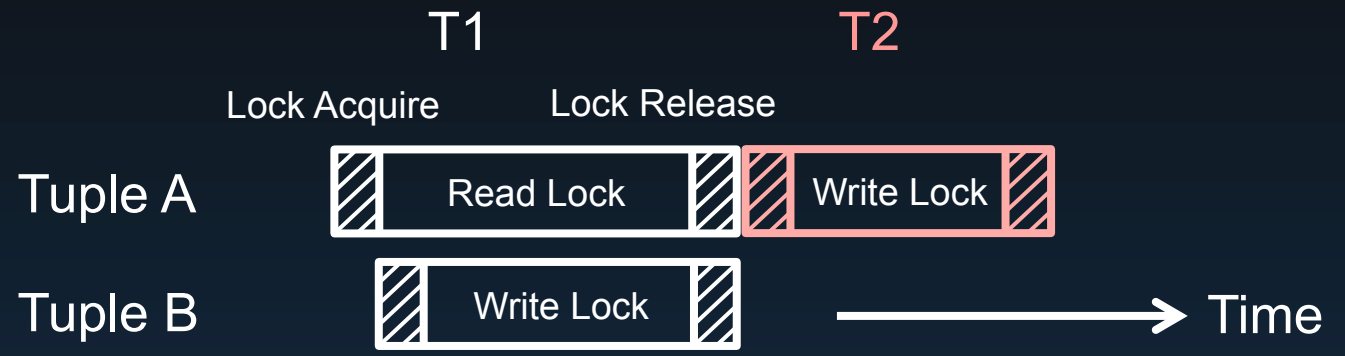
TicToc: Time Traveling Optimistic Concurrency Control

Xiangyao Yu[1], Andrew Pavlo[2], Daniel Sanchez[1], Srinivas Devadas[1]

Massachusetts Institute of Technology[1]
Carnegie Mellon University[2]

PHYSICAL VS LOGICAL TIME

Two-Phase Locking (2PL)



Timestamp Ordering (T/O)

T1 @ ts = 1 T2 @ ts = 2

TIMESTAMP ORDERING

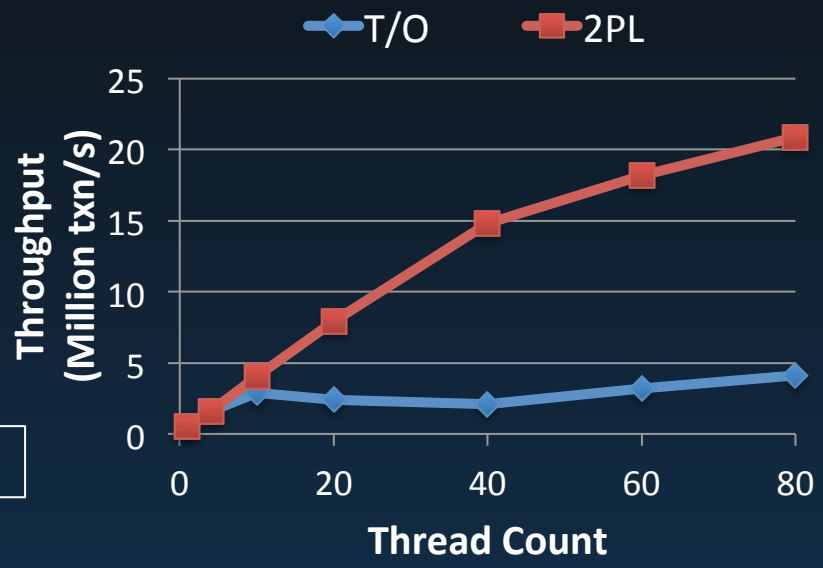
Timestamp Allocation

(txns have unique timestamps)

- Centralized Allocator
  - Timestamp allocation is a scalability bottleneck

```
ts = __sync_fetch_and_add(&glob_ts, 1)
```

