

# A Competitive Analysis for Balanced Transactional Memory Workloads

**Gokarna Sharma** and **Costas Busch**

Department of Computer Science  
Louisiana State University, USA

December 16, 2010

# Transactional Memory - Background

- The multi-core revolution is here.
  - ▶ Opportunities and challenges

# Transactional Memory - Background

- The multi-core revolution is here.
  - ▶ Opportunities and challenges
- How to handle access to shared data?
  - ▶ Traditional approaches using Locks, Monitors, ...

# Transactional Memory - Background

- The multi-core revolution is here.
  - ▶ Opportunities and challenges
- How to handle access to shared data?
  - ▶ Traditional approaches using Locks, Monitors, ...
- Transactional Memory (TM) as an alternative synchronization abstraction.
  - ▶ Simple, composable, ...

# Transactional Memory - Background

- The multi-core revolution is here.
  - ▶ Opportunities and challenges
- How to handle access to shared data?
  - ▶ Traditional approaches using Locks, Monitors, ...
- Transactional Memory (TM) as an alternative synchronization abstraction.
  - ▶ Simple, composable, ...
- A transaction consists of a sequence of read and write operations to a set of shared system resources.

# Transactional Memory - Background

- The multi-core revolution is here.
  - ▶ Opportunities and challenges
- How to handle access to shared data?
  - ▶ Traditional approaches using Locks, Monitors, ...
- Transactional Memory (TM) as an alternative synchronization abstraction.
  - ▶ Simple, composable, ...
- A transaction consists of a sequence of read and write operations to a set of shared system resources.
- Hardware, Software, and Hybrid TMs.
  - ▶ In this talk, we focus on **progress** in Software TM (STM) systems

# STM systems

- In STM systems, progress is ensure through some contention management policy.

# STM systems

- In STM systems, progress is ensured through some contention management policy.
- If transactions  $T$  and  $T'$  conflict, one has to abort; the aborted transaction then retries again until it eventually commits.

# STM systems

- In STM systems, progress is ensured through some contention management policy.
- If transactions  $T$  and  $T'$  conflict, one has to abort; the aborted transaction then retries again until it eventually commits.
- Each transaction consults with the contention manager (CM) for which choice to make.

# STM systems

- In STM systems, progress is ensure through some contention management policy.
- If transactions  $T$  and  $T'$  conflict, one has to abort; the aborted transaction then retries again until it eventually commits.
- Each transaction consults with the contention manager (CM) for which choice to make.
- Performance of a CM is generally measured by competitive ratio:

$$\text{competitive ratio} = \frac{\text{makespan of my CM}}{\text{makespan of optimal CM}}$$

# STM systems

- In STM systems, progress is ensure through some contention management policy.
- If transactions  $T$  and  $T'$  conflict, one has to abort; the aborted transaction then retries again until it eventually commits.
- Each transaction consults with the contention manager (CM) for which choice to make.
- Performance of a CM is generally measured by competitive ratio:  
$$\text{competitive ratio} = \frac{\text{makespan of my CM}}{\text{makespan of optimal CM}}$$
- Makespan primarily depends on the TM workload.
  - ▶ Arrival times, execution time durations, release times, read/write sets

# STM systems

- In STM systems, progress is ensured through some contention management policy.
- If transactions  $T$  and  $T'$  conflict, one has to abort; the aborted transaction then retries again until it eventually commits.
- Each transaction consults with the contention manager (CM) for which choice to make.
- Performance of a CM is generally measured by competitive ratio:  
$$\text{competitive ratio} = \frac{\text{makespan of my CM}}{\text{makespan of optimal CM}}$$
- Makespan primarily depends on the TM workload.
  - ▶ Arrival times, execution time durations, release times, read/write sets
- How to schedule transactions such that it reduces the makespan?

## Related Work

- Mostly empirical evaluation.

## Related Work

- Mostly empirical evaluation.
- Theoretical results on transactional contention managers
  - ▶ Guerraoui et al., PODC'05
    - ★ *Greedy*, Competitive ratio =  $O(s^2)$  ( $s$  is the number of shared resources).
  - ▶ Attiya et al., PODC'06
    - ★ Competitive ratio of *Greedy* is improved to  $O(s)$ .
  - ▶ Schneider and Wattenhofer, ISAAC'09
    - ★ *RandomizedRounds*, Competitive ratio =  $O(C \cdot \log n)$  ( $C$  is the maximum number of conflicting transactions for  $n$  transactions).
  - ▶ Attiya and Milani, OPODIS'09
    - ★ *Bimodal* scheduler, Competitive ratio =  $O(s)$  (for bimodal workloads).
  - ▶ Sharma et al., DISC'10
    - ★ Three new algorithms with competitive ratio very close to  $O(s)$  (for execution window model).

## Related Work

- Mostly empirical evaluation.
- Theoretical results on transactional contention managers
  - ▶ Guerraoui et al., PODC'05
    - ★ *Greedy*, Competitive ratio =  $O(s^2)$  ( $s$  is the number of shared resources).
  - ▶ Attiya et al., PODC'06
    - ★ Competitive ratio of *Greedy* is improved to  $O(s)$ .
  - ▶ Schneider and Wattenhofer, ISAAC'09
    - ★ *RandomizedRounds*, Competitive ratio =  $O(C \cdot \log n)$  ( $C$  is the maximum number of conflicting transactions for  $n$  transactions).
  - ▶ Attiya and Milani, OPODIS'09
    - ★ *Bimodal* scheduler, Competitive ratio =  $O(s)$  (for bimodal workloads).
  - ▶ Sharma et al., DISC'10
    - ★ Three new algorithms with competitive ratio very close to  $O(s)$  (for execution window model).
- Theoretical results on transactional schedulers
  - ▶ *ATS* [Yoo and Lee, SPAA'08], *Steal-On-Abort* [Ansari et al., HiPEAC'09], *Shrink* [Dragojević et al., PODC'09], etc.
    - ★ All are at least  $O(n)$ -competitive.

