STAR: Scaling Transactions through Asymmetric Replication

Yi Lu, Xiangyao Yu and Sam Madden

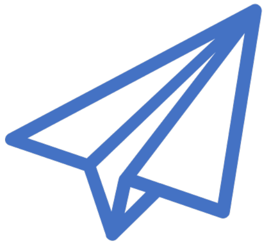
Slides: https://tiny.cc/star_slides

Background

Transactions make programing easier and are used everywhere

Financial services

Online shopping

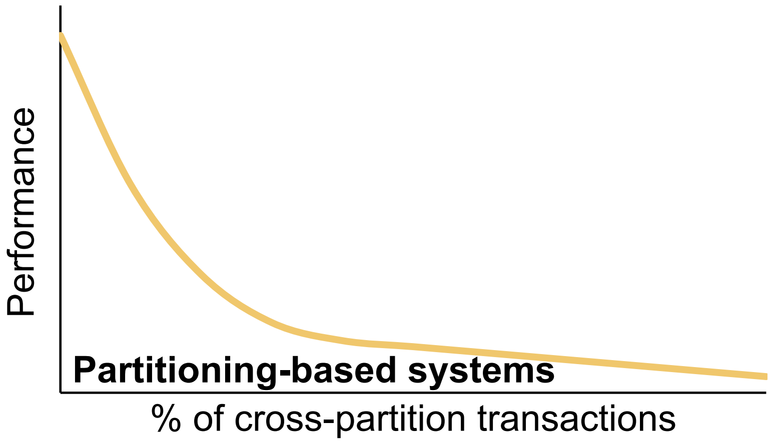


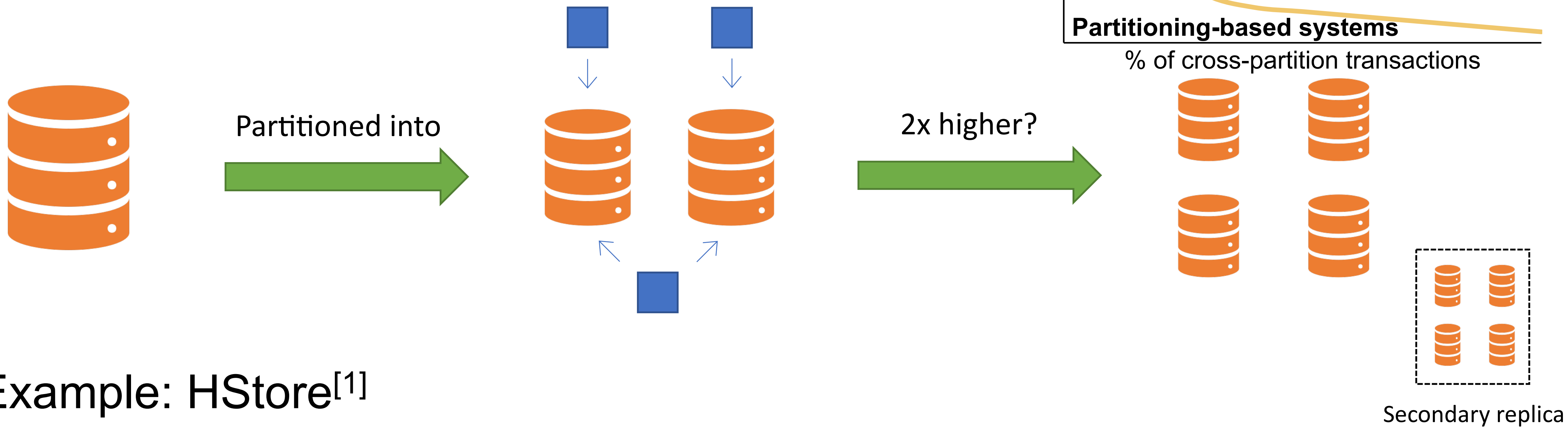
Ticket booking

High availability is crucial in modern OLTP applications

» Replication

Partitioning-based systems





Example: HStore[1]

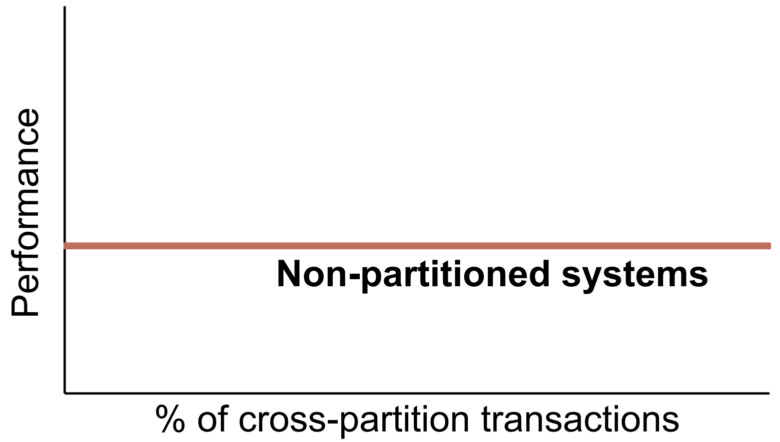
✔ Good fit for workloads with **single-partition** transactions