- Synchronized Clock
  - Clock skew causes unnecessary aborts

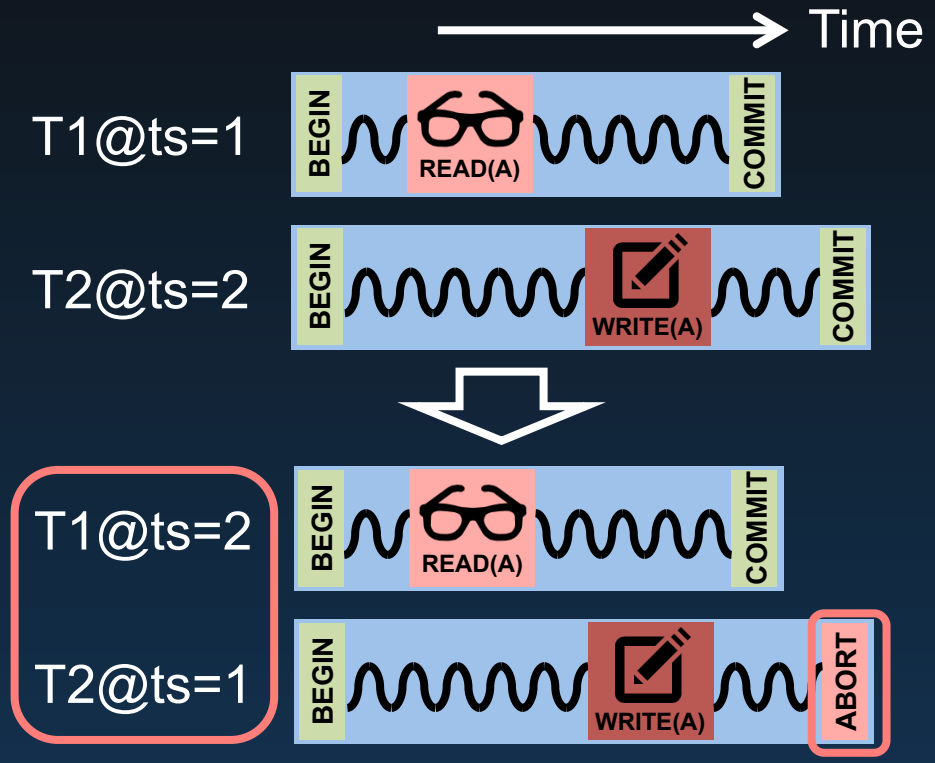


TIMESTAMP ORDERING

Timestamp Allocation
(txns have unique timestamps)

Static Timestamp Assignment

- Txns abort due to bad timestamp assignment



DATA DRIVEN TIMESTAMP MANAGEMENT

Traditional T/O

1. Acquire timestamp (TS)
2. Determine tuple visibility using TS

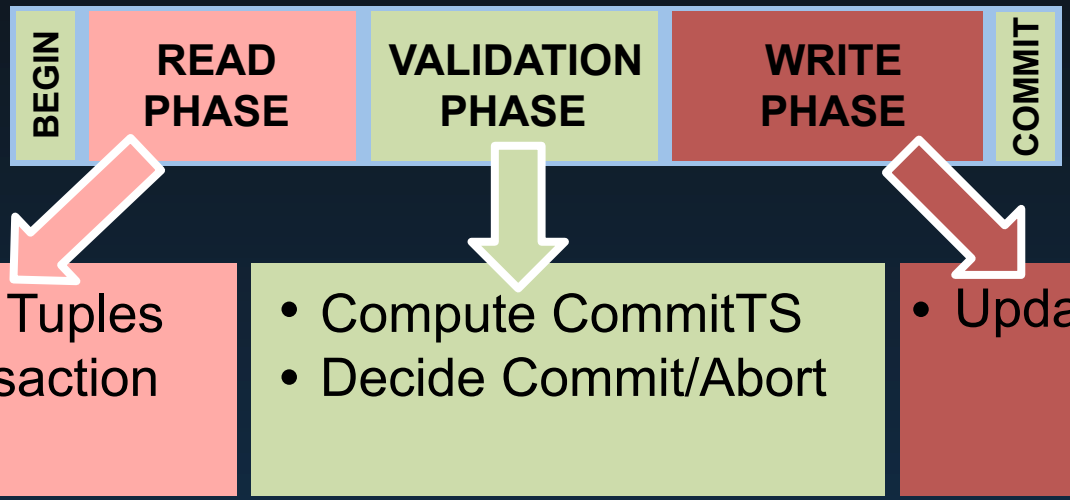
- ✗ Timestamp Allocation
- ✗ Static Timestamp Assignment

TicToc

1. Access tuples and remember their timestamp info.
2. Compute commit timestamp (CommitTS)

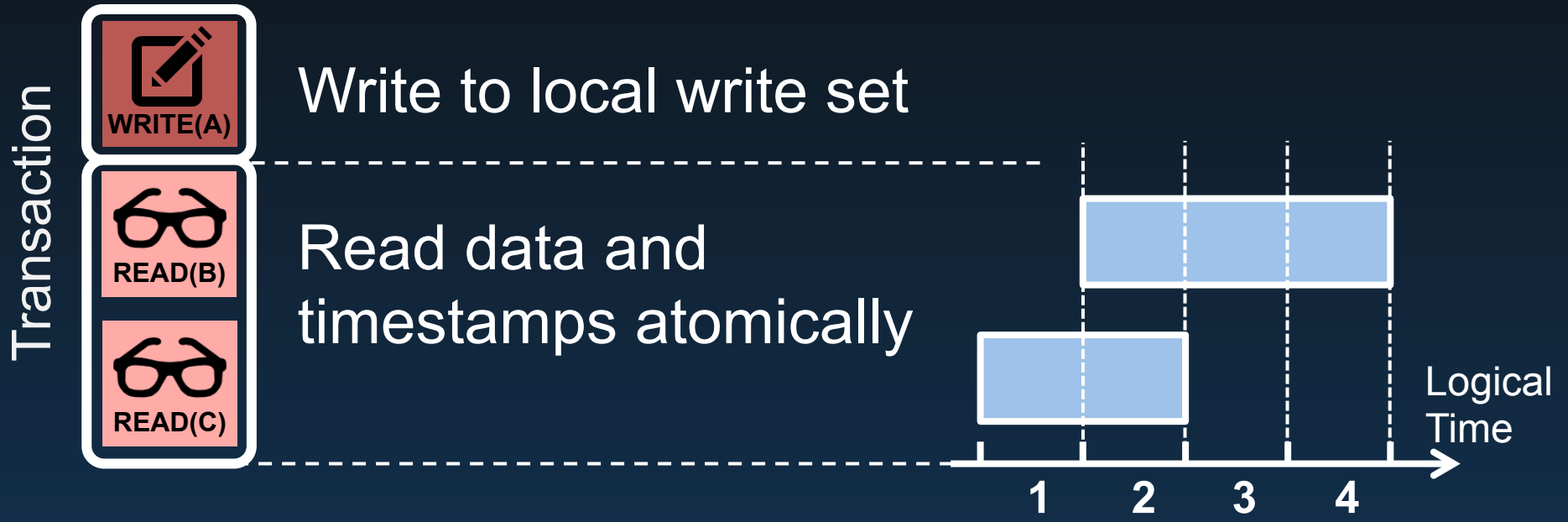
- ✓ No Timestamp Allocation
- ✓ Dynamic Timestamp Assignment

