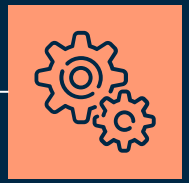
BENCHMARKING BLOCKCHAIN FRAMEWORKS

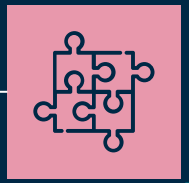
Team Romeo:
Chun-An Bau (2165883)
Guilhem Dubois (1897198)
Matyas Jesina (2165626)

TABLE OF CONTENTS

01 CONTEXT

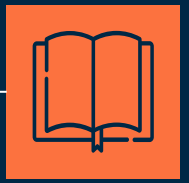


02 EXPERIMENTS



03 RESULTS

04 DISCUSSION



05 CONCLUSION

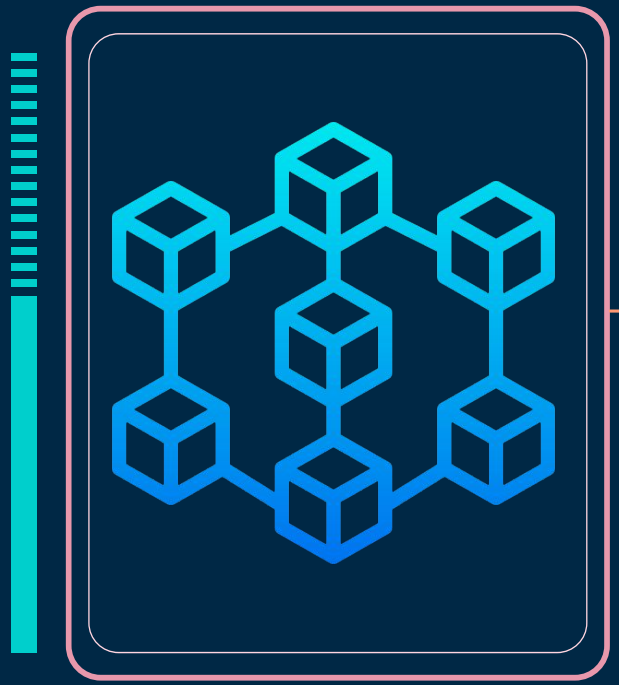
CONTEXT

01

Blockchain
Ethereum
Hyperledger Fabric
Blockbench

BLOCKCHAIN

- P2P distributed ledger
- Growing list of *blocks*
  - Contain hash of the previous block
- Why
  - Reduce cost
  - Improve traceability
  - Improve transparency
- Types
  - Public
  - Private



PUBLIC vs. PRIVATE BLOCKCHAIN

	Public	Private
Access	Anyone	Single organization
Authority	Decentralized	Partially decentralized
Transaction Speed	Slow	Fast
Transaction Cost	High	Low
Immutability	Full	Partial
Efficiency	Low	High

Reference: https://101blockchains.com/public-vs-private-blockchain/

Ethereum

- Both public and private blockchain
- PoW (Proof-of-Work)
  - Not secure enough
  - Require lots of computational power
- New Clique: PoA (Proof-of-Authority)
  - Blocks are sealed by approved signers
  - Dynamically add or remove signers

Hyperledger Fabric

- Private blockchain proposed by IBM
- Conquer limitations of other blockchains
  - Order-executive or hard-coded consensus
  - Better throughput and latency

BLOCKBENCH

- A Framework for analyzing private blockchains
- Simple APIs
- Evaluate by workloads that are based on real smart contracts
- Measure overall and component-wise performance
  - Throughput
  - Latency
  - Scalability
  - Fault-tolerance

EXPERIMENTS

02

Tools setup
Workloads

Tools setup

- Amazon EC2: Virtual machines in the cloud
  - 2 vCPUs
  - 8.0 GiB RAM
  - 25.0 GiB Storage
- Run blockchain networks using Docker
  - 5 Miners / Peers
- Run Blockbench
  - 50 transactions per second
  - Configurable workloads

Workloads

```
# Yahoo! Cloud System Benchmark
# Small

recordcount=2500
operationcount=10000
workload=ycsb

readallfields=true

readproportion=0
updateproportion=1
scanproportion=0
insertproportion=0

requestdistribution=zipfian
```