# Contributions

- We address the question “*Is there better than  $O(s)$ -competitive algorithm exists?*” for transaction contention management, under certain assumptions.
- Balanced transactional memory workloads.
- Two polynomial time contention management algorithms that achieve competitive ratio very close to  $O(\sqrt{s})$  in balanced workloads.
  - ▶ *Clairvoyant* algorithm - Competitive ratio =  $O(\sqrt{s})$ .
  - ▶ *Non-Clairvoyant* algorithm - Competitive ratio =  $O(\sqrt{s} \cdot \log n)$  w.h.p.
- Lower bound of  $O(\sqrt{s})$  for the transaction scheduling problem.

# Outline of the Talk

- 1 Balanced TM Workloads
- 2 Clairvoyant Algorithm
- 3 Non-Clairvoyant Algorithm
- 4 Hardness of Balanced Transaction Scheduling
- 5 Discussions

# Outline of the Talk

- 1 **Balanced TM Workloads**
- 2 Clairvoyant Algorithm
- 3 Non-Clairvoyant Algorithm
- 4 Hardness of Balanced Transaction Scheduling
- 5 Discussions

# Balanced Workloads

## Balancing ratio for a transaction $T_i$

- Let  $\mathcal{R}(T_i)$  denote the set of resources used by a transaction  $T_i$ .
- $\mathcal{R}(T_i) = \mathcal{R}_w(T_i) \cup \mathcal{R}_r(T_i)$ , where  $\mathcal{R}_w(T_i)$  are the resources which are to be written and  $\mathcal{R}_r(T_i)$  are the resources to be read by  $T_i$ .

# Balanced Workloads

## Balancing ratio for a transaction $T_i$

- Let  $\mathcal{R}(T_i)$  denote the set of resources used by a transaction  $T_i$ .
- $\mathcal{R}(T_i) = \mathcal{R}_w(T_i) \cup \mathcal{R}_r(T_i)$ , where  $\mathcal{R}_w(T_i)$  are the resources which are to be written and  $\mathcal{R}_r(T_i)$  are the resources to be read by  $T_i$ .
- The *balancing ratio*  $\beta(T_i) = \frac{|\mathcal{R}_w(T_i)|}{|\mathcal{R}(T_i)|}$ , i.e., the ratio of number of writes versus the total number of resources it accesses.
- $\frac{1}{s} \leq \beta(T_i) \leq 1$ .
- For *read-only* transactions,  $|\mathcal{R}_w(T_i)| = 0$ , while  $|\mathcal{R}_r(T_i)| = 0$  for *write-only* transactions.

# Balanced Workloads

## Balancing ratio for a transaction $T_i$

- Let  $\mathcal{R}(T_i)$  denote the set of resources used by a transaction  $T_i$ .
- $\mathcal{R}(T_i) = \mathcal{R}_w(T_i) \cup \mathcal{R}_r(T_i)$ , where  $\mathcal{R}_w(T_i)$  are the resources which are to be written and  $\mathcal{R}_r(T_i)$  are the resources to be read by  $T_i$ .
- The *balancing ratio*  $\beta(T_i) = \frac{|\mathcal{R}_w(T_i)|}{|\mathcal{R}(T_i)|}$ , i.e., the ratio of number of writes versus the total number of resources it accesses.
- $\frac{1}{s} \leq \beta(T_i) \leq 1$ .
- For *read-only* transactions,  $|\mathcal{R}_w(T_i)| = 0$ , while  $|\mathcal{R}_r(T_i)| = 0$  for *write-only* transactions.

## Balanced Workloads

A workload (set of transactions)  $\mathcal{T}$  is called *balanced* if balancing ratio  $\beta = \Theta(1)$ .

# Outline of the Talk

- 1 Balanced TM Workloads
- 2 Clairvoyant Algorithm
- 3 Non-Clairvoyant Algorithm
- 4 Hardness of Balanced Transaction Scheduling
- 5 Discussions

# Clairvoyant Algorithm (1/3)

- Assumptions:
  - ▶ We assume each transaction  $T_i$  knows execution time duration ( $\tau$ ) and number of resources it accesses ( $\mathcal{R}(T_i)$ ).
  - ▶ For simplicity, assume equal execution time duration  $\tau$  for all transactions.
- The algorithm is clairvoyant in the sense it uses explicitly the knowledge of conflict relations at each time step  $t$  to resolve conflicts.
- It is greedy, obeys *pending commit* property, and computes schedule in polynomial time.

# Clairvoyant Algorithm (2/3)

## Algorithm

- Divide writing transactions into  $\kappa = \lceil \log s \rceil + 1$  groups  $A_0, A_1, \dots, A_{\kappa-1}$  where  $A_i$  contains transactions that access resources in range  $[2^i, 2^{i+1} - 1]$ .

# Clairvoyant Algorithm (2/3)

## Algorithm

- Divide writing transactions into  $\kappa = \lceil \log s \rceil + 1$  groups  $A_0, A_1, \dots, A_{\kappa-1}$  where  $A_i$  contains transactions that access resources in range  $[2^i, 2^{i+1} - 1]$ .
- Order the groups such that  $A_i < A_k$  if  $i < k$ .