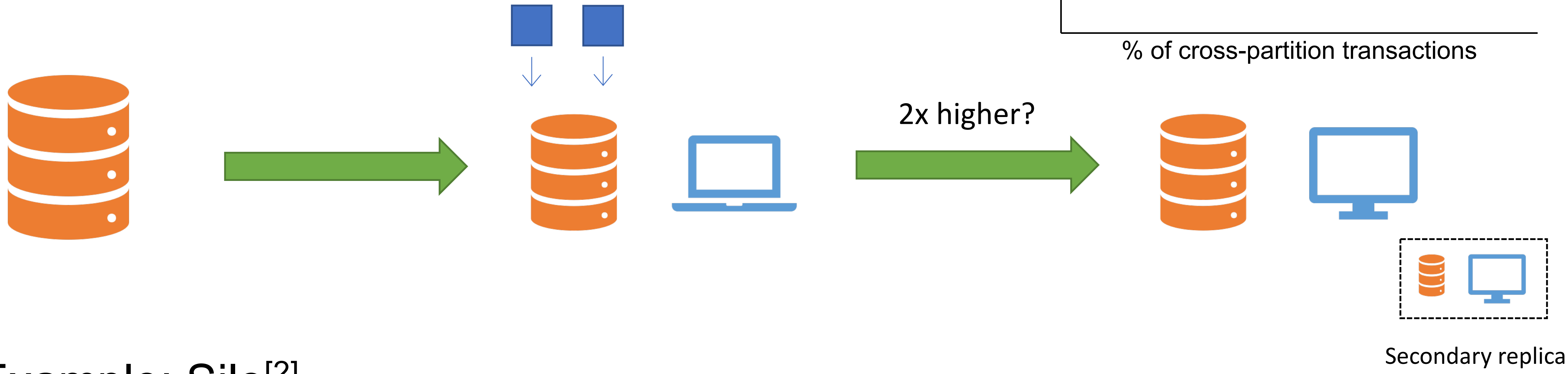
✖ Network communication and 2PC in cross-partition transactions

[1] Michael Stonebraker, Samuel Madden, Daniel J. Abadi, Stavros Harizopoulos, Nabil Hachem, Pat Helland

The End of an Architectural Era (It's Time for a Complete Rewrite). VLDB 2007: 1150-1160

Non-partitioned systems





Example: Silo[2]

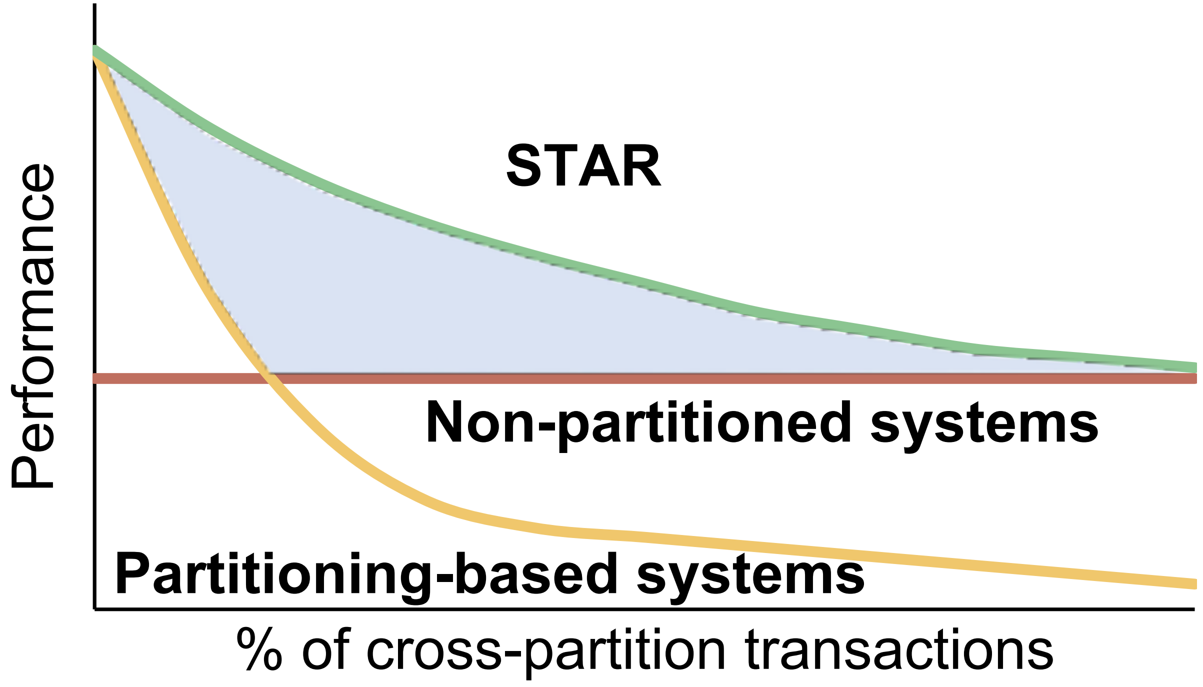
✔ Good fit for workloads with **cross-partition** transactions