- Macro-benchmark: Yahoo! Cloud Serving Benchmark
  - Benchmark framework based on key-values
- Record Count: Number of total requests sent
- Operation Count: Number of operations performed per transaction
- Update Proportion (write): Consensus protocol

Workloads

```
# Yahoo! Cloud System Benchmark
# Small

recordcount=2500
```

```
# Yahoo! Cloud System Benchmark
# Medium

recordcount=5000
```

```
# Yahoo! Cloud System Benchmark
# Large

recordcount=10000
```

SMALL

MEDIUM

LARGE

RESULTS

03

METRICS

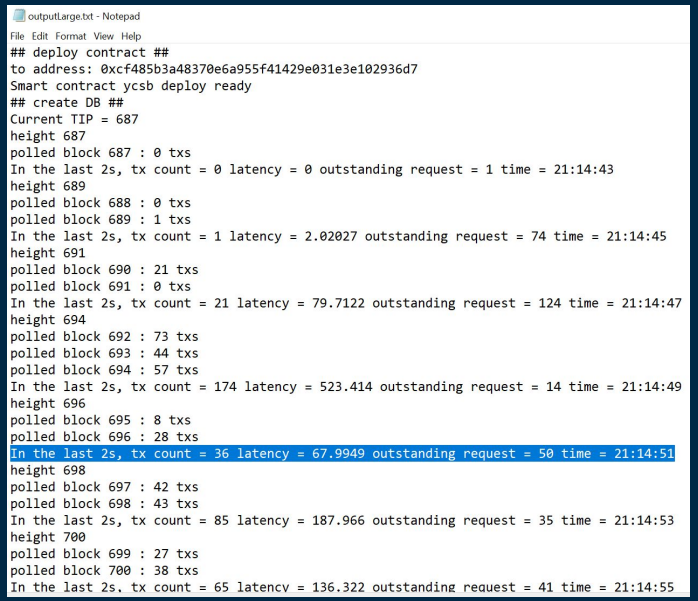
LATENCY

Definition: Response time per transaction

THROUGHPUT

Definition: Number of successful transactions per second

Raw output



outputLarge.txt - Notepad

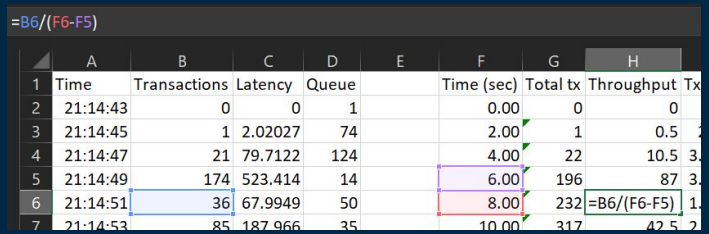
File Edit Format View Help

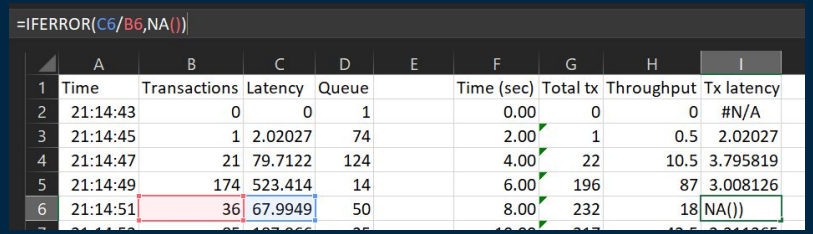
```
## deploy contract ##
to address: 0xcf485b3a48370e6a955f41429e031e3e102936d7
Smart contract ycsb deploy ready
## create DB ##
Current TIP = 687
height 687
polled block 687 : 0 txs
In the last 2s, tx count = 0 latency = 0 outstanding request = 1 time = 21:14:43
height 689
polled block 688 : 0 txs
polled block 689 : 1 txs
In the last 2s, tx count = 1 latency = 2.02027 outstanding request = 74 time = 21:14:45
height 691
polled block 690 : 21 txs
polled block 691 : 0 txs
In the last 2s, tx count = 21 latency = 79.7122 outstanding request = 124 time = 21:14:47
height 694
polled block 692 : 73 txs
polled block 693 : 44 txs
polled block 694 : 57 txs
In the last 2s, tx count = 174 latency = 523.414 outstanding request = 14 time = 21:14:49
height 696
polled block 695 : 8 txs
polled block 696 : 28 txs
In the last 2s, tx count = 36 latency = 67.9949 outstanding request = 50 time = 21:14:51
height 698
polled block 697 : 42 txs
polled block 698 : 43 txs
In the last 2s, tx count = 85 latency = 187.966 outstanding request = 35 time = 21:14:53
height 700
polled block 699 : 27 txs
polled block 700 : 38 txs
In the last 2s, tx count = 65 latency = 136.322 outstanding request = 41 time = 21:14:55
```