# Clairvoyant Algorithm (2/3)

## Algorithm

- Divide writing transactions into  $\kappa = \lceil \log s \rceil + 1$  groups  $A_0, A_1, \dots, A_{\kappa-1}$  where  $A_i$  contains transactions that access resources in range  $[2^i, 2^{i+1} - 1]$ .
- Order the groups such that  $A_i < A_k$  if  $i < k$ .
- Resolve the conflicts by assigning priority among transactions.

# Clairvoyant Algorithm (3/3)

## Conflict Resolution Routine

- Let  $\mathcal{T}_t$  : set of transactions that are pending at any time  $t$ .
- Let  $S_t$  : set of transactions started before  $t$  among pending transactions  $\mathcal{T}_t$  and are from lowest order group.
- Let  $I_t$  : maximal independent set in the conflict graph  $G$  at time  $t$  after removing conflicting transactions from  $\mathcal{T}_t$  which conflict with  $S_t$ .

# Clairvoyant Algorithm (3/3)

## Conflict Resolution Routine

- Let  $\mathcal{T}_t$  : set of transactions that are pending at any time  $t$ .
- Let  $S_t$  : set of transactions started before  $t$  among pending transactions  $\mathcal{T}_t$  and are from lowest order group.
- Let  $I_t$  : maximal independent set in the conflict graph  $G$  at time  $t$  after removing conflicting transactions from  $\mathcal{T}_t$  which conflict with  $S_t$ .
- High priority transactions:  $\Pi_t^h \leftarrow S_t \cup I_t$ .
- Low priority transactions:  $\Pi_t^l \leftarrow \mathcal{T}_t \setminus \Pi_t^h$ .

# Clairvoyant Algorithm (3/3)

## Conflict Resolution Routine

- Let  $\mathcal{T}_t$  : set of transactions that are pending at any time  $t$ .
- Let  $S_t$  : set of transactions started before  $t$  among pending transactions  $\mathcal{T}_t$  and are from lowest order group.
- Let  $I_t$  : maximal independent set in the conflict graph  $G$  at time  $t$  after removing conflicting transactions from  $\mathcal{T}_t$  which conflict with  $S_t$ .
  
- High priority transactions:  $\Pi_t^h \leftarrow S_t \cup I_t$ .
- Low priority transactions:  $\Pi_t^l \leftarrow \mathcal{T}_t \setminus \Pi_t^h$ .
  
- **On conflict** of transaction  $T_u$  with  $T_v$  at any time  $t$ 
  - ▶ **if**  $T_u \in \Pi_t^h \wedge T_v \in \Pi_t^l$ 
    - ★  $T_u$  aborts  $T_v$ ,
  - ▶ **otherwise**
    - $T_v$  aborts  $T_u$ .

## Analysis (1/4)

**Observation 1:** For the transactions in group  $A_i$ , the competitive ratio of Clairvoyant is  $CR_{Clairvoyant}(A_i) \leq \lambda_{\max} + 1$ , where  $\lambda_{\max} = 2^{i+1} - 1$ .

## Analysis (1/4)

**Observation 1:** For the transactions in group  $A_i$ , the competitive ratio of Clairvoyant is  $CR_{Clairvoyant}(A_i) \leq \lambda_{\max} + 1$ , where  $\lambda_{\max} = 2^{i+1} - 1$ .

### Intuition

- Let  $\gamma_i(R_v)$  denote the number of transactions in group  $A_i$  that write  $R_v$ . Let  $\gamma_{\max} = \max_{v \in [1,s]} \gamma_i(R_v)$ .

## Analysis (1/4)

**Observation 1:** For the transactions in group  $A_i$ , the competitive ratio of Clairvoyant is  $CR_{Clairvoyant}(A_i) \leq \lambda_{\max} + 1$ , where  $\lambda_{\max} = 2^{i+1} - 1$ .

### Intuition

- Let  $\gamma_i(R_v)$  denote the number of transactions in group  $A_i$  that write  $R_v$ . Let  $\gamma_{\max} = \max_{v \in [1,s]} \gamma_i(R_v)$ .
- A transaction  $T \in A_i$  conflicts with at most  $\lambda_{\max} \cdot \gamma_{\max}$  other transactions in the group. Hence,

$$makespan_{Clairvoyant}(A_i) \leq (\lambda_{\max} \cdot \gamma_{\max} + 1) \cdot \tau.$$

## Analysis (1/4)

**Observation 1:** For the transactions in group  $A_i$ , the competitive ratio of Clairvoyant is  $CR_{Clairvoyant}(A_i) \leq \lambda_{\max} + 1$ , where  $\lambda_{\max} = 2^{i+1} - 1$ .

### Intuition

- Let  $\gamma_i(R_v)$  denote the number of transactions in group  $A_i$  that write  $R_v$ . Let  $\gamma_{\max} = \max_{v \in [1,s]} \gamma_i(R_v)$ .
- A transaction  $T \in A_i$  conflicts with at most  $\lambda_{\max} \cdot \gamma_{\max}$  other transactions in the group. Hence,

$$makespan_{Clairvoyant}(A_i) \leq (\lambda_{\max} \cdot \gamma_{\max} + 1) \cdot \tau.$$

- There is a resource that is accessed by at least  $\gamma_{\max}$  transactions of  $A_i$  for write. Therefore,

$$makespan_{opt}(A_i) \geq \gamma_{\max} \cdot \tau.$$

# Analysis (1/4)

**Observation 1:** For the transactions in group  $A_i$ , the competitive ratio of Clairvoyant is  $CR_{Clairvoyant}(A_i) \leq \lambda_{\max} + 1$ , where  $\lambda_{\max} = 2^{i+1} - 1$ .