✖ Cannot employ multiple nodes for parallel transaction execution

[2] Stephen Tu, Wenting Zheng, Eddie Kohler, Barbara Liskov, Samuel Madden
Speedy transactions in multicore in-memory databases. SOSP 2013: 18-32

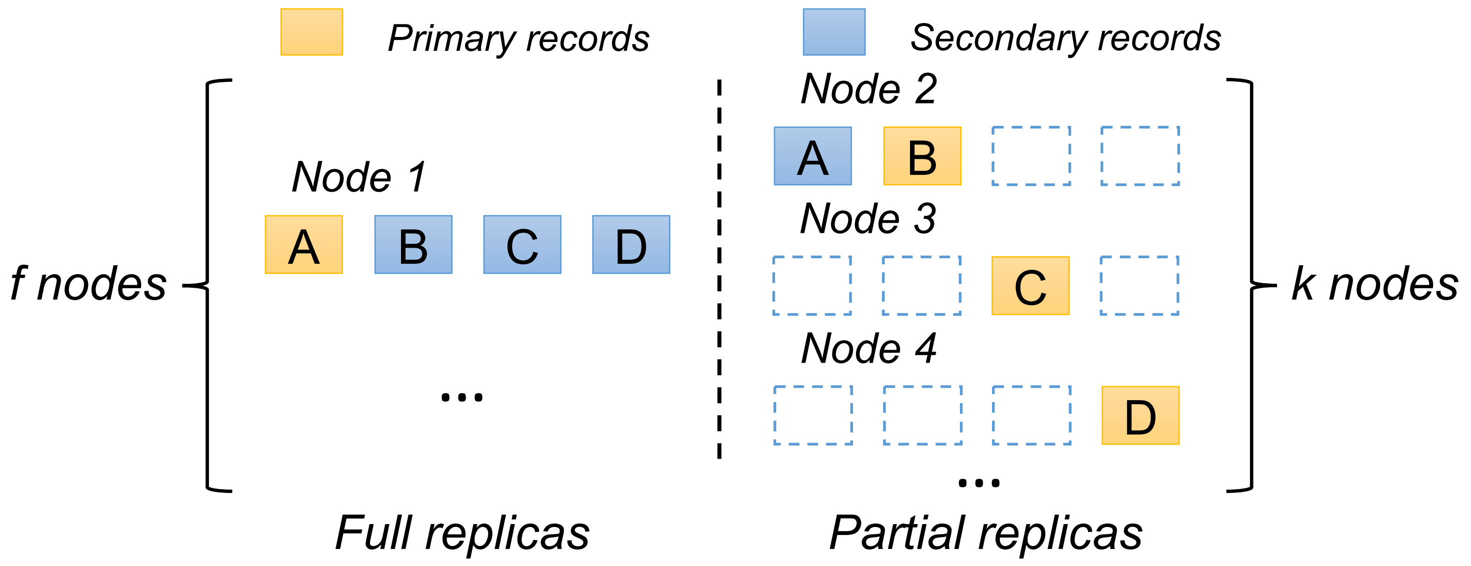
Our System: STAR

STAR uses partitioned and non-partitioned replicas to achieve the best of both worlds



Asymmetric replication

1. One of these replicas is complete
2. One of these replicas is partitioned across several nodes



Amazon EC2 and Google Cloud now provide high memory instances with 12 TB RAM, and 24 TB instances are coming in the fall of 2019.