OPTIMISTIC CONCURRENCY CONTROL (OCC)

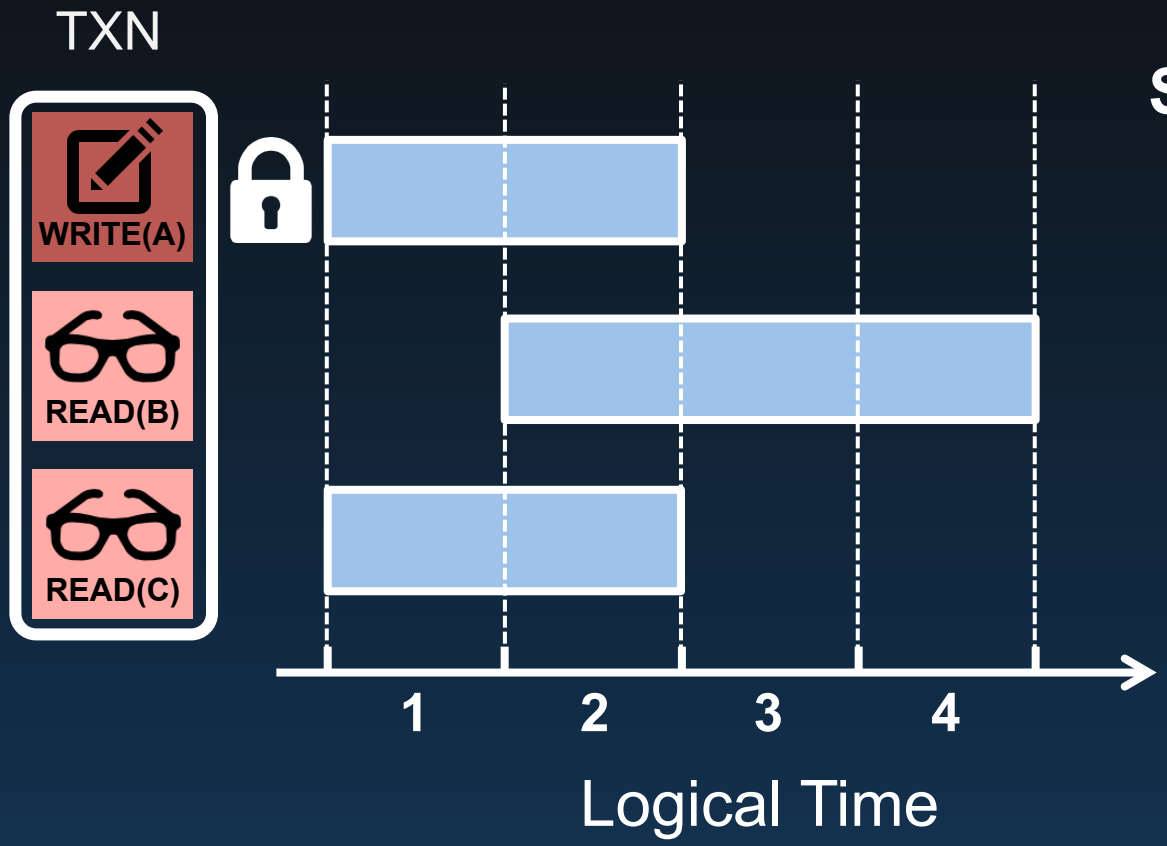


Tuple Format

READ PHASE

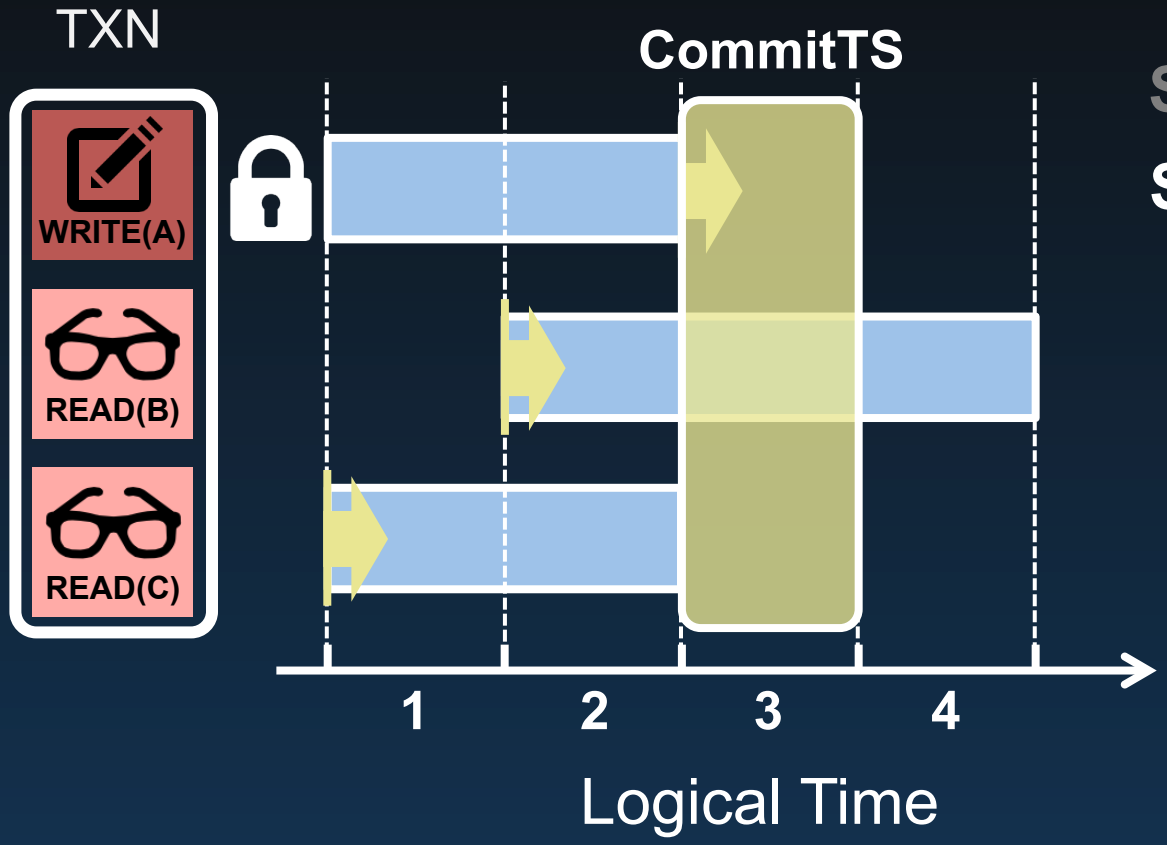

Transaction
WRITE(A)
READ(B)
READ(C)
Write to local write set
Read data and
timestamps atomically
Logical
Time
1
2
3
4

