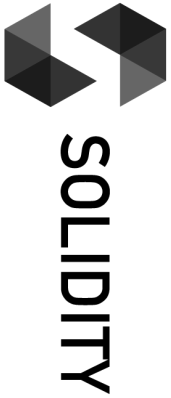
```
Do you want create the network from zero? (yes/no)
yes

How many validators do you want? (4, 8, 12)
4

Which consensus mechanism do you want to use? (1-IBFT, 2-PoS)
1
Network initialized correctly

Do you want run the newtork (yes/no)
yes
```

Project structure


```
.
├── README.md
├── build
├── contracts
├── polygon
├── quorum
├── simulation-web3py
└── statistics-web3py
```

contracts/CloudSLA.sol

Script function	Contract function
upload	UploadRequest UploadRequesAck UploadTransferAck
read	ReadRequest ReadRequestAck
delete	DeleteRequest Delete
file_check_undeleted_file	ReadRequest ReadRequestDeny
read_deny_lost_file_check	FileHashRequest DigestStore FileCheck



simulation-web3py

```
.
├── config
├── results
├── README.md
├── contract_functions.py
├── main.py
├── requirements.txt
├── run_simulation.sh
├── settings.py
├── utility.py
└── web3client.py
```

- Features
  - Interact with any blockchain that supports HTTP
  - Deploy multiple instance of a contract
  - Interact with contract functions
  - Use threads and async methods
  - Parametrized simulations

Concurrent requests

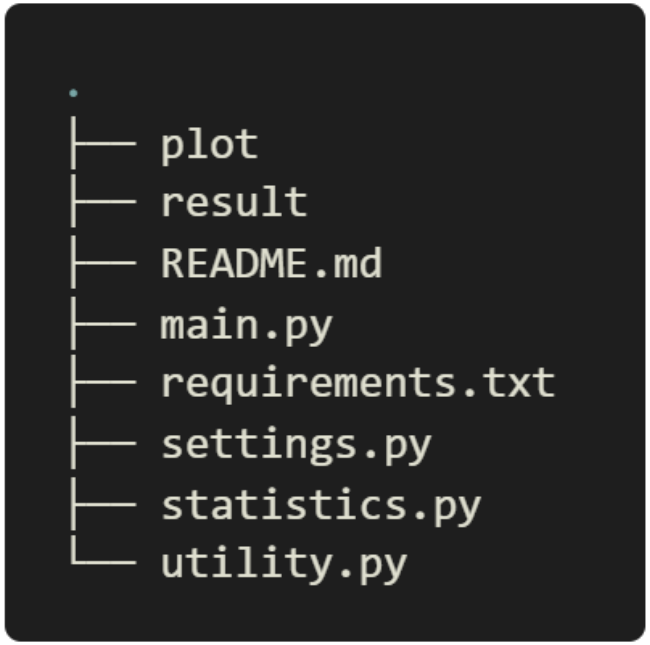
```
async def main():
    # ...
    while actual < start + args.time:
        thread = threading.Thread(
            target=between_callback,
            args=[idx, f'contracts[{idx % DEPLOYED_CONTRACTS}].{args.function}']
        )
        jobs.append(thread)

        actual = (datetime.now() - zero_time).total_seconds()
        rand = np.random.exponential(1 / args.lambda_p)
        await asyncio.sleep(rand)
        jobs[idx].start()
        idx += 1

    for j in jobs:
        j.join()
```

Save metrics

```python
async def get_time(func_to_run: str, process_count: int) -> pd.DataFrame:
    # ...
    try:
        if 'cloud_sla_creation_activation' in func_to_run:
            start_fun = datetime.now()
            cloud_address, function_status = await eval(func_to_run)
            end_fun = datetime.now()
        else:
            start_fun = datetime.now()
            function_status = await eval(func_to_run)
            end_fun = datetime.now()
    except ValueError as v:
        print(f'{type(v)} [get_time#{process_count}]: {v}')
        function_status = False
        end_fun = datetime.now()
    finally:
        duration_fun = end_fun - start_fun

        return pd.DataFrame({
            'id': [process_count],
            'start_fun': [(start_fun - zero_time).total_seconds()],
            'end_fun': [(end_fun - zero_time).total_seconds()],
            'time_fun': [duration_fun.total_seconds()],
            'address': [cloud_address],
            'status': function_status,
            'lambda': args.lambda_p,
            'num_run': args.num_run
        })
```

statistics-web3py

- Features
  - Read results from different csv files
  - Find transient for every simulation
  - Calculate steady-state metrics
  - Plot results in different way



