Rolis: A software approach to efficiently replicating multi-core transactions

Weihai Shen
Stony Brook University

Ansh Khanna
Stony Brook University

Sebastian Angel
University of Pennsylvania and
Microsoft Research

Siddhartha Sen
Microsoft Research

Shuai Mu
Stony Brook University

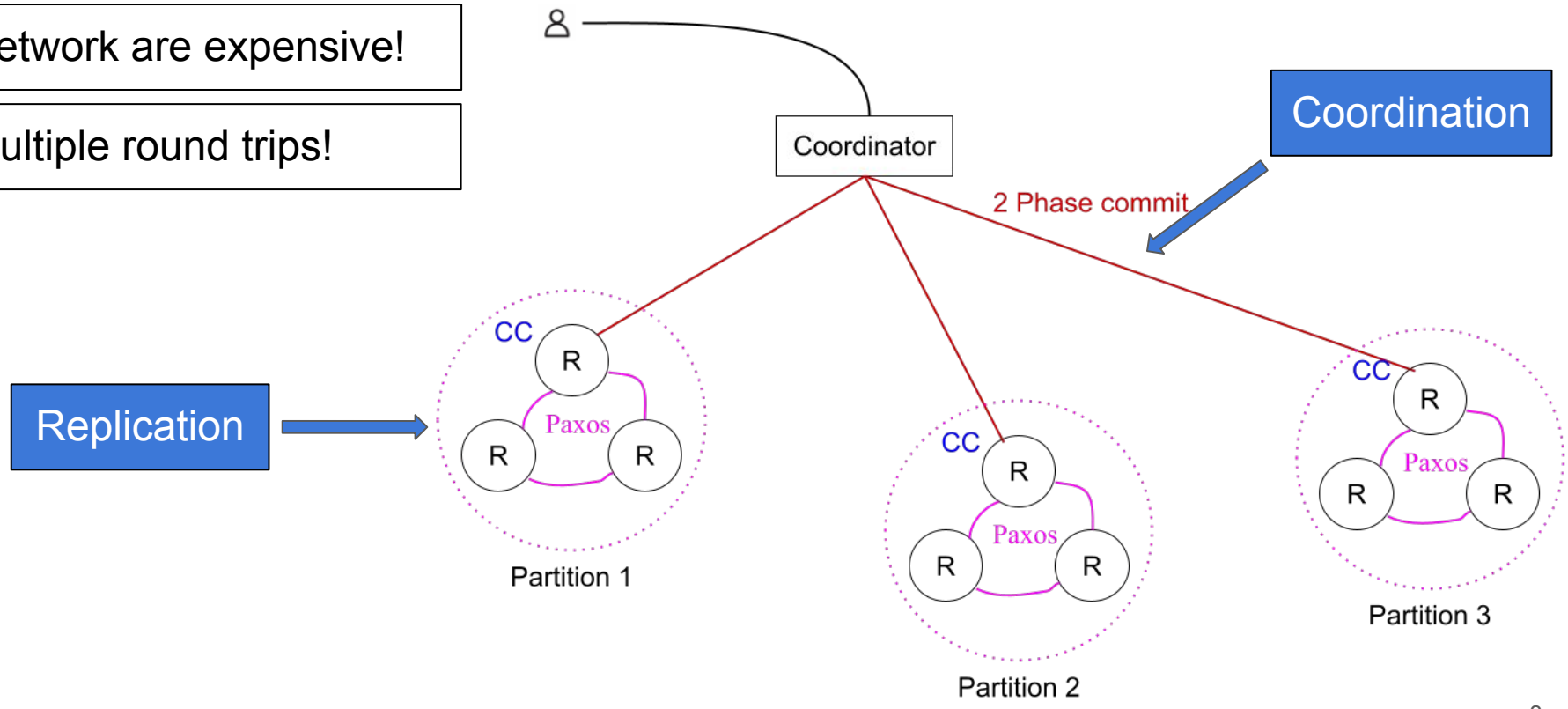
Trade-off between performance and fault-tolerance

	Single-node multi-core system *(e.g., **Silo, Cicada**)*	Distributed system *(e.g., **Spanner**)*
Performance	**Fast** *(over 1M)*	**Slow** *(several or tens of thousand)*
Fault tolerance	**Limited** *(can't tolerate node failures)*	**Strong** *(tolerate a minority of node failures)*

Why distributed systems are so slow

Network are expensive!

