Efficient means of Achieving Composability using Object based Conflicts on Transactional Memory

Sathya Peri, Ajay Singh and Archit Somani
(sathya_p, cs15mtech01001, cs15resch01001)@iith.ac.in
CSE Department, IIT Hyderabad, India

November 1, 2018
Content

Introduction to STM

Problem with Read-Write STM

Motivation towards Object based STM (OSTM)

HT-OSTM Design

Execution Under HT-OSTM

Proof Of Correctness

Results

Conclusion
Content

Introduction to STM

Problem with Read-Write STM

Motivation towards Object based STM (OSTM)

HT-OSTM Design

Execution Under HT-OSTM

Proof Of Correctness

Results

Conclusion
Transaction

Sequence of instructions guaranteed to execute atomically.
Software Transaction Memory Systems (STMs)

Introduction

Transaction
Sequence of instructions guaranteed to execute atomically.

History
Concurrent execution of transactions.

Figure: History of an STM
Software Transaction Memory Systems (STMs) are a convenient programming interface for a programmer to access shared memory using concurrent threads without worrying about concurrency issues.
Software Transaction Memory Systems (STMs) are a convenient programming interface for a programmer to access shared memory using concurrent threads without worrying about concurrency issues.

STMs export the following methods:

- t_begin()
- t_read()
- t_write()
- tryC() and tryA().

We refer to these as Read-Write STMs (or RWSTM).
STMs Contd.
Working of STM System

Figure: Working of STM System
Figure: Working of STM System
STMs Contd..
Working of STM System

Figure: Working of STM System
STMs Contd..
Working of STM System

Figure: Working of STM System
STMs Contd..
Working of STM System

Figure: Working of STM System
Figure: Working of STM System
STM Contd.. Working of STM System

Figure: Working of STM System
Figure: Working of STM System
Figure: Working of STM System
STMs Contd..  
Working of STM System

Figure: Working of STM System
Figure: Working of STM System
Correctness Criterion: Opacity [Guerraoui and Kapałka]

- A history $H$ is opaque if there exists a serial history $S$ s.t.
  1. Operations of $H$ and $S$ are same
  2. $S$ respects real time order $\prec_H^{RT}$ and
  3. $\forall \text{trans}(T_i) \in S$ is legal.
Correctness Criterion: Opacity [Guerraoui and Kapałka]

- A history $H$ is opaque if there exists a serial history $S$ s.t.
  1. Operations of $H$ and $S$ are same
  2. $S$ respects real time order $\prec^{RT}_H$ and
  3. $\forall \text{trans}(T_i) \in S$ is legal.

$T_1 \quad r_1(x,0) \quad r_1(y,10) \quad A_1$

$T_2 \quad w_2(x,10) \quad w_2(y,10) \quad C_2$

Figure: History $H$ is not Opaque
Correctness Criterion: Opacity [Guerraoui and Kapałka]

- A history $H$ is opaque if there exists a serial history $S$ s.t.
  1. Operations of $H$ and $S$ are same
  2. $S$ respects real time order $\prec_{RT}^H$ and
  3. $\forall \text{trans}(T_i) \in S$ is legal.

Figure: History $H$ is not Opaque

Figure: Opaque History $H(T_1, T_2)$
Content

Introduction to STM

Problem with Read-Write STM

Motivation towards Object based STM (OSTM)

HT-OSTM Design

Execution Under HT-OSTM

Proof Of Correctness

Results

Conclusion
Problem with Read-Write STM

Figure: A concurrent list
Problem with Read-Write STM

Figure: A concurrent list

T1: lookup(k₅), lookup(k₈) & T₂: delete(k₇)
Problem with Read-Write STM

Figure: A concurrent list

T1: lookup\( (k_5) \), lookup\( (k_8) \) & \( T_2: \) delete\( (k_7) \)

Figure: Transaction Tree Structure
Problem with Read-Write STM

Schedule could not be accepted by STM.
Problem with Read-Write STM

Figure: Cyclic Conflicts

Schedule could not be accepted by STM.

Figure: Cycle (Not Serial)
Content

Introduction to STM

Problem with Read-Write STM

Motivation towards Object based STM (OSTM)

HT-OSTM Design

Execution Under HT-OSTM

Proof Of Correctness

Results

Conclusion
Motivation towards Object based STM (OSTM)
Opportunity to Leverage CDS semantics

Figure: A concurrent list

Figure: Transaction Tree Structure [Weikum and Vossen]
Level-0 conflicts are irrelevant at Level-1.
Motivation towards Object based STM (OSTM)
Opportunity to Leverage CDS semantics

Figure: Pruned Tree
Motivation towards Object based STM (OSTM)
Opportunity to Leverage CDS semantics

Figure: Pruned Tree

Figure: Sequential Schedule
Motivation towards Object based STM (OSTM)
Opportunity to Leverage CDS semantics

Schedule can be accepted by STM.
Can we gain more concurrency yet ensuring composability and ease of programming?
Million Dollar Question

Can we gain more concurrency yet ensuring composability and ease of programming?

Solution: Yes
Can we gain more concurrency yet ensuring composability and ease of programming?

Solution: Yes

- Ease of programming $\Rightarrow$ STM Interface.
- Efficient Composition $\Rightarrow$ Object Level Semantics.
Object-based STMs (OSTM) operate on higher level objects rather than primitive reads & writes which act upon memory locations.
Object-based STMs (OSTM) operate on higher level objects rather than primitive reads & writes which act upon memory locations.

OSTM model exports:

- $STM_{\text{begin}}(), STM_{\text{push}}(), STM_{\text{pop}}(), STM_{\text{peek}}()$ and $STM_{\text{tryC}()}$ for stack.
- $STM_{\text{begin}}(), STM_{\text{insert}}(), STM_{\text{delete}}(), STM_{\text{lookup}}()$ and $STM_{\text{tryC}()}$ for set.
OSTM: Detail View

\[ T_i \quad \square \quad C_i \]

- **STM\_begin()**: Prepare a transaction
  - *Unique\_id.*
  - Init \textit{txlog}.

- **STM\_lookup()**: Validate at instant.
  Update \textit{txlog}.

- **STM\_insert()**: Execute w/o touching shared memory.
  Update \textit{txlog}.

- **STM\_Delete()**: Validate at instant.
  Modify at commit.
  Update \textit{txlog}.

- **STM\_tryC()**: Validation
  - *Interference validation*
  - *Time order validation*
  - *Intra Trans Validation*
  Commit into underlying data-structure.

**Figure**: Transaction lifecycle of OSTM
Introduction to STM

Problem with Read-Write STM

Motivation towards Object based STM (OSTM)

HT-OSTM Design

Execution Under HT-OSTM

Proof Of Correctness

Results

Conclusion
**Figure:** The underlying shared data-structure as hash table
HT-OSTM: Design
Underlying Data-Structure

Figure: The underlying shared data-structure as hash table

HT-OSTM Exported methods:

$STM\_begin()$, $STM\_lookup()$, $STM\_insert()$, $STM\_delete()$ and $STM\_tryC()$
Challenge I
Maintaining Information of Deleted Nodes for Satisfying Opacity

Figure: Time-Stamp of deleted node \( (k_7) \) is needed (Otherwise, not opaque).
Challenge I
Maintaining Information of Deleted Nodes for Satisfying Opacity

Figure: Time-Stamp of deleted node \((k_7)\) is needed (Otherwise, not opaque).

Figure: History is opaque.
Nodes are only logically deleted using **marked** field.

Each node stores **time-stamp** along with key, value & marked field.

Red color depicts dead node (deleted) and blue color depicts live node (not deleted).
Challenge I Contd..
Disadvantage of chaining with lazylist

Figure: Issue: Searching key $k_8$ over lazylist

- Red color depicts dead node (deleted) and blue color depicts live node (not deleted).
Disadvantage of chaining with lazy list

Figure: Issue: Searching key $k_8$ over lazylist

- Red color depicts dead node (deleted) and blue color depicts live node (not deleted).
Challenge I Contd..
Disadvantage of chaining with lazylist

Figure: Issue: Searching key $k_8$ over lazylist

- Red color depicts dead node (deleted) and blue color depicts live node (not deleted).
Challenge I Contd..
Disadvantage of chaining with lazylist

Figure: Issue: Searching key $k_8$ over lazylist

- Red color depicts dead node (deleted) and blue color depicts live node (not deleted).
Challenge I Contd..
Advantage of lazyrb-list over lazylist

- Each node is having two links: red link, blue link.
- Blue links are pointing to live nodes.
- Red links are pointing to live nodes as well as dead nodes.
- List invariants
  - Increasing order of keys.
  - Nodes accessible by blue links $\subseteq$ nodes accessible by red links.