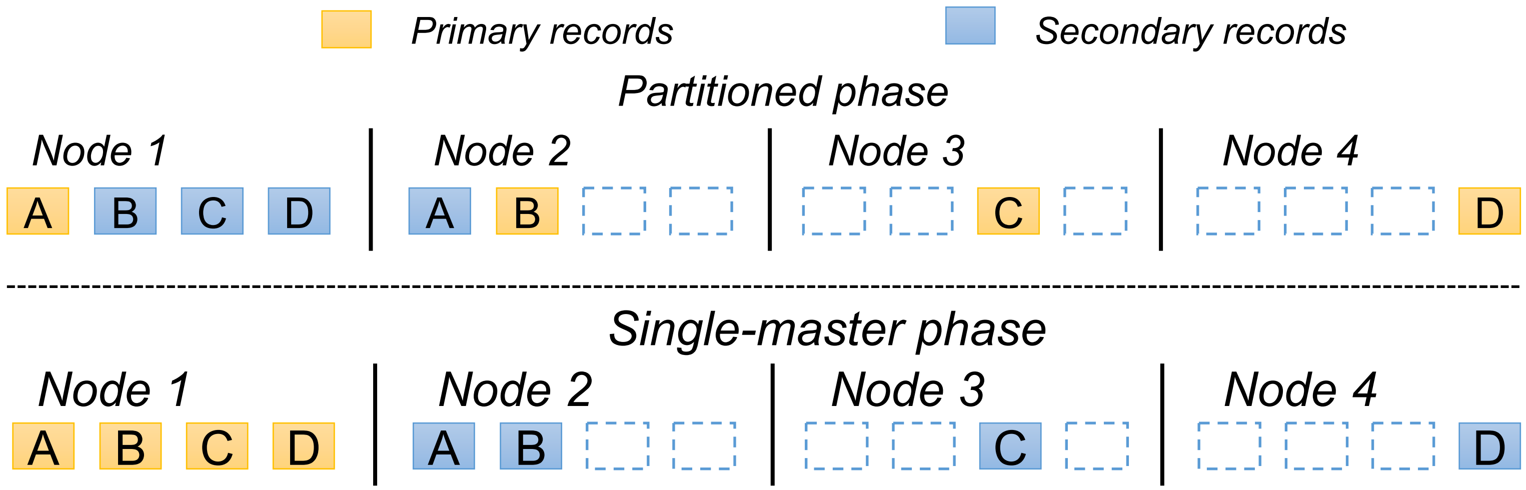
Partitioned phase and Single-master phase

T1: A = A + 1

T2: B = B + 1

T3: C = C + 1

T4: D = D + 1



T1: A = A + B + C

T2: B = B + C + D

Transactions only run over primary records.

The phase switching algorithm

Partitioned phase

Single-master phase

Start the **partitioned phase** execution
Sleep τ_p seconds

single-threaded execution per partition

--- Replication fence ---

Start the **single-master phase** execution
Sleep τ_s seconds

multi-threaded execution

--- Replication fence ---

Replication fence ensures replicas are consistent with one another before phase switching