## Intuition

- Let  $\gamma_i(R_v)$  denote the number of transactions in group  $A_i$  that write  $R_v$ . Let  $\gamma_{\max} = \max_{v \in [1, s]} \gamma_i(R_v)$ .
- A transaction  $T \in A_i$  conflicts with at most  $\lambda_{\max} \cdot \gamma_{\max}$  other transactions in the group. Hence,

$$makespan_{Clairvoyant}(A_i) \leq (\lambda_{\max} \cdot \gamma_{\max} + 1) \cdot \tau.$$

- There is a resource that is accessed by at least  $\gamma_{\max}$  transactions of  $A_i$  for write. Therefore,

$$makespan_{opt}(A_i) \geq \gamma_{\max} \cdot \tau.$$

- $CR_{Clairvoyant}(A_i) = \frac{makespan_{Clairvoyant}(A_i)}{makespan_{opt}(A_i)} \leq \lambda_{\max} + 1$ .

## Analysis (2/4)

**Observation 2:** For the transactions in group  $A_i$ , the competitive ratio of Clairvoyant is  $CR_{\text{Clairvoyant}}(A_i) \leq 2 \cdot \frac{s/\beta}{\lambda_{\max}}$ .

## Analysis (2/4)

**Observation 2:** For the transactions in group  $A_i$ , the competitive ratio of Clairvoyant is  $CR_{Clairvoyant}(A_i) \leq 2 \cdot \frac{s/\beta}{\lambda_{\max}}$ .

### Intuition

- Since the algorithm satisfies pending commit property, if  $T \in A_i$  does not commit, then some  $T' \in A_i$  must commit. Therefore,

$$makespan_{Clairvoyant}(A_i) \leq |A_i| \cdot \tau.$$

## Analysis (2/4)

**Observation 2:** For the transactions in group  $A_i$ , the competitive ratio of Clairvoyant is  $CR_{Clairvoyant}(A_i) \leq 2 \cdot \frac{s/\beta}{\lambda_{\max}}$ .

### Intuition

- Since the algorithm satisfies pending commit property, if  $T \in A_i$  does not commit, then some  $T' \in A_i$  must commit. Therefore,

$$makespan_{Clairvoyant}(A_i) \leq |A_i| \cdot \tau.$$

- $\mathcal{R}_w(T) \geq \beta \cdot \lambda_{\min} \geq \beta \cdot \lambda_{\max}/2$ . By the pigeonhole principle, there will be a resource  $R$  which is accessed by at least  $(|A_i| \cdot \beta \cdot \lambda_{\min})/s$  transactions for write. All these transactions need to be serialized due to conflicts with each other in accessing  $R$ . Hence,

$$makespan_{opt}(A_i) \geq \frac{|A_i| \cdot \beta \cdot \lambda_{\max}}{2s} \cdot \tau.$$

## Analysis (2/4)

**Observation 2:** For the transactions in group  $A_i$ , the competitive ratio of Clairvoyant is  $CR_{Clairvoyant}(A_i) \leq 2 \cdot \frac{s/\beta}{\lambda_{\max}}$ .

### Intuition

- Since the algorithm satisfies pending commit property, if  $T \in A_i$  does not commit, then some  $T' \in A_i$  must commit. Therefore,

$$makespan_{Clairvoyant}(A_i) \leq |A_i| \cdot \tau.$$

- $\mathcal{R}_w(T) \geq \beta \cdot \lambda_{\min} \geq \beta \cdot \lambda_{\max}/2$ . By the pigeonhole principle, there will be a resource  $R$  which is accessed by at least  $(|A_i| \cdot \beta \cdot \lambda_{\min})/s$  transactions for write. All these transactions need to be serialized due to conflicts with each other in accessing  $R$ . Hence,

$$makespan_{opt}(A_i) \geq \frac{|A_i| \cdot \beta \cdot \lambda_{\max}}{2s} \cdot \tau.$$

- $CR_{Clairvoyant}(A_i) = \frac{makespan_{Clairvoyant}(A_i)}{makespan_{opt}(A_i)} \leq 2 \cdot \frac{s/\beta}{\lambda_{\max}}$ .

## Analysis (3/4)

**Corollary 1 (From Observations 1 and 2):** For the transactions in group  $A_i$ , the competitive ratio of Clairvoyant is  $CR_{Clairvoyant}(A_i) \leq 2 \cdot \min \left\{ \lambda_{\max}, \frac{s/\beta}{\lambda_{\max}} \right\}$ .

## Analysis (3/4)

**Corollary 1 (From Observations 1 and 2):** For the transactions in group  $A_i$ , the competitive ratio of Clairvoyant is  $CR_{Clairvoyant}(A_i) \leq 2 \cdot \min \left\{ \lambda_{\max}, \frac{s/\beta}{\lambda_{\max}} \right\}$ .

**Observation 3:** For the transactions in group  $A_i$ , the competitive ratio of Clairvoyant is bounded by  $CR_{Clairvoyant}(A_i) \leq 16 \cdot \sqrt{\frac{s}{\beta}}$ .

## Analysis (3/4)

**Corollary 1 (From Observations 1 and 2):** For the transactions in group  $A_i$ , the competitive ratio of Clairvoyant is  $CR_{Clairvoyant}(A_i) \leq 2 \cdot \min \left\{ \lambda_{\max}, \frac{s/\beta}{\lambda_{\max}} \right\}$ .

**Observation 3:** For the transactions in group  $A_i$ , the competitive ratio of Clairvoyant is bounded by  $CR_{Clairvoyant}(A_i) \leq 16 \cdot \sqrt{\frac{s}{\beta}}$ .