Simulation parameters

Tested configuration		
Blockchain	**Consensus mechanisms**	**Interarrival time Poisson (λ)**
GoQuorum	IBFT, QBFT, Raft	2.0, 1.0, 0.5
Hyperledger Besu	IBFT, QBFT, Clique	2.0, 1.0, 0.5
Polygon	IBFT, PoS	2.0, 1.0, 0.5

- Gas limit: 0xf7b760
- Block period seconds: 5s
- Number of deployed contracts: 40

- Hardware setup
  - Intel Core i7 8750H
  - 2 x 8GB DDR4 @2667MHz
  - Windows 11 v21H2 (build 22000.556) – WSL2

Steady-state simulation

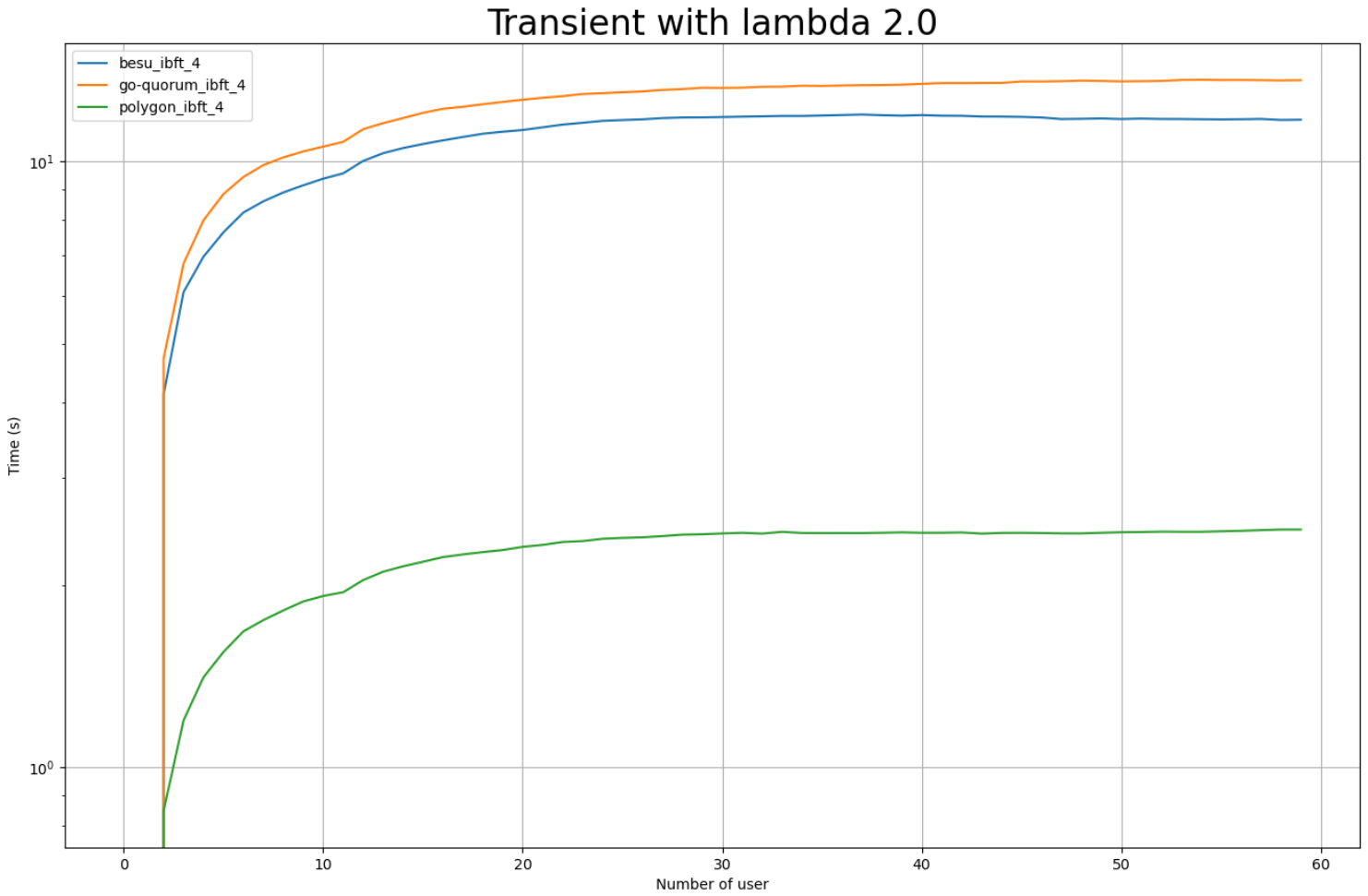
- Transient phase
  - Calculation of convergence of the mean of the distribution
  - Tested on upload function

