Execute smart contract transactions concurrently using optimistic Software Transactional Memory systems (STMs)

Om Sitapara, Harshit Patel

Sathya Peri, Parwat Singh Anjana

INTRODUCTION

- **Software Transaction Memory Systems(STMs)** are a convenient programming interface for a programmer to access shared memory using concurrent threads without worrying about concurrency issues.
- STMs export the following methods: t_begin, t_read, t_write, tryC, tryA.
- Currently in most commonly used blockchains the smart contract transactions are executed serially (both miner and validator) resulting in poor throughput.

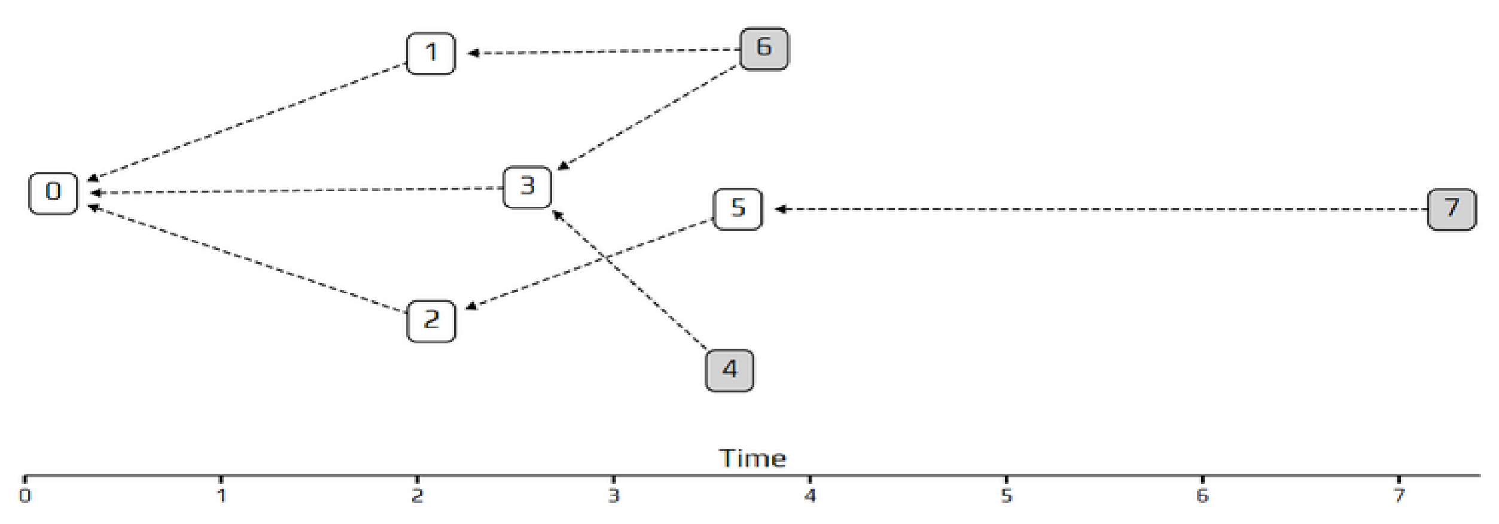
Block Chain	Number of transactions per second
BitCoin	7
Ethereum	15
IOTA (Tangle)	50
EOSIO	3000
Visa	24000

Objective

- So from the above table we can see that currently the throughput for blockchains is very less. Thus the objective is to add concurrency in smart contract execution to achieve higher efficiency and throughput.

