

PEEP: A Parallel Execution Engine for Permissioned Blockchain Systems

Zhihao Chen, Xiaodong Qi, Xiaofan Du, Zhao Zhang, Cheqing Jin
chenzh@stu.ecnu.edu.cn

Presented at DASFAA 2021



華東師範大學
EAST CHINA NORMAL UNIVERSITY



SCHOOL OF DATA
SCIENCE & ENGINEERING
数据科学与工程学院

Outline

- Introduction
- Contribution
- Related Work
- System Overview
- System Details
- Experiment
- Conclusion

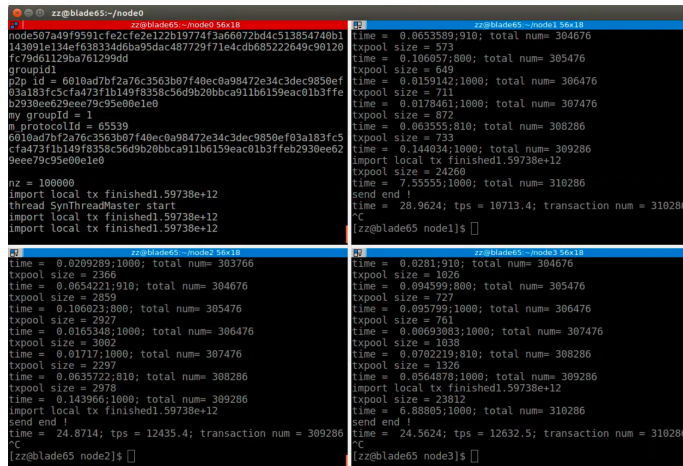
Introduction

- Blockchain provides data integrity, transparency and immutability to tackle trust problems among **mutually distrustful** parties
- **Permissioned blockchain** is being widely applied to support large-scale businesses in enterprise collaborations



Introduction

- Permissioned settings opens an opportunity for seeking **higher throughput** under stricter assumptions
- However, there is **a big performance gap** between efficient consensus and conventional serial execution



The image displays four terminal windows, each representing a node in a private network. Each window shows the output of a performance measurement script. The output includes metrics such as 'time', 'txpool size', 'total num', and 'transaction num'. The results indicate that the system achieved a throughput of approximately 15,000 transactions per second (TPS) across the four nodes.

```
node507a49f9591cfe2cfe2e122b19774f3a66072bd4c513854740b1
143091e134ef638334d6ba95dac487729f71e4cdeb685222649c90120
fc79d61129ba761299dd
group id = 1
b2p_id = 6010ad7b72a76c3563b07f48ec0a98472e34c3dec9850ef
03a183f3c5cfa473f1b149f8358c56d9b20bbca911b6159eac01b3ffe
b2930ee229ee79c95e0e1e0
my_group id = 1
n_protocol id = 65539
6010ad7b72a76c3563b07f48ec0a98472e34c3dec9850ef03a183f3c5
cfa473f1b149f8358c56d9b20bbca911b6159eac01b3ffe2930ee2
9ee79c95e0e1e0
hz = 100000
import local tx finished1.59738e+12
thread SynThreadMaster start
import local tx finished1.59738e+12
import local tx finished1.59738e+12
time = 0.0209289; total num= 303766
txpool size = 2366
time = 0.0654221; total num= 304676
txpool size = 2859
time = 0.186023; total num= 305476
txpool size = 2927
time = 0.0165348; total num= 306476
txpool size = 3092
time = 0.01717; total num= 307476
txpool size = 2297
time = 0.0635722; total num= 308286
txpool size = 2978
time = 0.143966; total num= 309286
import local tx finished1.59738e+12
send end !
time = 24.8714; tps = 12435.4; transaction num = 309286
^C
zz@blade65 node2]$
```

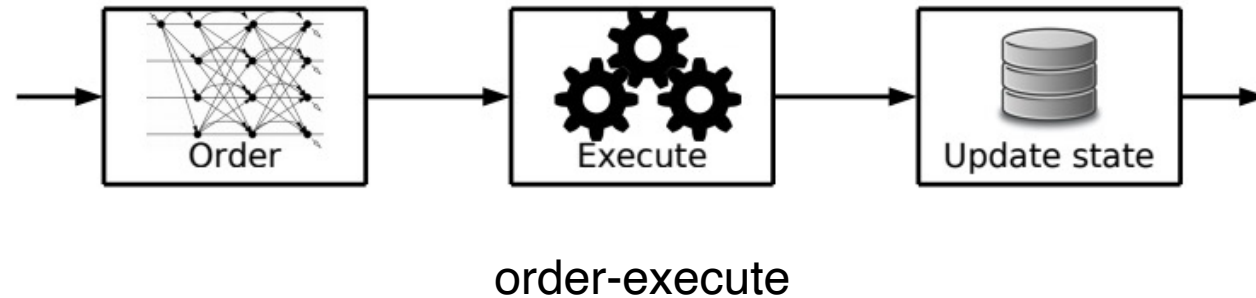
- We have measured the performance of **PBFT** on 4 nodes in a private network, which results in over **15K TPS**
- While the execution only reaches **half order-of-magnitude**

Contribution

- We propose **PEEP**, a novel execution engine **incorporate concurrency** for order-execute blockchain
- To further scale the transaction processing, we present a lock-free **parallel update algorithm** for state trie and realize the **non-blocked workflow**
- We implement a prototype for PEEP integrating above techniques and conduct extensive experiments