### Intuition

- Since  $\lambda_{\max} = (2^{i+1} - 1)$ , from Corollary 1

$$CR_{Clairvoyant}(A_i) \leq 2 \cdot \min \left\{ 2^{i+1} - 1, \frac{s/\beta}{2^{i+1}-1} \right\} \leq 4 \cdot \min \left\{ 2^i, \frac{s/\beta}{2^i} \right\}.$$

- Bitonic sequence with peak at  $i = \frac{\log(s/\beta)}{2}$ .
- Group  $A_i$  contains  $\kappa$  subgroups and at worst-case transactions commit according to the order from lowest to highest order, i.e.,

$$CR_{Clairvoyant}(A_i) \leq \sum_{i=0}^{\kappa-1} CR_{Clairvoyant}(A_i) = 16 \cdot \sqrt{\frac{s}{\beta}}.$$

## Analysis (4/4)

### Competitive Ratio of Clairvoyant

For the set of transactions  $\mathcal{T}$ , Algorithm Clairvoyant has competitive ratio

$$CR_{\text{Clairvoyant}}(\mathcal{T}) = O\left(\sqrt{\frac{s}{\beta}}\right).$$

## Analysis (4/4)

### Competitive Ratio of Clairvoyant

For the set of transactions  $\mathcal{T}$ , Algorithm Clairvoyant has competitive ratio

$$CR_{\text{Clairvoyant}}(\mathcal{T}) = O\left(\sqrt{\frac{s}{\beta}}\right).$$

### Balanced Workload

For balanced workload  $\mathcal{T}$  ( $\beta = \Theta(1)$ ), Algorithm Clairvoyant has competitive ratio

$$CR_{\text{Clairvoyant}}(\mathcal{T}) = O(\sqrt{s}).$$

# Outline of the Talk

- 1 Balanced TM Workloads
- 2 Clairvoyant Algorithm
- 3 Non-Clairvoyant Algorithm**
- 4 Hardness of Balanced Transaction Scheduling
- 5 Discussions

# Non-Clairvoyant Algorithm (1/3)

- Again, we assume equi-length transactions with known execution time ( $\tau$ ) and total number of resources each accesses ( $\mathcal{R}(T_i)$ ).
- This algorithm is similar to Clairvoyant but the difference is that conflicts are resolved using random priorities which are determined with out explicit knowledge of the conflict graph.
  - ▶ Called non-clairvoyant in this sense.
- The algorithm is also greedy, obeys pending commit property, and computes the schedule in polynomial time.

## Non-Clairvoyant Algorithm (2/3)

### Algorithm

- Similar to *Clairvoyant*, organize transactions into groups.
- Order the groups such that  $A_i < A_k$  if  $i < k$ .
- Lower order groups have always higher priority and then higher order groups.
- Let  $\hat{A}_t$  denote the lowest order group, transactions at  $\hat{A}_t$  win at the time of conflict.
- When transactions at same group conflict, use priorities based on some random number to resolve conflicts.

# Non-Clairvoyant Algorithm (3/3)

## Conflict Resolution Routine

- When a transaction  $T$  starts execution, it chooses uniformly at random a discrete number  $r(T) \in [1, n]$ .

# Non-Clairvoyant Algorithm (3/3)

## Conflict Resolution Routine

- When a transaction  $T$  starts execution, it chooses uniformly at random a discrete number  $r(T) \in [1, n]$ .
- In case of conflict of a transaction  $T_w$  with another transaction  $T_x$  in the same group
  - ▶ **If**  $r(T_x) < r(T_w)$ 
    - ★  $T_x$  aborts  $T_w$ ,
  - ▶ **otherwise**
    - ★  $T_w$  aborts  $T_x$ .

# Non-Clairvoyant Algorithm (3/3)

## Conflict Resolution Routine

- When a transaction  $T$  starts execution, it chooses uniformly at random a discrete number  $r(T) \in [1, n]$ .
- In case of conflict of a transaction  $T_w$  with another transaction  $T_x$  in the same group
  - ▶ **If**  $r(T_x) < r(T_w)$ 
    - ★  $T_x$  aborts  $T_w$ ,
  - ▶ **otherwise**
    - ★  $T_w$  aborts  $T_x$ .
- After every abort, the restarted transaction chooses again a new discrete number at random from  $[1, n]$ .

## Analysis (1/3)

**Observation 4:** Using random priorities, the time span a transaction  $T$  needs from when it is issued until commit is  $16 \cdot e \cdot (d_T + 1) \cdot \tau \cdot \ln n$  w.h.p., where  $d_T$  is the number of transactions conflicting with  $T$ .

## Analysis (1/3)

**Observation 4:** Using random priorities, the time span a transaction  $T$  needs from when it is issued until commit is  $16 \cdot e \cdot (d_T + 1) \cdot \tau \cdot \ln n$  w.h.p., where  $d_T$  is the number of transactions conflicting with  $T$ .

### Intuition

- The chance that  $T$ 's priority  $r(T)$  is different than among its neighbors is at least  $\frac{1}{e}$ .

## Analysis (1/3)

**Observation 4:** Using random priorities, the time span a transaction  $T$  needs from when it is issued until commit is  $16 \cdot e \cdot (d_T + 1) \cdot \tau \cdot \ln n$  w.h.p., where  $d_T$  is the number of transactions conflicting with  $T$ .

### Intuition

- The chance that  $T$ 's priority  $r(T)$  is different than among its neighbors is at least  $\frac{1}{e}$ .
- The chance that  $T$ 's priority  $r(T)$  is smallest and different than among its neighbors is at least  $\frac{1}{e \cdot (d_T + 1)}$ .

