Threads and Cores

Marc Tremblay, Ph.D.
Sun Fellow & Vice President
Chief Architect – Scalable Systems Group
Niagara – 32 threads

Core 8
- Thread 4
- Thread 3
- Thread 2
- