

# Transactional Memory

2nd edition

© Springer Nature Switzerland AG 2022  
Reprint of original edition ©Morgan & Claypool 2010

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means—electronic, mechanical, photocopy, recording, or any other except for brief quotations in printed reviews, without the prior permission of the publisher.

Transactional Memory, 2nd edition  
Tim Harris, James Larus, and Ravi Rajwar

ISBN: 978-3-031-00600-5 paperback  
ISBN: 978-3-031-01728-5 ebook

DOI 10.1007/978-3-031-01728-5

A Publication in the Springer series  
*SYNTHESIS LECTURES ON COMPUTER ARCHITECTURE*

Lecture #11  
Series Editor: Mark D. Hill, *University of Wisconsin*  
Series ISSN  
Synthesis Lectures on Computer Architecture  
Print 1935-3235 Electronic 1935-3243

# Synthesis Lectures on Computer Architecture

## Editor

**Mark D. Hill**, *University of Wisconsin*

Synthesis Lectures on Computer Architecture publishes 50- to 100-page publications on topics pertaining to the science and art of designing, analyzing, selecting and interconnecting hardware components to create computers that meet functional, performance and cost goals. The scope will largely follow the purview of premier computer architecture conferences, such as ISCA, HPCA, MICRO, and ASPLOS.

## Transactional Memory, 2nd edition

Tim Harris, James Larus, and Ravi Rajwar  
2010

## Computer Architecture Performance Evaluation Models

Lieven Eeckhout  
2010

## Introduction to Reconfigured Supercomputing

Marco Lanzagorta, Stephen Bique, and Robert Rosenberg  
2009

## On-Chip Networks

Natalie Enright Jerger and Li-Shiuan Peh  
2009

## The Memory System: You Can't Avoid It, You Can't Ignore It, You Can't Fake It

Bruce Jacob  
2009

## Fault Tolerant Computer Architecture

Daniel J. Sorin  
2009

## The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines

Luiz André Barroso and Urs Hölzle  
2009

**Computer Architecture Techniques for Power-Efficiency**

Stefanos Kaxiras and Margaret Martonosi

2008

**Chip Multiprocessor Architecture: Techniques to Improve Throughput and Latency**

Kunle Olukotun, Lance Hammond, and James Laudon

2007

**Transactional Memory**

James R. Larus and Ravi Rajwar

2006

**Quantum Computing for Computer Architects**

Tzvetan S. Metodi and Frederic T. Chong

2006

# Transactional Memory

2nd edition

Tim Harris  
Microsoft Research

James Larus  
Microsoft Research

Ravi Rajwar  
Intel Corporation

*SYNTHESIS LECTURES ON COMPUTER ARCHITECTURE #11*

## ABSTRACT

The advent of multicore processors has renewed interest in the idea of incorporating transactions into the programming model used to write parallel programs. This approach, known as transactional memory, offers an alternative, and hopefully better, way to coordinate concurrent threads. The ACI (atomicity, consistency, isolation) properties of transactions provide a foundation to ensure that concurrent reads and writes of shared data do not produce inconsistent or incorrect results. At a higher level, a computation wrapped in a transaction executes atomically – either it completes successfully and commits its result in its entirety or it aborts. In addition, isolation ensures the transaction produces the same result as if no other transactions were executing concurrently. Although transactions are not a parallel programming panacea, they shift much of the burden of synchronizing and coordinating parallel computations from a programmer to a compiler, to a language runtime system, or to hardware. The challenge for the system implementers is to build an efficient transactional memory infrastructure. This book presents an overview of the state of the art in the design and implementation of transactional memory systems, as of early spring 2010.

## KEYWORDS

transactional memory, parallel programming, concurrent programming, compilers, programming languages, computer architecture, computer hardware, nonblocking algorithms, lock-free data structures, cache coherence, synchronization

# Contents

<b>Preface</b> .....	xiii
<b>Acknowledgments</b> .....	xv
<b>1 Introduction</b> .....	1
1.1 Motivation .....	1
1.1.1 Difficulty of Parallel Programming .....	1
1.1.2 Parallel Programming Abstractions .....	3
1.2 Database Systems and Transactions .....	4
1.2.1 What Is a Transaction? .....	5
1.3 Transactional Memory .....	6
1.3.1 Basic Transactional Memory .....	7
1.3.2 Building on Basic Transactions .....	8
1.3.3 Software Transactional Memory .....	9
1.3.4 Hardware Transactional Memory .....	11
1.3.5 What is Transactional Memory Good For? .....	11
1.3.6 Differences Between Database Transactions and TM .....	12
1.3.7 Current Transactional Memory Systems and Simulators .....	13
<b>2 Basic Transactions</b> .....	17
2.1 TM Design Choices .....	19
2.1.1 Concurrency Control .....	20
2.1.2 Version Management .....	21
2.1.3 Conflict Detection .....	22
2.2 Semantics of Transactions .....	23
2.2.1 Correctness Criteria for Database Transactions .....	24
2.2.2 Consistency During Transactions .....	28