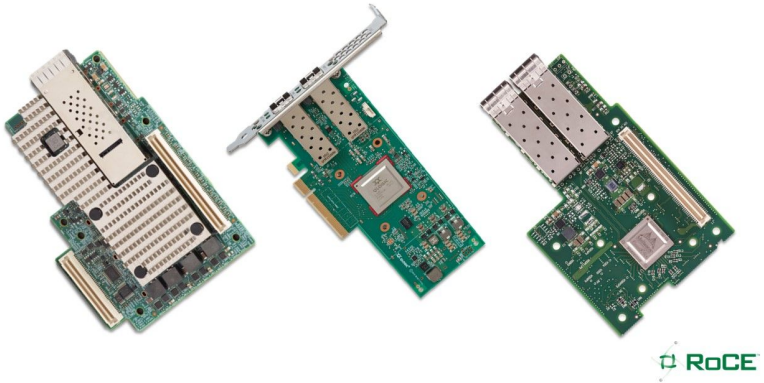
Multiple round trips!



Solutions to fix the high-cost network problem

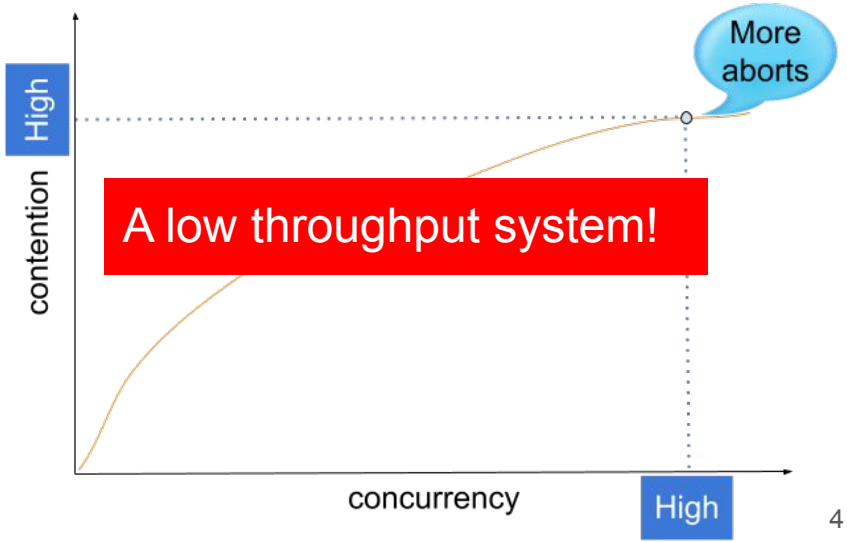
A recent popular solution

To use advanced kernel-bypass network hardware, e.g., RDMA or DPDK



A possible software-based solution

Adding more current transactions to mask the high-cost network?