	A	B	C	D
1	Time	Transactions	Latency	Queue
2	21:14:43	0	0	1
3	21:14:45	1	2.02027	74
4	21:14:47	21	79.7122	124
5	21:14:49	174	523.414	14
6	21:14:51	36	67.9949	50
7	21:14:53	85	187.966	35
8	21:14:55	65	136.322	41
9	21:14:57	88	175.772	26

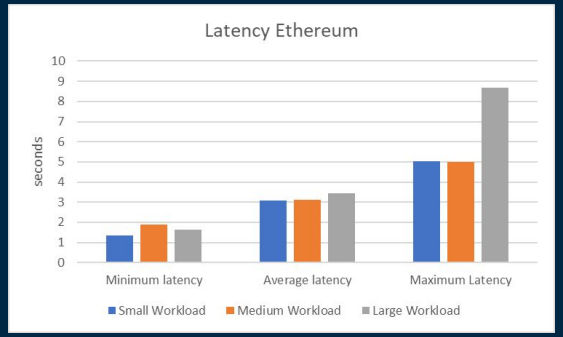
Data aggregation

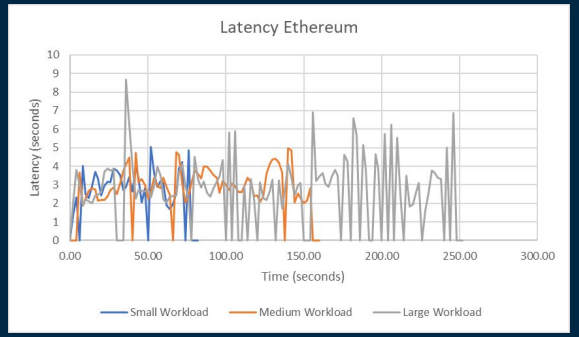
	A	B	C	D	E	F	G	H	I
1	Time	Transactions	Latency	Queue		Time (sec)	Total tx	Throughput	Tx latency
2	21:14:43	0	0	1		0.00	0	0	#N/A
3	21:14:45	1	2.02027	74		2.00	1	0.5	2.02027
4	21:14:47	21	79.7122	124		4.00	22	10.5	3.795819
5	21:14:49	174	523.414	14		6.00	196	87	3.008126
6	21:14:51	36	67.9949	50		8.00	232	18	1.888747
7	21:14:53	85	187.966	35		10.00	317	42.5	2.211365
8	21:14:55	65	136.322	41		12.00	382	32.5	2.097262
9	21:14:57	88	175.772	26		14.00	470	44	1.997409

