Transactional Contention Management as a Non-Clairvoyant Scheduling Problem

Hagit Attiya, Alessia Milani

Technion, Haifa-LABRI, University of Bordeaux

Optimism

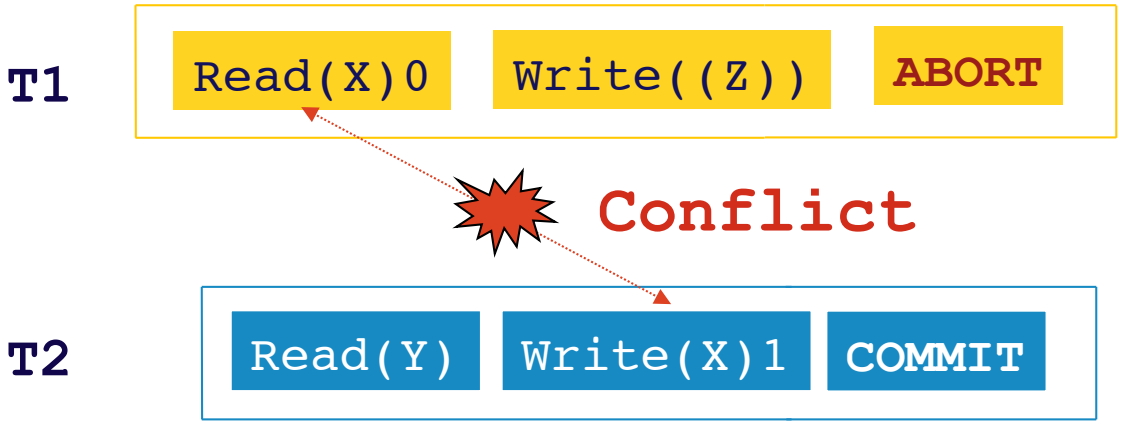
- Transactions (Txs) proceed until a conflict occurs
  - T_1 conflicts with an on-going T_2 if T_1 tries to write to a data item previously accessed by T_2

⇨ one transaction aborts or waits for the other to complete

  - If no conflict occurs, they run in parallel

Conservative approach

- A conflict does not imply a violation of serializability



- T1 and T2 can both commit without violating strict serializability

Contention manager mediates conflicts

- Decides which transaction aborts
  - E.g., the Greedy contention manager [Guerraoui et al. PODC 05]
    - Each Tx is assigned a unique timestamp reflecting Txs real-time order
    - If Txs T1 and T2 conflict, the Tx with the smaller timestamp aborts
- Decides when to restart aborted Txs
  - E.g., CAR-STM [Dolev et al. PODC 08]
    - the aborted Tx is not executed until the completion of the conflicting Tx ➡ unrelated Txs may be executed serially

Need for a "clever" contention manager

- Complete the work quickly
  - ➡ makespan : Worst-case total time to complete all transactions
- Not waste work
  - ➡ do not repeate conflicts
- What Works and Why?

What Works and Why? In Practice

[Scherer and Scott, CSJP 04]

- Extensive testing
  - Backoff
  - Aging
  - Randomized
  - Various priority
  - …
- None dominates on all benchmarks

What Works and Why? In Theory

- Contention Management as a Scheduling problem
  - Evaluate the throughput, measured by the makespan of a finite set of transactions
    - Worst-case total time to complete all transactions
  - Relative to the makespan guaranteed by an optimal off-line scheduler

Non-Clairvoyant Scheduling

- A scheduler *A* does not know Txs characteristics a priori
  - Txs arrive one by one, and their duration is unknown
- Evaluated in comparison with an optimal, clairvoyant scheduler
  - knows the set of Txs, their data set, their release times and duration
- Competitive ratio: $\max_{\Gamma} makespan_A(\Gamma) / makespan_{OPT}(\Gamma)$

A lower bound for CM [Attiya et al. PODC 06]

Theorem 1. The competitive ratio of any work conserving CM is $\Omega(s)$, where s is # of data items