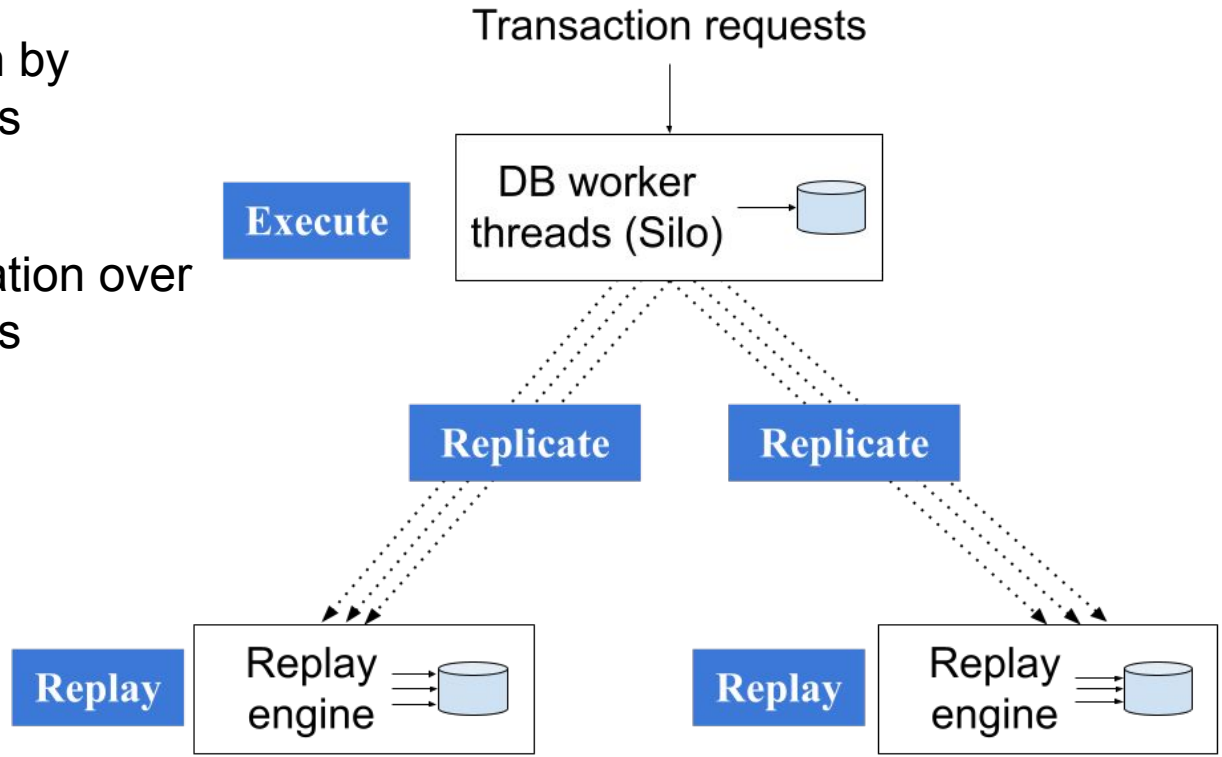


Rolis: a software-based solution

	Single-node multi-core system *(e.g., **Silo, Cicada**)*	Distributed system *(e.g., **Spanner**)*	**Rolis**
Performance	**Fast** *(over 1M on OLTP)*	**Slow** *(several thousand)*	**Fast**
Fault tolerance	**Limited** *(can't tolerate node failures)*	**Strong** *(tolerate a minority of node failures)*	**Strong**

Rolis: “execute-replicate-replay” model

1. **speculative** execution by multiple worker threads
2. **asynchronous** replication over multiple Paxos streams
3. replay **efficiently**