200s x 15 repetitions

- Steady-state phase
  - Compute the metrics over every run
  - Average the previously computed values
  - Tested on all functions

600s x 5 repetitions

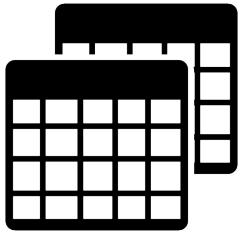
Transient phase - Plot



Steady-state phase

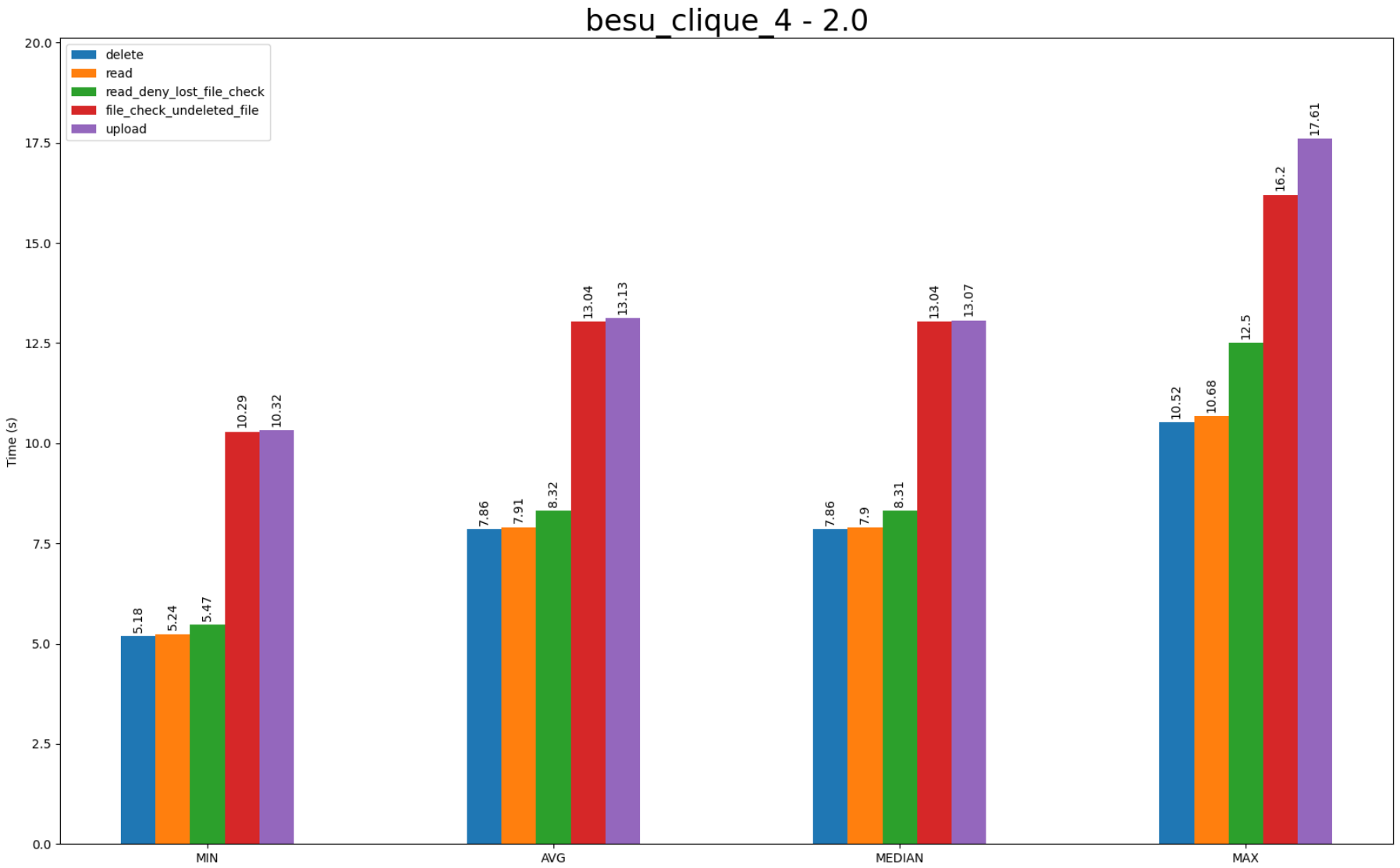
- Calculated metrics
  - minimum
  - average
  - median
  - maximum
  - average error (%)
  - number of users
  - t_students, accuracy 90%

- See the output datasets
  - steady_state_metrics.csv
  - transient_metrics.csv

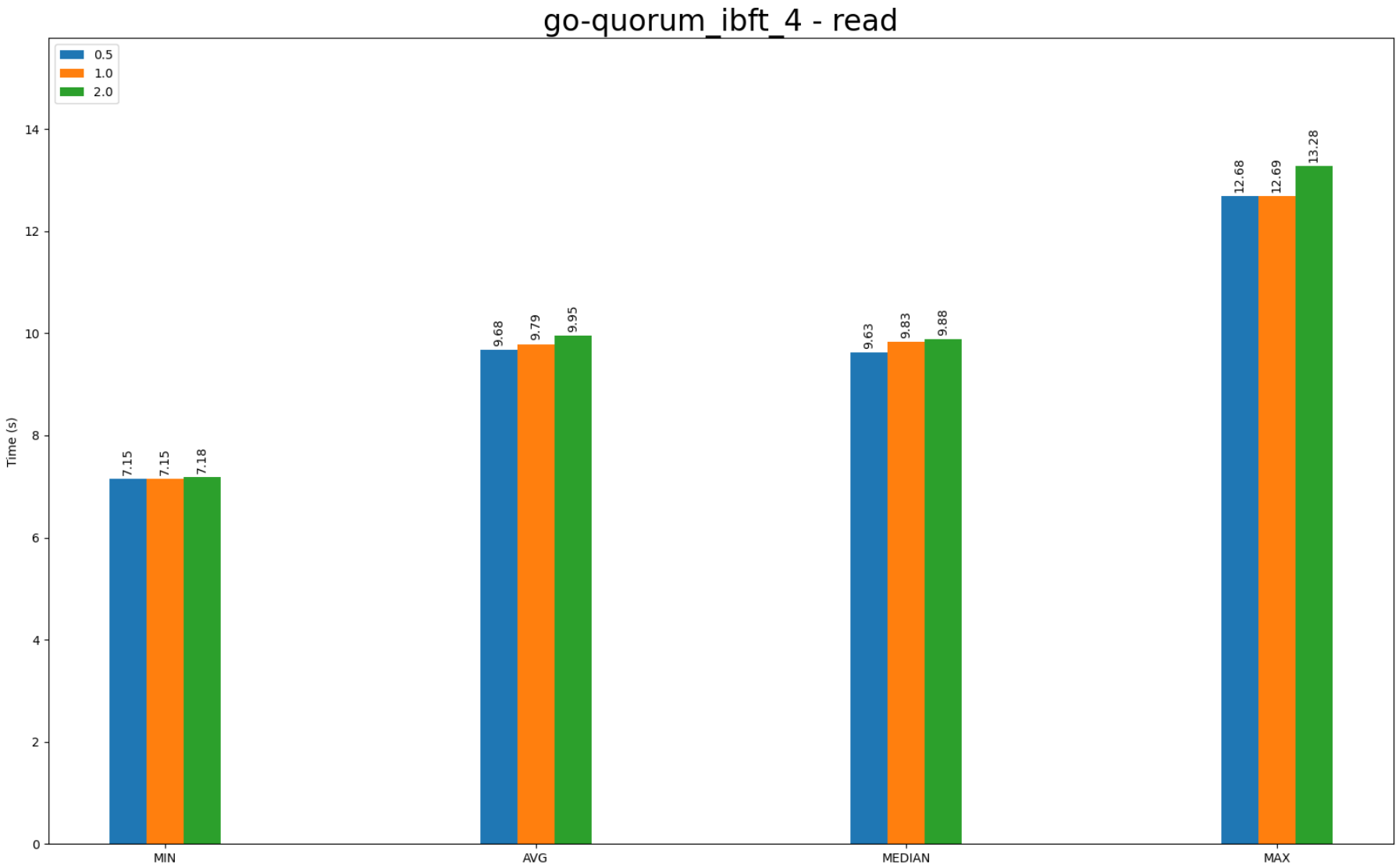


- Different plots
  1. Single experiment with fixed lambda
  2. Single experiment with fixed function
  3. All experiments with fixed lambda and function
  4. Percentage error for all functions with fixed lambda
  5. Number of users for all functions with fixed lambda

