NeuChain: A Fast Permissioned Blockchain System with Deterministic Ordering

Zeshun Peng, Yanfeng Zhang, Qian Xu, Haixu Liu,
Yuxiao Gao, Xiaohua Li, Ge Yu

Northeastern University, China

High Throughput Blockchain

- Existing blockchain systems cannot meet the demand of high throughput applications.

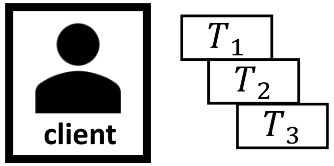
(a) Financial applications

(b) Internet of things

(c) Industrial supply chain

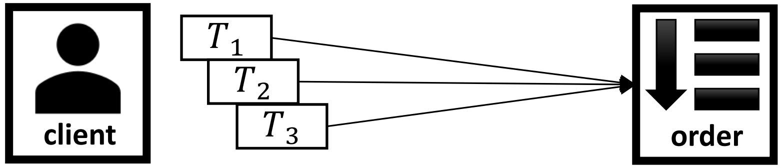
Architecture: Order-Execute-Validate (OEV)

- Examples: Bitcoin, Ethereum, Quorum, ResilientDB [VLDB'20], etc.



Architecture: Order-Execute-Validate (OEV)

- Examples: Bitcoin, Ethereum, Quorum, ResilientDB [VLDB'20], etc.

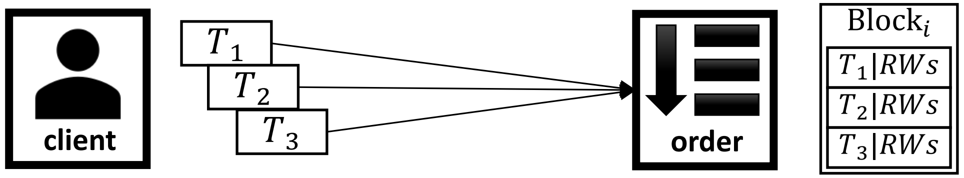


Workflow:

- The consensus leader receives clients' transactions.

Architecture: Order-Execute-Validate (OEV)

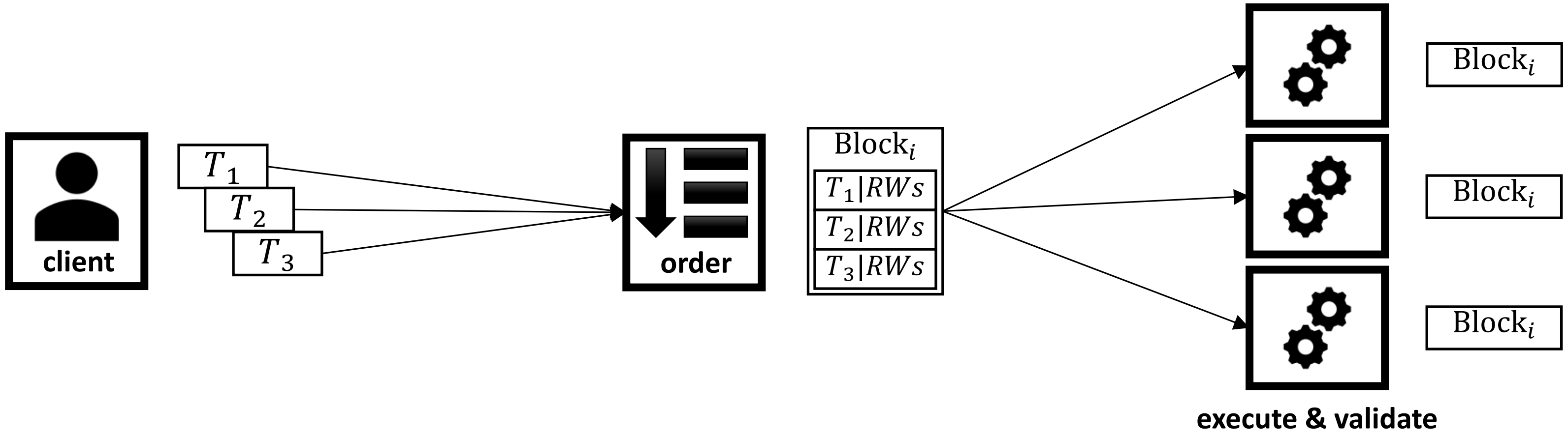
- Examples: Bitcoin, Ethereum, Quorum, ResilientDB [VLDB'20], etc.



Workflow:
- The consensus leader receives clients' transactions.
- The leader orders transactions into blocks.

Architecture: Order-Execute-Validate (OEV)

➢ Examples: Bitcoin, Ethereum, Quorum, ResilientDB [VLDB'20], etc.



Workflow:

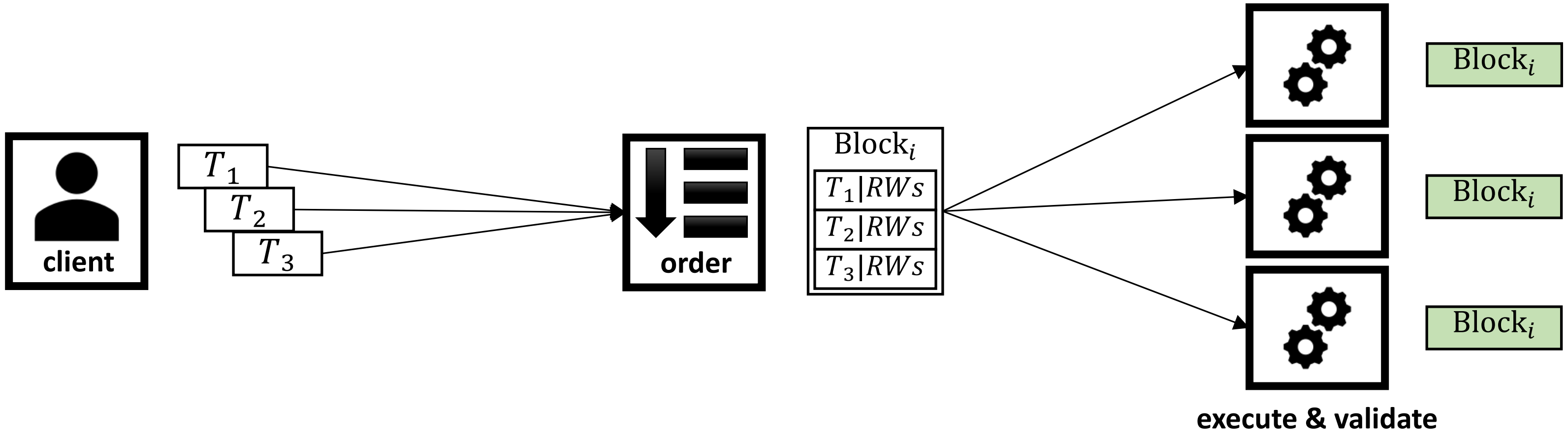
➢ The consensus leader receives clients' transactions.

➢ The leader orders transactions into blocks.

➢ Broadcasts the block to other peers.

Architecture: Order-Execute-Validate (OEV)

- Examples: Bitcoin, Ethereum, Quorum, ResilientDB [VLDB'20], etc.

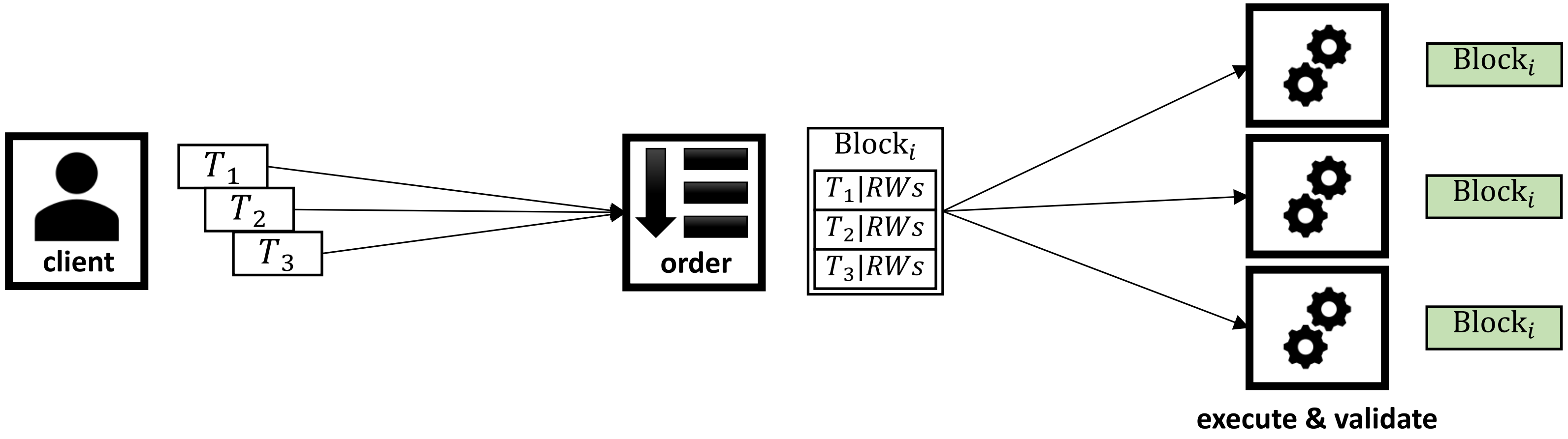


Workflow:

- The consensus leader receives clients' transactions.
- The leader orders transactions into blocks.
- Broadcasts the block to other peers.
- Validate the block by re-executing **in that order**.