VALIDATION PHASE



Step 1 Lock Write Set

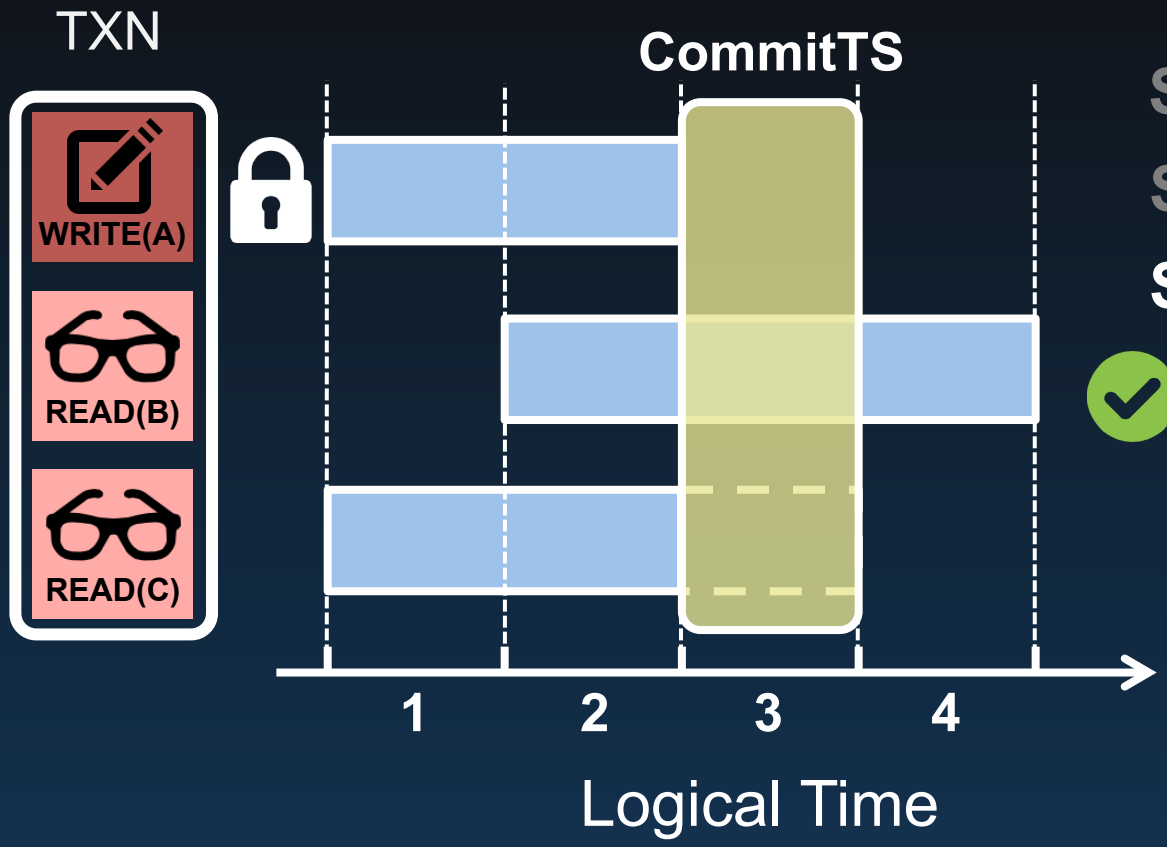
VALIDATION PHASE



Step 1 Lock Write Set

Step 2 Compute CommitTS

VALIDATION PHASE



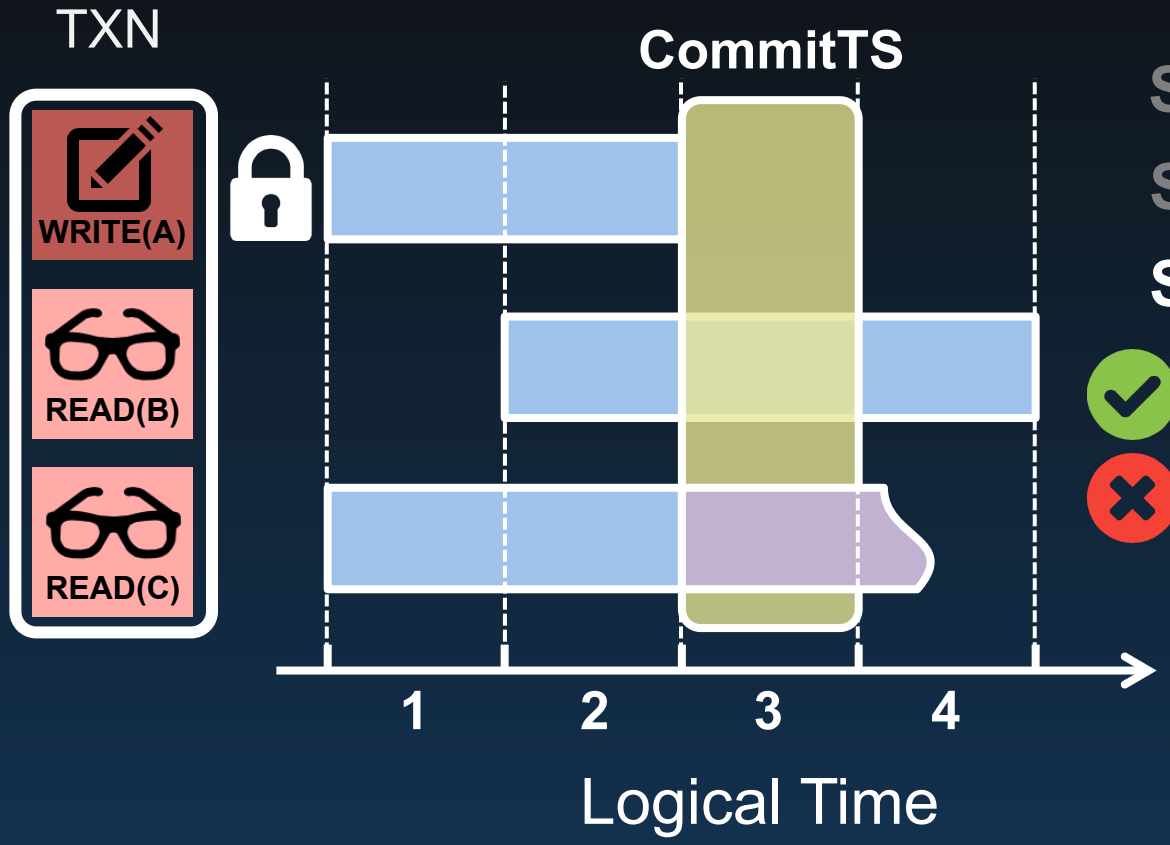
Step 1 Lock Write Set

Step 2 Compute CommitTS