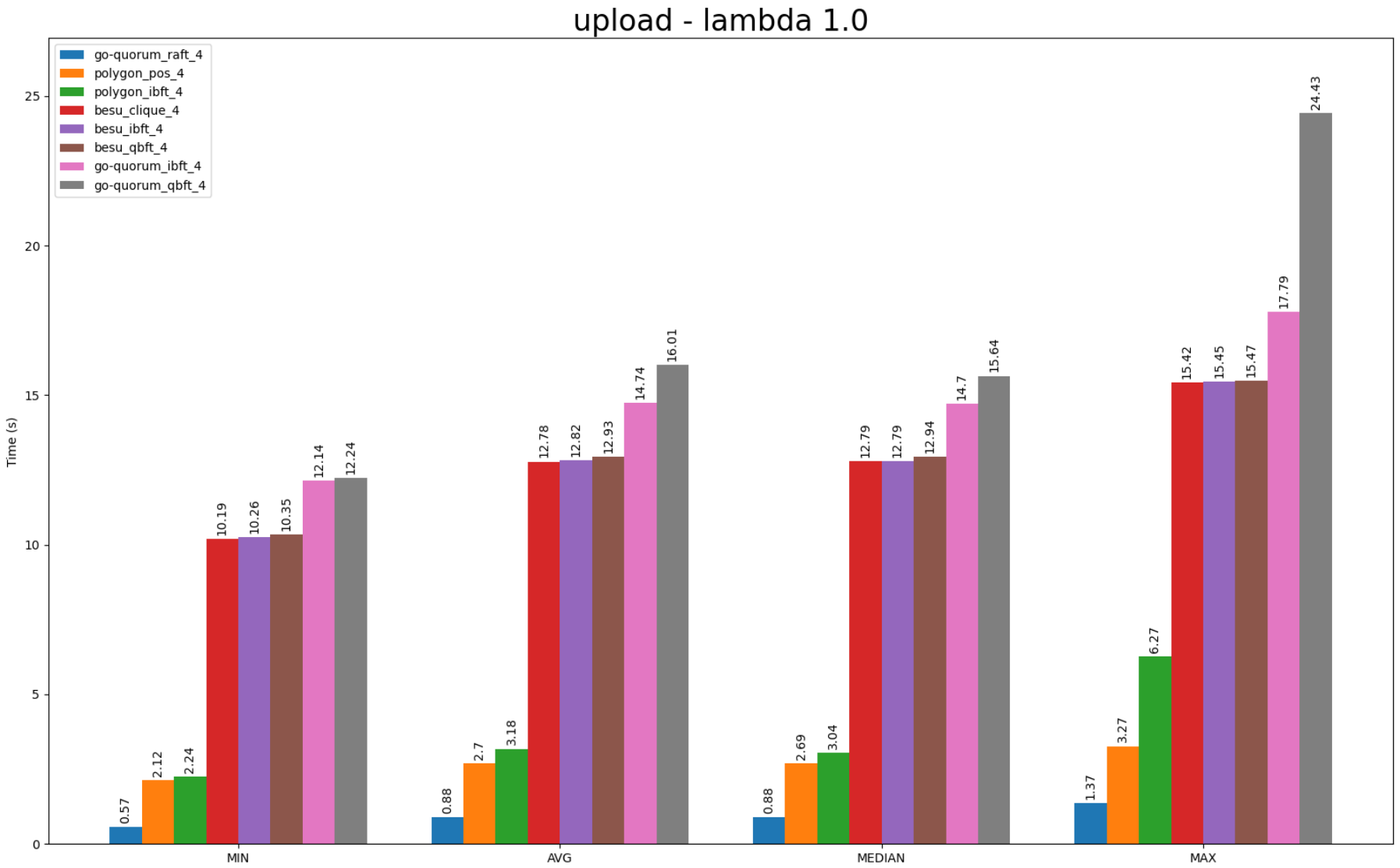
Plot type 1


besu_clique_4 - 2.0
delete
read
read_deny_lost_file_check
file_check_undeleted_file
upload
Time (s)
20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
5.18
5.24
5.47
10.29
10.32
7.86
7.91
8.32
13.04
13.13
7.86
7.9
8.31
13.04
13.07
10.52
10.68
12.5
16.2
17.61
MIN
AVG
MEDIAN
MAX