Block-Chains Studied

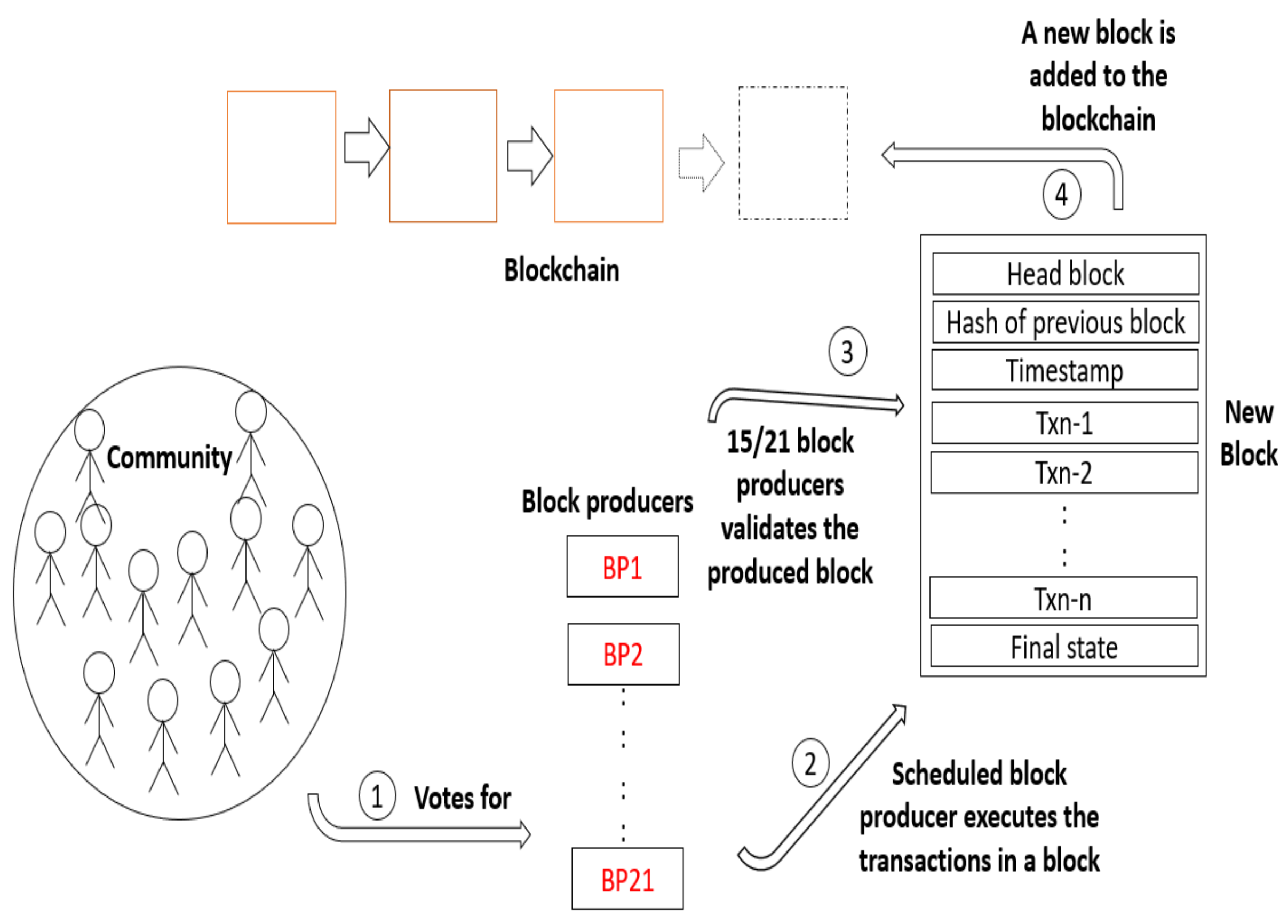
1. **IOTA (tangle):** Tangle is the data-structure behind IOTA which is a Block-DAG where vertices represent transactions and edges represent approvals[1].



- In IOTA there are no miners or validators. To add a transaction in the tangle, it has to validate two previous transactions which are represented by direct edges.

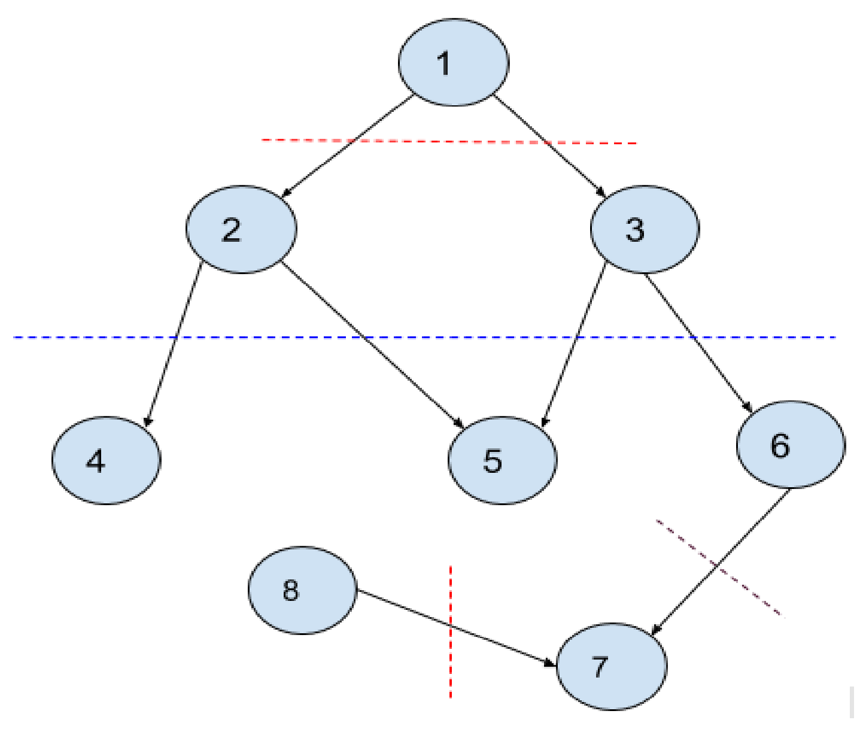
2. **EOSIO:** EOSIO consists of two main components

