NZTM: Nonblocking Zero-indirection Transactional Memory

Fuad Tabba¹ Cong “James” Wang¹
James R. Goodman¹ Mark Moir²

¹Department of Computer Science
University of Auckland

²Sun Microsystems Laboratories

The Second ACM SIGPLAN Workshop on Transactional Computing, 2007
Outline

Motivation

NZSTM - Design and Implementation
  Data Structures
  Making NZSTM Nonblocking
  NZTM - The Hybrid Approach

Our Results
  Test Environment
  Main Results

Conclusion
  Ongoing and Future Work
Why NZTM?

Goal
Efficient nonblocking transactional memory.

Our Solution

- **Object-based**: suitable for object-oriented languages.
- **Zero-Indirection**: each level is a potential cache-miss.
- **Hybrid**: complete STM that can exploit best-effort HTM to improve performance.
DSTM

TMObject

* Start

* Transaction
  * Old
  * New

Locator

Status

Transaction

Data

Data

Tabba, Wang, Goodman, Moir

University of Auckland and Sun Microsystems

NZTM: Nonblocking Zero-indirection Transactional Memory
RSTM

RSTM Object

* New
Visible Reader 1
...
Visible Reader N

Clean Bit

RSTM Object

* Transaction
* Old
New Data
Old Data

Transaction
Status

Tabba, Wang, Goodman, Moir

University of Auckland and Sun Microsystems

NZTM: Nonblocking Zero-indirection Transactional Memory
**DSTM2**

**DSTM2 Shadow Factory Object**

```
+-----------------------------------------------------+
| * Transaction                                       |
+-----------------------------------------------------+
| field1                                              |
+-----------------------------------------------------+
| shadow1                                             |
| field2                                              |
+-----------------------------------------------------+
| shadow2                                             |
| field3                                              |
+-----------------------------------------------------+
| shadow3                                             |
| DSTM2 Shadow Object                                 |
| Status Transaction                                  |
+-----------------------------------------------------+
```
NZSTM

NZObject

* Writer TXN
* Readers List
* Clone(*destination, *source)
* Old Data

Data

NZObject

Tabba, Wang, Goodman, Moir

University of Auckland and Sun Microsystems

NZTM: Nonblocking Zero-indirection Transactional Memory
Transactions and Aborting

Aborting an NZSTM Transaction

* Writer TXN
* Readers List
* Clone(*destination, *source)
* Old Data

Modified Data

NZObject

Aborted Transaction

Backup Data
Transactions and Aborting

NZSTM Transaction

<table>
<thead>
<tr>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>{Active, Committed, Aborted}</td>
</tr>
</tbody>
</table>

Abort Now Please: true/false

Transaction
Making NZSTM Nonblocking

Displacing Data

NXTM: Nonblocking Zero-indirection Transactional Memory

Tabba, Wang, Goodman, Moir

University of Auckland and Sun Microsystems
NZTM - The Hybrid Approach

Integrating Hardware and Software

- Compatible with best-effort hardware schemes.
- Attempts transactions in hardware first, then falls back onto software.
- We hope that hardware transactions would be the common case.
Benchmarks and Experiments

Benchmarks
- Linked List
- Red-black Tree
- Chained Hash Table
- Web Cache Simulation

Experiments
- Single Global Lock
- LogTM
- NZSTM
- NZTM
Results

Chained Hash Table

![Graph showing results for different thread counts and memory models.]

- **LOGTM**
- **NZTM/HTM/16**
- **NZTM/HTM/3**
- **LOCK**
- **NZSTM**

Tabba, Wang, Goodman, Moir

University of Auckland and Sun Microsystems

NZTM: Nonblocking Zero-indirection Transactional Memory
Ongoing and Future Work

- More comprehensive evaluation.
- Investigate different design alternatives and optimizations.
- Currently implementing invisible reads.
Making NZSTM Nonblocking - Alternative Method

Using Short Hardware Transactions per Store

- Begin Hardware Transaction
- Confirm transaction is Active
- Modify data in-place
- Commit Hardware Transaction
Results

Linked List

![Graph showing results of different transactional memory systems for a linked list operation. The graph plots the number of transactions per 1000 cycles against the number of threads. The systems compared are LOGTM, NZTM/HTM/16, NZTM/HTM/3, LOCK, and NZSTM. The graph demonstrates the performance variation among these systems as the number of threads increases.]
Results

Red-black Tree

The graph shows the comparison of different transactional memory systems for Red-black Tree operations as a function of the number of threads. The systems compared include LOGTM, NZTM/HTM/16, NZTM/HTM/3, LOCK, and NZSTM. The x-axis represents the number of threads, while the y-axis represents the transactions per 1000 cycles. The graph illustrates how each system performs under varying thread counts, with LOGTM generally showing the highest performance.
Results

LFUCache

The graph shows the transactional memory performance across different thread counts and various memory management techniques. The x-axis represents the number of threads, while the y-axis represents the transactions per 1000 cycles.

- **LOGTM** (solid red line)
- **NZTM/HTM/16** (dashed orange line)
- **NZTM/HTM/3** (dashed blue line)
- **LOCK** (dashed purple line)
- **NZSTM** (solid green line)

The graph indicates that NZSTM offers the best performance, followed by LOGTM, with other techniques showing varying degrees of performance.