Step 3 Validate Read Set

Case 1: latest version

VALIDATION PHASE



Step 1 Lock Write Set

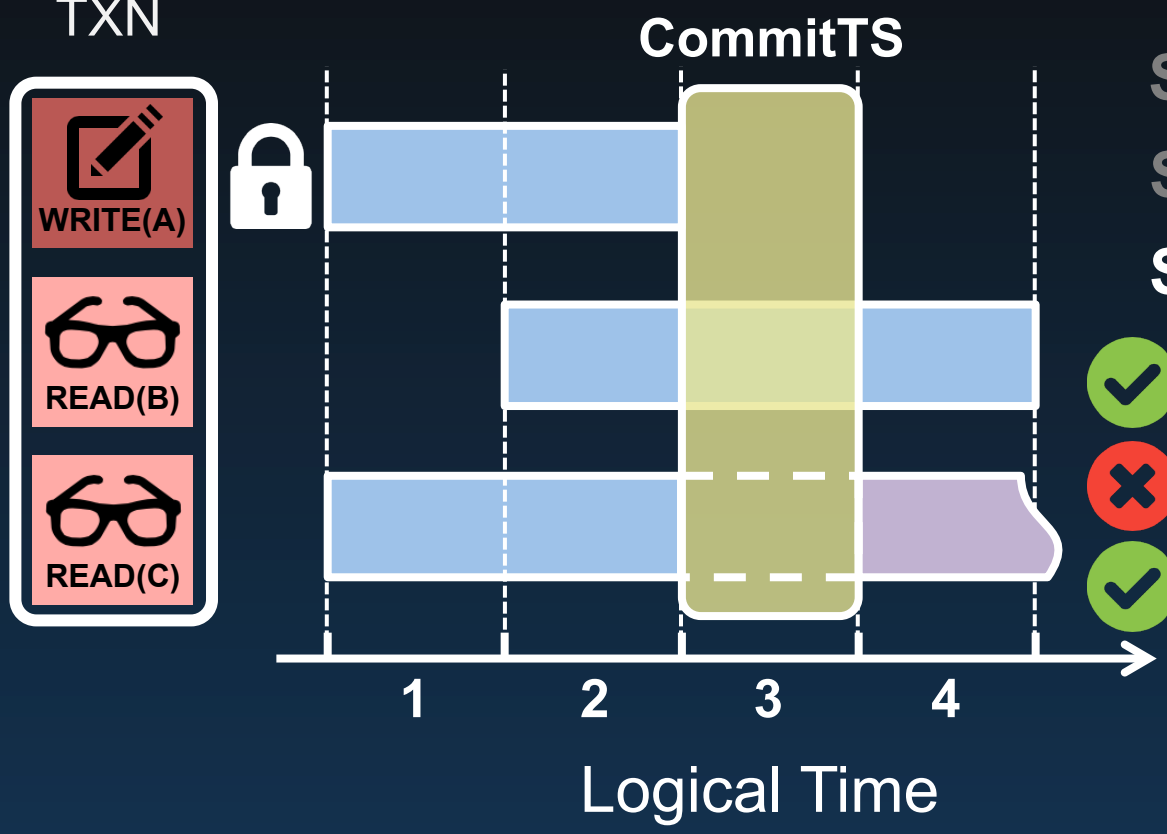
Step 2 Compute CommitTS

Step 3 Validate Read Set

Case 1: latest version

Case 2: New version at/before CommitTS