- 1) EOSIO Blockchain 2) EOSIO-cdt (contract development toolkit)
- EOSIO-cdt uses eosio-cpp compiler to convert smart contracts written in C++ to WASM. (web assembly code)
- A producer produces a block for 0.5 seconds which is then validated by other producers using DPOS-3.0 consensus protocol


A new block is added to the blockchain
4
Blockchain
Head block
Hash of previous block
Timestamp
Txn-1
Txn-2
Txn-n
Final state
New Block
3
15/21 block producers validates the produced block
Community
Block producers
BP1
BP2
BP21
1 Votes for
2 Scheduled block producer executes the transactions in a block

Methodology

- STM produces a conflict-list ensuring serializability using which we create a Block graph that captures the conflict relations among the transactions.
- Block graph is generated concurrently as the transactions are executed.
- This block graph will further be used by the validators to execute the transactions and reach the same final state as the miners.
- Miner stores this block graph in the block and pass it to the validators.



Phase No	Transactions executed Concurrently
1	1,8
2	2,3
3	4,5,6
4	7

Experimental Details

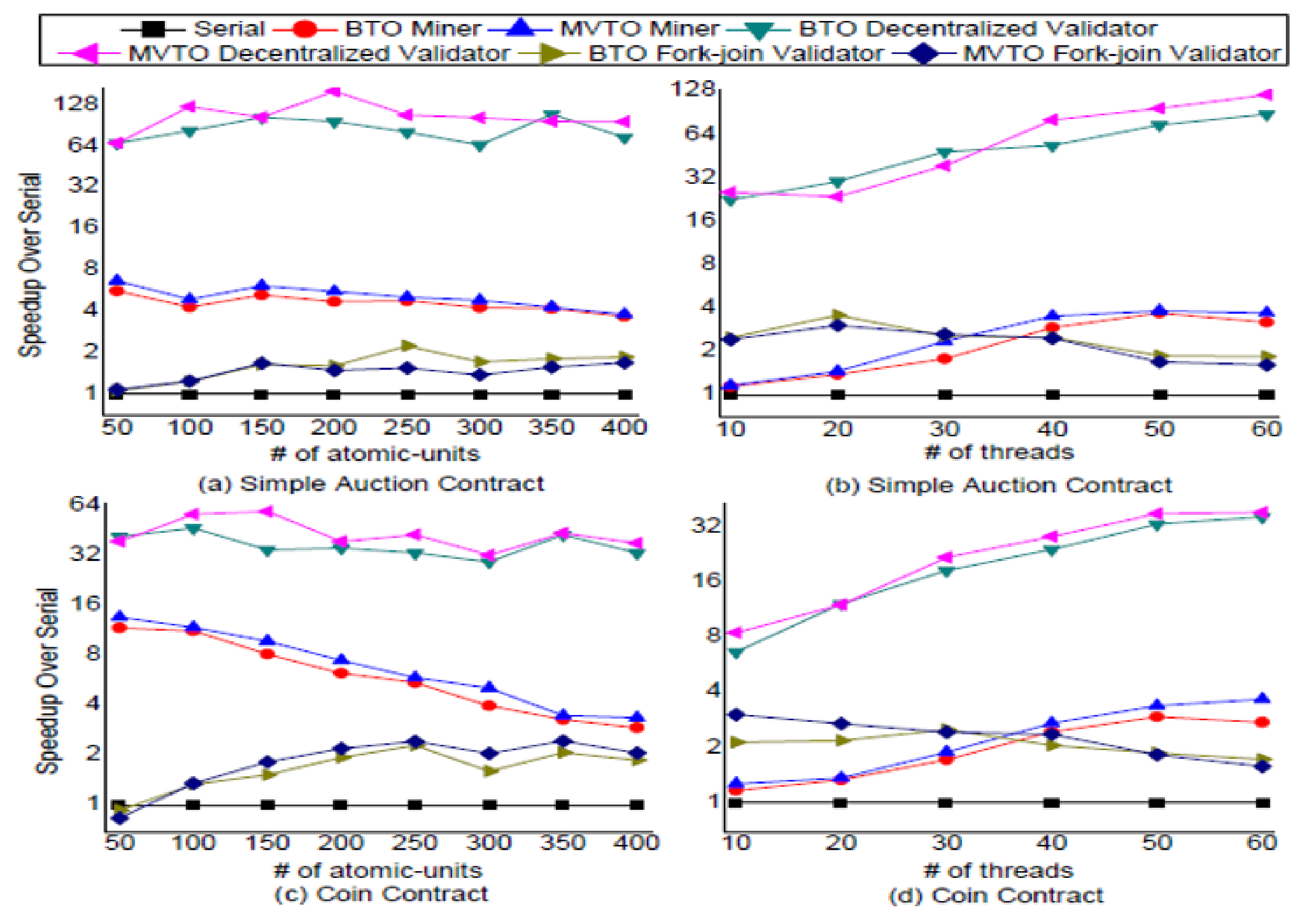
- EOSIO has been used as it is written in C++ and smart contracts are also written is C++.
- Setup the new experimental EOSIO blockchain network and Esoio.cdt (contract development toolkit).
- Modify the EOSIO blockchain such that it will add the block graph to the block proposed.
- Integrating the lock-free graph library with EOSIO.
- Integrating STM library with Eosio and writing an on-chain smart contract with STM.
- Created an on-chain smart contract simulating the exact working of off-chain smart contract.
- Implemented a new method to execute this on-chain smart contracts transactions both sequentially and concurrently.