Architecture: Order-Execute-Validate (OEV)

- Examples: Bitcoin, Ethereum, Quorum, ResilientDB [VLDB'20], etc.



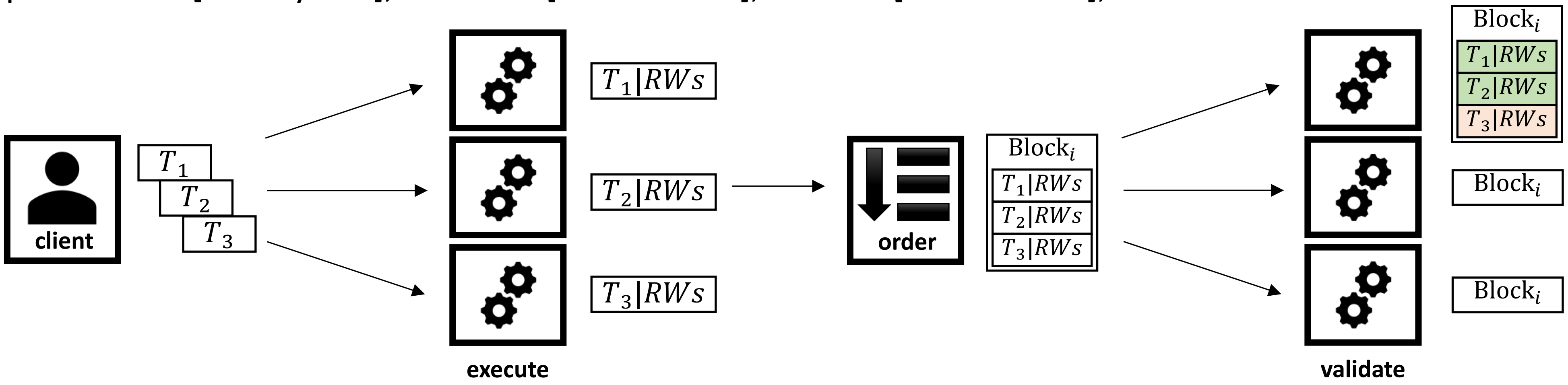
Workflow:
- The consensus leader receives clients' transactions.
- The leader orders transactions into blocks.
- Broadcasts the block to other peers.
- Validate the block by re-executing **in that order**.

Advantage:
- Simple and widely used.
- Low abort rate due to sequential execution.

However:
- Serial execution (and validation) **limits throughput**.
- Consensus leader could be **a network bottleneck**.

Architecture: Execute-Order-Validate (EOV)

- Examples: Fabric [EuroSys'18], Fabric++ [SIGMOD'19], Fabric# [SIGMOD'20], etc.

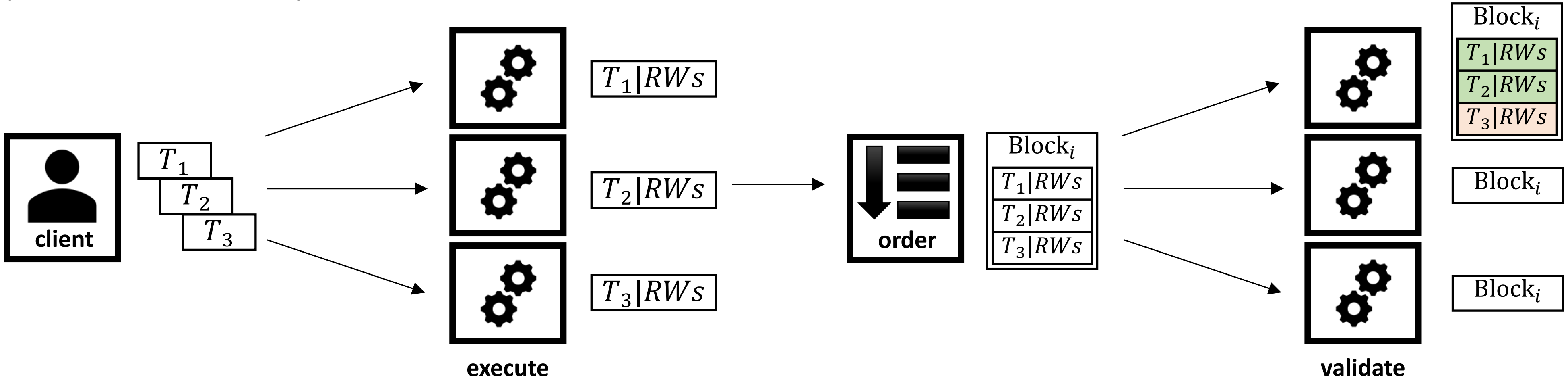


Workflow:

- Peers execute transactions **concurrently**.
- The ordering leader orders transactions into blocks.
- Peers validate the read-write sets in a block **in that order**.

Architecture: Execute-Order-Validate (EOV)

- Examples: Fabric [EuroSys'18], Fabric++ [SIGMOD'19], Fabric# [SIGMOD'20], etc.



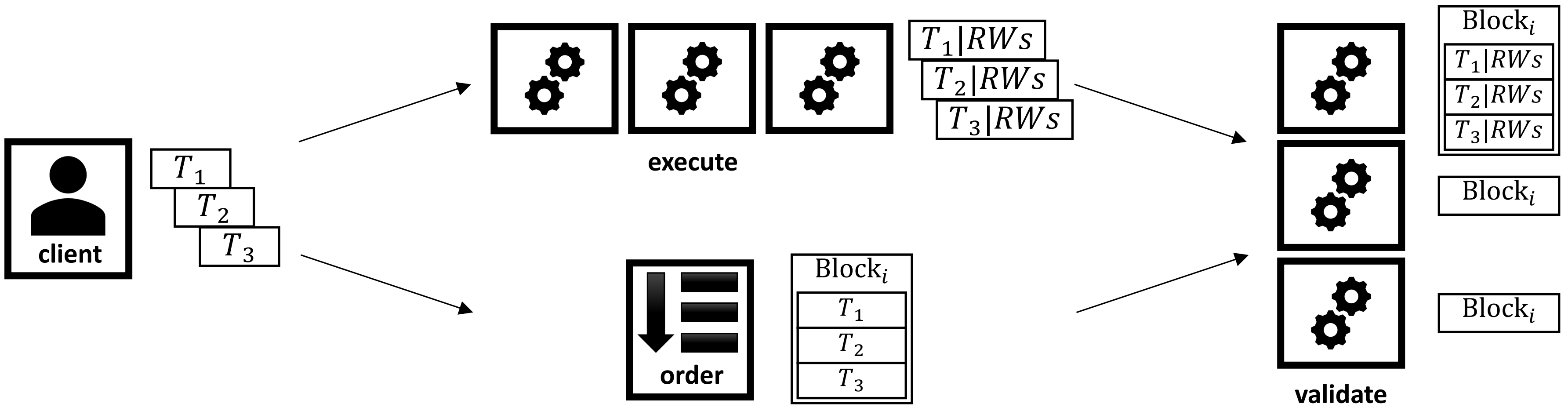
Workflow:
- Peers execute transactions **concurrently**.
- The ordering leader orders transactions into blocks.
- Peers validate the read-write sets in a block **in that order**.

Advantage:
- Tolerating non-deterministic transactions.
- Allow concurrent execution of transactions.

However:
- High abort rates due to additional inter-block conflicts.
- Explicit order among transactions.

Architecture: Order-Execute-Parallel-Validate (OEPV)

➢ Examples: FabricSSI [1], BIDL [SOSP'21], etc.



Workflow:

➢ Execute transactions while ordering.

➢ Validate based on the ordering result.

Advantage:

➢ Reduce overall latency.

However:

➢ Inherits the ordering phase of the EOV architecture.