It always lets a maximal set of non-conflicting transactions run

Read-Dominated Workloads

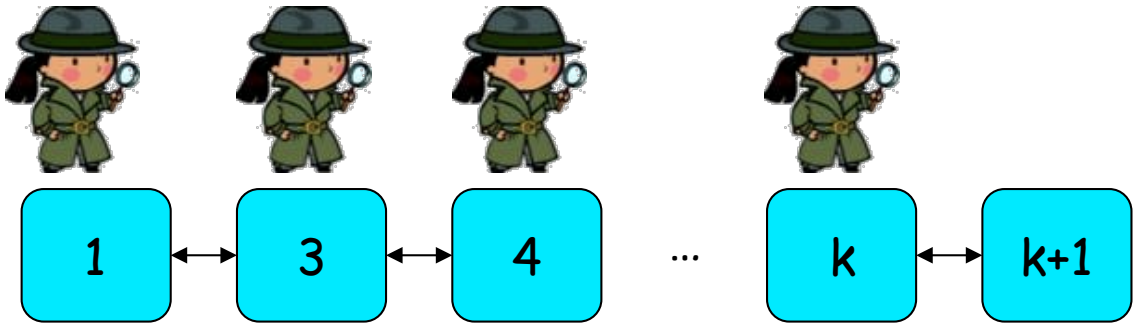
- Existing results hold for write-dominated workloads
  - Transactions need exclusive access for most of their duration (*early-write* transactions)

[Guerraoui et al. PODC 05, Attiya et al. PODC 06]

- What about read-dominated workloads?

[Attiya & Milani. OPODIS 09]

  - Read-only transactions
  - *Late-write* transactions



Extending the lower bound to read-dominated workload

Theorem 3. There is a read-dominated workload, s.t. the competitive ratio of any deterministic CM is $\Omega(s)$

- It holds also for CM that has a more careful approach than being conservative
- Transactions :
  - have the same duration, are available at time 0
  - But may have a different data set if executed at different times or restarted

Lower Bound : Workload

	1	2	…	q=s/2
1	$R_1 \ldots R_q\ R_{q+1}\ W_{q+1}$	$R_1 \ldots R_q\ R_{q+1}\ W_{q+1}$	…	$R_1 \ldots R_q\ R_{q+1}\ W_{q+1}$
2	$R_1 \ldots R_q\ R_{q+2}\ W_{q+2}$	$R_1 \ldots R_q\ R_{q+2}\ W_{q+2}$	…	$R_1 \ldots R_q\ R_{q+2}\ W_{q+2}$
⋮	⋮	⋮	…	⋮
i	$R_1 \ldots R_q\ R_{q+i}\ W_{q+i}$	$R_1 \ldots R_q\ R_{q+i}\ W_{q+i}$	…	$R_1 \ldots R_q\ R_{q+i}\ W_{q+i}$
⋮	⋮	⋮	…	⋮
q	$R_1 \ldots R_q\ R_{2q}\ W_{2q}$	$R_1 \ldots R_q\ R_{2q}\ W_{2q}$	…	$R_1 \ldots R_q\ R_{2q}\ W_{2q}$
1	$R_1\ R_2 \ldots R_{q-1} R_q$	$R_1\ R_2 \ldots R_{q-1} R_q$	…	$R_1\ R_2 \ldots R_{q-1} R_q$
2	$R_1\ R_2 \ldots R_{q-1} R_q$	$R_1\ R_2 \ldots R_{q-1} R_q$	…	$R_1\ R_2 \ldots R_{q-1} R_q$
⋮	$R_1\ R_2 \ldots R_{q-1} R_q$	$R_1\ R_2 \ldots R_{q-1} R_q$	…	$R_1\ R_2 \ldots R_{q-1} R_q$
m-q	$R_1\ R_2 \ldots R_{q-1} R_q$	$R_1\ R_2 \ldots R_{q-1} R_q$	…	$R_1\ R_2 \ldots R_{q-1} R_q$

