

# TRAMP Position Paper

David A. Wood

## Wisconsin Multifacet Group

Currently proposed TM systems simplify some aspects of concurrent programming, but are not sufficiently robust for general programmers. For example, STMs provide only weak isolation (a.k.a., weak atomicity [1]), allowing non-transactional code to observe and corrupt transactions. Moreover, high software overheads make current STMs perform too poorly for general use. Conversely, currently proposed HTMs promise good performance, but impose restrictions not unlike those for spinlocks: limited execution duration, no page faults, and no blocking system calls. With these limitations, programmers using transactions trade one set of problems (lock order and granularity) for another.

The Wisconsin Multifacet group seeks to create a *robust transactional environment* for concurrent programming. We aim to develop a class of TM systems that make few demands on programmers, relaxing restrictions on how much or which code they execute within a transaction. These systems should also support graceful integration with existing non-transactional operating systems (OSs) and applications, preserving industry's multi-billion dollar investment. We propose doing this using virtual machine technology, which allows the guest OS to be initially oblivious to transactions but then incrementally evolve to use them internally. We seek simple hardware mechanisms that can be used for TM and re-targeted to other important purposes.

Our recent experience developing *Log-based Transactional Memory (LogTM)* [2,3,4] has led us to deconstruct the transactional memory functions into separate components for *version management/rollback* (storing both old version to restore on abort and the new versions to instantiate on commit), *access summary* (recording transaction read/write sets), and *access checks* (detect conflicts between transactions). This deconstruction enables a class of TM implementations with different cost/performance characteristics, ranging from all software to largely hardware. The LogTM-SE represents a sweet-spot in this spectrum. We believe that further deconstruction of these TM mechanisms may lead to further insights and implementations.

Programmers have long benefited from systems that *virtualize* resources, thus freeing them from managing hardware limits. Virtualization is typically implemented with optional virtual machine monitor (hypervisor), operating system, and hardware mechanisms. Most systems deconstruct program state into its memory state (managed by virtual memory policies and mechanisms), thread state (managed by saving and restoring registers), and system state (managed in kernel memory). TM, however, adds new program state (e.g., read/write sets and buffered updates) that many current HTMs either fail to virtualize or do so using monolithic solutions.

We will extend the benefits of virtualization to HTM programmers by virtualizing TM components. By avoiding monolithic solutions, our approach fits well with existing systems (protecting current investments) and manages complexity (divide and conquer). We are currently seeking to achieve robust transactions by extending a production OS (OpenSolaris) to virtualize our TM mechanisms, thereby hiding thread switches and paging from TM programmers.

Current HTMs, including LogTM-SE, favor transactions with a small memory footprint and short running time. The bias is sufficiently strong that programmers may never explore the potential uses of large, long-running transactions (e.g., where conflicts between transactions signal assumption violations). To enable the exploration of other uses for robust transactions, we are exploring a new TM system that handles prolonged transactions with precision (few false conflicts) and low overhead.

## References

- [1] Colin Blundell, E Christopher Lewis, and Milo M.K. Martin. Deconstructing Transactional Semantics: The Subtleties of Atomicity. In *Workshop on Duplicating, Deconstructing, and Debunking (WDDD)*, June 2005.
- [2] Kevin E. Moore, Jayaram Bobba, Michelle J. Moravan, Mark D. Hill, and David A. Wood. LogTM: Log-Based Transactional Memory. In *Proceedings of the Twelfth IEEE Symposium on High-Performance Computer Architecture*, pages 258–269, February 2006.
- [3] Michelle J. Moravan, Jayaram Bobba, Kevin E. Moore, Luke Yen, Mark D. Hill, Ben Liblit, Michael M. Swift, and David A. Wood. Supporting Nested Transactional Memory in LogTM. In *Proceedings of the Twelfth International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 359–370, October 2006.
- [4] 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 Thirteenth IEEE Symposium on High-Performance Computer Architecture*, pages 261–272, February 2007.