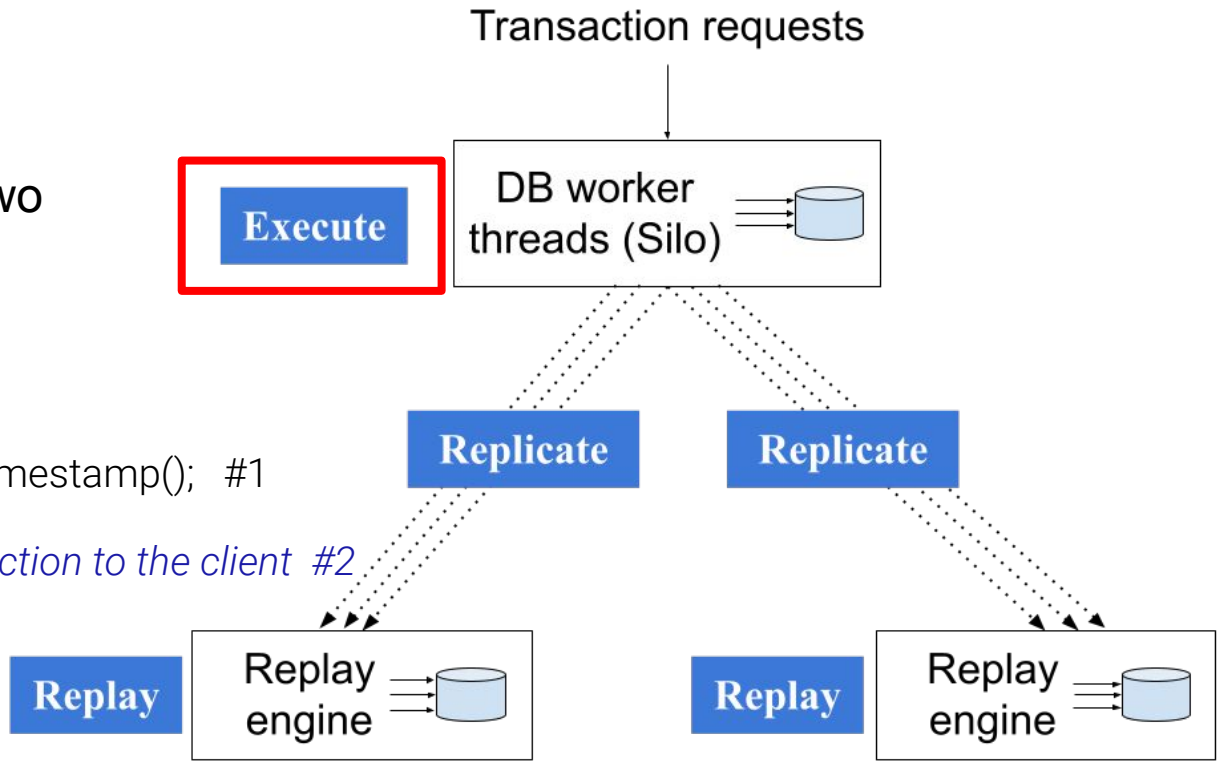
Speculative execution

- Atop of Silo
- A building block with two small modifications

```
void *txn = db->new_txn() ;
db->commit_txn(txn) ;

int64 timestamp = global_timestamp();   #1

// defer releasing this transaction to the client  #2
```