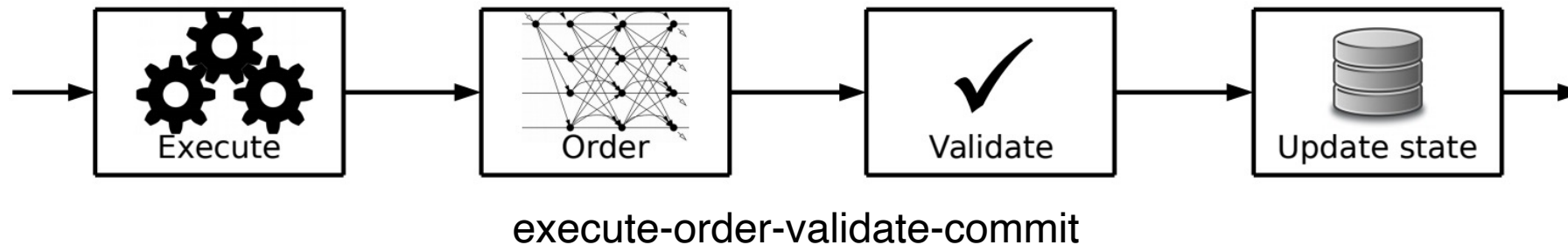
Related Work (Optimized execution)

- Two-phase execution framework
 - **blocked** & poor performance
- Parblockchain exploits transaction **parallelism** (order-execute paradigm)
 - priori knowledge of transactions
 - using dependency graph



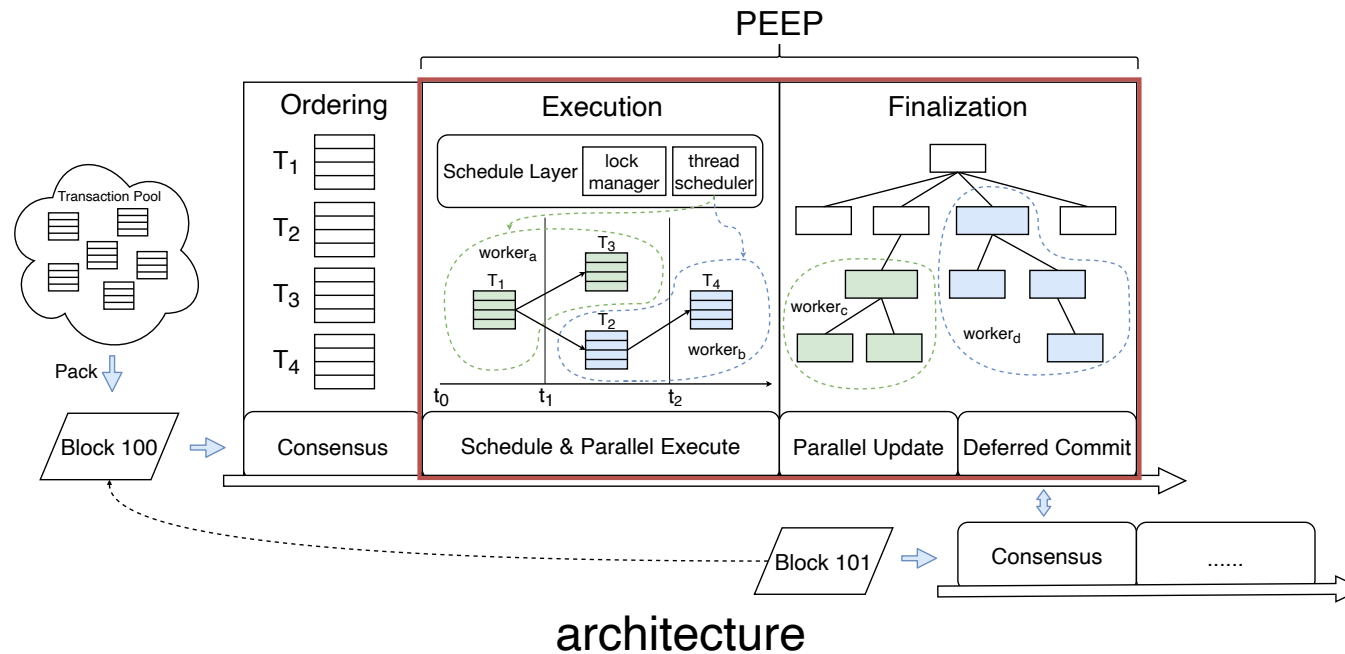
Related Work (Optimized execution)

- Fabric incorporate concurrency (execute-order-validate-commit paradigm)
 - serial validation
 - high abort rates for hotspot workloads
 - enhanced works still inherits the limitations of serial validation