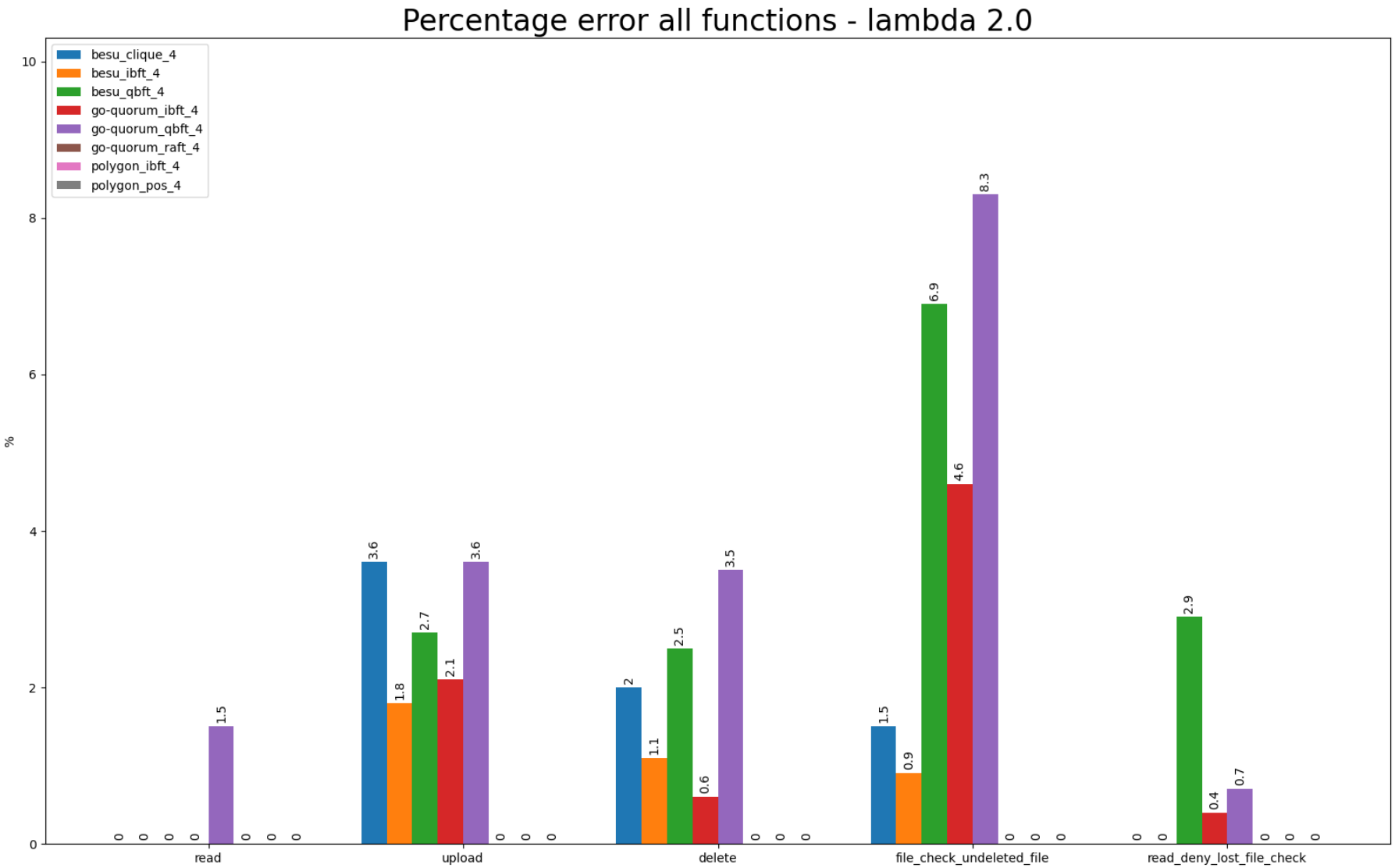
Plot type 2


go-quorum_ibft_4 - read
0.5
1.0
2.0
Time (s)
0
2
4
6
8
10
12
14
7.15
7.15
7.18
9.68
9.79
9.95
9.63
9.83
9.88
12.68
12.69
13.28
MIN
AVG
MEDIAN
MAX