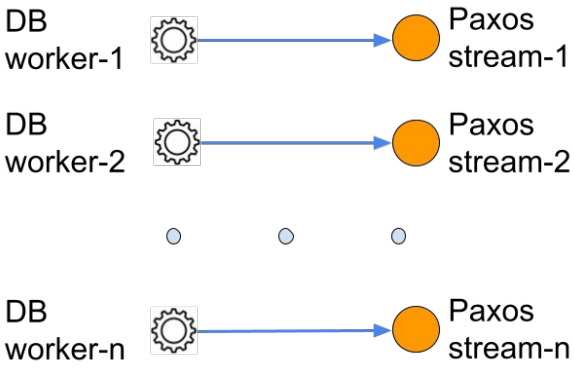


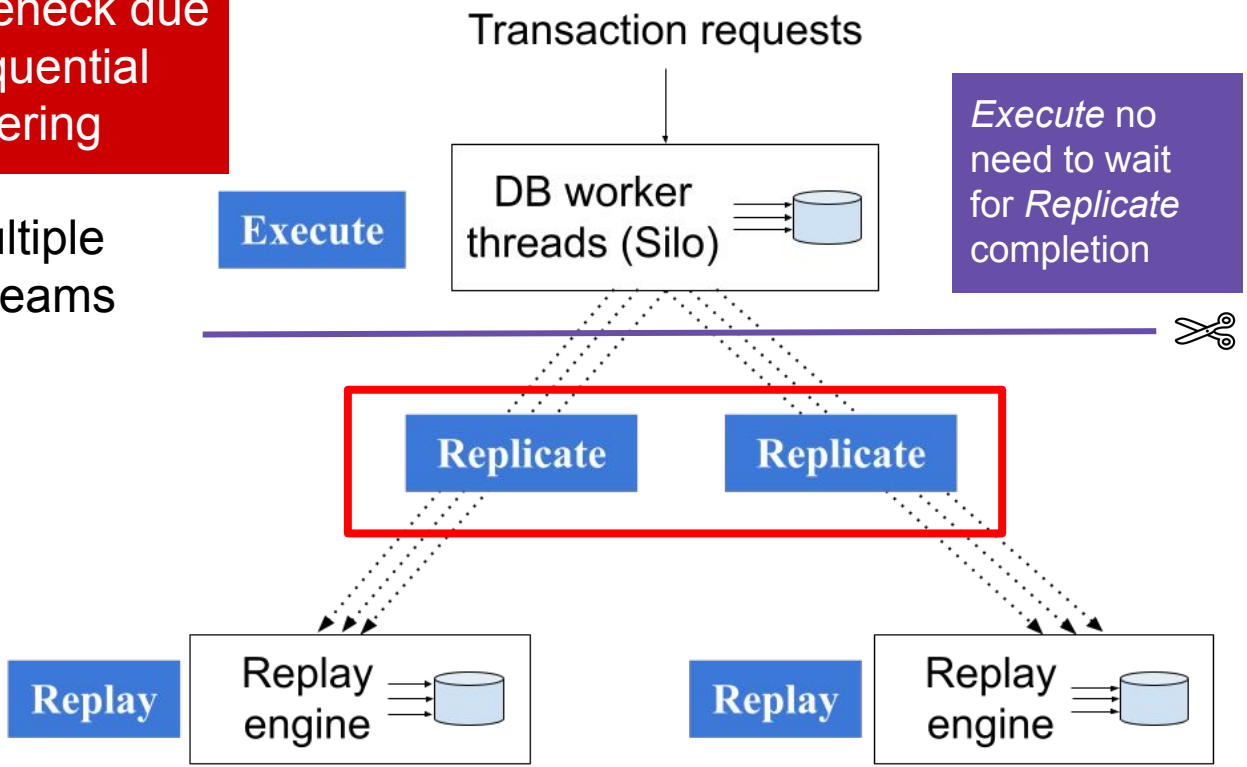
Asynchronous replication

- Paxos!

the bottleneck due to sequential ordering

- Instead, Rolis uses multiple independent Paxos streams





Replay

- 3 friends: Alice, Bob, Charlie
- tx1: Alice transfers 100$ to Bob (*timestamp: 10*)

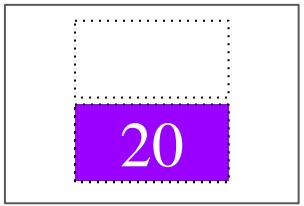
Paxos stream-1

- tx2: then Bob transfers this 100$ to Charlie (*timestamp: 20*)

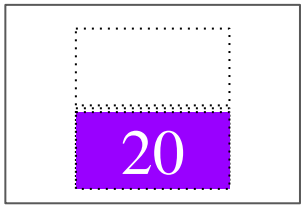
Paxos stream-2

tx1 → tx2

Leader



R1



R2

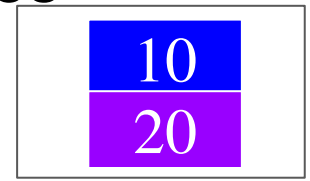
Outcome: tx1 is lost, tx2 is durable

Watermark: tracking dependencies

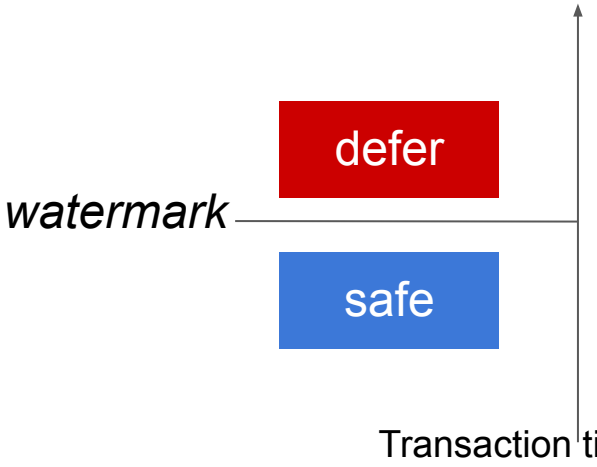
Most recent durable timestamp:

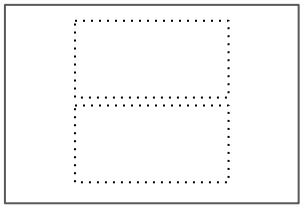
- Paxos stream-1: 0
- Paxos stream-2: 0

watermark = 0 (*smallest one*)