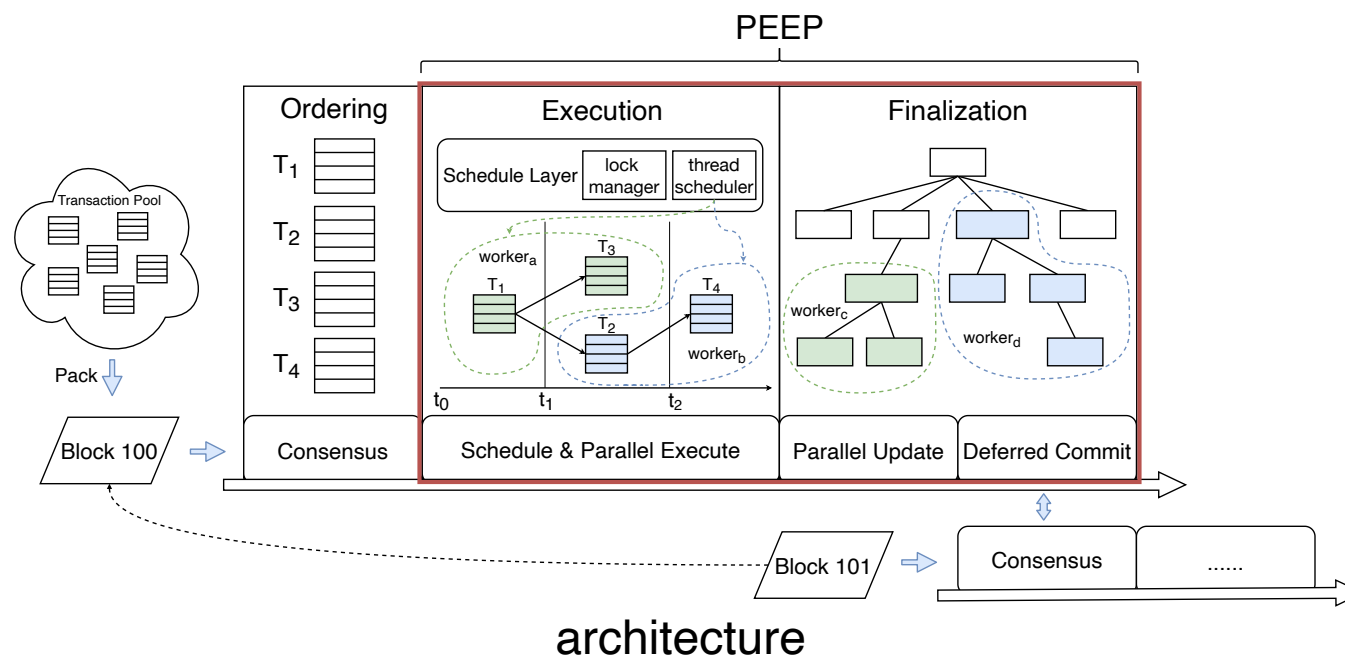
System Overview

- Assumptions & Architecture
 - **order-execute** workflow
 - **early** read/write-set acquisition
 - adversary model



System Overview

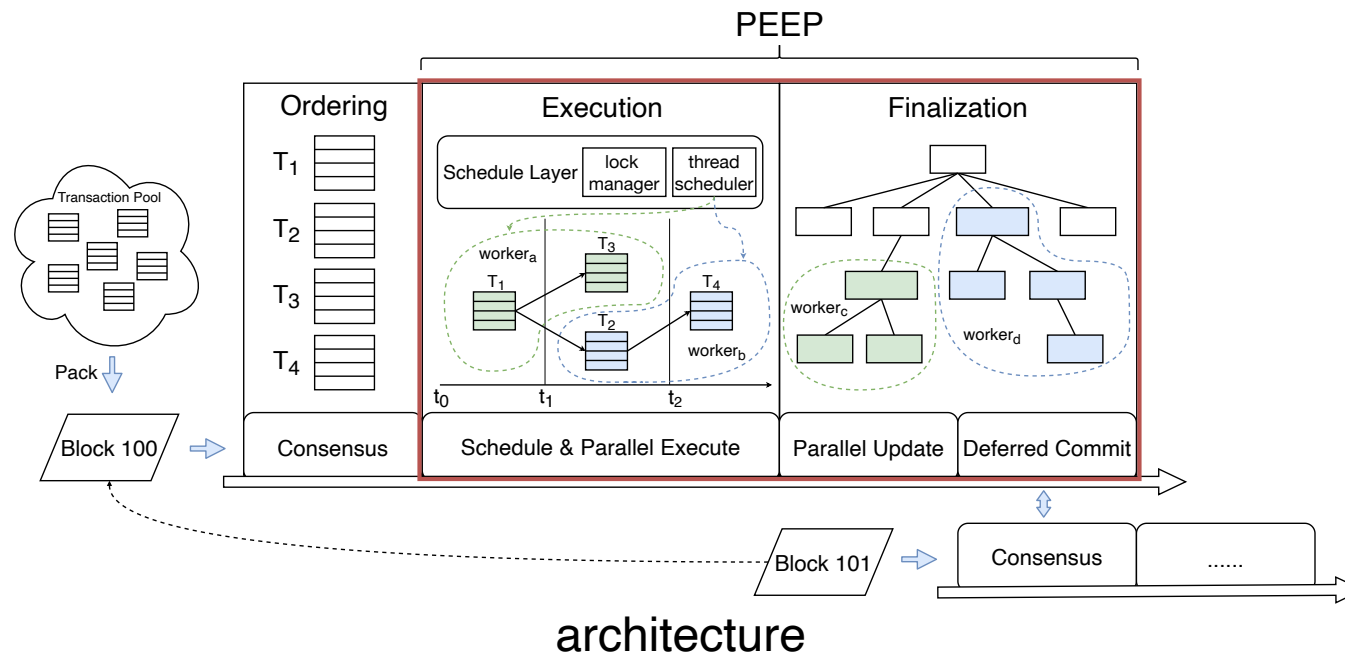
- Ordering
 - merely **order** the transactions through consensus