VALIDATION PHASE



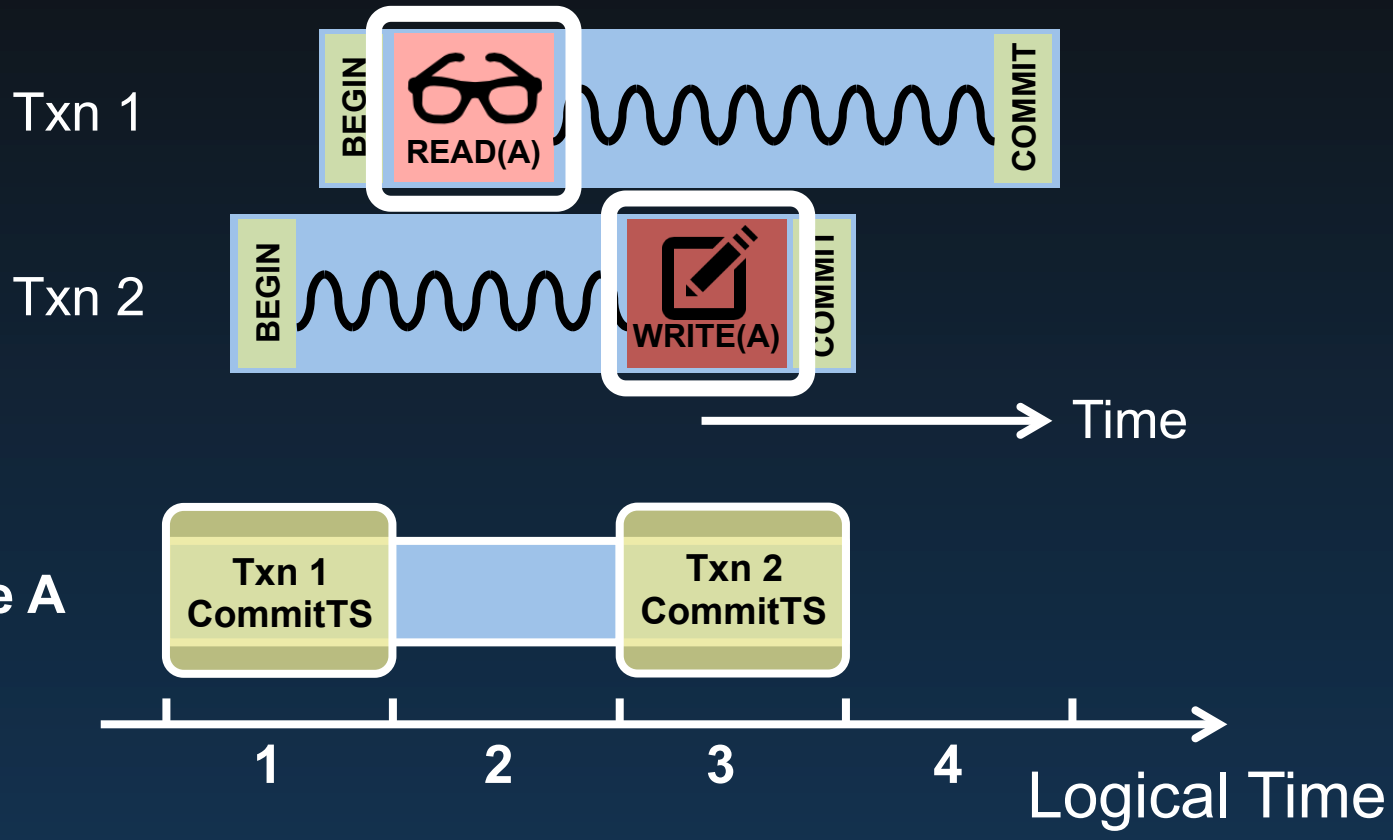
Step 1 Lock Write Set

Step 2 Compute CommitTS

Step 3 Validate Read Set

- Case 1: latest version
- Case 2: New version at/before CommitTS
- Case 3: New version after CommitTS

