RETROSPECTIVE: Bulk Disambiguation of Speculative Threads in Multiprocessors

Luis Ceze*, James M. Tuck†, Călin Caşcaval‡, Josep Torrellas§
*University of Washington
†North Carolina State University
‡Google Research
§University of Illinois at Urbana-Champaign

I. Motivation

In the early 2000s, the computer architecture community was debating solutions to the diminishing performance returns of monolithic superscalar processors. The need to continue delivering higher performance, better energy efficiency, and new capabilities started a number of efforts on exploiting parallelism in future architectures. Chip Multiprocessors (CMPs) were the putative next-generation architecture, but they raised serious concerns along multiple fronts—in particular, performance and programmability. Single-thread performance was paramount for the workloads at the time; if cores were to grow relatively weaker as compared to their monolithic alternative, how would single-thread performance continue to improve? Parallel programming and its associated challenges of data races, non-determinism, and synchronization overhead was another concern; would all programmers need to be parallel programmers?

These concerns motivated significant interest in techniques such as Thread-Level Speculation (TLS), Transactional Memory (TM), and Checkpointed Multiprocessors. Moreover, many of the ideas that had traditionally been studied in the context of large-scale shared-memory multiprocessors, such as scalable synchronization and distributed cache coherence, could now be applied to a single-chip architecture—offering the tantalizing ability to integrate substantial functionality into lower-cost platforms.

As work to integrate TLS and TM into a CMP cache hierarchy developed, several authors, including ourselves, published TM and TLS designs. A common thread across these designs was the need to efficiently manage conflicts between addresses accessed by different threads. For example, the Transactional Memory Coherence and Consistency (TCC) project at Stanford [4] required a committing thread to broadcast the list of addresses it had written, so that all other threads would detect conflicts with their own accesses.

In this timeframe, the DARPA HPCS program emerged to help create the next generation of high productivity computer systems—raising the bar not only on performance, but also on programmability. In this program, Josep Torrellas collaborated with IBM as part of the IBM-led PERCS project. Dr. Balaram Sinharoy, one of the leading designers of IBM's POWER processors, was interested in what TLS could offer if added to POWER processors. He had discussions with Josep Torrellas and visited the University of Illinois.

The genesis of the idea behind our ISCA 2006 paper came as part of evaluating TLS when Luis Ceze interned with Călin Caşcaval at IBM Research during the summer of 2005. The challenges of incorporating TLS into a real processor became apparent: changes to the cache hierarchy were difficult and needed to be avoided, which meant that versioning information, speculative read bits, and speculative write bits commonly used to track dependence information across threads could not be held in the caches. They needed to be stored somewhere else. From that design constraint, the idea of *Signatures* emerged.

With the involvement of James Tuck and Josep Torrellas, the idea was broadened and refined to also make it applicable to TM, as TM required similar hardware primitives as TLS.

II. Design

The key idea of the paper is to hash-encode the set of addresses accessed by a thread in a concise signature, and add hardware support for operations that efficiently process signatures. The signatures are a superset encoding that can efficiently track many addresses in a single register using hundreds of bits or more. What made the signatures especially useful was what we called *Bulk operations*: operations that allowed many addresses to be compared or operated on simultaneously. Being a set, signatures are operated on efficiently using bitwise operations—e.g., bitwise-and for intersection and bitwise-or for union. These simple yet powerful operations made it possible to build even more sophisticated functionalities on signatures. For example, using combinations of signatures and Bulk operations together, we implemented mechanisms for disambiguating the addresses accessed by different threads, invalidating stale state in caches, making the state of committing threads visible to all other threads, discarding incorrect state when threads are squashed, and managing the speculative state of multiple cooperating threads.

Traditionally, mechanisms to implement such features are complex, often distributed, and error prone. In contrast, Bulk operations provide substantial conceptual and implementation simplicity. For example, we simplified the implementation of thread commit in TCC by broadcasting the signatures of the committing threads to both detect conflicts with other threads

and selectively invalidate the cached state of other threads. We envisioned a Bulk Disambiguation Module that included the signature registers and various functional units to operate on them. This unit worked together with, but independently of the L1 cache.

III. Evolution

Our paper garnered attention from industry and academia shortly after publication. On the industry side, there was a flurry of TM designs by several processor vendors. For example, one of the architectures that considered speculation was the IBM Blue Gene/Q (BGQ). In the BGQ design, both TLS and TM were considered [7] and, to our recollection, early discussions included the use of signatures. During that time, IBM filed several patents that included signatures for cache coherence and speculation support [3], [1]. Patents from Sun/Oracle employees who were involved in the pioneering Sun Rock processor cited our paper [5]. Josep Torrellas presented the work at Intel and it was received with significant interest.