System Overview

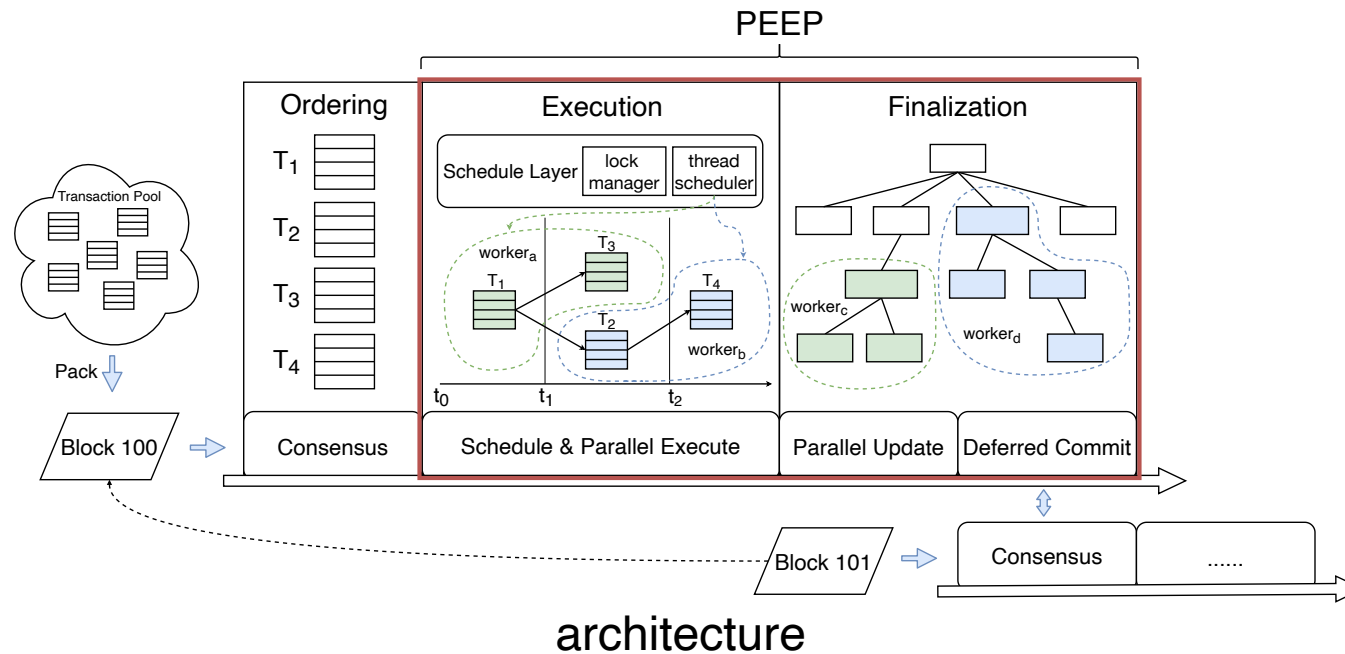
- Execution

- allow **parallel execution** with **deterministic** serial order ← defined by ordering
- incur no additional network communication among replicas for negotiation of the final result



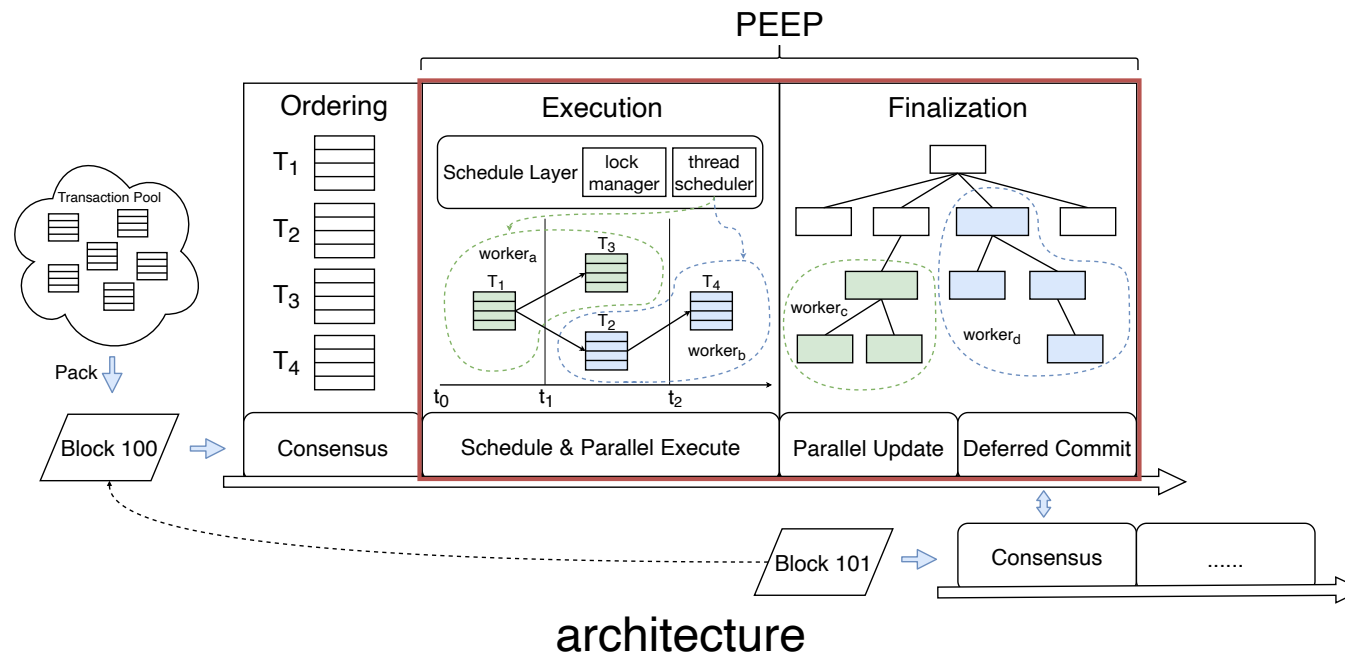
System Overview

- Finalization
 - take full advantage of the batch property
 - **parallelizes** the updating on state trie