2.2.3	Problems with Mixed-Mode Accesses .....	30
2.2.4	Handling Mixed-Mode Accesses: Lock-Based Models .....	35
2.2.5	Handling Mixed-Mode Accesses: TSC .....	38
2.2.6	Nesting .....	41
2.3	Performance, Progress and Pathologies .....	44
2.3.1	Progress Guarantees .....	45
2.3.2	Conflict Detection and Performance .....	48
2.3.3	Contention Management and Scheduling .....	51
2.3.4	Reducing Conflicts Between Transactions .....	54
2.3.5	Higher-Level Conflict Detection .....	57
2.4	Summary .....	59
<b>3</b>	<b>Building on Basic Transactions .....</b>	<b>61</b>
3.1	Basic Atomic Blocks .....	61
3.1.1	Semantics of Basic Atomic Blocks .....	64
3.1.2	Building Basic Atomic Blocks Over TM .....	68
3.1.3	Providing Strong Guarantees Over Weak TM Systems .....	69
3.2	Extending Basic Atomic Blocks .....	72
3.2.1	Condition Synchronization .....	72
3.2.2	Exceptions and Failure Atomicity .....	78
3.2.3	Integrating Non-TM Resources .....	80
3.2.4	Binary Libraries .....	81
3.2.5	Storage Allocation and GC .....	82
3.2.6	Existing Synchronization Primitives .....	84
3.2.7	System Calls, IO, and External Transactions .....	87
3.3	Programming with TM .....	89
3.3.1	Debugging and Profiling .....	89
3.3.2	TM Workloads .....	90
3.3.3	User Studies .....	93
3.4	Alternative Models .....	94
3.4.1	Transactions Everywhere .....	94



3.4.2	Lock-Based Models over TM .....	96
3.4.3	Speculation over TM .....	98
3.5	Summary .....	99
<b>4</b>	<b>Software Transactional Memory .....</b>	<b>101</b>
4.1	Managing STM Logs and Metadata .....	103
4.1.1	Maintaining Metadata .....	103
4.1.2	Undo-Logs and Redo-Logs .....	106
4.1.3	Read-Sets and Write-Sets .....	108
4.2	Lock-Based STM Systems with Local Version Numbers .....	108
4.2.1	Two-Phase Locking with Versioned Locks .....	109
4.2.2	Optimizing STM Usage .....	113
4.2.3	Providing Opacity .....	114
4.2.4	Discussion .....	116
4.3	Lock-Based STM Systems with a Global Clock .....	116
4.3.1	Providing Opacity Using a Global Clock .....	117
4.3.2	Timebase Extension .....	121
4.3.3	Clock Contention vs False Conflict Tradeoffs .....	121
4.3.4	Alternative Global Clock Algorithms .....	123
4.4	Lock-Based STM Systems with Global Metadata .....	123
4.4.1	Bloom Filter Conflict Detection .....	124
4.4.2	Value-Based Validation .....	126
4.5	Nonblocking STM Systems .....	128
4.5.1	Per-object Indirection .....	128
4.5.2	Nonblocking Object-Based STM Design Space .....	131
4.5.3	Nonblocking STM Systems Without Indirection .....	132
4.6	Additional Implementation Techniques .....	136
4.6.1	Supporting Privatization Safety and Publication Safety .....	136
4.6.2	Condition Synchronization .....	140
4.6.3	Irrevocability .....	141
4.7	Distributed STM Systems .....	142

4.7.1	STM for Clusters .....	142
4.7.2	STM-Based Middleware .....	143
4.7.3	STM for PGAS Languages .....	144
4.8	STM Testing and Correctness .....	144
4.9	Summary .....	145
<b>5</b>	<b>Hardware-Supported Transactional Memory .....</b>	<b>147</b>
5.1	Basic Mechanisms for Conventional HTMs .....	148
5.1.1	Identifying Transactional Locations .....	148
5.1.2	Tracking Read-Sets and Managing Write-Sets .....	149
5.1.3	Detecting Data Conflicts .....	151
5.1.4	Resolving Data Conflicts .....	152
5.1.5	Managing Architectural Register State .....	152
5.1.6	Committing and Aborting HTM Transactions .....	153
5.2	Conventional HTM Proposals .....	154
5.2.1	Explicitly Transactional HTMs .....	154
5.2.2	Implicitly Transactional HTM Systems .....	159
5.2.3	Hybrid TMs: Integrating HTMs and STMs .....	164
5.2.4	Software and Design Considerations .....	168
5.3	Alternative Mechanisms for HTMs .....	170
5.3.1	Software-Resident Logs for Version Management .....	170
5.3.2	Signatures for Access Tracking .....	174
5.3.3	Conflict Detection via Update Broadcasts .....	179
5.3.4	Deferring Conflict Detection .....	182
5.4	Unbounded HTMs .....	184
5.4.1	Combining Signatures and Software-Resident Logs .....	185
5.4.2	Using Persistent Meta-Data .....	187
5.4.3	Using Page Table Extensions .....	194
5.5	Exposing Hardware Mechanisms to STMs .....	197
5.5.1	Accelerating Short Transactions and Filtering Redundant Reads .....	197
5.5.2	Software Controlled Cache Coherence .....	198