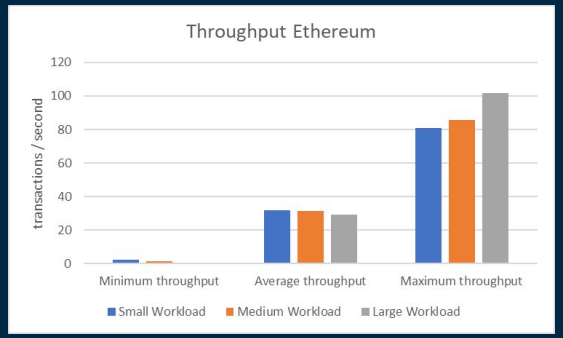


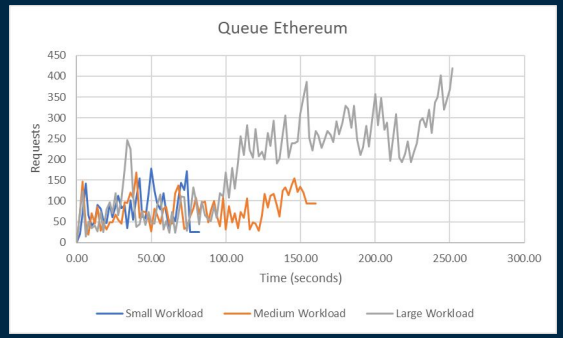


Results (Ethereum)

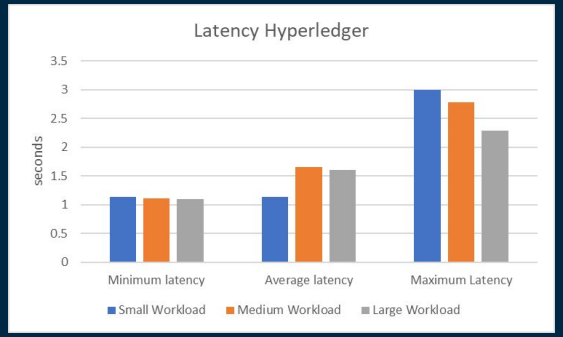


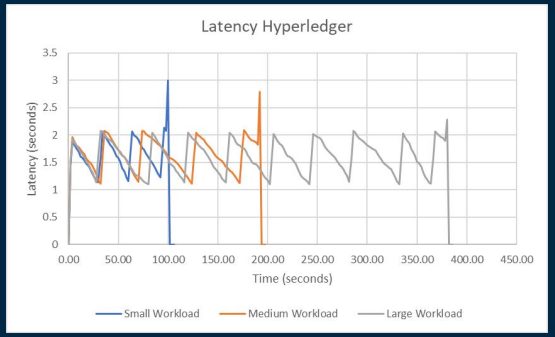


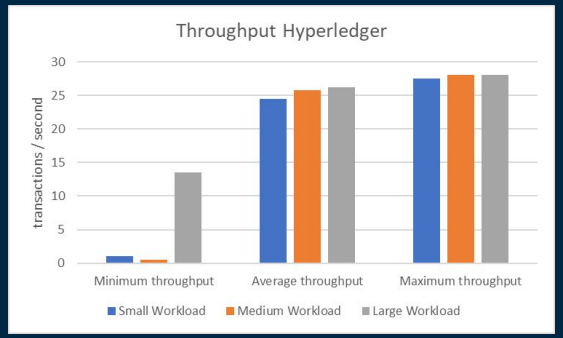


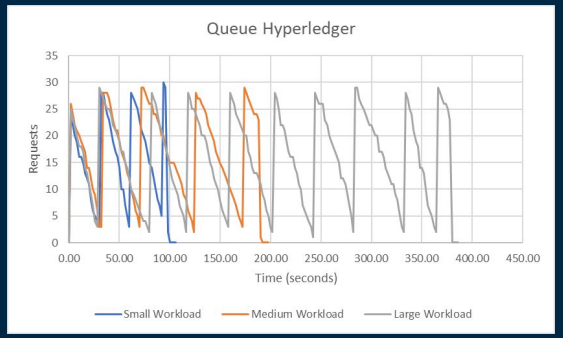


Results (Hyperledger)







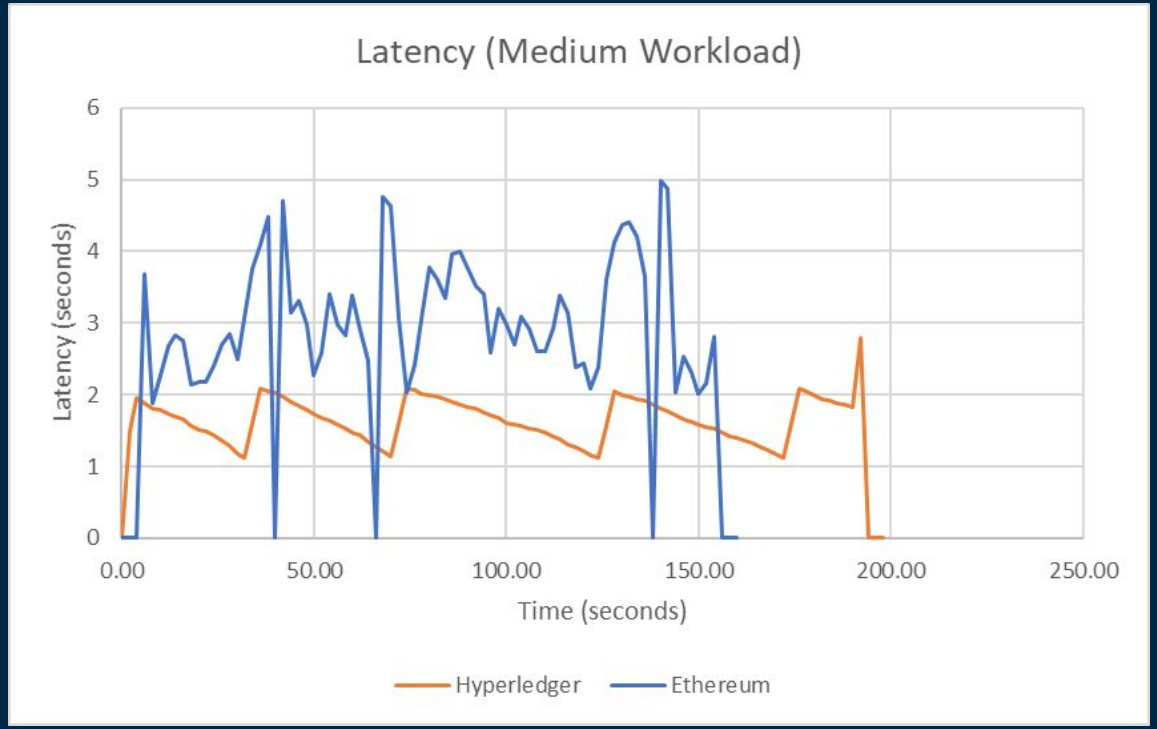


DISCUSSION

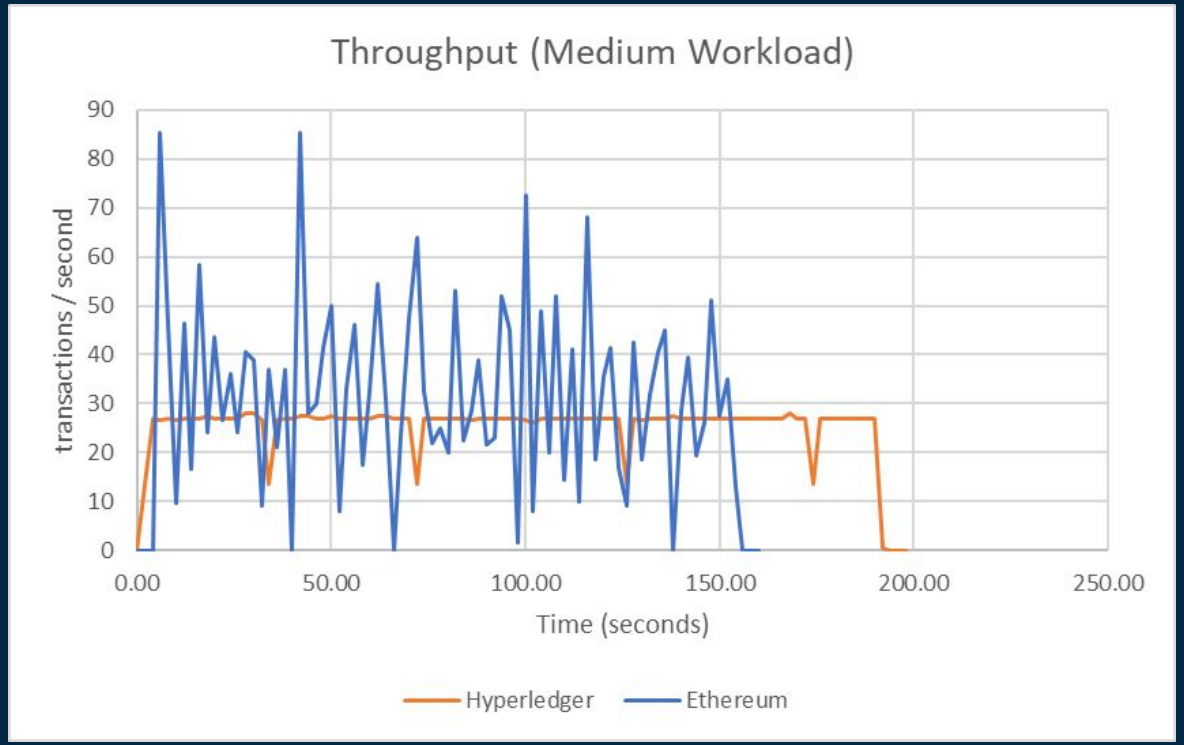
Comparison between the two projects

04

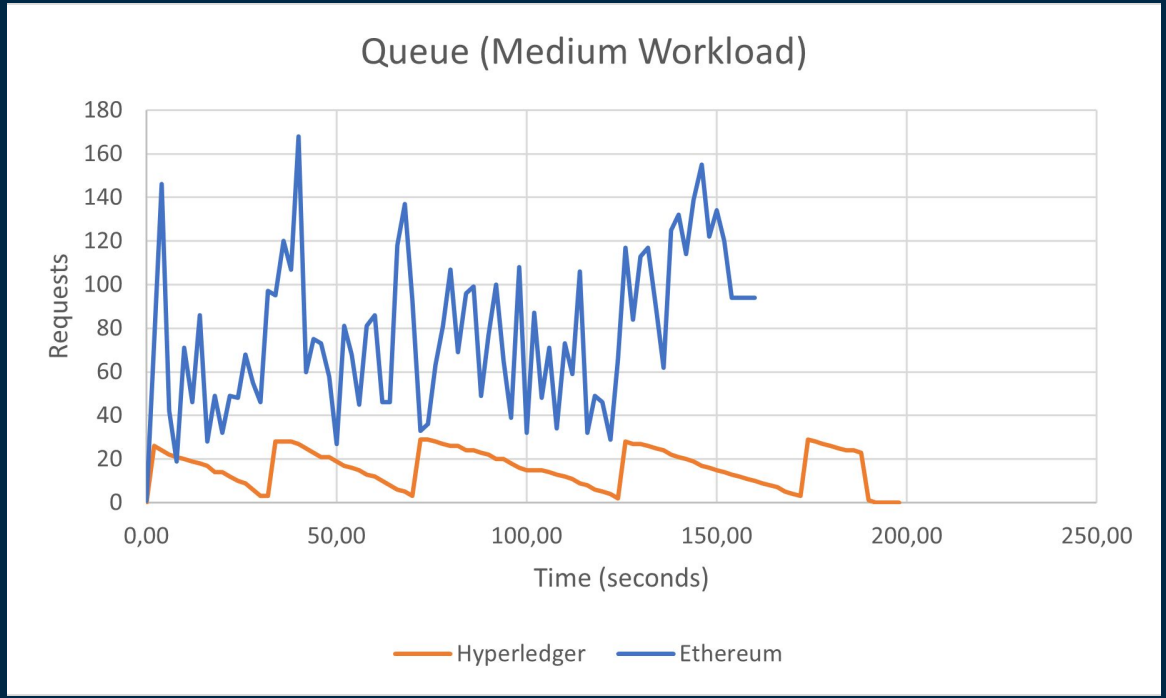
Comparison (Latency)



Comparison (Throughput)



Comparison (Queue)



CONCLUSION

05

Conclusion

- Benchmarks were mostly successful with only minor issues
- Both Hyperledger Fabric and Ethereum are performant and suitable for their own environments
  - Hyperledger Fabric achieved better latency and smaller queues