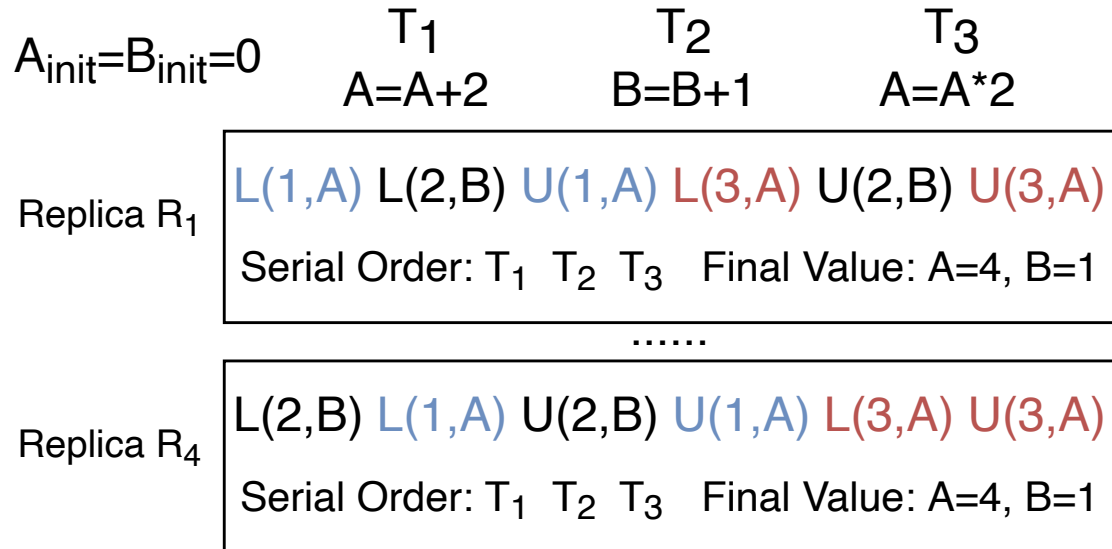
System Overview

- Non-Blocked Workflow
 - deferred commit strategy
 - better utilization of various resources