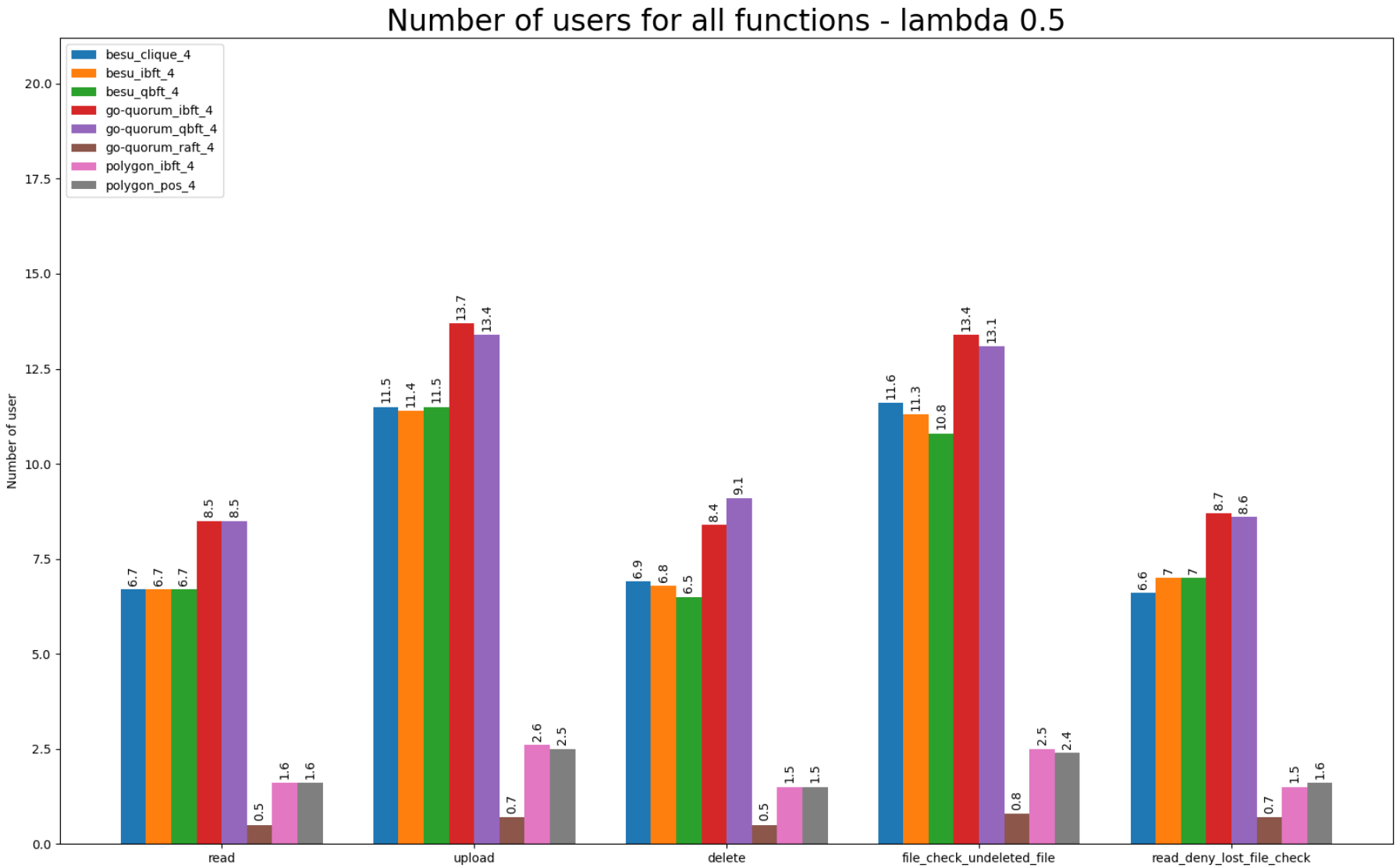
Plot type 3


upload - lambda 1.0
go-quorum_raft_4
polygon_pos_4
polygon_ibft_4
besu_clique_4
besu_ibft_4
besu_qbft_4
go-quorum_ibft_4
go-quorum_qbft_4
Time (s)
0
5
10
15
20
25
0.57
2.12
2.24
10.19
10.26
10.35
12.14
12.24
0.88
2.7
3.18
12.78
12.82
12.93
14.74
16.01
0.88
2.69
3.04
12.79
12.79
12.94
14.7
15.64
1.37
3.27
6.27
15.42
15.45
15.47
17.79
24.43
MIN
AVG
MEDIAN
MAX

Plot type 4


Percentage error all functions - lambda 2.0
besu_clique_4
besu_ibft_4
besu_qbft_4
go-quorum_ibft_4
go-quorum_qbft_4
go-quorum_raft_4
polygon_ibft_4
polygon_pos_4
%
0
2
4
6
8
10
read
upload
delete
file_check_undeleted_file
read_deny_lost_file_check
0
0
0
0
1.5
0
0
0
3.6
1.8
2.7
2.1
3.6
0
0
0
2
1.1
2.5
0.6
3.5
0
0
0
1.5
0.9
6.9
4.6
8.3
0
0
0
0
0
2.9
0.4
0.7
0
0
0

Plot type 5


Number of users for all functions - lambda 0.5
besu_clique_4
besu_ibft_4
besu_qbft_4
go-quorum_ibft_4
go-quorum_qbft_4
go-quorum_raft_4
polygon_ibft_4
polygon_pos_4
Number of user
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
6.7 6.7 6.7 8.5 8.5 0.5 1.6 1.6
11.5 11.4 11.5 13.7 13.4 0.7 2.6 2.5
6.9 6.8 6.5 8.4 9.1 0.5 1.5 1.5
11.6 11.3 10.8 13.4 13.1 0.8 2.5 2.4
6.6 7 7 8.7 8.6 0.7 1.5 1.6
read
upload
delete
file_check_undeleted_file
read_deny_lost_file_check

Issues & difficulties

- Gathering data takes a lot of time for each configuration
- Async part of the library Web3.py is under developing
- More contracts are needed to execute transactions with status true
- Hyperledger Besu and GoQuorum are resource expensive

Conclusion

The proposed script allowed us to interact with all different blockchains that support HTTP provider connection.

- The results show us that all the blockchain work fine with a minimum number of contracts deployed equal to 40, if this number decrease only Polygon network is capable to sign transaction properly.
- As we can see from the plots the fastest configuration are **polygon_pos_4**, **polygon_ibft_4** and **go-quorum_raft_4**, this can derive from the internal implementation of the blockchain and the consensus algorithm.

Code available at cloud-chain-simulation repository.

References

[1] Gabriele D'Angelo, Stefano Ferretti, and Moreno Marzolla. 2018. A Blockchain-based Flight Data Recorder for Cloud Accountability. In Proceedings of the 1st Workshop on Cryptocurrencies and Blockchains for Distributed Systems (CryBlock'18). Association for Computing Machinery, New York, NY, USA, 93–98.

[2] Getting Started with ConsenSys Quorum

[3] GoQuorum Enterprise Ethereum Client

[4] Hyperledger Besu Ethereum Client

[5] Polygon Edge

[6] Web3.py