[1] Senthil Nathan, et, al. Blockchain Meets Database: Design and Implementation of a Blockchain Relational Database, VLDB'20

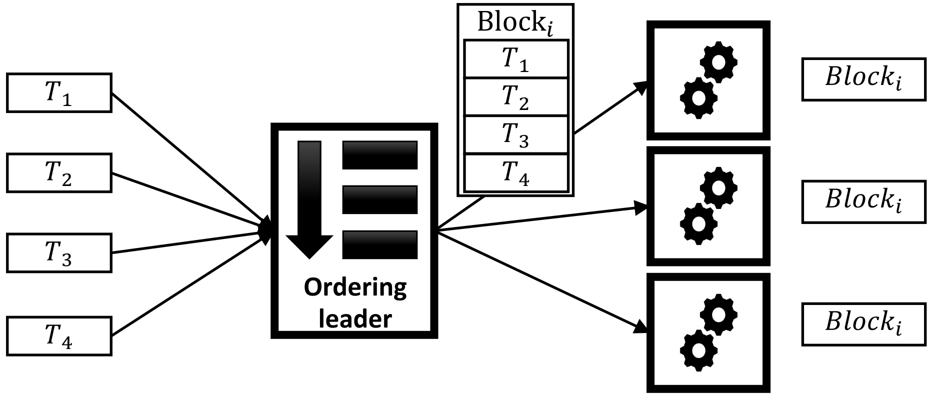
Summary of Blockchain Architectures

Systems	Architecture	Type	Consensus			Transaction Processing	
			Content	Protocol	Participant	Conflict type	Concurrency control
Bitcoin [39]	OEV	permissionless	block	PoW	all nodes	intra-block	-
Ethereum [65]	OEV	permissionless	block	PoW PoS	all nodes	intra-block	-
Quorum [7]	OEV	permissioned	block	PBFT Raft	all nodes	intra-block	-
ResilientDB [32]	OEV	permissioned	Tx batch	GEOBFT	all nodes	intra-block	-
PoE [31]	OEV	permissioned	Tx batch	PoE	all nodes	intra-block	-
Monoxide [64]	shard+OEV	permissionless	block	PoW	group of nodes	cross-shard	eventual atomicity
ByShard [34]	shard+OEV	permissioned	block	PBFT	group of nodes	cross-shard	cross-shard 2PC
SharPer [13]	shard+OEV	permissioned	block	PBFT	group of nodes	cross-shard	cross-shard commit protocol
Rivet [22]	shard+OEV	permissioned	block	HotStuff	reference shard	cross-shard	optimistic
Fabric [14]	EOV	permissioned	Tx batch	PBFT Raft	order nodes	inter&intra-block	MVOCC
FastFabric [30]	EOV	permissioned	Tx header	PBFT Raft	order nodes	inter&intra-block	MVOCC
Fabric++ [55]	EOV	permissioned	Tx batch	PBFT Raft	order nodes	inter&intra-block	MVOCC+reorder
Fabric# [52]	EOV	permissioned	Tx batch	PBFT Raft	order nodes	inter&intra-block	OCC+SSI+reorder
SlimChain [69]	EOV	both	block	PoW Raft	consensus nodes	inter&intra-block	OCC+SSI
Basil [56]	EOV	permissioned	KVs	PBFT	clients	inter&intra-block	MVTSO
Fabric SSI [40]	OEPV	permissioned	block	PBFT Raft	order nodes	intra-block	SSI
BIDL [45]	OEPV	permissioned	block	PBFT Raft	order nodes	intra-block	-
NeuChain (ours)	EV	permissioned	block number Tx batch	PBFT Raft	epoch servers client proxies	intra-block	deterministic

Drawbacks of the Explicit Ordering Phase

Traditional blockchains all have an ordering phase, which could **limit the throughput**.

➢ The ordering leader must replicate blocks to all followers.



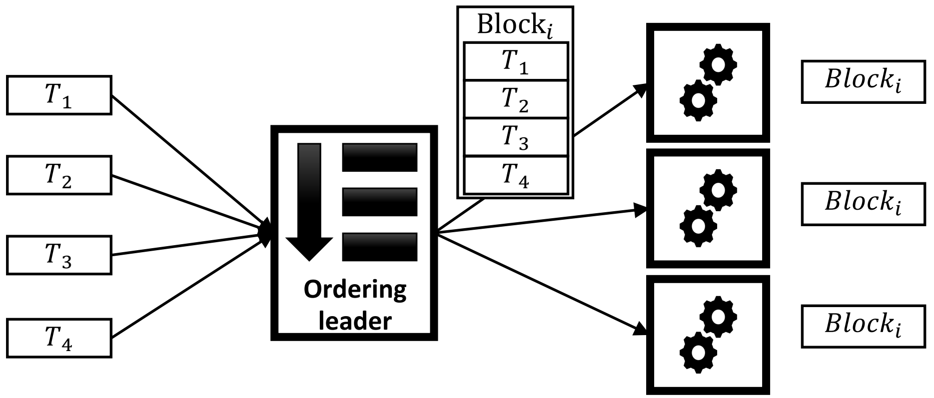
Replicate a block to followers

➢ **The maximum bandwidth** of the leader could be a bottleneck.

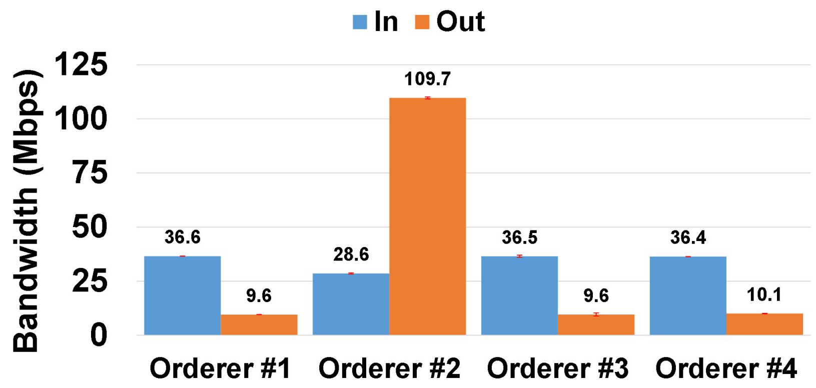
Drawbacks of the Explicit Ordering Phase

Traditional blockchains all have an ordering phase, which could **limit the throughput**.

➢ The ordering leader must replicate blocks to all followers.



Replicate a block to followers



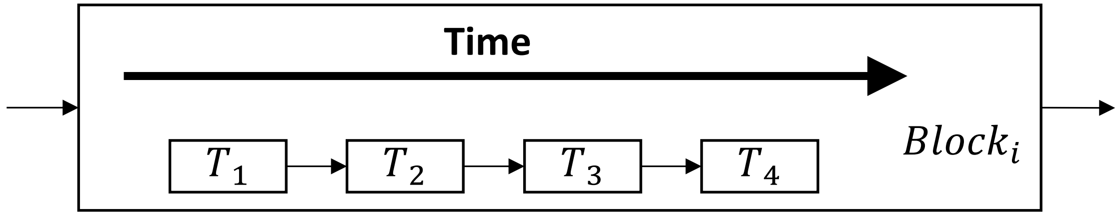
Bandwidth consumption of Fabric

➢ **The maximum bandwidth** of the leader could be a bottleneck.
➢ E.g. the outbound bandwidth of orderer #2 becomes a bottleneck.

Drawbacks of the Explicit Ordering Phase

Traditional blockchains all have an ordering phase, which could **limit the throughput**.

- **Execute (or validate) serially** based on the explicit order.



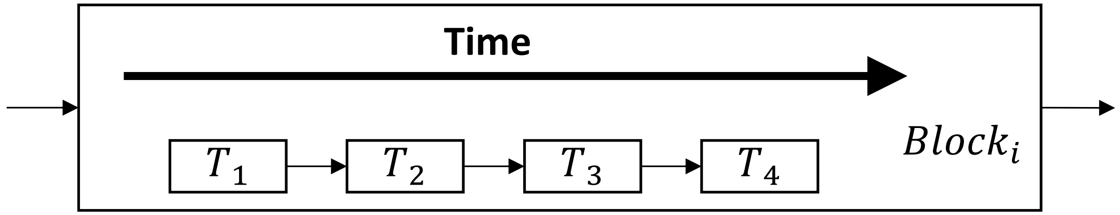
Validate serially to ensure determinism

- **Serial execution (or validation)** could be a bottleneck.