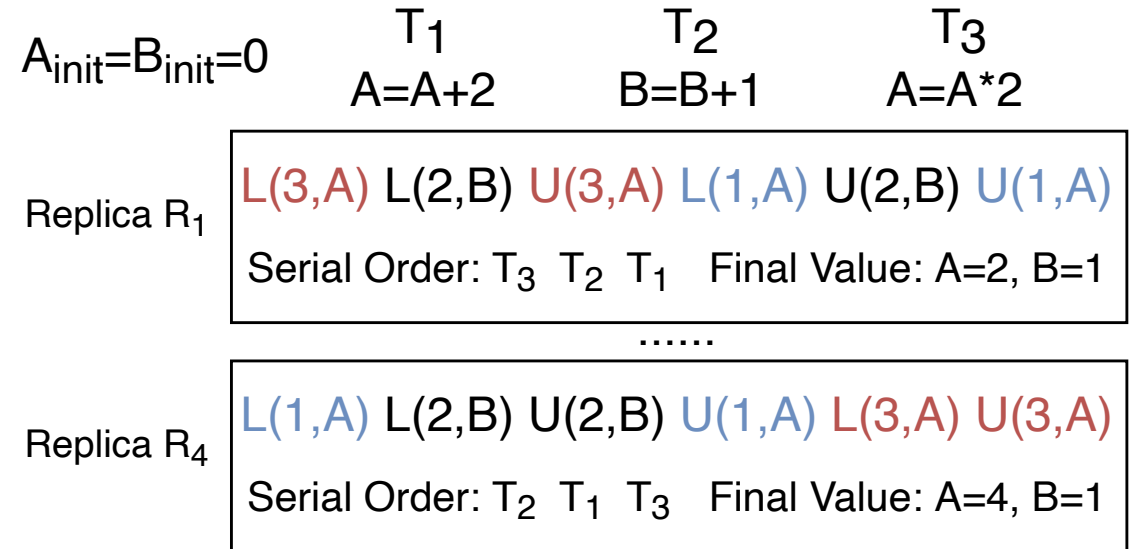


System Details

- Parallel execution
 - challenge: **eliminate non-determinism** among replicas



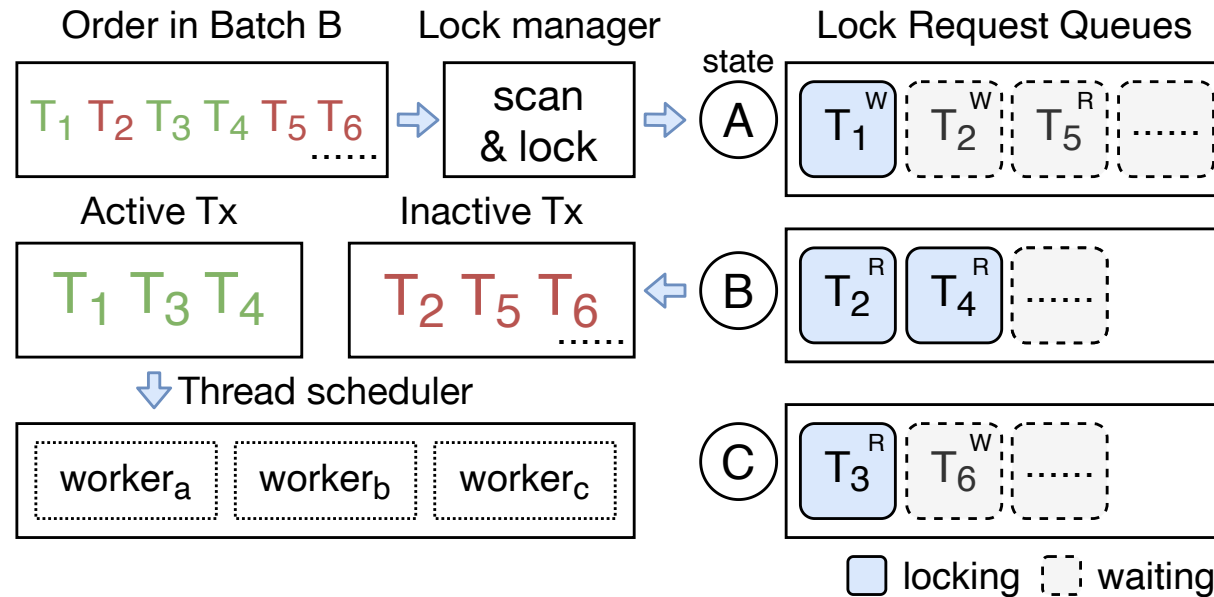
determinism



non-determinism

System Details

- Parallel execution
 - solution: schedule transactions based on **ordered locking mechanism**
 - single schedule thread and multiple worker threads



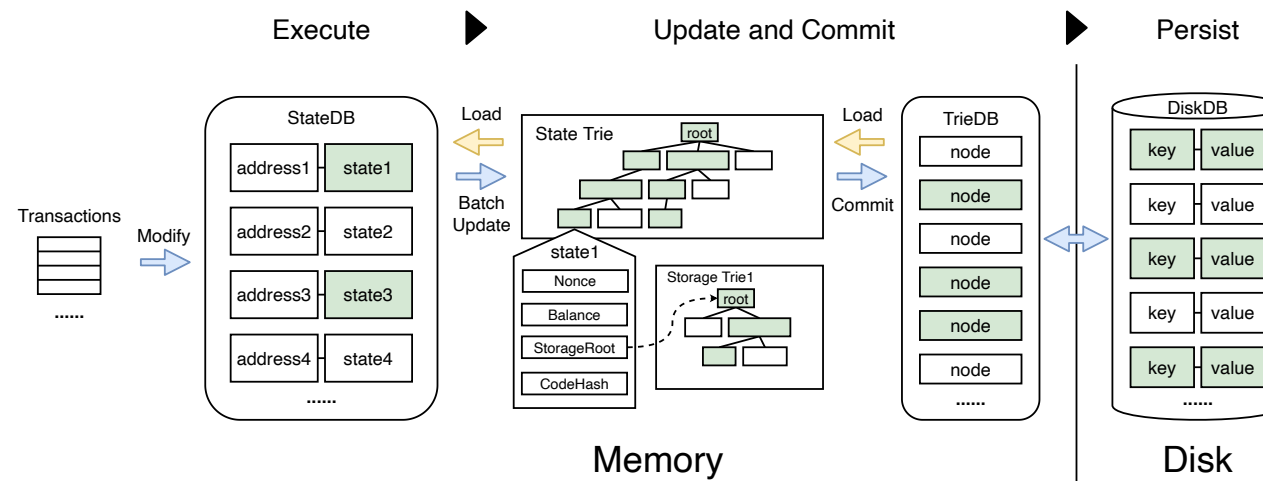
lock example of ordered locking

System Details

- Advantages & Discussion
 - improve the overall performance
 - **prevent** non-deterministic behaviors
 - allow each replica to execute transactions in parallel **independently**
 - incur **non-logical aborts** mentioned as inherent limitations in Fabric
 - only acquire the **key** in prior

System Details

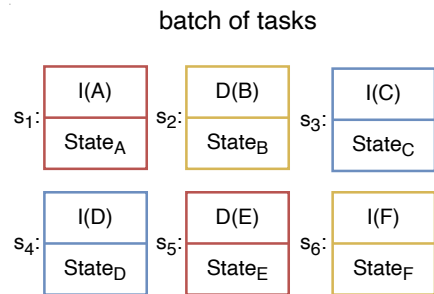
- Parallel Finalization
 - few works explore concurrency on Merkle trees
 - utilize the **batch property**
 - design a **lock-free parallel update** algorithm on state trie



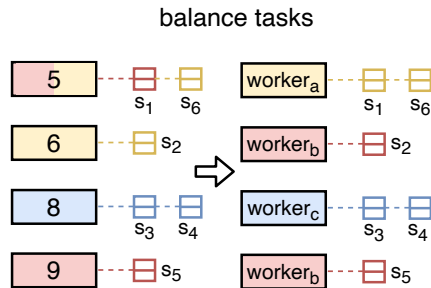
The role of state trie

System Details

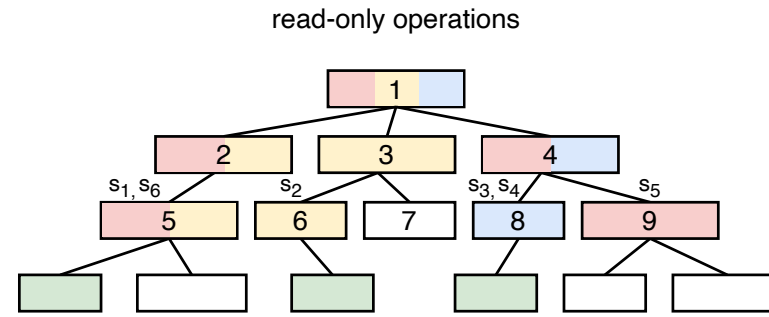
- Parallel Finalization
 - regard each update $\langle k, v \rangle$ as a task
 - detect the potential contention among tasks



