Lecture Notes: Wednesday, May 14, 2008

Sun's ROCK Processor and the Adaptive Transactional Memory Test Platform

Note: This information is mainly based on speculation of what the upcoming ROCK processor is. It might not be accurate and the end product might be different.

Background

The ROCK Processor is a multicore multithreaded SPARC processor. It's aimed at higher thread performance, better floating points performance and more scalability than Niagra processors. It targets database servers, and floating point high performance workloads.

It's expected that it will run at 2.3 GHz and will have 16 cores capable of running two threads each, arranged in 4 core clusters. Cores in a cluster share a 32KB instruction cache, two 32KB data caches, and two floating point units.

More importantly, as far as Compsci703 is concerned, it's excpected to support best-effort hardware transactional memory.

What is best-effort HTM?

Best effort HTM supports the most common and easy to implement transactions, but doesn't guarantee the support all transactions regardless of their size and duration. The chip *reserves the right* to simply fail transactions that exceed on-chip resources for HTM or that encounter difficult instructions or situations. In other words, there are little or no guarantees about what kind of transaction could eventually succeed.

It's easier for the processor designer to design, but more difficult for the programmer to utelize.

Why not support full-blown hardware transaction with stronger guarnatees? It's more difficult to design, and as of yet it's not clear that there will be enough demand or use of such a scheme. Having a starting point will lend itself so future incremental improvements.

ISA Additions in ROCK

ROCK adds three instructions (technically two new instructions and one new status register) to support transactional memory:-

Instruction	*Description*
`chkpt <fail_pc>`	Begins a hardware transaction. Branches to the specified address if and only if it fails.
`commit`	Commits the transaction
`rd %cps, <dest_reg>`	Reads the checkpoint status register, used to determine *why* a transaction has failed

Reasons for transaction failures to commit

Broadly speaking, a transaction may fail due to a conflict with another transaction, an exception, an interrupt, executing a banned transaction and exceeding chip limits.

In other words, a context switch, a function call, a TLB miss, too many reads or too many writes are all reasons a transaction might fail.

It is important therefore, that a transaction not be retried indefinitely unless there are guarantees that it would eventually succeed.

Other Notes

ROCK doesn't have a contention management scheme, whenever there's a conflict, the requester always wins. This is similar to the *aggressive* contention manager in DSTM.

ROCK doesn't take advantage of cache incoherence for bufferring writes. Rather it stores all writes into a store queue, which means that the transaction is also limited by the number of writes to the store queue.

Uses of ROCK Transactions

Explicit Speculative Lock Elision

By modifying the locking libraries, whether it's the Pthread library or the JVM's locking primitives, all locks could be replaced with a ROCK transaction, whereby it would speculatively try to execute the critical section for a limited number of times, and fall back onto acquiring the lock in case of failure.

Note that even without contention, running a critical section speculatively could improve performance since the cache lines holding the lock variables don't need to be modified by the cache coherence protocol, but only read (shared).

Improving Existing STMs

Some blocking STMs, such as TL2, DSTM2   Shadow Factory, and the blocking version of NZSTM require that a lock is held for a short period of time or that the system block for a short period.

If the short blocking critical section is replaced with a ROCK transaction then that should improve performance. Moreover, if there are guarantees that a certain type of transaction will eventually complete then the blocking STM could be essentially made nonblocking.

Hybrid Transactional Memory

Future topic...

References

http://en.wikipedia.org/wiki/Rock_processor

Moir, Moore & Nussbaum: The Adaptive Transactional Memory Test Platform (Required)

Dice, Herlihy, Lea, Lev, Luchangco, Mesard, Moir, Moore & Nussbaum: Applications of the Adaptive Transactional Memory Test Platform (Recommended)