# Analysis (1/3)

**Observation 4:** Using random priorities, the time span a transaction  $T$  needs from when it is issued until commit is  $16 \cdot e \cdot (d_T + 1) \cdot \tau \cdot \ln n$  w.h.p., where  $d_T$  is the number of transactions conflicting with  $T$ .

## Intuition

- The chance that  $T$ 's priority  $r(T)$  is different than among its neighbors is at least  $\frac{1}{e}$ .
- The chance that  $T$ 's priority  $r(T)$  is smallest and different than among its neighbors is at least  $\frac{1}{e \cdot (d_T + 1)}$ .
- If we conduct  $16 \cdot e \cdot (d_T + 1) \cdot \ln n$  trials with success probability  $\frac{1}{e \cdot (d_T + 1)}$ , then the number of successes  $Z$  is less than  $8 \cdot \ln n$  becomes:

$$\Pr(Z < 8 \cdot \ln n) < e^{-2 \cdot \ln n} = 1/n^2,$$

using Chernoff bound.

# Analysis (1/3)

**Observation 4:** Using random priorities, the time span a transaction  $T$  needs from when it is issued until commit is  $16 \cdot e \cdot (d_T + 1) \cdot \tau \cdot \ln n$  w.h.p., where  $d_T$  is the number of transactions conflicting with  $T$ .

## Intuition

- The chance that  $T$ 's priority  $r(T)$  is different than among its neighbors is at least  $\frac{1}{e}$ .
- The chance that  $T$ 's priority  $r(T)$  is smallest and different than among its neighbors is at least  $\frac{1}{e \cdot (d_T + 1)}$ .
- If we conduct  $16 \cdot e \cdot (d_T + 1) \cdot \ln n$  trials with success probability  $\frac{1}{e \cdot (d_T + 1)}$ , then the number of successes  $Z$  is less than  $8 \cdot \ln n$  becomes:

$$\Pr(Z < 8 \cdot \ln n) < e^{-2 \cdot \ln n} = 1/n^2,$$

using Chernoff bound.

## Analysis (2/3)

**Observation 5:** For transactions in group  $A_i$ , the competitive ratio of Non-Clairvoyant is  $CR_{\text{Non-Clairvoyant}}(A_i) \leq 32 \cdot e \cdot \lambda_{\max} \cdot \ln n$  w.h.p.

## Analysis (2/3)

**Observation 5:** For transactions in group  $A_i$ , the competitive ratio of Non-Clairvoyant is  $CR_{\text{Non-Clairvoyant}}(A_i) \leq 32 \cdot e \cdot \lambda_{\max} \cdot \ln n$  w.h.p.

**Observation 6:** For transactions in group  $A_i$ , the competitive ratio of Non-Clairvoyant is  $CR_{\text{Non-Clairvoyant}}(A_i) \leq 32 \cdot e \cdot \frac{s/\beta}{\lambda_{\max}} \cdot \ln n$  w.h.p.

## Analysis (2/3)

**Observation 5:** For transactions in group  $A_i$ , the competitive ratio of Non-Clairvoyant is  $CR_{\text{Non-Clairvoyant}}(A_i) \leq 32 \cdot e \cdot \lambda_{\max} \cdot \ln n$  w.h.p.

**Observation 6:** For transactions in group  $A_i$ , the competitive ratio of Non-Clairvoyant is  $CR_{\text{Non-Clairvoyant}}(A_i) \leq 32 \cdot e \cdot \frac{s/\beta}{\lambda_{\max}} \cdot \ln n$  w.h.p.

**Corollary 2 (From Observations 5 and 6):** For the transactions in group  $A_i$ , the competitive ratio of Non-Clairvoyant is

$$CR_{\text{Non-Clairvoyant}}(A_i) \leq 32 \cdot e \cdot \min \left\{ \lambda_{\max}, \frac{s/\beta}{\lambda_{\max}} \right\} \cdot \ln n \text{ w.h.p.}$$

## Analysis (3/3)

**Observation 7:** For the transactions in group  $A_i$ , then the competitive ratio of Non-Clairvoyant is bounded by  $CR_{Non-Clairvoyant}(A_i) \leq 256 \cdot e \cdot \sqrt{\frac{s}{\beta}} \cdot \ln n$  w.h.p.

## Analysis (3/3)

**Observation 7:** For the transactions in group  $A_i$ , then the competitive ratio of Non-Clairvoyant is bounded by  $CR_{Non-Clairvoyant}(A_i) \leq 256 \cdot e \cdot \sqrt{\frac{s}{\beta}} \cdot \ln n$  w.h.p.

### Competitive Ratio of Non-Clairvoyant

For a set of transactions  $\mathcal{T}$ , Algorithm Non-Clairvoyant has competitive ratio  $CR_{Non-Clairvoyant}(\mathcal{T}) = O\left(\sqrt{\frac{s}{\beta}} \cdot \log n\right)$  w.h.p.

## Analysis (3/3)

**Observation 7:** For the transactions in group  $A_i$ , then the competitive ratio of Non-Clairvoyant is bounded by  $CR_{Non-Clairvoyant}(A_i) \leq 256 \cdot e \cdot \sqrt{\frac{s}{\beta}} \cdot \ln n$  w.h.p.

### Competitive Ratio of Non-Clairvoyant

For a set of transactions  $\mathcal{T}$ , Algorithm Non-Clairvoyant has competitive ratio  $CR_{Non-Clairvoyant}(\mathcal{T}) = O\left(\sqrt{\frac{s}{\beta}} \cdot \log n\right)$  w.h.p.