Figure: Searching key $k_8$ over lazyrb-list
Each node is having two links: red link, blue link.

- Blue links are pointing to live nodes.
- Red links are pointing to live nodes as well as dead nodes.

List invariants
- Increasing order of keys.
- Nodes accessible by blue links $\subseteq$ nodes accessible by red links.

**Figure:** Searching key $k_8$ over lazyrb-list
Content

Introduction to STM

Problem with Read-Write STM

Motivation towards Object based STM (OSTM)

HT-OSTM Design

Execution Under HT-OSTM

Proof Of Correctness

Results

Conclusion
Execution Under HT-OSTM

Figure: Underlying zoomed in lazyrb-list

Figure: Example History
Execution Under HT-OSTM

**Figure:** Underlying zoomed in lazyrb-list

**Figure:** Current Execution

**Figure:** Example History

**Figure:** Transaction Log
Execution Under HT-OSTM

Figure: Underlying lazyrb-list: \( l_1(k_{10}) \) find region.

Figure: Current Execution

Figure: Transaction Log

<table>
<thead>
<tr>
<th>T_1</th>
<th>opn</th>
<th>key</th>
<th>P</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>lookup</td>
<td>( k_{10} )</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Execution Under HT-OSTM

Figure: Underlying lazy-skip list: $l_1(k_{10})$ find region.

Figure: Current Execution

Figure: Transaction Log

Archit Somani, IIT Hyderabad, India
Execution Under HT-OSTM

Figure: Underlying lazy-skip list: \( l_1(\kappa_{10}) \) find region.

Figure: Current Execution

Figure: Example History

Figure: Transaction Log

Archit Somani, IIT Hyderabad, India
Execution Under HT-OSTM

Figure: Underlying list: $l_1(k_{10})$: $k_{10}$ not found.

Figure: Current Execution

Figure: Example History

Figure: $T_1$: $l_1(k_{10})$: Log updated.
Execution Under HT-OSTM

**Figure: Underlying list:** $l_1(k_{10})$: $k_{10}$ added.

**Figure: Current Execution**

| $T_1$ | $l_1(k_{10})$ |

**Figure: Transaction Log**

<table>
<thead>
<tr>
<th>$T_1$</th>
<th>opn</th>
<th>key</th>
<th>P</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>lookup</td>
<td>$k_{10}$</td>
<td>$k_8$</td>
<td>$+\infty$</td>
<td></td>
</tr>
<tr>
<td>lookup</td>
<td>$k_8$</td>
<td>$k_8$</td>
<td>$+\infty$</td>
<td></td>
</tr>
</tbody>
</table>

**Figure: Example History**

$T_1 \mid l_1(k_{10}) \rightarrow d_1(k_6) \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \righta
Execution Under HT-OSTM

Figure: Underlying zoomed in lazy-skip list

Figure: Example History

Figure: Current Execution

Figure: $T_2$: $i_2(k_5)$: Log created.
Execution Under HT-OSTM

Figure: Underlying zoomed in lazy-skip list

Figure: Example History

Figure: Current Execution

Figure: $T_1$: $d_1(k_6)$: Log created.
Execution Under HT-OSTM

**Figure: Underlying list:** $d_1(k_6)$: $k_6$ find region.

**Figure: Example History**

**Figure: Current Execution**

**Figure: Transaction Log**

Archit Somani, IIT Hyderabad, India
Execution Under HT-OSTM

Figure: Underlying list: \( d_1(k_6) \): 
\( k_6 \) find region.

Figure: Current Execution

Figure: Transaction Log

Figure: Example History
Execution Under HT-OSTM

**Figure: Underlying list:** $d_1(k_6)$: $k_6$ find region.

**Figure: Example History**

**Figure: Current Execution**

**Figure: Transaction Log**
Execution Under HT-OSTM

**Figure:** Underlying list: \( d_1(k_6) \): 
\( k_6 \) find region.

**Figure:** Current Execution

**Figure:** Example History

**Figure:** Transaction Log
Execution Under HT-OSTM

**Figure: Underlying list: $d_1(k_6)$:**

$k_6$ found.