In academia, our paper motivated or influenced a long series of Bloom-filter-based disambiguation work by the community covering TM, cache design, and memory models. Following the publication of our paper, we conceived the BulkSC [2] checkpointed multiprocessor, which was published in the following year at ISCA. In BulkSC, cores continuously execute atomic blocks of instructions called Chunks, maintain cache coherence with signature operations, and provide high-performance sequential consistency. BulkSC formed the foundation for many follow-up works that provide scalable checkpointed multiprocessing and deterministic parallel processing, both by the authors and by others.

One important influence of our work was on tackling the overheads in TM. For example, the LogTM-SE [8] proposal from the University of Wisconsin employed signatures to enhance their TM design. Many other TM designs adopted signatures. Yet other works considered ways of using signatures to reduce aborts, better capture locality, or tailor them to specific workloads and applications.

The influence of signature-based operations on parallel systems extended broadly to aspects such as data race detection, cache design, coherence support, and memory models. Furthermore, some works exposed signatures to programmers to perform profiling, compiler-directed runtime disambiguation, and program optimizations [6]. More recently, work integrating durable transactions for non-volatile memory and TM has built on BulkSC ideas.

IV. The Future

When we wrote the paper, we had every expectation that TM and coarse-grain speculation would become popular techniques in commercial computer systems. Indeed, in the years that followed, interest in these techniques was very high, and companies such as IBM, Intel, and Sun developed hardware support for TM. In particular, the Intel hardware TM architecture (TSX) received a lot of attention. As years went by, however, these designs failed to gain traction and interest waned. Recently, Intel has discontinued support for TSX.

While one can speculate on the various reasons for such an outcome, a major reason has to be the imbalance between the relatively high hardware complexity of TM implementations and the small set of existing applications that can use TM to substantially improve performance or programmability. Indeed, TM hardware intimately interacts with complicated mechanisms in multiprocessor memory systems such as cache coherence, which increases the chance of introducing hardware bugs. In addition, recent disclosures of security vulnerabilities enabled by speculation hardware have prompted additional concerns.

Another reason may be the emergence of broad classes of applications that expose regular, data-level parallelism, and of compute engines that can exploit it. A good example is machine learning applications and GPUs. GPUs provide performance scaling for these applications without requiring speculative techniques.

Nonetheless, TM, TLS, and Checkpointed Multiprocessors remain important computing techniques that can boost application speed or programmability. Their time may still come.

In the meantime, signatures and simple Bulk operations on them are being employed in a variety of uses in processor, memory, and storage systems hardware. They are useful in various search, comparison, and matching mechanisms, such as those found in data coherence and virtual memory support. As future computer architecture structures become more complicated, the conceptual and implementation simplicity of signatures and Bulk operations will be increasingly valued.

References

[1] D. Ahn, L. H. Ceze, A. Gara, M. Ohmacht, and Z. Xiaotong, "Reader set encoding for directory of shared cache memory in multiprocessor system," 2014, US Patent 8751748.

[2] L. Ceze, J. Tuck, P. Montesinos, and J. Torrellas, "BulkSC: Bulk enforcement of sequential consistency," in *Proceedings of the 34th annual international symposium on Computer architecture*, 2007, pp. 278–289.

[3] A. Gara and V. Salapura, "Architectural support for thread level speculative execution," 2006, US Patent 7350027B2.

[4] L. Hammond, V. Wong, M. Chen, B. D. Carlstrom, J. D. Davis, B. Hertzberg, M. K. Prabhu, H. Wijaya, C. Kozyrakis, and K. Olukotun, "Transactional memory coherence and consistency," in *Proceedings of the 31st Annual International Symposium on Computer Architecture*, 2004.

[5] M. S. Moir and D. Dice, "System and method for integrating best effort hardware mechanisms for supporting transactional memory," Jun. 14 2016, US Patent 9,367,363.

[6] J. Tuck, W. Ahn, L. Ceze, and J. Torrellas, "SoftSig: Software-Exposed Hardware Signatures for Code Analysis and Optimization," in *13th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2008.

[7] A. Wang, M. Gaudet, P. Wu, J. N. Amaral, M. Ohmacht, C. Barton, R. Silvera, and M. Michael, "Evaluation of Blue Gene/Q Hardware Support for Transactional Memories," in *Proceedings of the 21st International Conference on Parallel Architectures and Compilation Techniques*, 2012, p. 127–136.

[8] L. Yen, J. Bobba, M. R. Marty, K. E. Moore, H. Volos, M. D. Hill, M. M. Swift, and D. A. Wood, "LogTM-SE: Decoupling Hardware Transactional Memory from Caches," in *2007 IEEE 13th International Symposium on High Performance Computer Architecture*, Feb 2007, pp. 261–272.