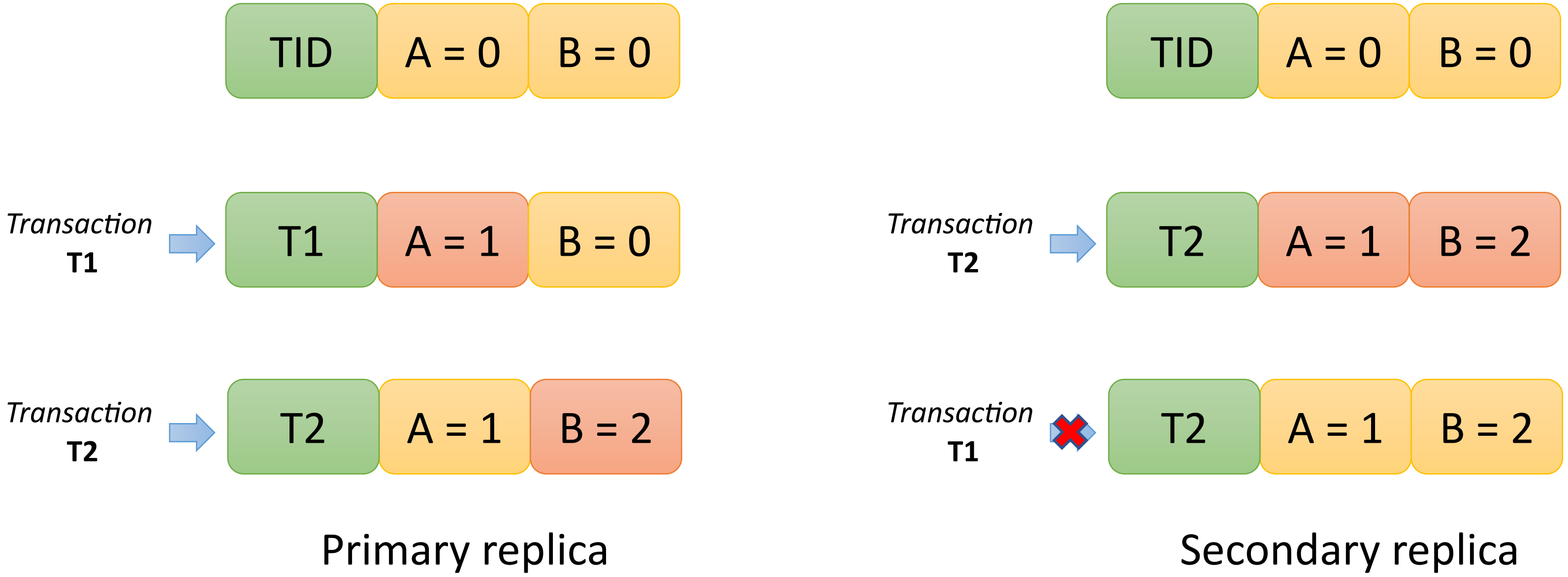
Replication between replicas

Value replication:

✓ each write, tagged with a TID, has the value of a whole record

T1: **A = B + 1**

T2: **B = A + 1**



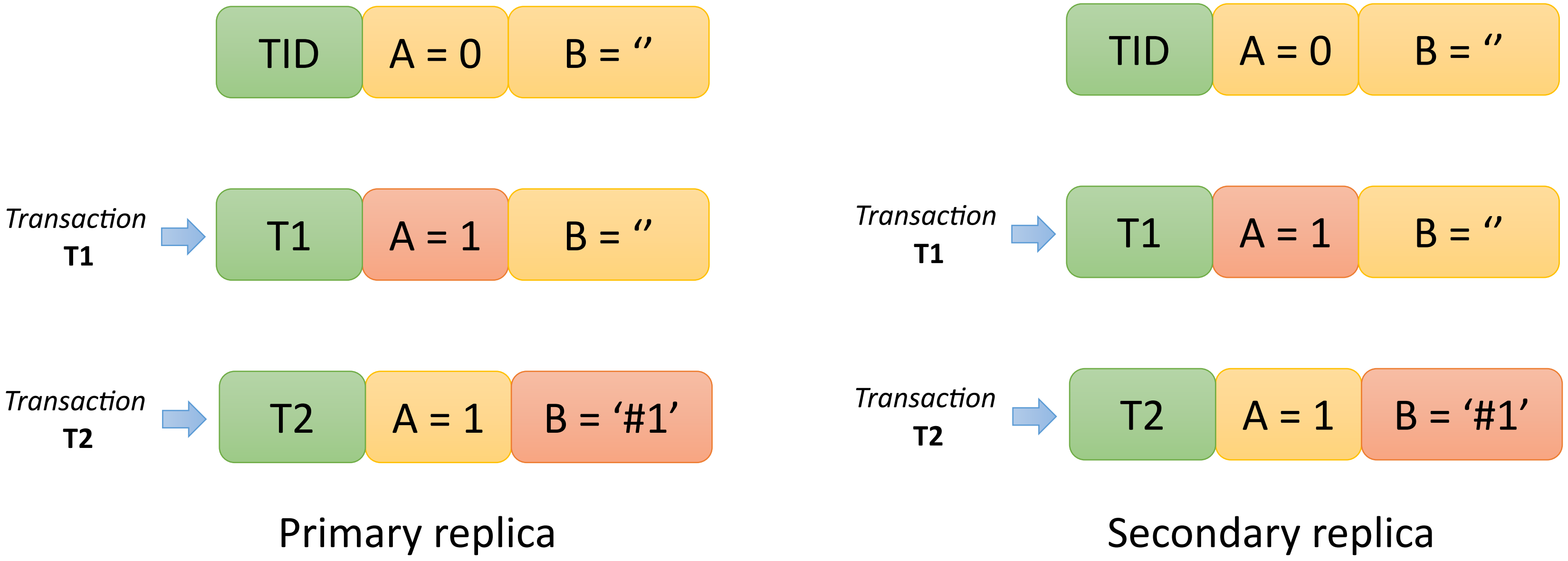
Optimization: replicating operations

Operation replication:

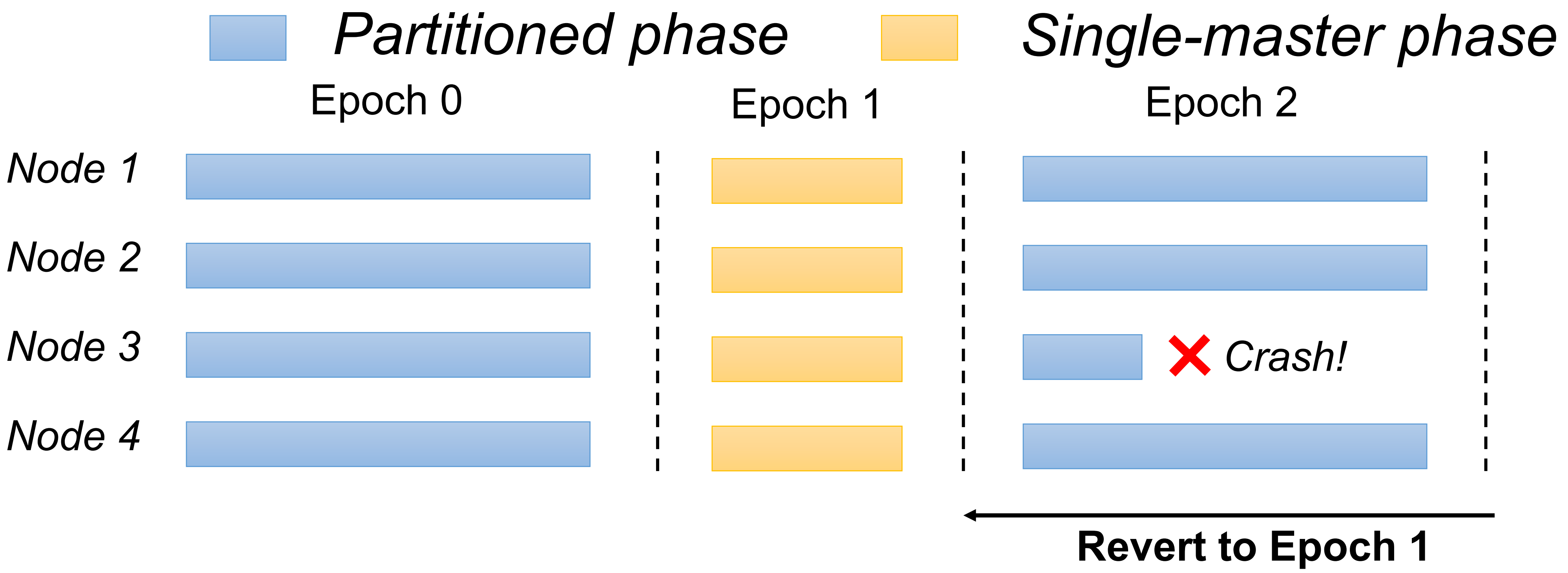
✓ The replication strictly follows the commit order in the single-master phase

T1: **A = len(B) + 1**

T2: B = B + $\underbrace{\textbf{'\#' + str(A)}}_{\text{operation}}$



Fault tolerance



Failure detection happens in replication fence

Experiments

A cluster of four m5.4xlarge nodes running on Amazon EC2

Benchmarks:

» YCSB

» TPC-C

Concurrency control algorithms:

» PB. OCC

» Dist. OCC

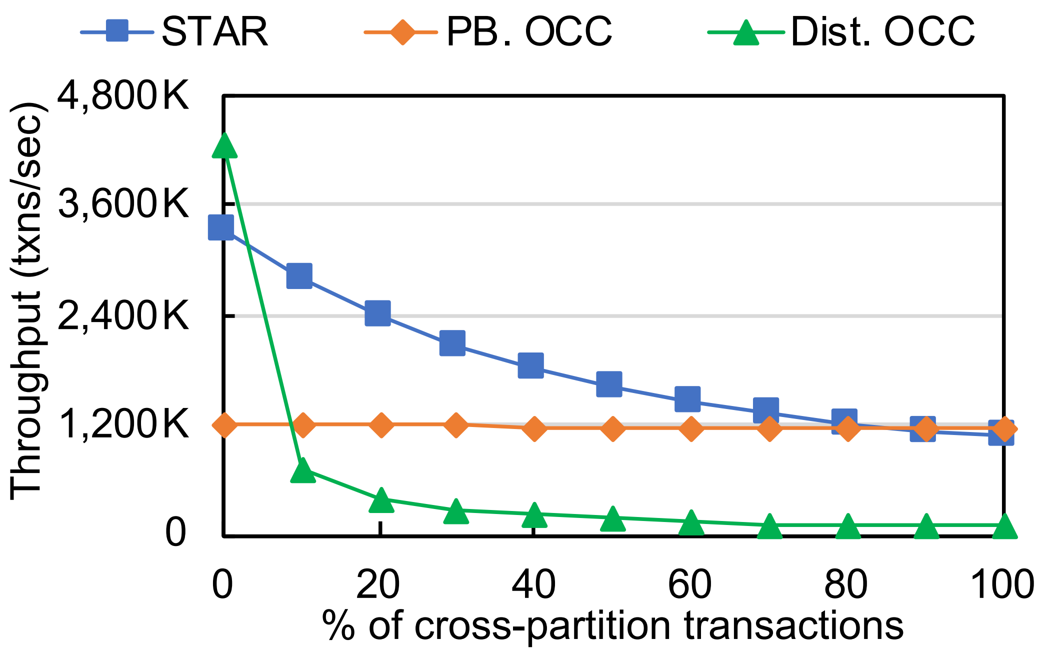
Synchronous replication

Asynchronous replication with epoch-based group commit

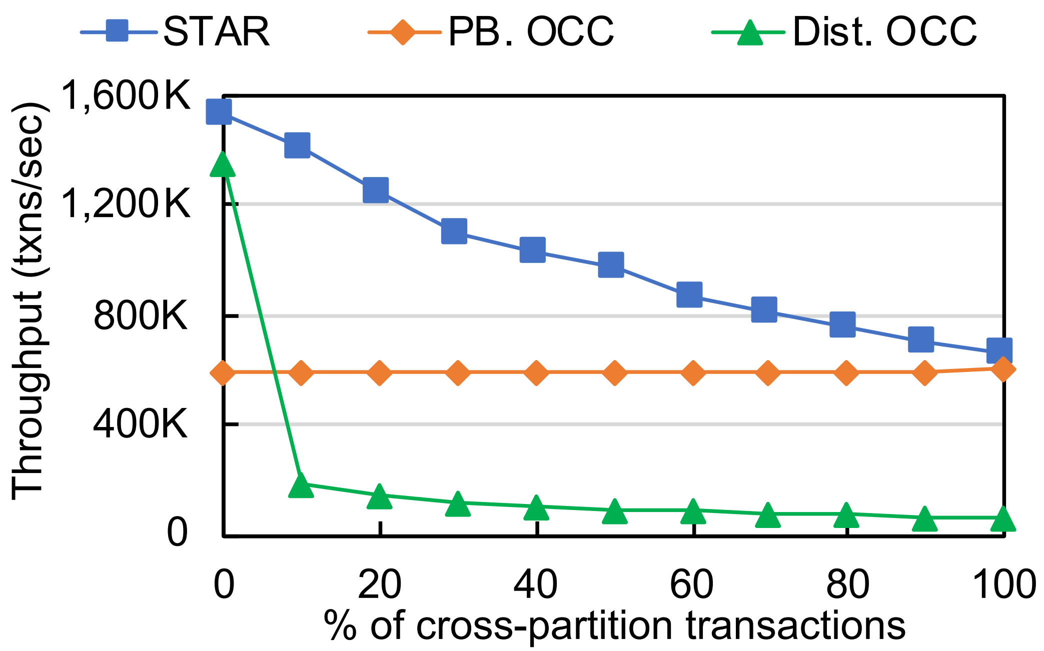
See our paper for comparison with Dist. S2PL and Calvin

Throughput comparison

Asynchronous replication and epoch-based group commit



YCSB



TPC-C

Latency comparison

They all have similar latency due to epoch-based group commit

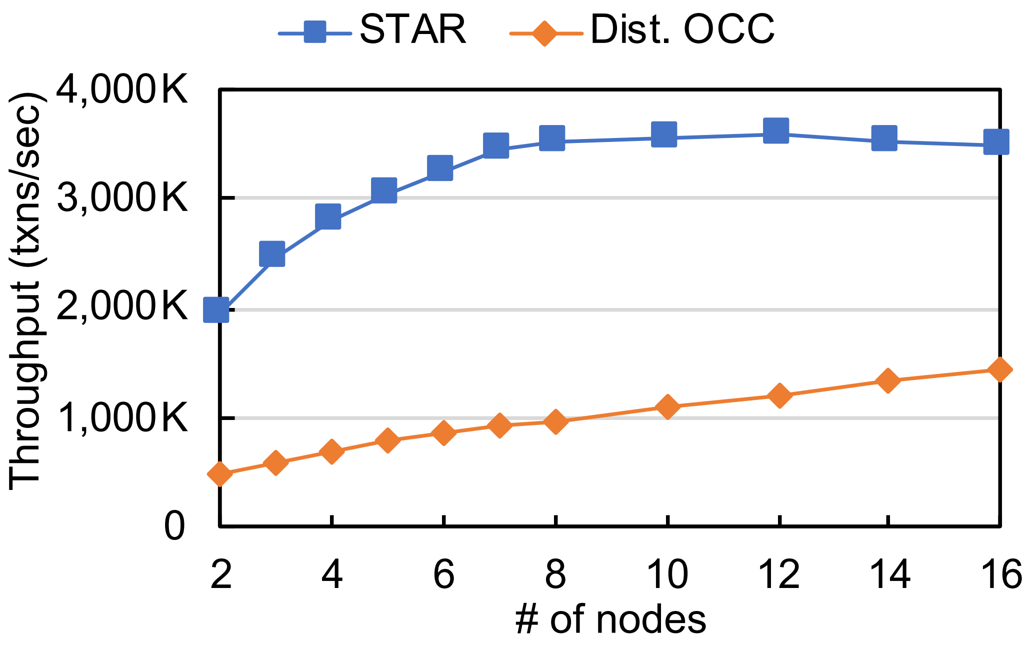
STAR	PB. OCC	Dist. OCC
6.2/9.4	5.5/11.3	6.4/11.4