### Balanced Workload

For balanced workload  $\mathcal{T}$  ( $\beta = \Theta(1)$ ), Algorithm Non-Clairvoyant has competitive ratio  $CR_{Non-Clairvoyant}(\mathcal{T}) = O(\sqrt{s} \cdot \log n)$  w.h.p.

# Outline of the Talk

- 1 Balanced TM Workloads
- 2 Clairvoyant Algorithm
- 3 Non-Clairvoyant Algorithm
- 4 **Hardness of Balanced Transaction Scheduling**
- 5 Discussions

# Hardness of Balanced Transaction Scheduling

## We prove the following theorem

Unless  $\text{NP} \subseteq \text{ZPP}$ , we cannot obtain a polynomial time transaction scheduling algorithm such that for every input instance with  $\beta = 1$  of the TRANSACTION SCHEDULING problem the algorithm achieves competitive ratio smaller than  $O((\sqrt{s})^{1-\epsilon})$  for any constant  $\epsilon > 0$ .

# Hardness of Balanced Transaction Scheduling

## We prove the following theorem

Unless  $\text{NP} \subseteq \text{ZPP}$ , we cannot obtain a polynomial time transaction scheduling algorithm such that for every input instance with  $\beta = 1$  of the TRANSACTION SCHEDULING problem the algorithm achieves competitive ratio smaller than  $O((\sqrt{s})^{1-\epsilon})$  for any constant  $\epsilon > 0$ .

## Proof intuition

- Reduce the NP-Complete graph coloring problem, VERTEX COLORING, to the transaction scheduling problem, TRANSACTION SCHEDULING.
- Use following result [Feige and Kilian, CCC'96]
  - ▶ No better than  $O(n^{1-\epsilon})$  approximation exists for VERTEX COLORING for any constant  $\epsilon > 0$ , unless  $\text{NP} \subseteq \text{ZPP}$ .

## Proof (1/4)

- A VERTEX COLORING problem instance asks whether a given graph  $G$  is  $k$ -colorable.
- A valid  $k$ -coloring is an assignment of integers  $\{1, 2, \dots, k\}$  to the vertices of  $G$  such that neighbors receive different integers.
- An algorithm approximates chromatic number  $\chi(G)$  with approximation ratio  $q(G)$  if it outputs  $u(G)$  such that  $\chi(G) \leq u(G)$  and  $u(G)/\chi(G) \leq q(G)$ .
- A TRANSACTION SCHEDULING problem asks whether a set of transactions  $\mathcal{T}$  with a set of resources  $\mathcal{R}$  has makespan  $k$  time steps.

## Proof (2/4)

### Polynomial time reduction to TRANSACTION SCHEDULING

- Consider an input graph  $G = (V, E)$  of the VERTEX COLORING problem, where  $|V| = n$  and  $|E| = s$ .

## Proof (2/4)

### Polynomial time reduction to TRANSACTION SCHEDULING

- Consider an input graph  $G = (V, E)$  of the VERTEX COLORING problem, where  $|V| = n$  and  $|E| = s$ .
- Construct a set of transactions  $\mathcal{T}$  such that
  - ▶ For each  $v \in V$ , there is a resp. transaction  $T_v \in \mathcal{T}$ ;  $|\mathcal{T}| = |V| = n$ .
  - ▶ For each  $e \in E$ , there is a resp. resource  $R_e \in \mathcal{R}$ ;  $|\mathcal{R}| = |E| = s$ .
  - ▶ Assume all transaction operations are writes, i.e.,  $\beta = 1$ .

## Proof (2/4)

### Polynomial time reduction to TRANSACTION SCHEDULING

- Consider an input graph  $G = (V, E)$  of the VERTEX COLORING problem, where  $|V| = n$  and  $|E| = s$ .
- Construct a set of transactions  $\mathcal{T}$  such that
  - ▶ For each  $v \in V$ , there is a resp. transaction  $T_v \in \mathcal{T}$ ;  $|\mathcal{T}| = |V| = n$ .
  - ▶ For each  $e \in E$ , there is a resp. resource  $R_e \in \mathcal{R}$ ;  $|\mathcal{R}| = |E| = s$ .
  - ▶ Assume all transaction operations are writes, i.e.,  $\beta = 1$ .
- Let  $G'$  be the conflict graph for the set of transactions  $\mathcal{T}$ 
  - ▶  $G'$  is isomorphic to  $G$ .
  - ▶ Assuming  $G$  has a valid  $k$ -coloring, it implies a schedule with makespan  $k$  for  $\mathcal{T}$ .
  - ▶ Symmetrically, a schedule with makespan  $k$  for  $\mathcal{T}$  implies a valid  $k$ -coloring in  $G$ .

## Proof (3/4)

### Hardness of TRANSACTION SCHEDULING problem

- Reduction is gap preserving with gap preserving parameter  $\rho = 1$ .

## Proof (3/4)

### Hardness of TRANSACTION SCHEDULING problem

- Reduction is gap preserving with gap preserving parameter  $\rho = 1$ .
- TRANSACTION SCHEDULING is in NP.
- From the reduction of the VERTEX COLORING problem, TRANSACTION SCHEDULING is NP-Complete.

# Proof (4/4)

## Proof Analogy

- From the above reduction, an approximation ratio  $q(G)$  of the VERTEX COLORING problem implies the existence of a scheduling algorithm  $\mathcal{A}$  with competitive ratio  $CR_{\mathcal{A}}(\mathcal{T}) = q(G)$  of the respective TRANSACTION SCHEDULING problem instance, and vice-versa.