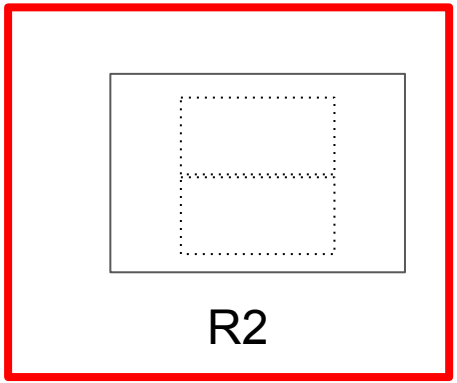


Leader





R1



R2

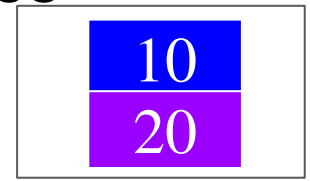
Watermark: tracking dependencies

Most recent durable timestamp:

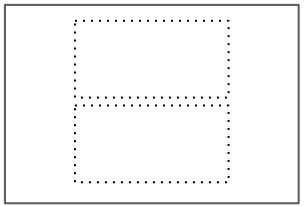
- Paxos stream-1: 0
- Paxos stream-2: 0

---

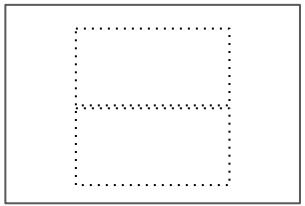
watermark = 0 (*smallest one*)



Leader



R1



R2

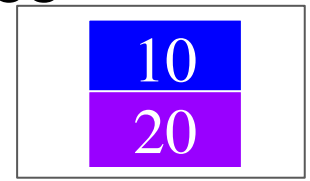
Watermark: tracking dependencies

Most recent durable timestamp:

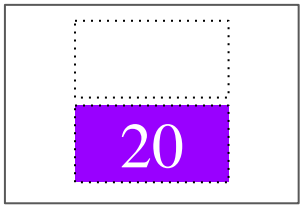
- Paxos stream-1: 0
- Paxos stream-2: 20

---

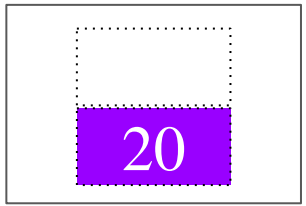
watermark = 0 (*smallest one*)



Leader



R1



R2

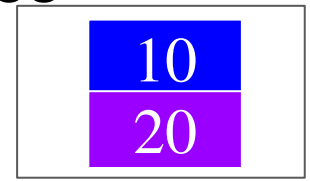
Watermark: tracking dependencies

Most recent durable timestamp:

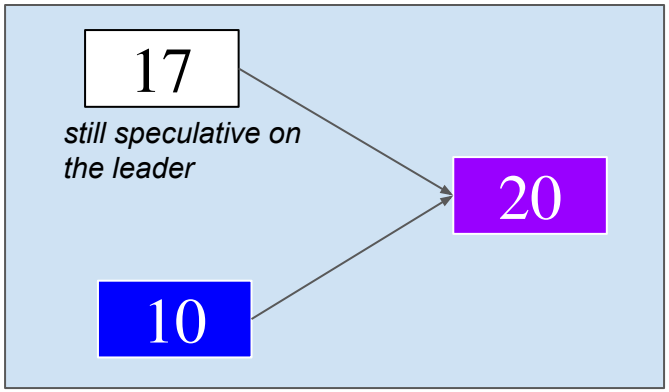
- Paxos stream-1: 10
- Paxos stream-2: 20

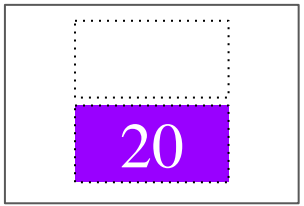
---

watermark = 10 (*smallest one*)