Step 1

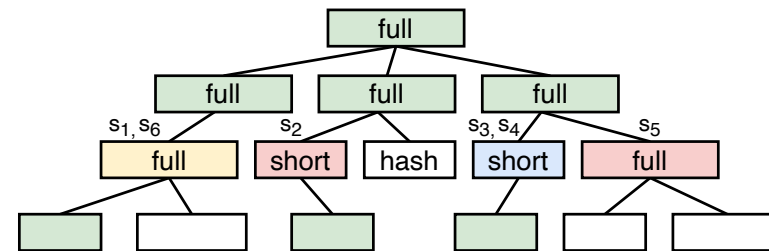


Step 3



Step 2

handle modification



Step 4

System Details

- Deferred commit strategy
 - **reform** the state root padding
 - leading to a **non-blocked** workflow
 - **various** system resources of each replica are **fully utilized**, e.g., computation、disk I/O、 network I/O
 - at the acceptable cost of a **deferred authenticate query**

Experiment

- Implementation
 - ordering:
 - ✦ we **integrate** an open-source **PBFT** realization in Hyperledger Fabric 0.6
 - execution:
 - ✦ we set an **EVM**(Ethereum VM) pool with **multiple instances** to provide the execution environment
 - finalization:
 - ✦ we reuse the **MPT** component in Ethereum and **implement** our parallel update algorithm upon it

Experiment

- Setup
 - All experiments are conducted on four qualified machines with an Intel Core i7-7700HQ CPU @ 2.80GHz of 4 cores, 16GB RAM and 1TB disk space
 - Benchmark:
 - ✦ a **SmallBank** contract as Macro-benchmark in BlockBench is deployed on one account
 - ✦ 1 million transactions are sent by 200 thousand accounts to invoke this smart contract under Zipfian distribution.
 - ✦ a block is **limited** to contain at most 1 thousand transactions

