Power Reduction with Transactional Memory

Tali Moreshet*, Maurice Herlihy†,
R. Iris Bahar* and Richard Weiss‡

*Brown University, Division of Engineering
†Brown University, Department of Computer Science
‡Hampshire College, School of Cognitive Science

Motivation

- N threads running on N parallel processors execute code
- Only one thread is allowed in the critical section at a time
- Coarse grain lock
  - Easy to implement
  - Not scalable
  - Limits parallelism
- Fine grain lock
  - Hard to program
  - Scalable
  - Enables parallelism

```
increment()
{
    tmp = value;          }
    tmp = tmp + 1;        }  critical section
    value = tmp;          }
    return value;         }
}
```

Lock Types

- Spin lock
  - On failure: repeatedly test lock (spinning, busy-wait)
  - Many main memory references
- Queue lock
  - Queue of threads waiting on a lock
  - Each thread spins on the lock of its predecessor
  - Fewer main memory references
  - Expensive to set up

Transactional Model

- Locks are conservative
- Locks are expensive
- Alternative to locks
- Transaction: Critical section *lock()* → *unlock()*
- Speculative execution – optimistic
  - No conflicts → commit
  - Conflicts detected → roll back, reissue
- Hardware requirements
  - Additional memory or dedicated cache (victim cache)
    - Storage area for old transaction data
  - Changes to cache coherence protocol
    - Data within a transaction not visible to others
    - Requests for ownership deferred

Transactional Modes

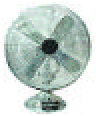
- WRITE
  - Transaction may modify memory location
  - No concurrent accesses
- READ
  - Transaction cannot modify memory location
  - May be read by concurrent transactions
  - Enables concurrent accesses to a tree-like data structure
- Other modes are useful for certain specialized cases
  - TEMP allows to read a memory location and then release it
  - Decreases the number of memory accesses

Alternative Techniques

- Lock-free - first priority
  Fall back - locking (in case of failure)
- Prioritize lock acquire requests
  Delay the low priority requests
- Predict data for critical section
  Forward with lock transfer
- Our work based on:
  - "Transactional Lock-free Execution of Lock-based Programs", Ravi Rajwar and James Goodman, ASPLOS 2002.
  - "Transactional Memory: Architectural Support for Lock-Free Data Structures", Maurice Herlihy, J. Eliot B. Moss, ISCA 1993.
- What about power?

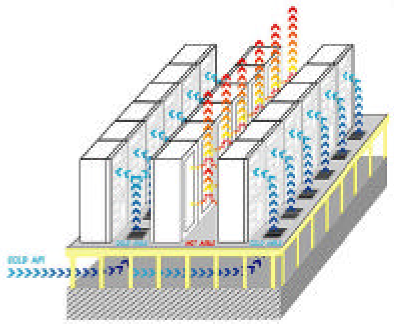
Data Center

Power Consumption



- Large embedded systems
  - Disk arrays
  - Multiprocessor blade servers
  - Multi-CPU network routers
- Data center:
  Frames of tightly packed boards with multiple CPUs and memories
- Cooling problems
  - Fans within a frame
  - Outside air conditioning
- Power supply problems
  - Increased by cooling power requirements
  - Require specially equipped building to meet power demands

Hot-Cold Aisle Cabinet



Transactional Memory and Power

- Main memory accesses
  - Reduce performance
  - High power consumption
- Transactional memory
  - No locks → Fewer memory accesses

    But…
  - May require roll-back and re-execution

    → Re-fetch data from main memory

    OR

    → Fetch data from other local cache

    → Write buffer holding old data
- Other synchronization techniques share similar power issues

Method and Goals

- Our goal:
  Compare power dissipation of locking and transactional models
- Benchmarks:
  - Synthetic micro benchmarks
  - Larger benchmarks from SPLASH (?)
- Is one approach better than the others when power is considered?
- The relationship between power and performance is not well understood