Makespan of Non-Clairvoyant Scheduler

Work conserving CM must select an independent set of m Txs

e.g., 1 row plus m-q read-only Txs

	1	2	…	q
1	$R_1\ R_q\ R_{q+1}\ W_{q+1}$	$R_1\ R_q\ R_{q+1}\ W_{q+1}$	…	$R_1\ R_q\ R_{q+1}\ W_{q+1}$
2	$R_1\ R_q\ R_{q+2}\ W_{q+2}$	$R_1\ R_q\ R_{q+2}\ W_{q+2}$	…	$R_1\ R_q\ R_{q+2}\ W_{q+2}$
⋮	⋮	⋮	…	⋮
i	$R_1\ R_q\ R_{q+i}\ W_{q+i}$	$R_1\ R_q\ R_{q+i}\ W_{q+i}$	…	$R_1\ R_q\ R_{q+i}\ W_{q+i}$
⋮	⋮	⋮	…	⋮
q	$R_1\ R_q\ R_{2q}\ W_{2q}$	$R_1\ R_q\ R_{2q}\ W_{2q}$	…	$R_1\ R_q\ R_{2q}\ W_{2q}$
q+1	$R_1\ R_2 \dots R_{q-1} R_q$	$R_1\ R_2 \dots R_{q-1} R_q$	…	$R_1\ R_2 \dots R_{q-1} R_q$
q+ 2	$R_1\ R_2 \dots R_{q-1} R_q$	$R_1\ R_2 \dots R_{q-1} R_q$	…	$R_1\ R_2 \dots R_{q-1} R_q$
⋮	$R_1\ R_2 \dots R_{q-1} R_q$	$R_1\ R_2 \dots R_{q-1} R_q$	…	$R_1\ R_2 \dots R_{q-1} R_q$
m	$R_1\ R_2 \dots R_{q-1} R_q$	$R_1\ R_2 \dots R_{q-1} R_q$	…	$R_1\ R_2 \dots R_{q-1} R_q$

Makespan of Non-Clairvoyant Scheduler

Only one Tx in a given row can commit

Restarted Txs all request the same data item

	1	2	…	q
1	$R_1\ R_q\ R_{q+1}\ W_{q+1}$	$R_1\ R_q\ R_{q+1}\ W_{q+1}$	…	$R_1\ R_q\ R_{q+1}\ W_{q+1}$
2	$R_1\ R_q\ R_{q+2}\ W_{q+2}$	$R_1\ R_q\ R_{q+2}\ W_{q+2}$	…	$R_1\ R_q\ R_{q+2}\ W_{q+2}$
⋮	⋮	⋮	…	⋮
i	$R_1\ R_q\ R_{q+i}\ W_{q+i}$	$R_1\ R_q\ R_{q+i}\ W_{q+i}$	…	$R_1\ R_q\ R_{q+i}\ W_{q+i}$
⋮	⋮	⋮	…	⋮
q	$R_1\ R_q\ R_{2q}\ W_{2q}$	$R_1\ R_q\ R_{2q}\ W_{2q}$	…	$R_1\ R_q\ R_{2q}\ W_{2q}$
q+1	$R_1\ R_2 \dots R_{q-1} R_q$	$R_1\ R_2 \dots R_{q-1} R_q$	…	$R_1\ R_2 \dots R_{q-1} R_q$
q+ 2	$R_1\ R_2 \dots R_{q-1} R_q$	$R_1\ R_2 \dots R_{q-1} R_q$	…	$R_1\ R_2 \dots R_{q-1} R_q$
⋮	$R_1\ R_2 \dots R_{q-1} R_q$	$R_1\ R_2 \dots R_{q-1} R_q$	…	$R_1\ R_2 \dots R_{q-1} R_q$
m	$R_1\ R_2 \dots R_{q-1} R_q$	$R_1\ R_2 \dots R_{q-1} R_q$	…	$R_1\ R_2 \dots R_{q-1} R_q$

Makespan of Non-Clairvoyant Scheduler

At time q, still ≈ q^2 late-write Txs to be executed

We have to execute them serially

q=s/2⇨

Makespan

$s^2/4$

	1	2	…	q
1		$R_1…R_q\ R_1\ W_1$	…	$R_1…R_q\ R_1\ W_1$
2		$R_1…R_q\ R_1\ W_1$	…	$R_1…R_q\ R_1\ W_1$
⋮		⋮	…	⋮
i		$R_1…R_q\ R_1\ W_1$	…	$R_1…R_q\ R_1\ W_1$
⋮		⋮	…	⋮
q		$R_1…R_q\ R_1\ W_1$	…	$R_1…R_q\ R_1\ W_1$
q+1			…	
q+ 2				
⋮			…	
m			…	

To remove the work-conserving assumption :
A Tx that starts after time q is [$R_1…R_q\ R_1\ W_1$]

