The Distributed Computing Column

by

Panagiota Fatourou

Department of Computer Science, University of Crete
P.O. Box 2208 GR-714 09 Heraklion, Crete, Greece
and
Institute of Computer Science (ICS)
Foundation for Research and Technology (FORTH)
N. Plastira 100. Vassilika Vouton
GR-700 13 Heraklion, Crete, Greece
faturu@csd.uoc.gr

Announcing the 2012 Edsger W. Dijkstra Prize in Distributed Computing

Marcos K. Aguilera, Microsoft Research
Dahlia Malkhi, Microsoft Research
Keith Marzullo, UCSD
Alessandro Panconesi, Sapienza, U. of Rome
Andrzej Pelc, U. Quebec
Roger Wattenhofer, ETH Zurich

The ACM-EATCS Edsger W. Dijkstra Prize in Distributed Computing is awarded to outstanding papers on the principles of distributed computing, whose significance and impact on the theory or practice of distributed computing have been evident for at least ten years. The prize is sponsored jointly by the ACM

Bulletin of the EATCS no 108, pp. 90–92, October 2012

Symposium on Principles of Distributed Computing (PODC) and the EATCS Symposium on Distributed Computing (DISC).

The 2012 Prize Committee, composed of Marcos K. Aguilera (chair), Dahlia Malkhi, Keith Marzullo, Alessandro Panconesi, Andrzej Pelc, and Roger Wattenhofer, has selected

Maurice Herlihy, J. Eliot B. Moss, Nir Shavit, and Dan Touitou

to receive the 2012 Edsger W. Dijkstra Prize in Distributed Computing for the following two outstanding papers:

> Maurice Herlihy and J. Eliot B. Moss. Transactional Memory: Architectural Support for Lock-Free Data Structures. *20th Annual International Symposium on Computer Architecture,* pages 289–300, May 1993.
>
> Nir Shavit and Dan Touitou. Software Transactional Memory. *Distributed Computing* 10(2):99–116, February 1997. (An earlier version appearing in the 14th ACM Symposium on Principles of Distributed Computing, pages 204–213, August 1995.)

These papers established the abstraction of Transactional Memory, which has fundamentally changed parallel computing both in its theoretical foundations and in its practice.

As with many influential papers, the work by Herlihy and Moss presents a beguilingly simple idea: extend load-linked and store-conditional to allow a processor to update a collection of locations atomically. This idea arose from deep insights:

- By allowing the creator of a concurrent data structure to focus on what should be atomic rather than how it should be made atomic, transactional memory significantly raises the level of abstraction for parallel programs, thereby eliminating much of the complexity of lock-free programming.
- Because actual dynamic conflicts among operations are rare in well-written programs, a speculative implementation of atomicity can enjoy a significant performance advantage over more conservative approaches.
- Given that cache coherence protocols already track conflicts among processors, multi-location atomic update can be realized in hardware by introducing a small "transactional cache" and by making simple modifications to standard cache coherence protocols.

This last insight notwithstanding, Herlihy and Moss's proposal proved too ambitious for the hardware of the day, and their work was largely ignored within

the architecture community for most of the following decade. Within the theory community, however, it inspired multiple explorations of the limits of software emulation, most notably the Software Transactional Memory work of Shavit and Touitou.

Building on earlier universal non-blocking constructions, Shavit and Touitou showed how to achieve lock freedom without the need for costly recursive helping, and thus provide effective non-blocking multi-word operations purely in software. It was the first work to demonstrate that software transactions could, under the right circumstances, outperform conservative locking.

In terms of fostering research, transactional memory has become a truly transformative idea. For example, two years ago, the second edition of the monograph by Harris, Larus, and Rajwar on Transactional Memory listed over 350 papers in the field. Google Scholar reports almost 1400 citations to Herlihy and Moss, and almost 1000 to Shavit and Touitou. The annual TRANSACT workshop, sponsored by ACM SIGPLAN, is now planning its eighth incarnation. In terms of practice, software architects have developed dozens of runtime implementations, both blocking and non-blocking, of dazzling algorithmic variety. At least four major compilers, including gcc, now support transactional memory in C++. Hardware implementations have been developed by Azul, Sun (Oracle), AMD (on paper), IBM, and Intel. The IBM and Intel implementations, in particular, ensure that hardware support is here to stay.

These two papers started the distributed computing research community along the path towards the design of general multi-word transactions; ones that in the future will most likely be based on a combination of hardware, software, and language techniques. Transactional memory serves as an outstanding example of how the distributed computing community has influenced the world.