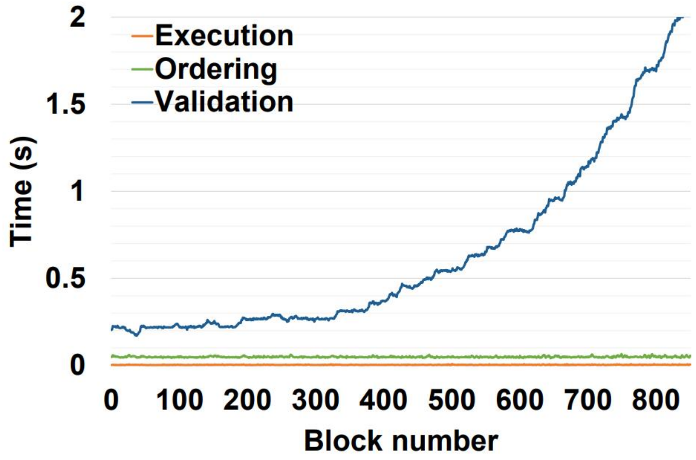
Drawbacks of the Explicit Ordering Phase

Traditional blockchains all have an ordering phase, which could **limit the throughput**.

- **Execute (or validate) serially** based on the explicit order.



Validate serially to ensure determinism



Runtime breakdown of Fabric

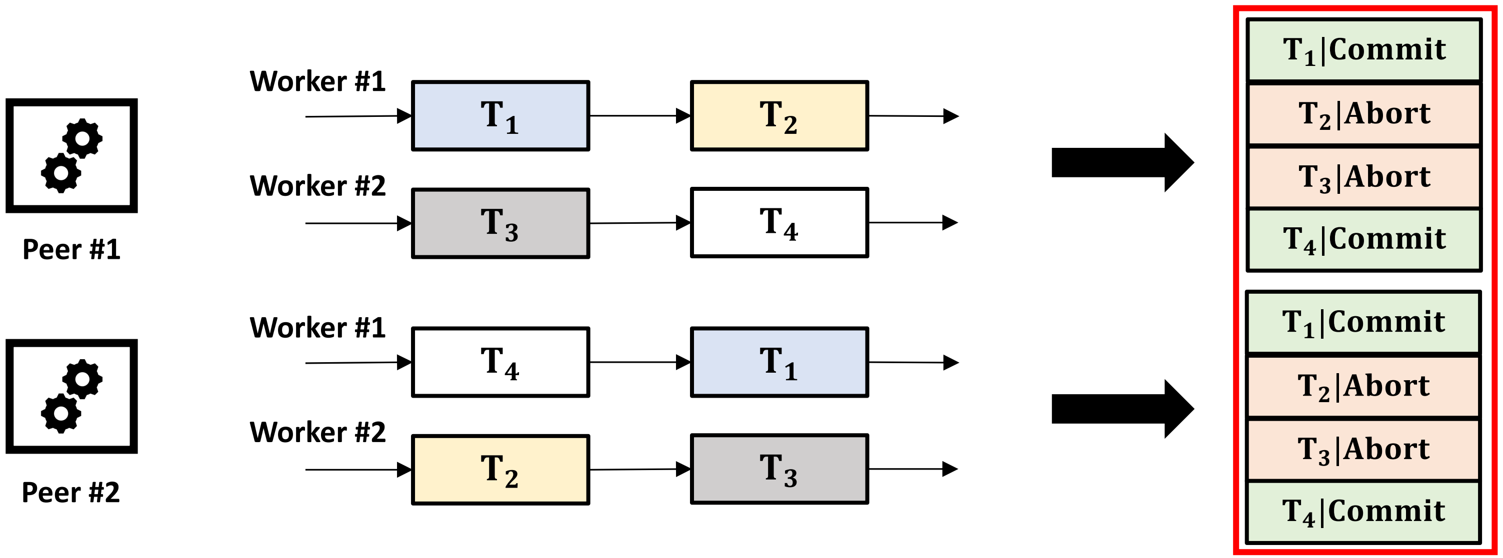
- **Serial execution (or validation)** could be a bottleneck.
- E.g. Validate the rw-set serially in a block **limit the parallelism**.

The deterministic execution technique

- Use the **deterministic execution technique** to eliminate the ordering phase.

The deterministic execution technique

- Use the **deterministic execution technique** to eliminate the ordering phase.
- **Concurrent execution** does not affect the result.



- The two peers execute the transactions in a different order.
- **The final execution result is the same.**

Eliminate the Explicit Ordering Phase

- Use the **deterministic execution technique** to eliminate the ordering phase.

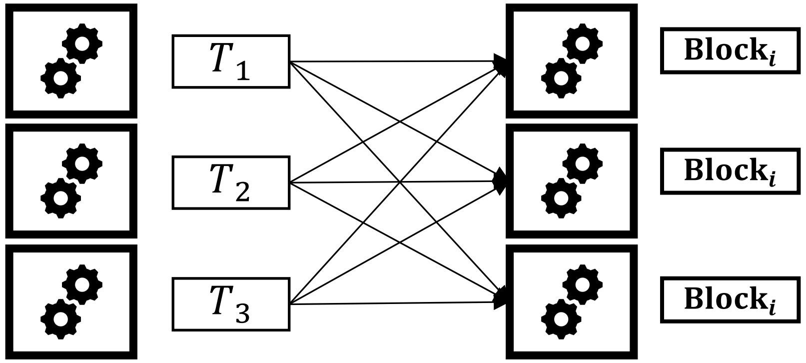
Replace the **explicit order** (specified by the ordering node) with an **implicit (rule-based) order**.

Eliminate the Explicit Ordering Phase

- Use the **deterministic execution technique** to eliminate the ordering phase.

Replace the **explicit order** (specified by the ordering node) with an **implicit (rule-based) order**.

- **Independently broadcast** their collected requests (with their own consensus instance).



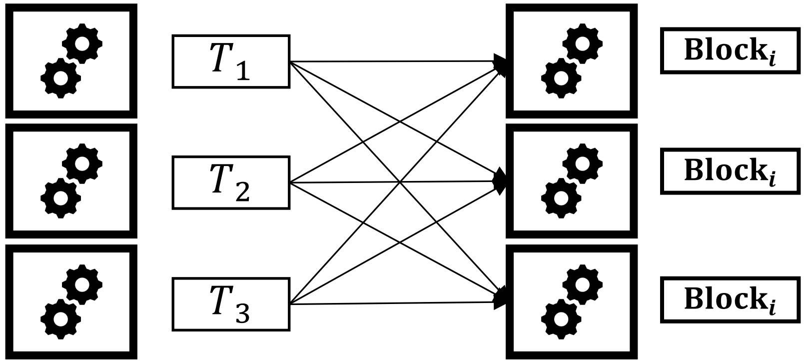
Eliminate the ordering service

Eliminate the Explicit Ordering Phase

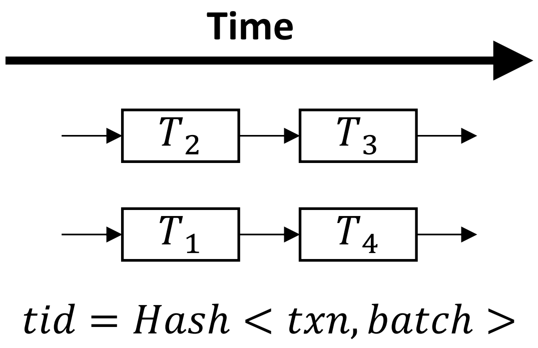
- Use the **deterministic execution technique** to eliminate the ordering phase.

Replace the **explicit order** (specified by the ordering node) with an **implicit (rule-based) order**.

- **Independently broadcast** their collected requests (with their own consensus instance).
- Allow **concurrent execution** of transactions.



Eliminate the ordering service



Deterministic execution based on *tids*

Eliminate the Explicit Ordering Phase

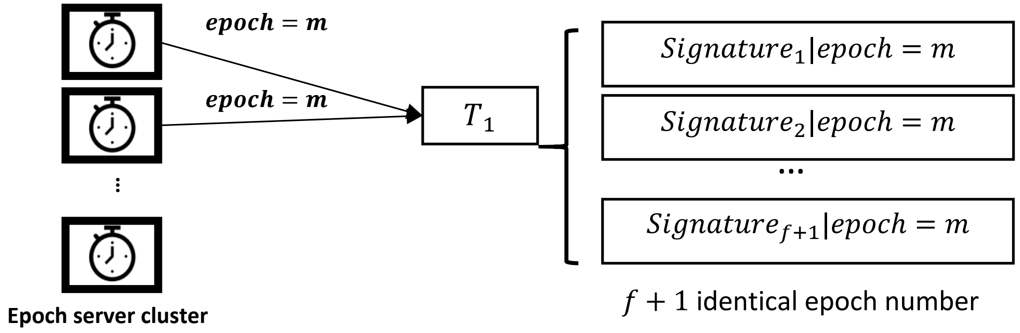
How to **determine the implicit (rule-based) ordering** of transactions **in an untrusted environment**?

- ❖ Inter-block order
- ❖ Intra-block order

Eliminate the Explicit Ordering Phase

How to **determine the implicit (rule-based) ordering** of transactions **in an untrusted environment**?

❖ Inter-block order: **global epoch number**.



➢ Transactions with the same epoch form into a block.