50th percentile/99th percentile

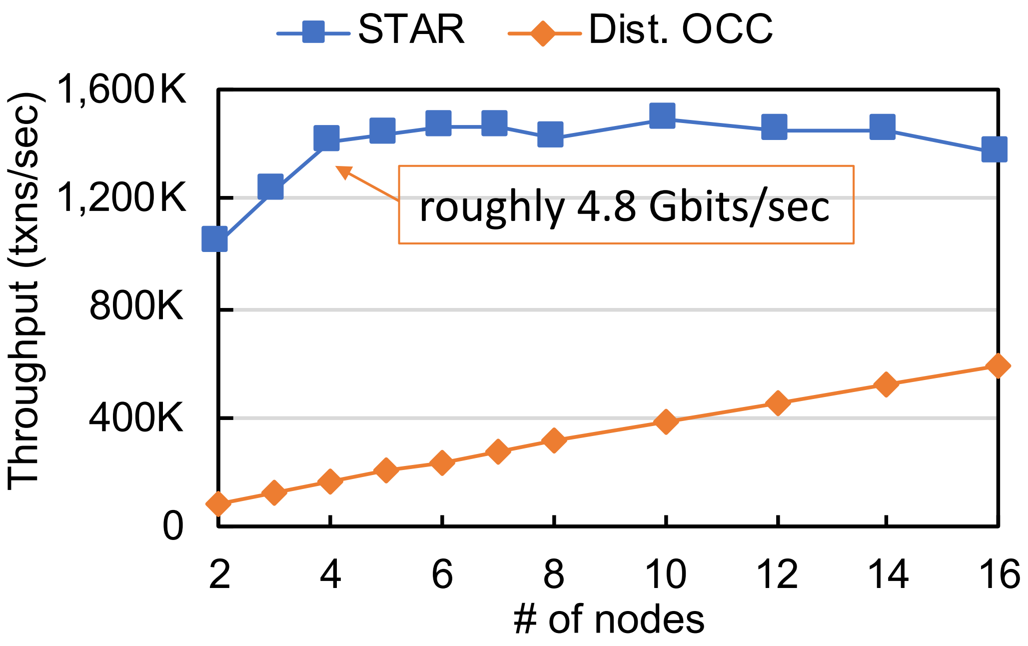
Scalability experiment

STAR scales out until the network saturates.

Other systems achieve much lower throughput with the same number of nodes.



YCSB



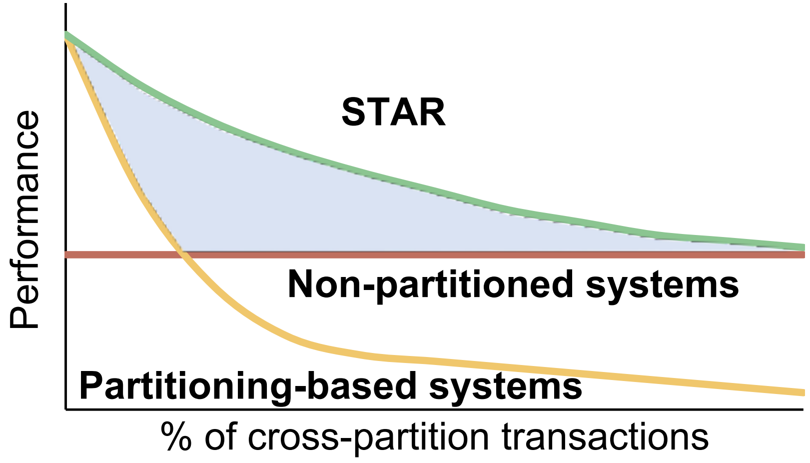
TPC-C

Conclusions

STAR employs a new phase-switching scheme

- » single-partition transactions are run on multiple machines in parallel
- » cross-partition transactions are run on a single machine by re-mastering records on the fly

STAR avoids cross-node communication and 2PC for distributed transactions.



Thank you

Scan the QR code to
access our paper.

Paper: https://tiny.cc/star_paper

Slides: https://tiny.cc/star_slides

Code: https://tiny.cc/star_git