Makespan of the Clairvoyant Scheduler

Each column is an independent set of Txs

At time q, all Txs are committed

q=s/2⇨

Makespan s

Competitive ratio s/2

	1	2	…	q
1	$R_1\ R_q\ R_{q+1}\ W_{q+1}$	$R_1\ R_q\ R_{q+1}\ W_{q+1}$	…	$R_1\ R_q\ R_{q+1}\ W_{q+1}$
2	$R_1\ R_q\ R_{q+2}\ W_{q+2}$	$R_1\ R_q\ R_{q+2}\ W_{q+2}$	…	$R_1\ R_q\ R_{q+2}\ W_{q+2}$
⋮	⋮	⋮	…	⋮
i	$R_1\ R_q\ R_{q+i}\ W_{q+i}$	$R_1\ R_q\ R_{q+i}\ W_{q+i}$	…	$R_1\ R_q\ R_{q+i}\ W_{q+i}$
⋮	⋮	⋮	…	⋮
q	$R_1\ R_q\ R_{2q}\ W_{2q}$	$R_1\ R_q\ R_{2q}\ W_{2q}$	…	$R_1\ R_q\ R_{2q}\ W_{2q}$
q+1	$R_1\ R_2 \dots R_{q-1}R_q$	$R_1\ R_2 \dots R_{q-1}R_q$	…	$R_1\ R_2 \dots R_{q-1}R_q$
q+ 2	$R_1\ R_2 \dots R_{q-1}R_q$	$R_1\ R_2 \dots R_{q-1}R_q$	…	$R_1\ R_2 \dots R_{q-1}R_q$
⋮	$R_1\ R_2 \dots R_{q-1}R_q$	$R_1\ R_2 \dots R_{q-1}R_q$	…	$R_1\ R_2 \dots R_{q-1}R_q$
m	$R_1\ R_2 \dots R_{q-1}R_q$	$R_1\ R_2 \dots R_{q-1}R_q$	…	$R_1\ R_2 \dots R_{q-1}R_q$

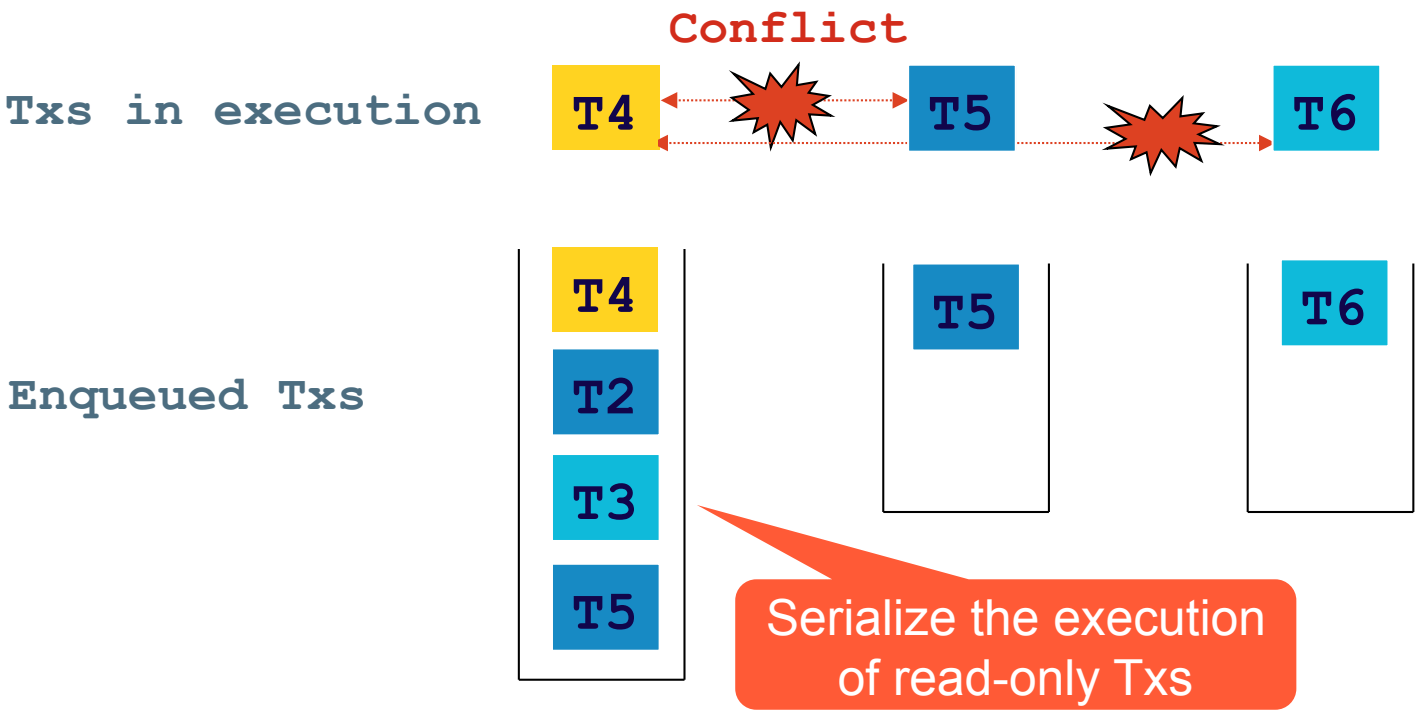
A lower bound for conservative CM

Theorem 4. There is a late-write workload,such that the competitive ratio of any deterministic conservative scheduler is $\Omega(m)$