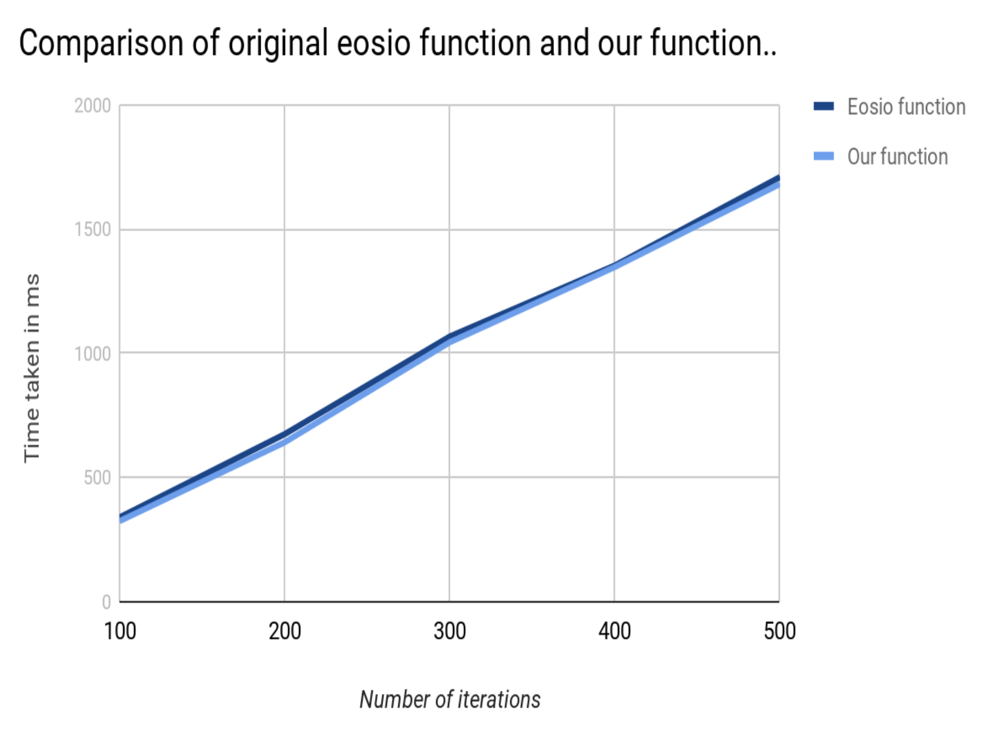
➢ **CHALLENGES:**

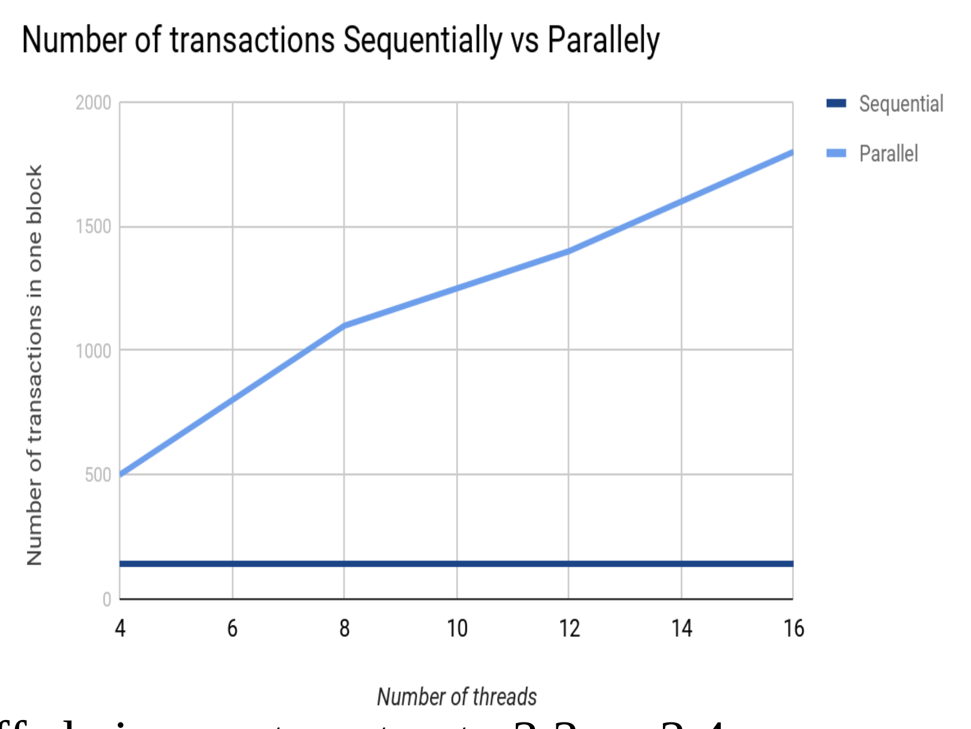
- Unable to integrate the STM library with Eosio-cdt because WASM does not supports atomic instructions.

Experimental Evaluation

- Results obtained by simulating STM [2]







- Time taken by for one transaction using off chain smart contract : 3.3ms-3.4ms
- Number of transactions being processed in one block sequentially : 145-150
- Number of transactions being processed in one block concurrently: 1100-1200

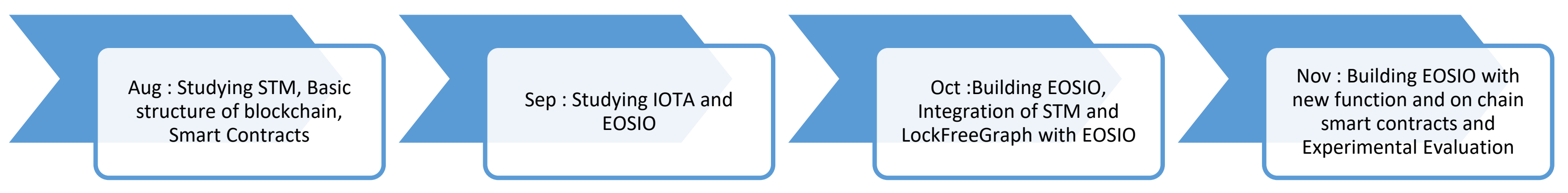
Conclusions

- The result of STM simulation shows that it can achieve higher throughput by providing an efficient framework to parallelize smart contract transactions.
- Thus we conclude that the results obtained by parallelizing the new implemented function will be same as parallelizing the already written function is EOSIO.
- IOTA contains a single transaction in a block, hence cannot be integrated with STM.

Future Work

- To write a real world smart contract with STM and try to parallelize it when there are dependency among the transactions.
- Validating the transactions using the block graph.
- Compare the performance and measure the speed up of using the STM inside smart contracts.
- This approach can be extended to any blockchain containing multiple transactions in a block.

Timeline


Aug : Studying STM, Basic structure of blockchain, Smart Contracts
Sep : Studying IOTA and EOSIO
Oct :Building EOSIO, Integration of STM and LockFreeGraph with EOSIO
Nov : Building EOSIO with new function and on chain smart contracts and Experimental Evaluation

References

- [1]https://blog.iota.org/the-tangle-an-illustrated-introduction-c0a86f994445
- [2] Parwat Singh Anjana, Sweta Kumari, Sathya Peri, Sachin Rathor, and Archit Somani. An efficient framework for optimistic concurrent execution of smart contracts. In 2019 *27th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP)*, pages 83–92. IEEE, 2019