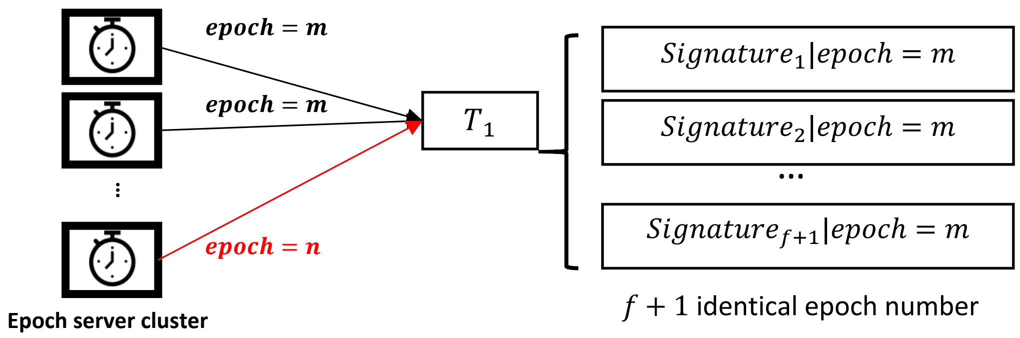
➢ **Increase the epoch through consensus**.

❖ Intra-block order

Eliminate the Explicit Ordering Phase

How to **determine the implicit (rule-based) ordering** of transactions **in an untrusted environment**?

❖ Inter-block order: **global epoch number**.



➢ Transactions with the same epoch form into a block.

➢ **Increase the epoch through consensus.**

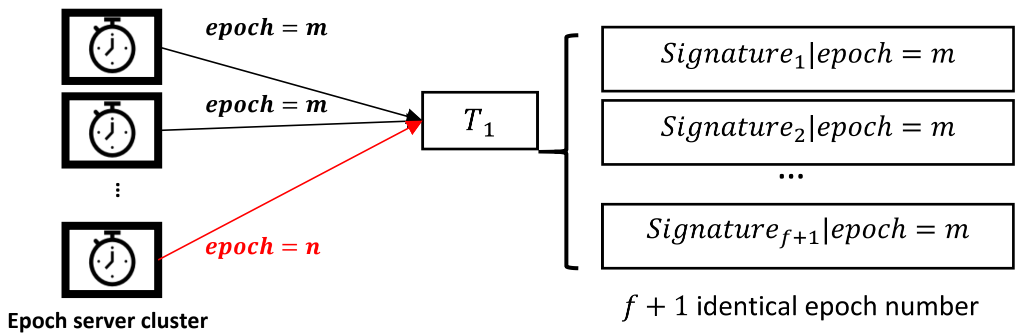
➢ Collect ***f + 1*** valid replies for each transaction batch.

❖ Intra-block order

Eliminate the Explicit Ordering Phase

How to **determine the implicit (rule-based) ordering** of transactions **in an untrusted environment**?

❖ Inter-block order: **global epoch number**.



- Transactions with the same epoch form into a block.
- **Increase the epoch through consensus.**
- Collect ***f + 1*** valid replies for each transaction batch.

❖ Intra-block order: **unique transaction ID**.

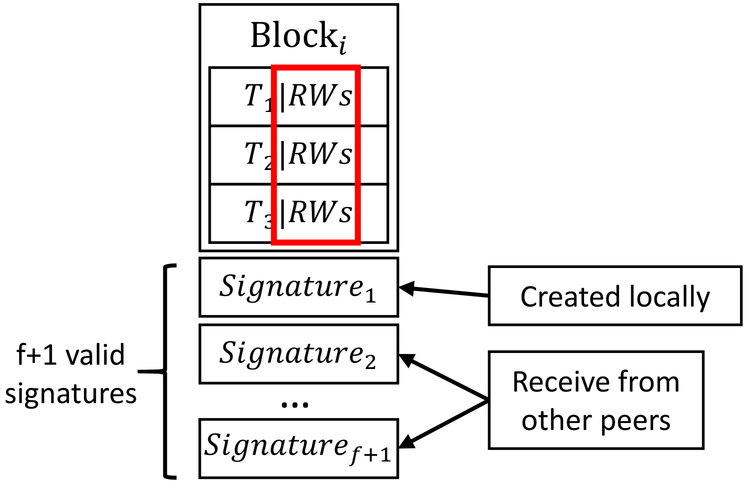
$$tid = Hash < txn,\ txn\ batch >$$

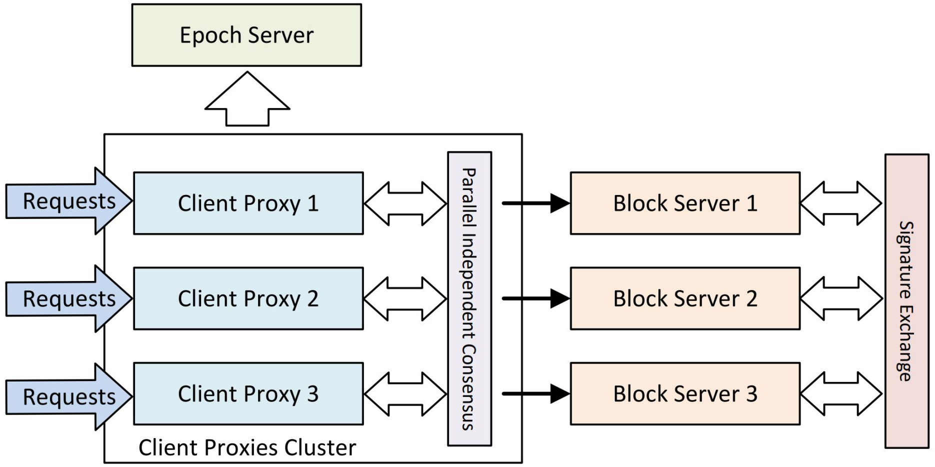
- Transactions are scheduled based on *tids*.
- The *tids* are generated using a hash function.
- **Immutable and unpredictable**.

Eliminate the Explicit Ordering Phase

How to **validate the execution result** in a block?

❖ Collect f+1 valid block signatures.

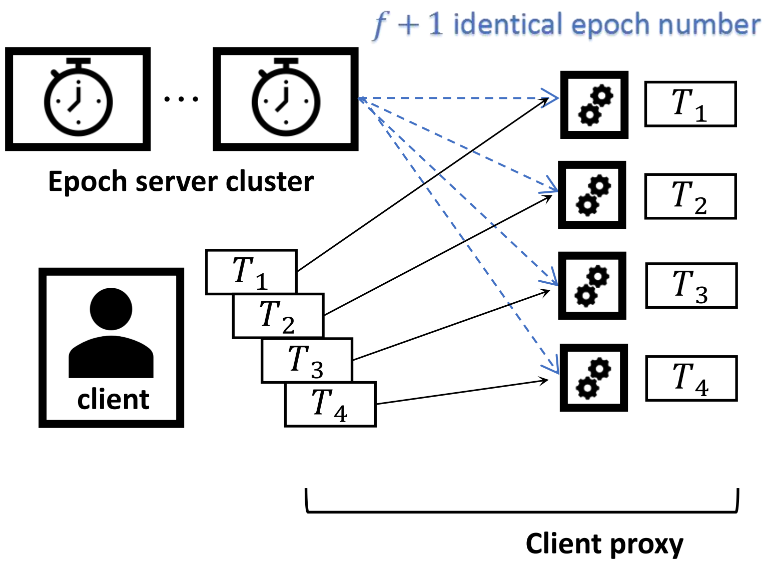




➢ The signature can only be verified if the execution results are the same.

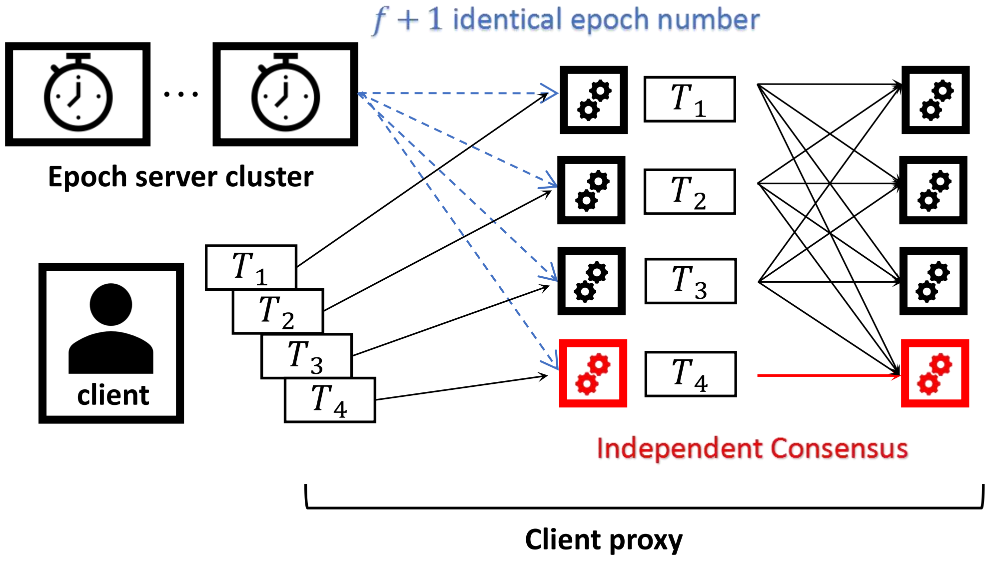
➢ Following this idea, we propose the **execute-validate** (EV) blockchain architecture.