On Bimodal Workloads

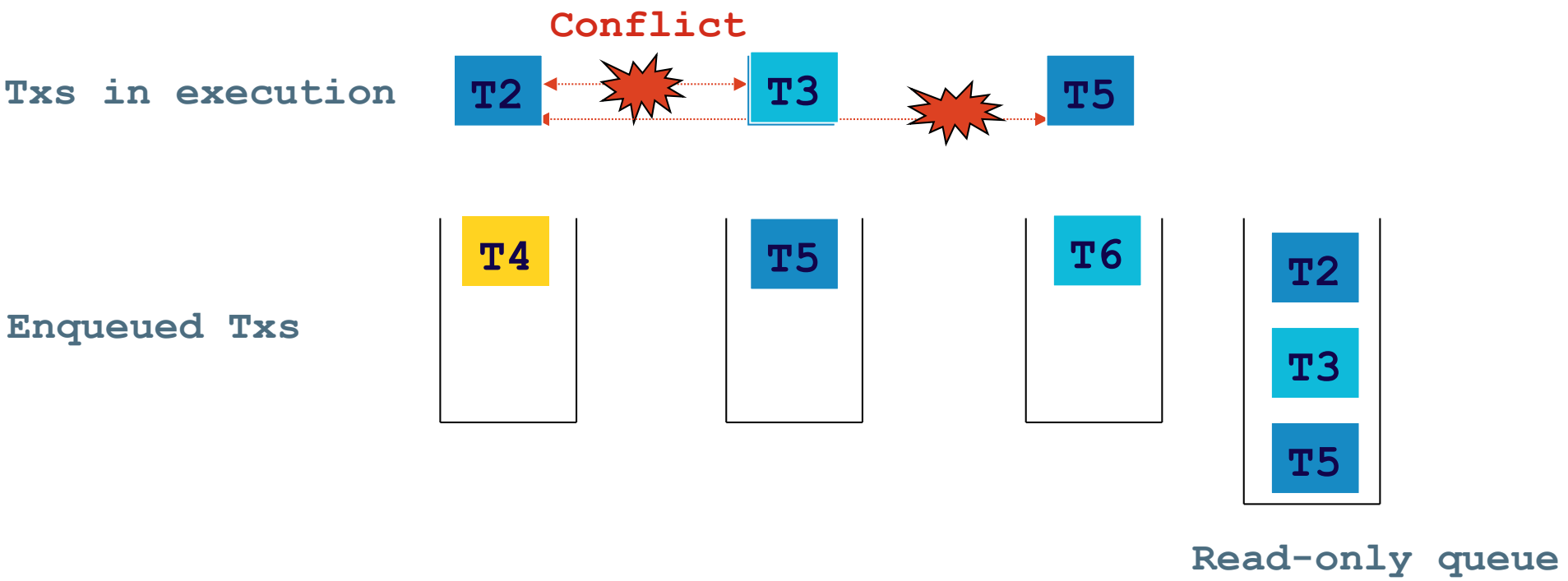
- If a transaction writes, it writes from the very beginning
- Recent contention managers try to avoid repeated conflicts by serializing conflicting Txs
  - CAR-STM, Steal On Abort, ATS
  - They are conservative
    - $\Omega(m)$ competitive ratio for read-dominated workloads (by Theorem 4)
  - also $\Omega(m)$ competitive ratio for bimodal workloads

CAR-STM scheduler



- T1 is a writing Tx
- T2 and T3 and T5 are read-only Txs

Bimodal scheduler



- T1 is a writing Tx
- T2 and T3 and T5 are read-only Txs

Summary

WORKLOADS	Any scheduler		
WRITE-DOMINATED Early write	**Θ(s)** **[Attiya et al.]** (1)		
BIMODAL : Early write + read-only	**Ω(s)** **derived from [Attiya et al.]**	**Ω(m)** **CAR-STM, ATS, SoA**	**O(s)** **Bimodal** (3)
READ-DOMINATED : Late write + read-only	**Ω(s)**	**Ω(m)** (2)	**O(m)** **trivial**

"Conservative" schedulers

late-write Txs are more difficult to handle than read-only Txs