Experiment

- Overall Performance

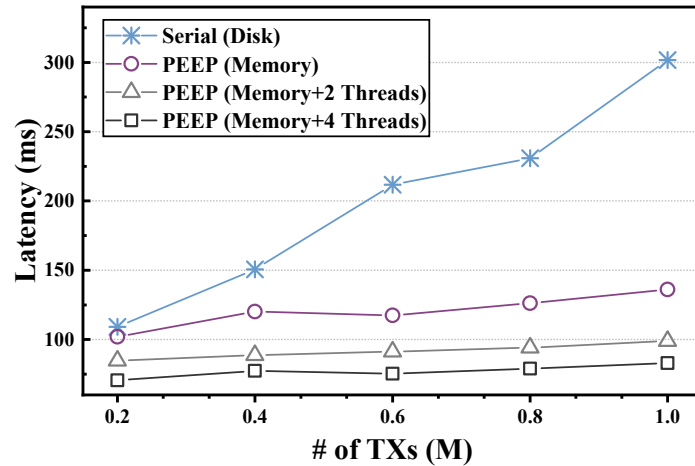


Fig1. Latency

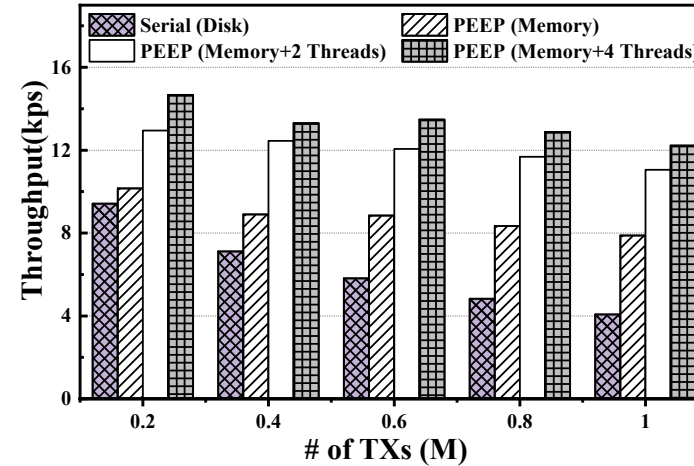


Fig2. Throughput

- against the number of transactions without consensus
- 50% overall enhancement

Experiment

- Analysis of parallel execution

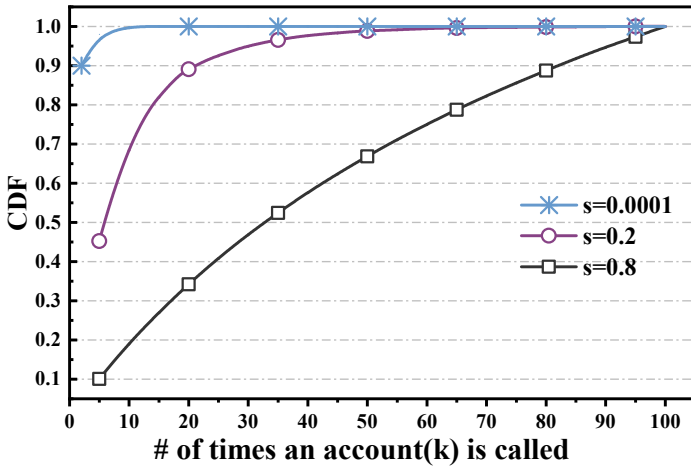


Fig3. CDF of # account calls

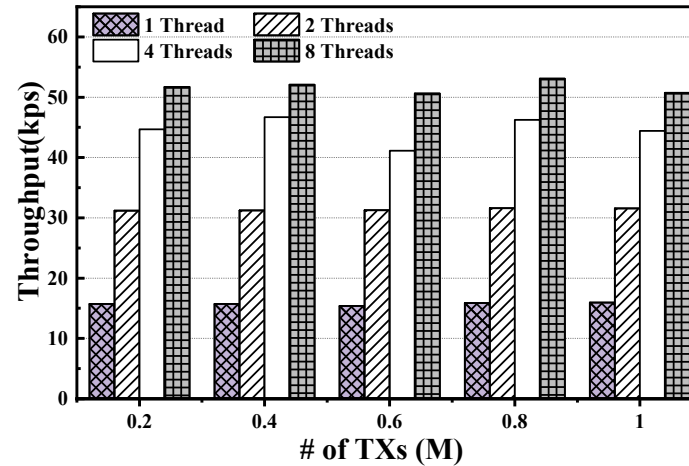


Fig4. Throughput

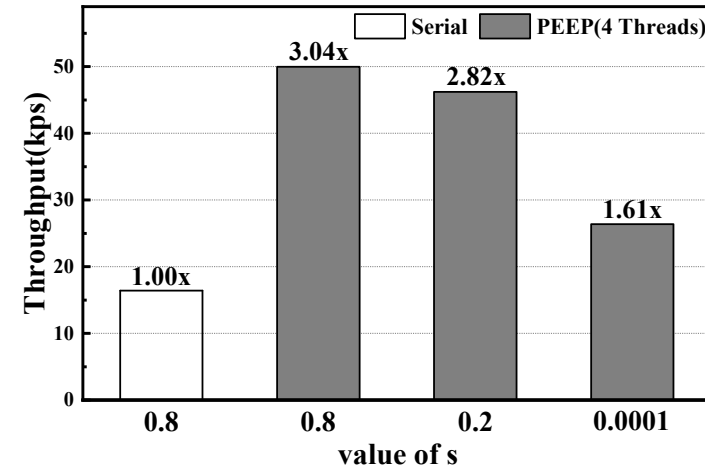


Fig5. Performance degradation

- significant improvement
- degrade under extreme contention

Experiment

- Analysis of parallel finalization

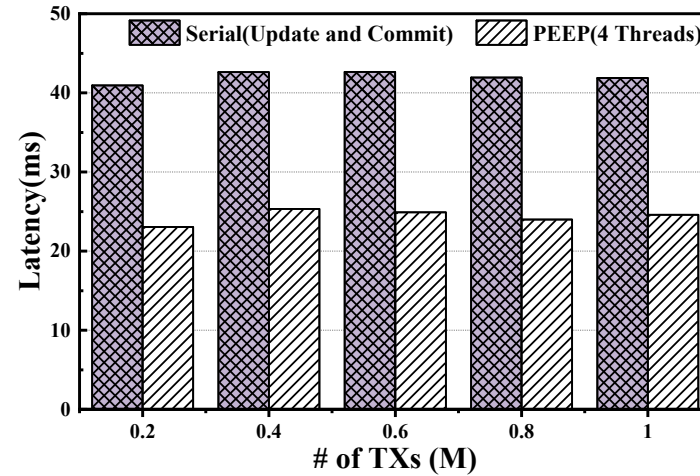


Fig6. Latency per block

- empower the finalization by **one order of magnitude**

Experiment

- Analysis of deferred commit strategy

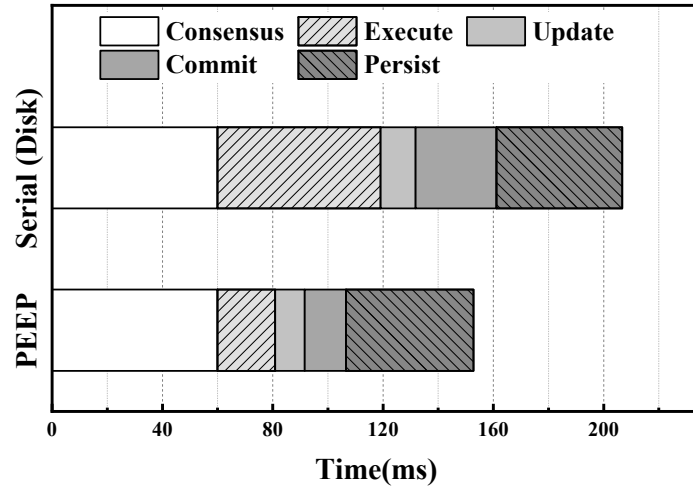


Fig7. Time of each sub-phases

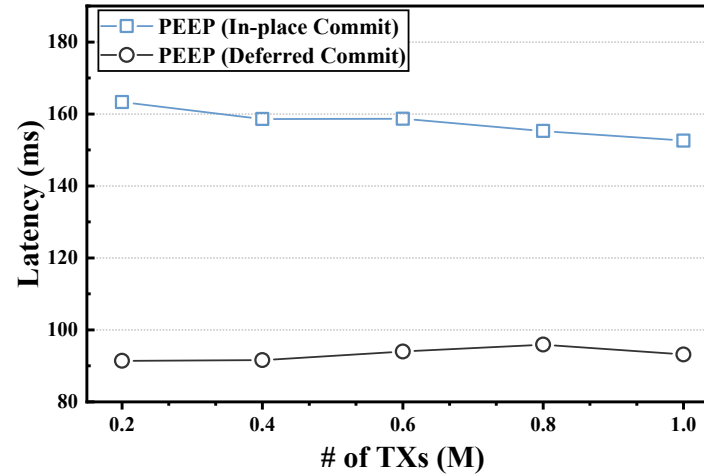


Fig8. Lantency

- non-blocked workflow
- **early** confirmation

Conclusion

- We introduces PEEP, a parallel execution engine designed for permissioned blockchain systems for higher throughput
- Optimized execution and finalization with a deferred strategy
 - experimental results support the practicability and effectiveness
- Future works
 - explore pipelined workflow
 - design more efficient authenticated data structures

THANKS !