**Figure: Current Execution**

**Figure: $T_1$: $d_1(k_6)$ updated log.**

Archit Somani, IIT Hyderabad, India
Execution Under HT-OSTM

Figure: Underlying zoomed in lazy-skip list

Figure: Example History

Figure: Current Execution

Figure: Transaction Log
Execution Under HT-OSTM

**Figure: Underlying list:** \( i_2(k_{10}) \): \( k_{10} \).

**Figure: Current Execution**

**Figure: Example History**

**Figure: \( T_2: i_2(k_{10}) \) log created.**
Figure: Underlying list: \( \text{tryC}_1() \): \( d_1(k_6) \).

Figure: Example History

Figure: Current Execution

Figure: Transaction Log
Execution Under HT-OSTM

Figure: Underlying list: \( \text{try}C_1() \): \( d_1(k_6) \) find region.

Figure: Current Execution

Figure: Transaction Log
Execution Under HT-OSTM

Figure: Underlying list: \( \text{try}C_1() \):
\( d_1(k_6) \) find region.

Figure: Example History

Figure: Current Execution

Figure: Transaction Log
Execution Under HT-OSTM

Figure: Underlying list: \( \text{tryC}_1() \):
\( d_1(k_6) \) find region.

Figure: Example History

Figure: Current Execution

Figure: Transaction Log
Execution Under HT-OSTM

Figure: Underlying list: \( \text{try}C_1() \):
\( d_1(k_6) \) found.

Figure: Example History

Figure: Current Execution

Figure: Transaction Log
Execution Under HT-OSTM

Figure: Underlying list: \( \text{try}C_1() \): \( d_1(k_6) \) deleted.

Figure: Current Execution

Figure: Example History

Figure: Transaction Log

Archit Somani, IIT Hyderabad, India
Execution Under HT-OSTM

**Figure: Underlying list:** $\text{try}C_2()$:

$i_2(k_5)$ find region.

**Figure: Example History**

**Figure: Current Execution**

**Figure: Transaction Log**
Execution Under HT-OSTM

Figure: Underlying list: \( \text{try}C_2() \): \( i_2(k_5) \) find region.

Figure: Current Execution

Figure: Transaction Log
Execution Under HT-OSTM

**Figure:** Underlying list: \( \text{try}C_2() \):
\( i_2(k_5) \) find region.

**Figure:** Example History

**Figure:** Current Execution

**Figure:** Transaction Log
Execution Under HT-OSTM

Figure: Underlying list: \texttt{tryC}_2():: \texttt{i}_2(k_5) \textit{find region.}

Figure: Example History

Figure: Current Execution

Figure: Transaction Log
Execution Under HT-OSTM

Figure: Underlying list: tryC2():
i2(k5) find region.

Figure: Example History

Figure: Current Execution

Figure: i2(k5) log updated.
Execution Under HT-OSTM

Figure: Underlying list: \textit{tryC}_2() 
\(k_5\) added.

Figure: Example History

Figure: Current Execution

Figure: Transaction Log
Execution Under HT-OSTM

**Figure: Underlying list:** $\text{tryC}_2()$: $i_2(k_{10})$ find region.

**Figure: Example History**

**Figure: Current Execution**

**Figure: Transaction Log**
**Execution Under HT-OSTM**

**Figure: Underlying list:** \( \text{tryC}_2() \): 
\( i_2(k_{10}) \) find region.

**Figure: Example History**

**Figure: Current Execution**

**Figure: Transaction Log**
Execution Under HT-OSTM

Figure: Underlying list: \( \text{tryC}_2() \):

\( i_2(k_{10}) \) find region.

Figure: Example History

Figure: Current Execution

Figure: Transaction Log
Execution Under HT-OSTM

**Figure:** Underlying list: \( \text{try}C_2() \): \( i_2(k_{10}) \) find region.

**Figure:** Example History

**Figure:** Current Execution

**Figure:** \( i_2(k_{10}) \) updated log.
Execution Under HT-OSTM

Figure: Underlying list: \textit{tryC}_2(): $k_{10}$ added

Figure: Example History

Figure: Current Execution

Figure: Transaction Log

Archit Somani, IIT Hyderabad, India
Content

Introduction to STM

Problem with Read-Write STM

Motivation towards Object based STM (OSTM)

HT-OSTM Design

Execution Under HT-OSTM

Proof Of Correctness

Results

Conclusion
Proof Of Correctness
Operation level and Transaction level

Figure: Overlapping Operations
Proof Of Correctness
Operation level and Transaction level

Figure: Overlapping Operations

Figure: Operations are Linearized
Proof Of Correctness
Operation level and Transaction level

**Figure: Overlapping Operations**

**Figure: Operations are Linearized**

**Figure: Serial History (T₁, T₂)**
Proof Of Correctness
Operation level and Transaction level

Figure: Overlapping Operations

Figure: Operations are Linearized

Figure: Serial History (T₁, T₂)

Figure: Serial History (T₂, T₁)
Proof Of Correctness

Theorem (1)

Consider a history $H$ generated by HT-OSTM, there exists a sequential & legal history $H'$ equivalent to $H$ such that the conflict-graph of $H'$ is acyclic.