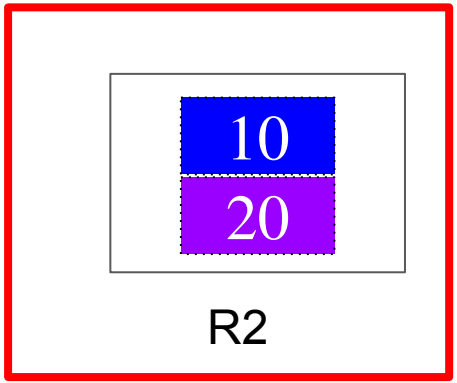


Leader





R1



R2

Evaluation

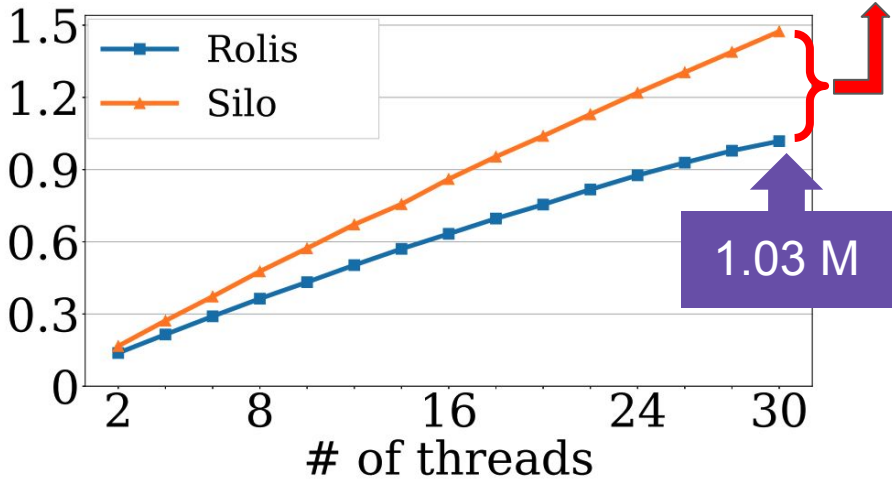
Benchmarks:

- TPC-C → **complex**
- YCSB++ → **simpler**

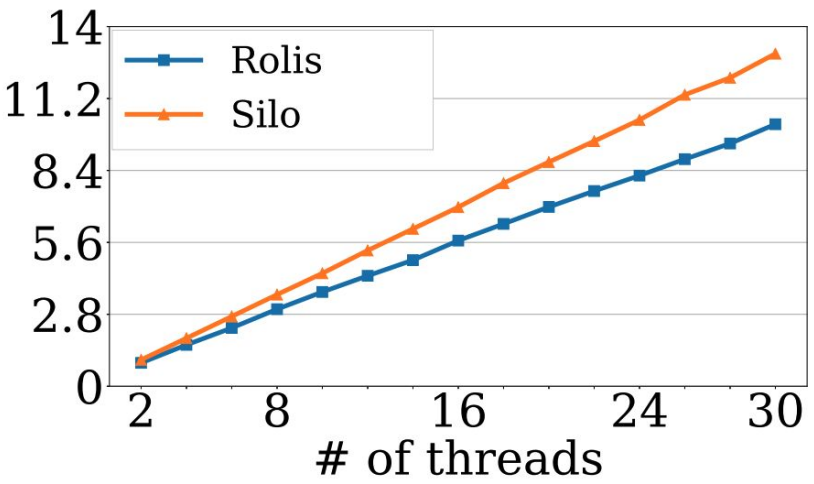
Comparisons		
Baseline	Software-based	Hardware-based
Silo *(SOSP '13)*	• 2PL *(from Janus OSDI '16)* • Calvin *(SIGMOD '12)*	Meerkat *(Eurosys '20)*