TIME TRAVELING



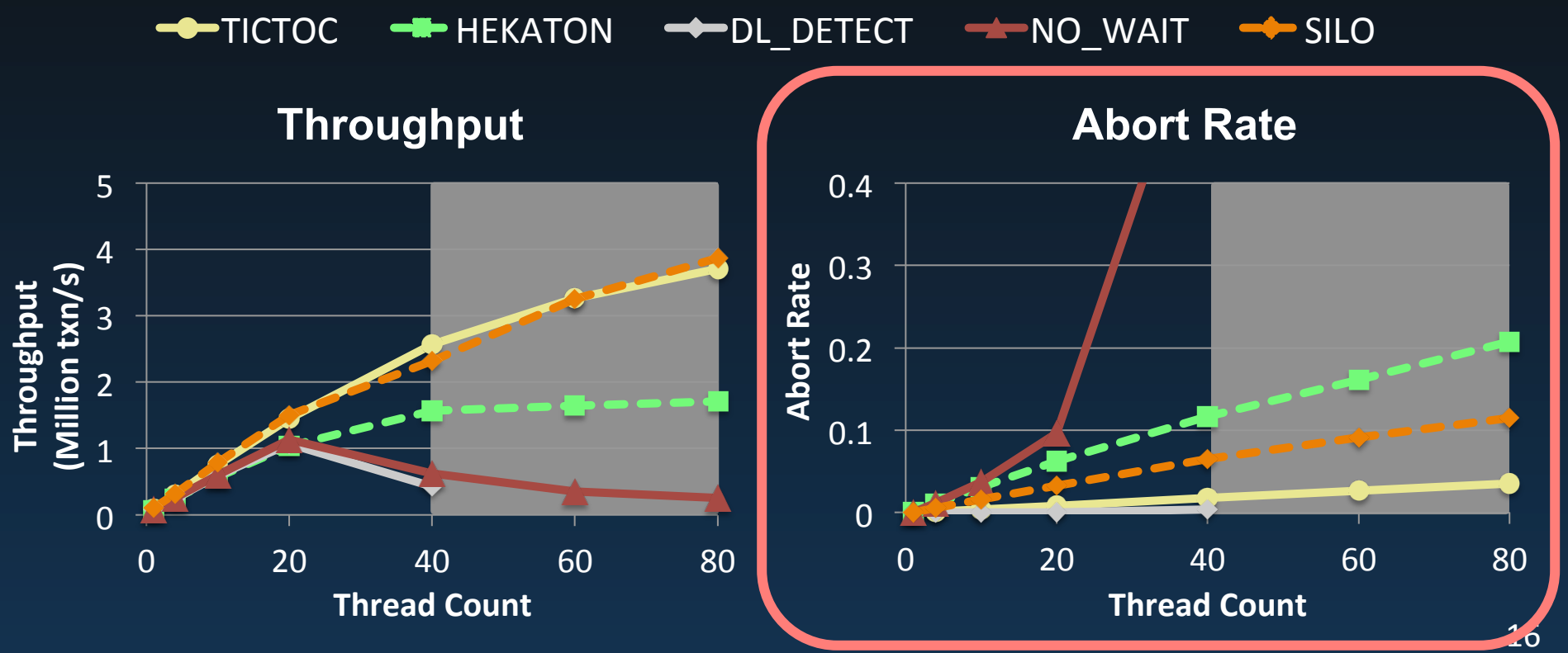
TICTOC OPTMIZATIONS

- No-wait locking in validation phase
- Preemptive aborts
- Timestamp history
- Lower isolation levels