Theorem (2)

A legal HT-OSTM history $H$ is opaque iff $CG(H)$ is acyclic.

Please refer the arxiv link—>
https://arxiv.org/abs/1709.00681

Archit Somani, IIT Hyderabad, India
Proof Of Correctness

Theorem (1)
Consider a history $H$ generated by HT-OSTM, there exists a sequential & legal history $H'$ equivalent to $H$ such that the conflict-graph of $H'$ is acyclic.

Theorem (2)
A legal HT-OSTM history $H$ is opaque iff $CG(H)$ is acyclic.

Please refer the arxiv link—>
Introduction to STM

Problem with Read-Write STM

Motivation towards Object based STM (OSTM)

HT-OSTM Design

Execution Under HT-OSTM

Proof Of Correctness

Results

Conclusion
Experimental Setup

Setup

- Intel(R) Xeon(R) CPU E5-2690 v4 @ 2.60GHz, 56 NUMA CPUs.
- HT-OSTM vs Basic Time stamp ordering Protocol(BTO) [Weikum et al.] / Elastic STM(ESTM) [Gramoli et al.].
- list-OSTM vs Boosted list(BST) [Herlihy et al.] / NOrec STM list(NTM) [Michael et al.] / lock-free transactional list(LFT) [Zhang et al.]
- Time/thread and Number of aborts/thread
Experimental Setup

Setup

- Intel(R) Xeon(R) CPU E5-2690 v4 @ 2.60GHz, 56 NUMA CPUs.
- HT-OSTM vs Basic Time stamp ordering Protocol (BTO) [Weikum et al.] / Elastic STM (ESTM) [Gramoli et al.].
- list-OSTM vs Boosted list (BST) [Herlihy et al.] / NOrec STM list (NTM) [Michael et al.] / lock-free transactional list (LFT) [Zhang et al.]
- Time/thread and Number of aborts/thread

Parameters

- Lookup Intensive Workload: lookup: 70%, insert: 10% & delete: 20%
- Update Intensive Workload: lookup: 50%, insert: 25% & delete: 25%
- Key range: 1000 and Operations/transaction: 10
Results

*HT-OSTM* against ESTM, RWSTM

**Figure:** time vs #threads OR #aborts vs #threads
Results

$list-OSTM$ against LTM, NTM and BST

Figure: time vs #threads OR #aborts vs #threads
Content

Introduction to STM

Problem with Read-Write STM

Motivation towards Object based STM (OSTM)

HT-OSTM Design

Execution Under HT-OSTM

Proof Of Correctness

Results

Conclusion
Conclusion and Future Work

OSTM = Efficiency + Composition + Programmer friendly.

- HT-OSTM based on OSTM is *opaque*.
- HT-OSTM shows speedup of 3 to 6 times better than state of the art ESTM & RWSTM.
- list-OSTM outperforms state of the art LFT, NTM and BST by 30% to 80% across all workloads and scenarios.
- HT-OSTM and list-OSTM are having negligible aborts in comparison to other techniques.

Future Work

- The OSTM model can be extended to other data structures like Queue, Stack etc.
- The OSTM model can be extended to multi-version OSTM (MV-OSTM) for achieving higher concurrency.
OSTM = Efficiency + Composition + Programmer friendly.

- HT-OSTM based on OSTM is opaque.
- HT-OSTM shows speedup of 3 to 6 times better than state of the art ESTM & RWSTM.
- list-OSTM outperforms state of the art LFT, NTM and BST by 30% to 80% across all workloads and scenarios.
- HT-OSTM and list-OSTM are having negligible aborts in comparison to other techniques.

Future Work

- The OSTM model can be extended to other data structures like Queue, Stack etc.
- The OSTM model can be extended to multi-version OSTM (MV-OSTM) for achieving higher concurrency.
Thank You!

Q&A