# Proof (4/4)

## Proof Analogy

- From the above reduction, an approximation ratio  $q(G)$  of the VERTEX COLORING problem implies the existence of a scheduling algorithm  $\mathcal{A}$  with competitive ratio  $CR_{\mathcal{A}}(\mathcal{T}) = q(G)$  of the respective TRANSACTION SCHEDULING problem instance, and vice-versa.
- Since  $s = |\mathcal{R}| = |E| \leq n^2$ , an  $(\sqrt{s})^{1-\epsilon}$  competitive ratio of  $\mathcal{A}$  implies at most an  $(n)^{1-\epsilon}$  approximation ratio of VERTEX COLORING problem.
- Using [Feige and Kilian, CCC'96] result, we obtain the theorem.

# Proof (4/4)

## Proof Analogy

- From the above reduction, an approximation ratio  $q(G)$  of the VERTEX COLORING problem implies the existence of a scheduling algorithm  $\mathcal{A}$  with competitive ratio  $CR_{\mathcal{A}}(\mathcal{T}) = q(G)$  of the respective TRANSACTION SCHEDULING problem instance, and vice-versa.
- Since  $s = |\mathcal{R}| = |E| \leq n^2$ , an  $(\sqrt{s})^{1-\epsilon}$  competitive ratio of  $\mathcal{A}$  implies at most an  $(n)^{1-\epsilon}$  approximation ratio of VERTEX COLORING problem.
- Using [Feige and Kilian, CCC'96] result, we obtain the theorem.
- The  $O(\sqrt{s})$  bound of Algorithm Clairvoyant, for  $\beta = \Theta(1)$ , is tight.

# Outline of the Talk

- 1 Balanced TM Workloads
- 2 Clairvoyant Algorithm
- 3 Non-Clairvoyant Algorithm
- 4 Hardness of Balanced Transaction Scheduling
- 5 Discussions

# Variable Length Transactions

- Groups and subgroups according to execution time duration and number of resources (Results on the paper).
- Number of groups  $\ell = \lceil \log(\frac{\tau_{\max}}{\tau_{\min}}) \rceil + 1$ , where  $\tau_{\max}$  is the longest and  $\tau_{\min}$  is the shortest transaction duration.
- Each group is again divided into  $\kappa$  subgroups (as described in this talk).

# Variable Length Transactions

- Groups and subgroups according to execution time duration and number of resources (Results on the paper).
- Number of groups  $\ell = \lceil \log(\frac{\tau_{\max}}{\tau_{\min}}) \rceil + 1$ , where  $\tau_{\max}$  is the longest and  $\tau_{\min}$  is the shortest transaction duration.
- Each group is again divided into  $\kappa$  subgroups (as described in this talk).

## Competitive Ratios

For a set  $\mathcal{T}$  of variable length transactions, Algorithm Clairvoyant has competitive ratio  $CR_{Clairvoyant}(\mathcal{T}) = O\left(\ell \cdot \sqrt{\frac{s}{\beta}}\right)$  and Algorithm Non-Clairvoyant has competitive ratio  $CR_{Non-Clairvoyant}(\mathcal{T}) = O\left(\ell \cdot \sqrt{\frac{s}{\beta}} \cdot \log n\right)$  w.h.p.

# Variable Length Transactions

- Groups and subgroups according to execution time duration and number of resources (Results on the paper).
- Number of groups  $\ell = \lceil \log(\frac{\tau_{\max}}{\tau_{\min}}) \rceil + 1$ , where  $\tau_{\max}$  is the longest and  $\tau_{\min}$  is the shortest transaction duration.
- Each group is again divided into  $\kappa$  subgroups (as described in this talk).

## Competitive Ratios

For a set  $\mathcal{T}$  of variable length transactions, Algorithm Clairvoyant has competitive ratio  $CR_{Clairvoyant}(\mathcal{T}) = O\left(\ell \cdot \sqrt{\frac{s}{\beta}}\right)$  and Algorithm Non-Clairvoyant has competitive ratio  $CR_{Non-Clairvoyant}(\mathcal{T}) = O\left(\ell \cdot \sqrt{\frac{s}{\beta}} \cdot \log n\right)$  w.h.p.

## Balanced Workload

For balanced workload  $\mathcal{T}$  ( $\beta = \Theta(1)$ ) and when  $\ell = O(1)$ , Algorithm Clairvoyant has competitive ratio  $CR_{Clairvoyant}(\mathcal{T}) = O(\sqrt{s})$  and Algorithm Non-Clairvoyant has competitive ratio  $CR_{Non-Clairvoyant}(\mathcal{T}) = O(\sqrt{s} \cdot \log n)$  w.h.p.

# Summary

- We introduced new kind of TM workloads, called balanced workloads.
- We proposed and analyzed two new randomized contention managers that exhibit competitive ratios very close to  $O(\sqrt{s})$  in balanced workloads.
- We proved the lower bound of  $O(\sqrt{s})$  for transaction scheduling problem using the polynomial time reduction from the vertex coloring problem to the transaction scheduling problem.

# Summary

- We introduced new kind of TM workloads, called balanced workloads.
- We proposed and analyzed two new randomized contention managers that exhibit competitive ratios very close to  $O(\sqrt{s})$  in balanced workloads.
- We proved the lower bound of  $O(\sqrt{s})$  for transaction scheduling problem using the polynomial time reduction from the vertex coloring problem to the transaction scheduling problem.

## Open Question

Is it possible to have a contention manager which still achieves  $O(\sqrt{s})$  or smaller than  $O(s)$  competitive ratio bound in balanced workload model after removing the assumptions that each transaction knows its execution time duration and the total number of shared resources it accesses?

Questions???