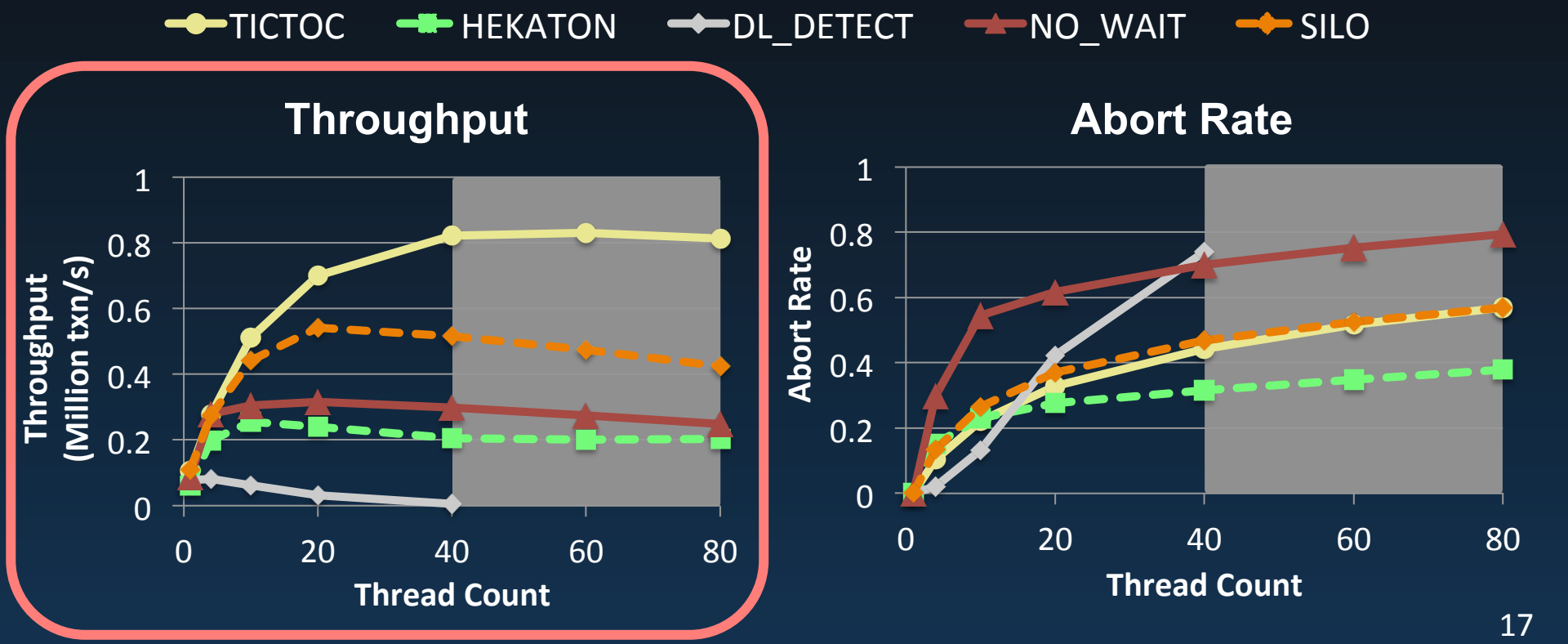
EXPERIMENTAL SETUP

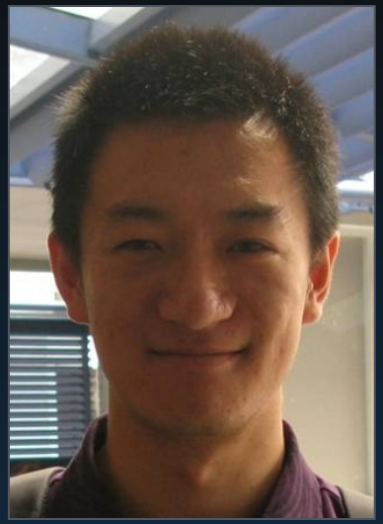
- 4-socket, 40-core Machine (80 threads w/ hyper-threading)
- Main Memory DBx1000
  - No logging
  - No B-tree (Hash indexing)
- Concurrency Control Algorithms
  - MVCC: HEKATON
  - OCC: SILO
  - 2PL: DL_DETECT, NO_WAIT

YSCB Medium Contention

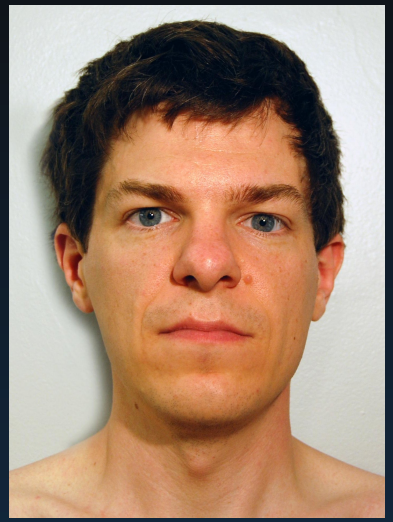


YSCB High Contention

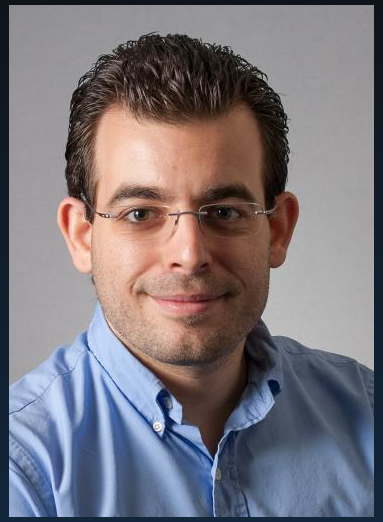




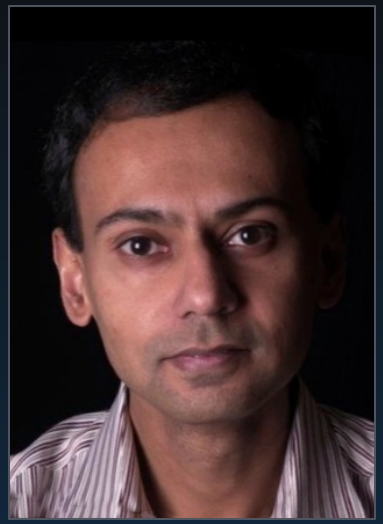
Xiangyao Yu
yxy@mit.edu



Andy Pavlo
pavlo@cs.cmu.edu



Daniel Sanchez
sanchez@mit.edu



Srinivas Devadas
devadas@mit.edu

DBx1000 (https://github.com/yxymit/DBx1000)

Carnegie Mellon University