Ordering-Free Execute-Validate (EV) Execution Flow



- Workflow of the EV architecture:
  - Groups transactions into batches.
  - Gets **f+1 valid epoch numbers**.

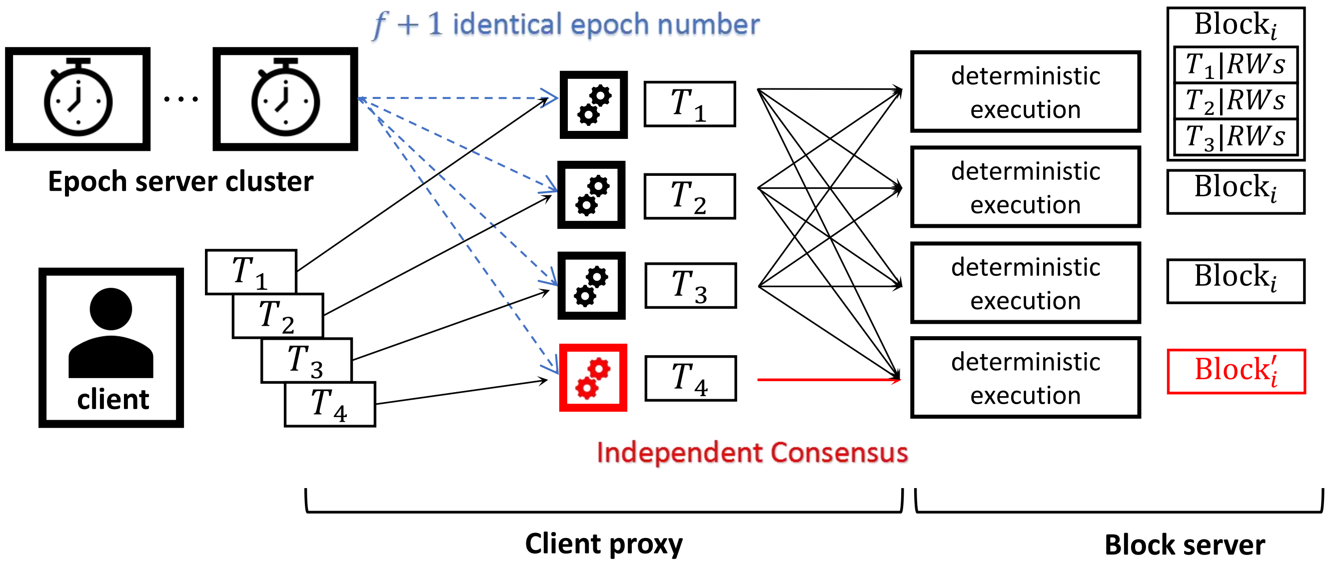
Ordering-Free Execute-Validate (EV) Execution Flow



❖ Workflow of the EV architecture:

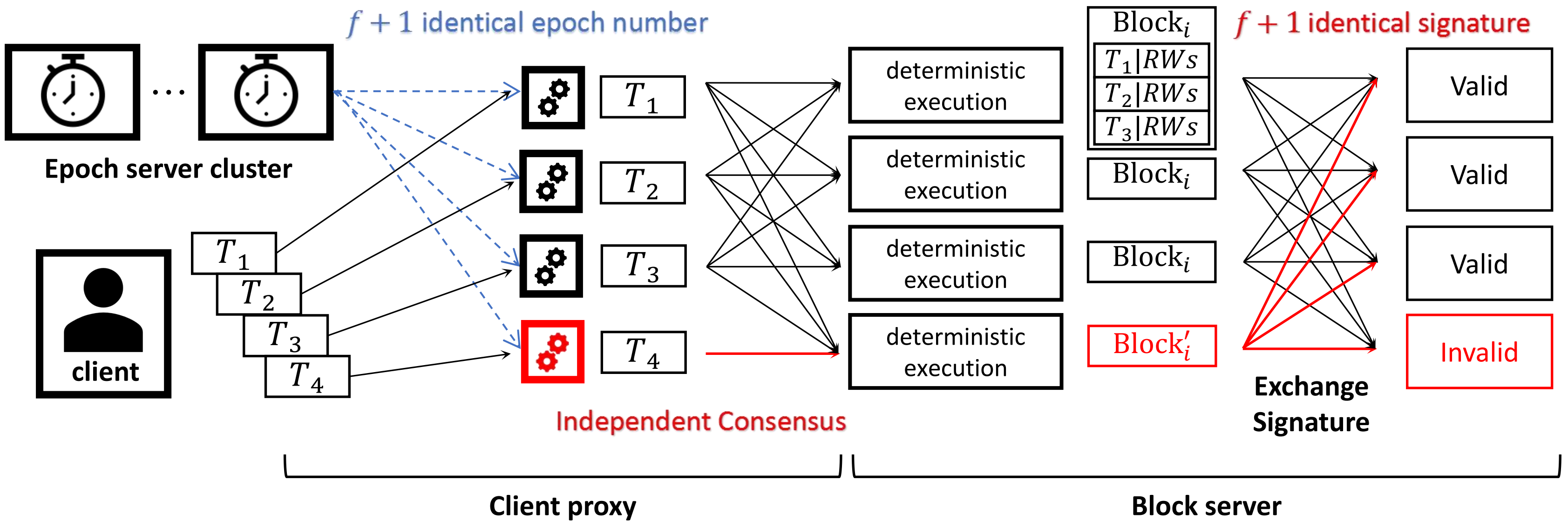
- Groups transactions into batches.
- Gets **f+1 valid epoch numbers**.
- Broadcasts to other client proxies.

Ordering-Free Execute-Validate (EV) Execution Flow



- Workflow of the EV architecture:
  - Groups transactions into batches.
  - Gets **f+1 valid epoch numbers**.
  - Broadcasts to other client proxies.
  - **Executes** all transactions **deterministically.**

Ordering-Free Execute-Validate (EV) Execution Flow



❖ Workflow of the EV architecture:

- Groups transactions into batches.
- Gets **f+1 valid epoch numbers**.
- Broadcasts to other client proxies.
- **Executes** all transactions **deterministically.**
- Exchange block signatures.

Experimental Setups

Baseline: **Fabric**, **FastFabric**, **Meepo**, **ResilientDB**, **Basil**, and NeuChain variants (**OEV**, **EOV**, and **OEPV**).

Platforms: (16 vCPUs, 32GB RAM, 100Mbps cross-region / 5Gbps local bandwidth)

- geo-distributed cluster: 4 regions (each region: 1 epoch server, 1 peer)
- local cluster: 4 epoch servers, 8 peers

Workload:

YCSB: Zipfian skewness factor 0.99; 10 columns and 1,000,000 rows; 100 bytes per column.

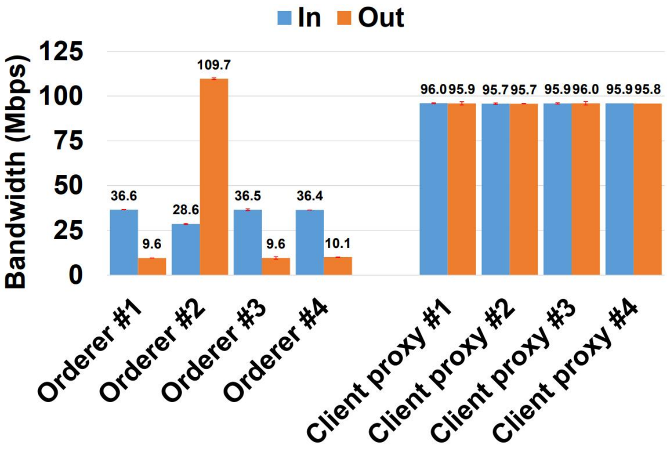
YCSB-A (50% read and 50% write), **YCSB-B** (95% read and 5% write), and **YCSB-C** (100% read)

Smallbank: uniform distribution; 100,000 accounts.

