Chapter 1

Hardware Approaches to Transactional Memory in Chip Multiprocessors

J. Rubén Titos-Gil
Chalmers University of Technology
(Sweden)
ruben.titos@chalmers.se

Manuel E. Acacio
Universidad de Murcia
(Spain)
meacacio@ditec.um.es

1.1 Introduction

Multicores are nowadays at the heart of almost every computational system, from the smartphone in our pocket, to the server-class machines in datacenters that provide us with a myriad of cloud services. With the advent of chip multiprocessors, the shift to mainstream parallel architectures is inevitable, and both programmers and architects are presented with immense opportunities and enormous challenges. Despite the fact that multiprocessor systems have existed for a long time, multi-threaded programming has not been much of a focus. Instead, multiprocessors were of interest only to the small community of high-performance computing (HPC), and so was parallel programming, which was mostly ignored by software vendors, and not widely investigated nor taught. As a matter of fact, most software development over time has been predicated on single-core hardware, and the collective knowledge of software developers across organizations has been based primarily on single processor platforms.

Now that the *free lunch* is over [76], programmers must change the way they create applications to fully leverage multicore hardware. At every layer of the computing stack, whether the targeted platform is a handheld device or a warehouse-scale computer, programmers are being pushed towards unfamiliar programming models in order to deliver parallel software that takes advantage of the newly available computational resources and meets the demands of the end user. In the context of datacenters, the task is even more daunting because of the massive scale and complex architecture of these systems where efficient exploitation of parallelism is paramount at every level. Ideally, parallel software developed for these large-scale clusters should be able to harness the potential of their multicore build-

[87] M.M. Waliullah and Per Stenstrom. Reducing roll-back overhead in transactional memory systems by checkpointing conflicting accesses. In *Proceedings of the 22nd International Parallel and Distributed Processing Symposium*. 2008. 1.7

[88] Steven C. Woo, Moriyoshi Ohara, Evan Torrie, Jaswinder Pal Singh, and Anoop Gupta. The SPLASH-2 programs: Characterization and methodological considerations. In *Proceedings of the 22nd International Symposium on Computer Architecture*, pages 24–36, 1995. 1.6

[89] Luke Yen, Jayaram Bobba, Michael R. Marty, Kevin E. Moore, Haris Volos, Mark D. Hill, Michael M. Swift, and David A. Wood. LogTM-SE: Decoupling hardware transactional memory from caches. In *Proceedings of the 13th Symposium on High-Performance Computer Architecture*, pages 261–272, 2007. 1.2.3, 1.4.2, 1.4.3, 1.7

[90] Luke Yen, Stark C. Draper, and Mark D. Hill. Notary: Hardware techniques to enhance signatures. In *Proceedings of the 41st International Symposium on Microarchitecture*, pages 234–245, 2008. 1.7

[91] Richard Yoo, Christopher Hughes, Konrad Lai, and Ravi Rajwar. Performance evaluation of intel transactional synchronization extensions for high performance computing. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage, and Analysis (SC)*, 2013. 1.1, 1.2.4, 1.4.1, 1.6, 1.8