5.5.3	Exposed Signatures to STMs.....	199
5.5.4	Exposing Metadata to STMs.....	200
5.6	Extending HTM: Nesting, IO, and Synchronization.....	201
5.7	Summary.....	203
<b>6</b>	<b>Conclusions .....</b>	<b>205</b>
	<b>Bibliography.....</b>	<b>209</b>
	<b>Authors' Biographies .....</b>	<b>245</b>

# Preface

This book presents an overview of the state of the art in transactional memory, as of early 2010. Substantial sections of this book have been revised since the first edition. There has been a vast amount of research on TM in the last three years (quantitatively, 210 of the 351 papers referred to in this book were written in 2007 or later). This work has expanded the range of implementation techniques that have been explored, the maturity of many of the implementations, the experience that researchers have writing programs using TM, and the insights from formal analysis of TM algorithms and the programming abstractions built over them.

At a high level, readers familiar with the first edition will notice two broad changes:

First, we have expanded the discussion of programming with TM to form two chapters. This reflects a separation between the lower level properties of transactions (Chapter 2) versus higher-level language constructs (Chapter 3). In early work, these notions were often combined with research papers introducing both a new TM algorithm and a new way of exposing it to the programmer. There is now a clearer separation, with common TM algorithms being exposed to programmers through many different interfaces, and with individual language features being implemented over different TMs.

The second main difference is that we have re-structured the discussions of STM (Chapter 4) and HTM (Chapter 5) so that they group work thematically rather than considering work chronologically on a paper-by-paper basis. In each case, we focus on detailed case studies that we feel are representative of major classes of algorithms or of the state-of-the-art. We try to be complete, so please let us know if there is work that we have omitted.

This book does not contain the answers to many questions. At this point in the evolution of the field, we do not have enough experience building and using transactional memory systems to prefer one approach definitively over another. Instead, our goal in writing this book is to raise the questions and provide an overview of the answers that others have proposed. We hope that this background will help consolidate and advance research in this area and accelerate the search for answers.

In addition, this book is written from a practical viewpoint, with an emphasis on the design and implementation of TM systems, and their integration into programming languages. Some of the techniques that we describe come from research that was originally presented in a more formal style; we provide references to the original papers, but we do not attempt a formal presentation in this book. A forthcoming book examines TM from a theoretical viewpoint [117].

There is a large body of research on techniques like thread-level speculation (TLS) and a history of cross-fertilization between these areas. For instance, Ding *et al.*'s work on value-based validation inspired techniques used in STM systems [88], whereas STM techniques using eager

version management inspired Oancea *et al.*'s work on in-place speculation [234]. Inevitably, it is difficult to delineate exactly what work should be considered "TM" and what should not. Broadly speaking, we focus on work providing shared-memory synchronization between multiple explicit threads; we try, briefly, to identify links with other relevant work where possible.

The bibliography that we use is available online at <http://www.cs.wisc.edu/trans-memory/biblio/index.html>; we thank Jayaram Bobba and Mark Hill for their help in maintaining it, and we welcome additions and corrections.

Tim Harris, James Larus, and Ravi Rajwar  
June 2010

# Acknowledgments

This book has benefited greatly from the assistance of a large number of people who discussed transactional memory in its many forms with the authors and influenced this book—both the first edition and this revised edition. Some people were even brave enough to read drafts and point out shortcomings (of course, the remaining mistakes are the authors’ responsibility).

Many thanks to: Adam Welc, Al Aho, Ala Alameldeen, Amitabha Roy, Andy Glew, Annette Bieniusa, Arch Robison, Bryant Bigbee, Burton Smith, Chris Rossbach, Christos Kotse-lidis, Christos Kozyrakis, Craig Zilles, Dan Grossman, Daniel Nussbaum, David Callahan, David Christie, David Detlefs, David Wood, Ferad Zyulkyarov, Gil Neiger, Goetz Graefe, Haitham Akkary, James Cownie, Jan Gray, Jesse Barnes, Jim Rose, João Cachopo, João Lourenço, Joe Duffy, Justin Gottschlich, Kevin Moore, Konrad Lai, Kourosh Gharachorloo, Krste Asanovic, Mark Hill, Mark Moir, Mark Tuttle, Martín Abadi, Maurice Herlihy, Michael Scott, Michael Spear, Milind Girkar, Milo Martin, Nathan Bronson, Nir Shavit, Pascal Felber, Paul Petersen, Phil Bernstein, Richard Greco, Rob Ennals, Robert Geva, Sanjeev Kumar, Satnam Singh, Scott Ananian, Shaz Qadeer, Simon Peyton Jones, Steven Hand, Suresh Jagannathan, Suresh Srinivas, Tony Hosking, Torvald Riegel, Vijay Menon, Vinod Grover, and Virendra Marathe.

Tim Harris, James Larus, and Ravi Rajwar  
June 2010