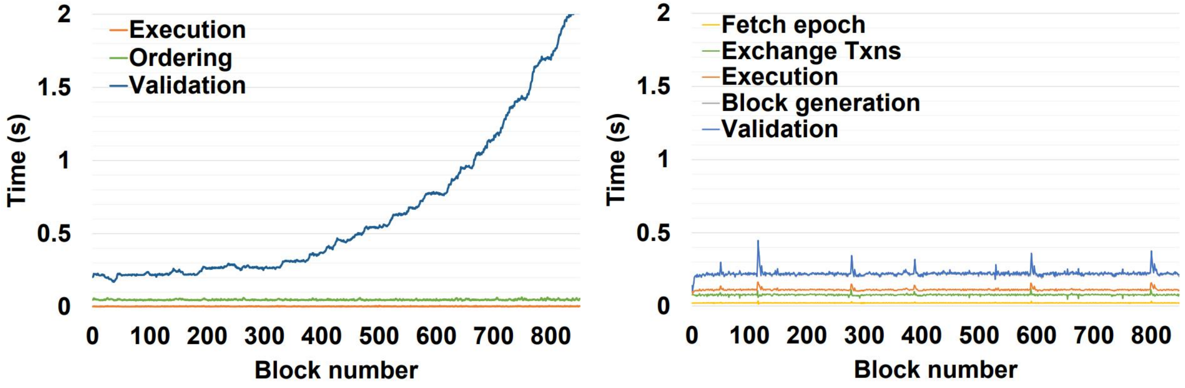
Questions:

- The effectiveness of the EV architecture.
- The effectiveness of optimizations.
- The robustness of NeuChain under malicious attacks

Effectiveness of the EV Architecture



Bandwidth consumption on geo-distributed cluster



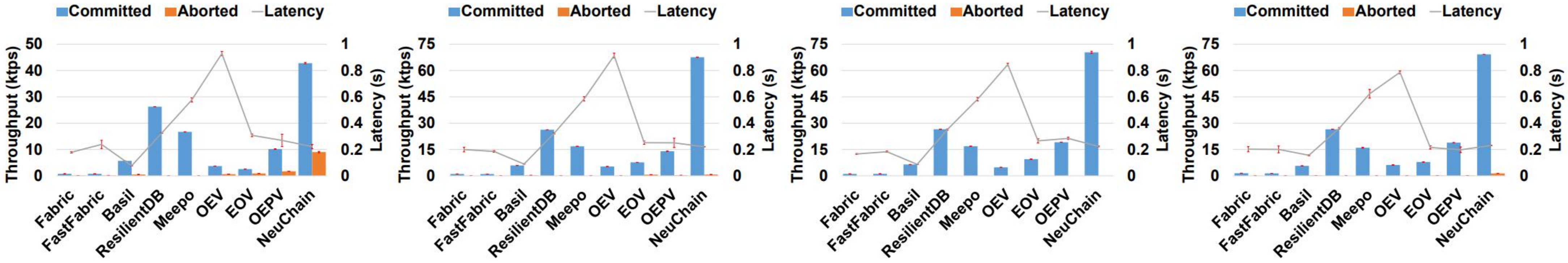
(a) Fabric (b) NeuChain

Breakdown on local cluster

On geo-distributed cluster, where network is the bottleneck, the EV architecture **utilizes the bandwidth** of all nodes.

On local cluster, where transaction processing is the bottleneck, the deterministic execution **increases the concurrency.**

Overall Performance on Geo-distributed Cluster



(a) YCSB-A (b) YCSB-B (c) YCSB-C (d) SmallBank

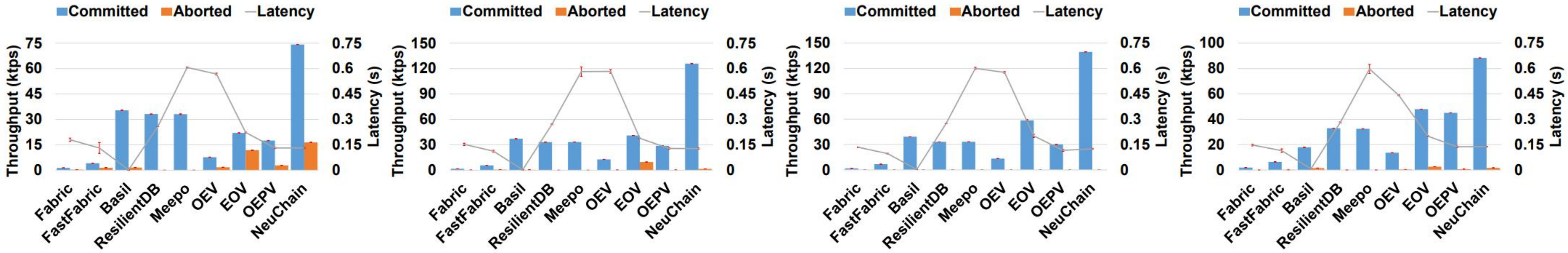
Figure 6: Performance comparison of different blockchains (on geo-distributed cluster).

On geo-distributed cluster, NeuChain exhibits the highest throughput under all workloads.

- No need to re-execute transactions (compared with OEV architecture).
- Blocks only contain user transactions (compared with EOV architecture).
- All peers can propose transactions (compared with OEPV architecture).

However, cross-range epoch number acquisition requires an additional RTT (20ms).

Overall Performance on Local Cluster



(a) YCSB-A throughput (b) YCSB-B throughput (c) YCSB-C throughput (d) SmallBank throughput

Figure 7: Performance comparison of different blockchains (on local cluster).

On local cluster, the performance of NeuChain mainly benefits from concurrent execution.

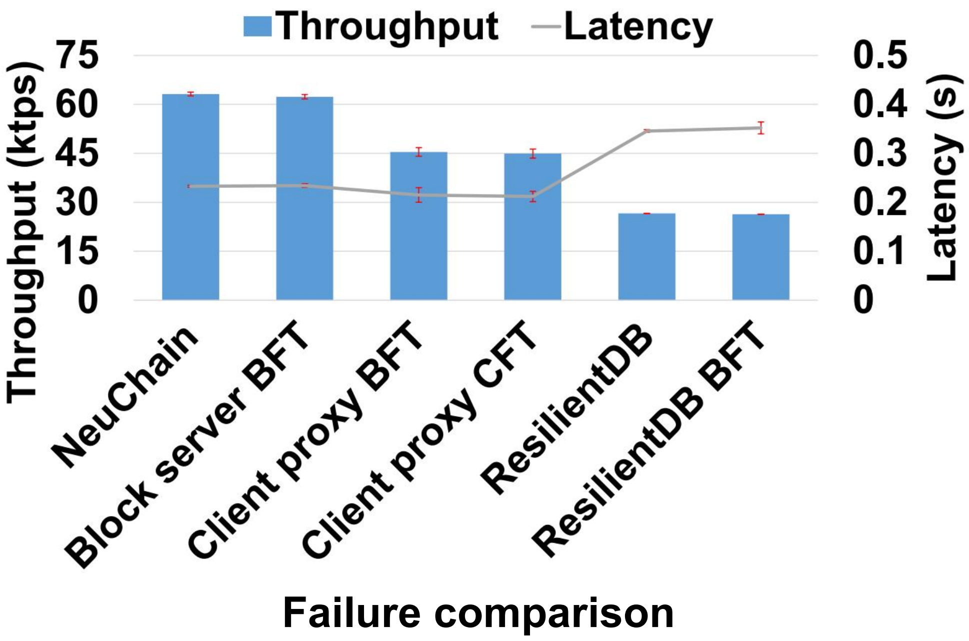
However, NeuChain only allows writing one value once in each block, which cause a higher abort rate.

Robustness under Malicious Attacks

We provide three kinds of failures:

- **Block server BFT:** A block server provides others with a fake block signature.
- **Client proxy BFT:** A client proxy sends fake messages to others.
- **Client proxy CFT:** We kill a client proxy to simulate crash failure.

➢ NeuChain is robust against these failures.
➢ The malicious client proxy is forbidden to submit transactions, reducing overall throughput.



