15-740: Computer Architecture
Project Proposal:
Speculation and Cache Problems in STAMPede

Patrick Riley `<pfr+@cs.cmu.edu>`
Joseph Vanderwaart `<joev+@cs.cmu.edu>`
Kevin Watkins `<kw+@cs.cmu.edu>`
`http://www.cs.cmu.edu/~kw/15-740`

Fall 2000
Submitted: October 24, 2000

1 Project Description

Cache management is one of the challenging aspects of thread-level data speculation. In the STAMPede project, the caches are used to maintain the speculatively read and modified data. This means that there must be careful cache management to allow for squashing failed speculation and efficient committing of successful speculation.

One problem is that a thread may need to evict a line from the cache which has been speculatively modified. This line cannot be sent back to the main shared memory because it is unknown whether the thread speculation will be successful. This project will use two techniques to try and address this problem

Victim Cache A victim cache holds cache lines which have recently been evicted. In the single processor case, this can help prevent thrashing. Here, a victim cache could catch speculative modified lines before they go out to the main shared memory. If a thread has to evict a speculatively modified line, it can be caught in the victim cache, allowing the thread to proceed speculatively.

Suspending Violations If the current data access of a thread would cause a speculatively modified cache line to be evicted, then the speculation of the thread will fail. Rather than failing and throwing away all of the work already accomplished, the thread could be stalled until it is no longer speculative.

We also hope to spend some time comparing these two approaches and exploring their interaction.

2 Schedule

This project divides fairly naturally into three task areas:

Victim Cache Coding Implementing the victim cache on the simulator. Joe Vanderwaart will be the primary person for this task.

Suspending Violations Coding Implementing the suspending violations in the simulator. Kevin Watkins will will be the primary person for this task. This is probably the largest of the task because something may need to be done at the "hardware" layer so that it is aware of the home free token. The other group members will assist with this task as required.

Testing Setup Getting and setting up tests for various benchmarks in order to evaluate the above ideas. It is not entirely clear at this point how much work this will require, since it depends largely on what has already been setup and is usable for the simulator. Patrick Riley will be the primary person for this task.

Here is our projected time line:

Week	Date	Description	Note
1	Oct 23	Setting up the simulator; getting familiar with the STAMPede setup	
2	Oct 30	Coding for all three tasks (victim cache, suspending violations, testing setup)	
3	Nov 6	Continue Coding (and begin some testing for debugging purposes)	
4	Nov 13	More extensive testing on (hopefully) complete code	
5	Nov 20	Explore the interaction of the two techniques	Milestone
6	Nov 27	Final testing and write-up	
7	Dec 4	Celebrating being done	Project Due

3 Milestone

For the milestone, we will have the following finished:

- The victim cache code will be complete.
- The suspending violations code will be completely functional, though there may still be improvements we will want to make.
- We will have preliminary results for the two techniques individually on real benchmark data.

4 Literature Search

The cache coherence scheme of STAMPede is described by Steffan *et. al.* in [7]. A somewhat more detailed presentation of the protocol is given in the technical report [6]. We believe that these two sources, together with the source code for the STAMPede simulator (see next section), will provide sufficient background on the existing cache coherence mechanism.

We have also found some papers describing other, different approaches to resolving dependences in the presence of thread-level speculation. Most [1, 2, 3, 4] describe modifications or extensions to hardware to support speculation, but at least one [5] attempts to solve the problem completely in software using special data structures for variables shared between processors.

5 Resources Needed

We will be using a simulator already developed for the STAMPede project, to which Greg Steffan was kind enough to give us access. It runs on both SGI and x86 Linux platforms so it will not be difficult to find equipment on which to perform the simulations.

6 Progress So Far

We have set up a CVS repository and web page which will support communication throughout the project. We have contacted Greg Steffan in order to get access to the simulator. We do not anticipate needing access to any other external resources.

References

[1] M. Cintra, J. Martínez, and J. Torrellas. Architectural support for scalable speculative parallelization in shared-memory multiprocessors. In *Proceedings of ISCA 27*, June 2000.

[2] Sridhar Gopal, T. N. Vijaykumar, James E. Smith, and Gurindar S. Sohi. Speculative versioning cache. In *Proceedings of the 4th International Symposium on High-Performance Computer Architecture*, February 1998.

[3] M. Gupta and R. Nim. Techniques for speculative run-time parallelization of loops. In *Supercomputing '98*, November 1998.

[4] Venkata Krishnan and Josep Torrellas. A chip-multiprocessor architecture with speculative multithreading. *IEEE Transactions on Computers*, pages 866–880, September 1999.

[5] Peter Rundberg. A low-overhead software approach to thread-level data dependence speculation on multiprocessors. Technical Report 00-13, Chalmers University of Technology, Göteborg, Sweden, July 2000.

[6] J. Gregory Steffan, Christopher B. Colohan, and Todd C. Mowry. Extending cache coherence to support thread-level data speculation on a single chip and beyond. Technical Report CMU-CS-98-171, Carnegie Mellon University, Pittsburgh, PA, December 1998.

[7] J. Gregory Steffan, Christopher B. Colohan, Antonia Zhai, and Todd C. Mowry. A scalable approach to thread-level speculation. In *Proceedings of ISCA 27*, June 2000.