Evaluation: scalability of

31 % drop due to serialization and replication



(a) TPC-C



(b) YCSB++

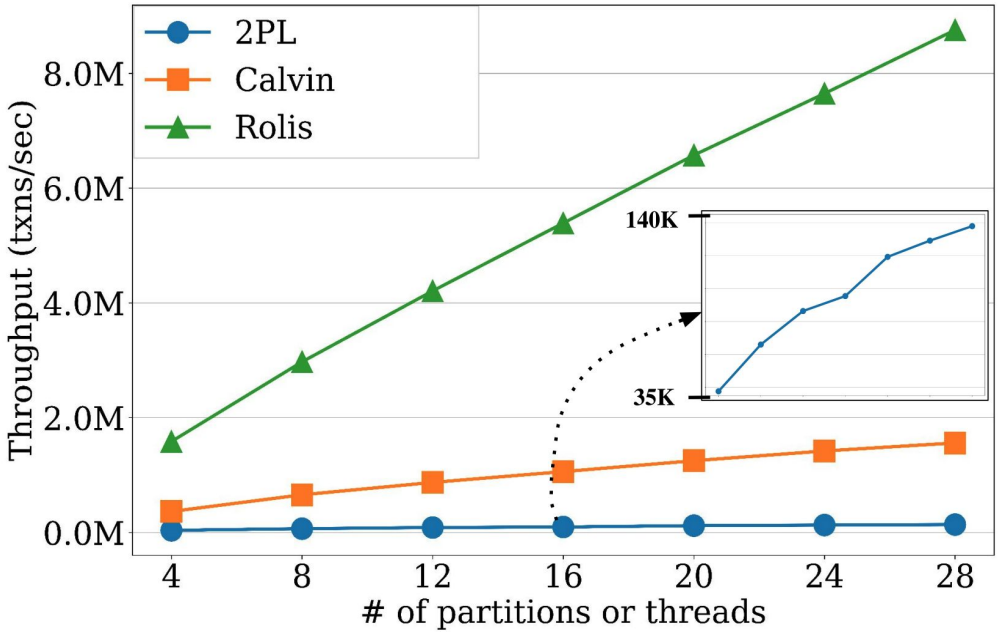
Evaluation: software-based comparisons

- 2 phase-locking

  intensive coordination among replicas and holds all locks before transaction execution

- Calvin: a deterministic database

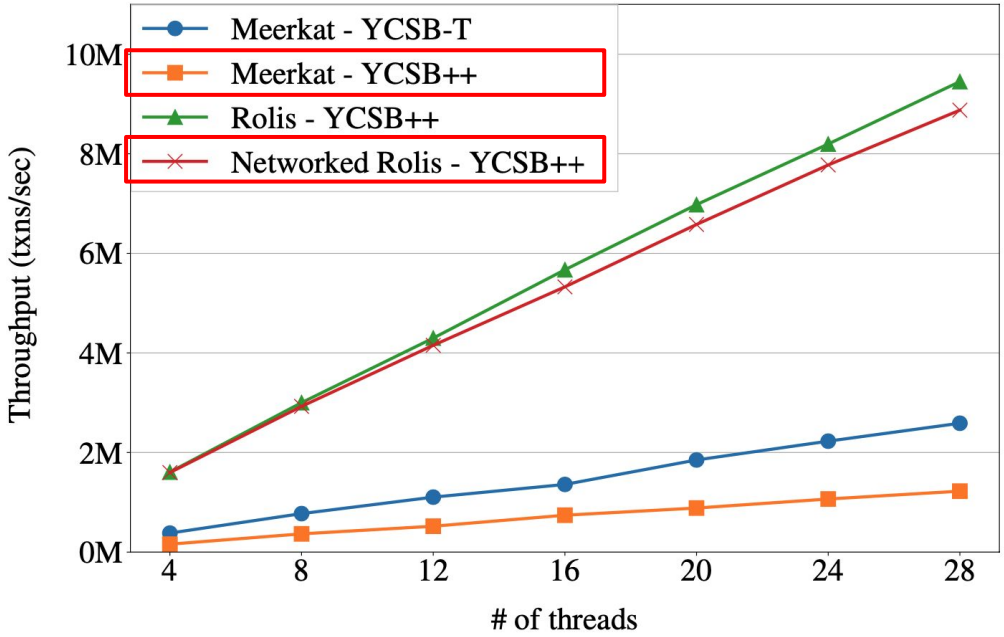
  a central sequencer to determine the order



Evaluation: hardware-based comparison

- A networked Rolis
- Still not an exact apple-to-apple comparison

Rolis client	Meerkat client
Stored procedure transactions	Interactive transactions



Summary

- Introduce Rolis's "execute-replicate-replay" model
- How Rolis tracks dependencies and replays transactions
- Show evaluations: 1.03M throughput on TPC-C

Questions?

Q & A or email at
weihshen@cs.stonybrook.edu

https://github.com/stonysystems/rolis