Summary

- **Providing insight into the existing blockchain architectures**

Summary

- **Providing insight into the existing blockchain architectures**

- **Proposing an ordering-free EV architecture**

Summary

- **Providing insight into the existing blockchain architectures**
- **Proposing an ordering-free EV architecture**
- **Delivering a fast permissioned blockchain system**

Summary

- **Providing insight into the existing blockchain architectures**
- **Proposing an ordering-free EV architecture**
- **Delivering a fast permissioned blockchain system**
- **Providing robustness under malicious attacks**

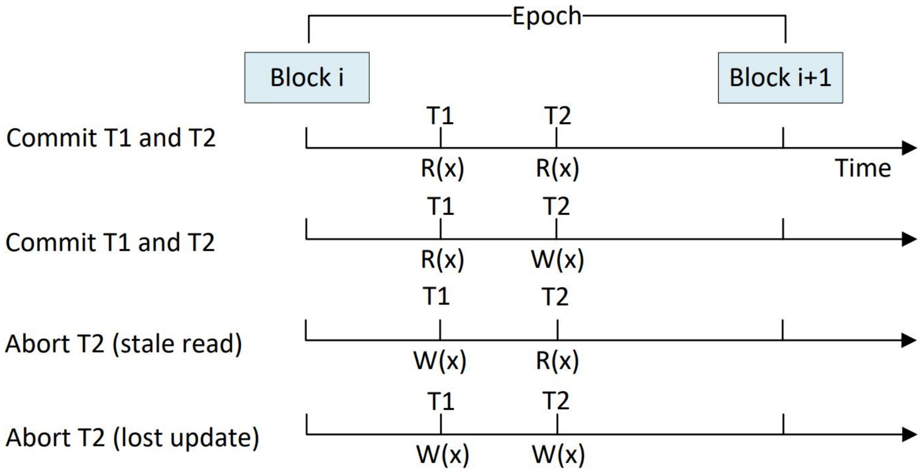
Summary

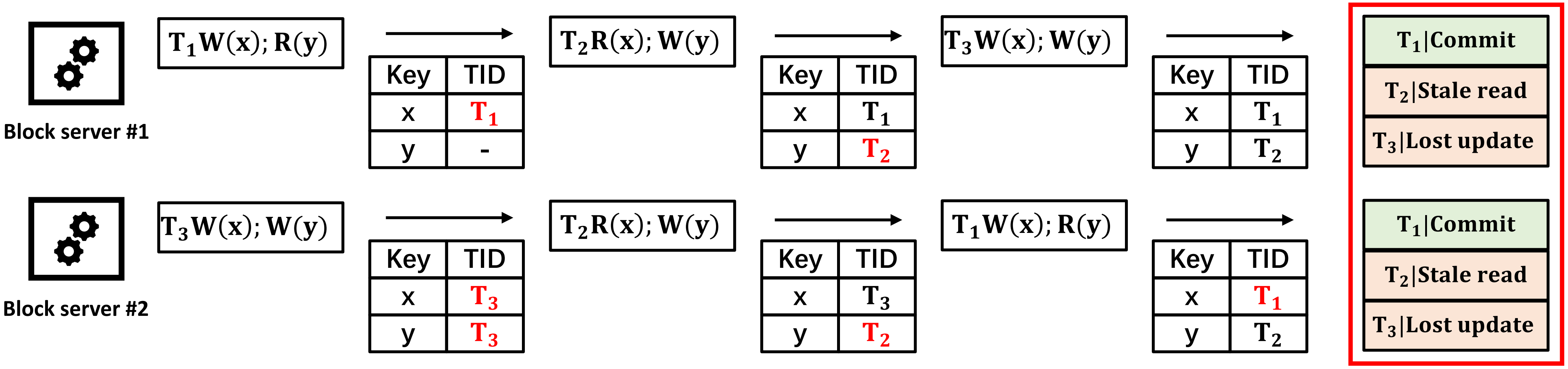
- **Providing insight into the existing blockchain architectures**
- **Proposing an ordering-free EV architecture**
- **Delivering a fast permissioned blockchain system**
- **Providing robustness under malicious attacks**

Deterministic Transaction Processing

- A **reserve table**[2] records write operation.
- 4 kinds of transaction read-write orders.
- Transaction abort if **stale read** or **lost update.**

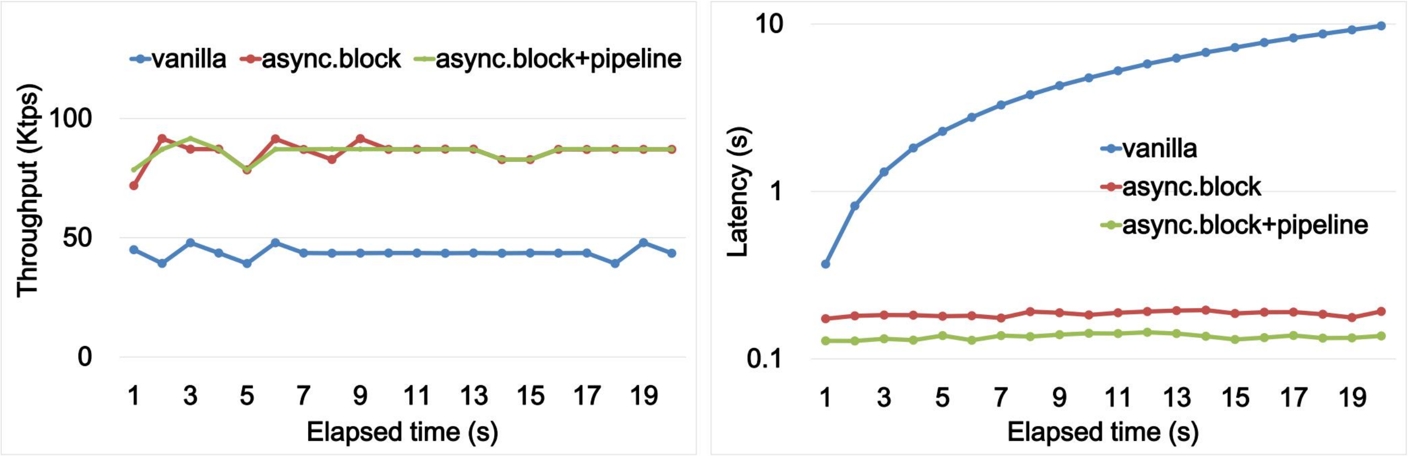
Based on the deterministic reservation result, **the execution is deterministic**.





[2] Yi Lu, et, al., Aria: a fast and practical deterministic OLTP database. VLDB'20

Effectiveness of Optimizations



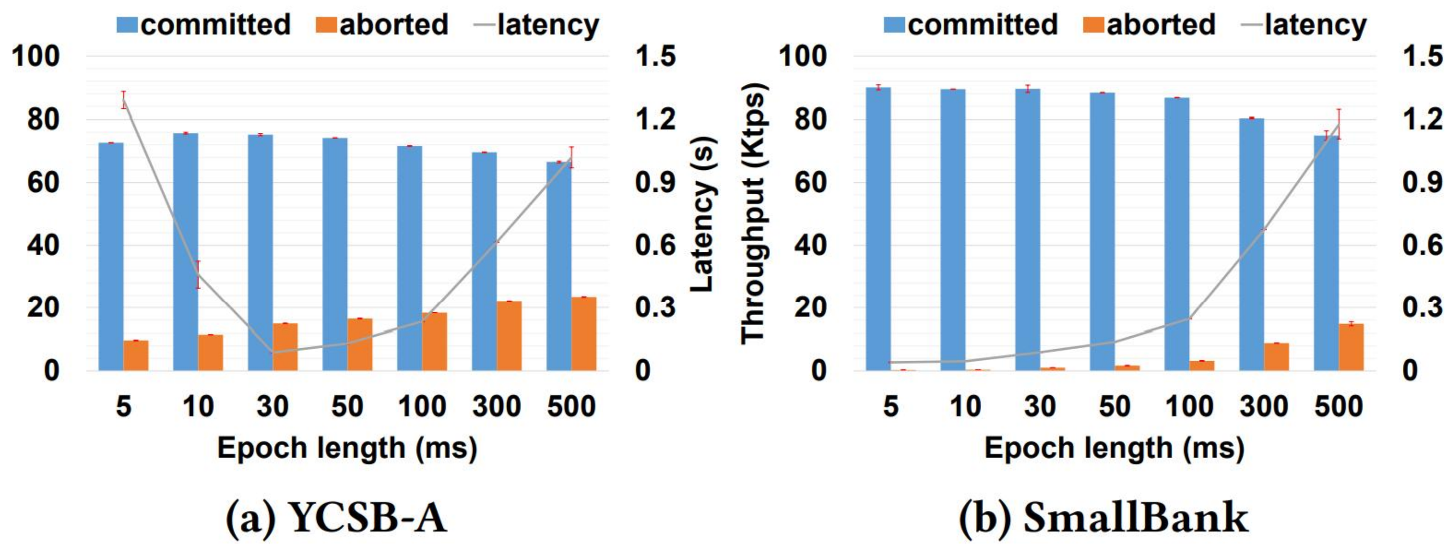
(a) Throughput (b) Latency (log-scaled)

Effect of optimizations

(a) The asynchronous block generation has greatly improved the performance.

(b) The pipelining technique further reduced the latency.

Performance when varying epoch length



Effect of optimizations

(a) The frequent data exchanges (due to short epoch) are expensive.

(b) The cost of Merkle tree generation is exponentially increased with the increase of block size (due to long epoch).