Transactional Memory

2nd edition

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 hardwarecomponents 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
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

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 Kotselidis, 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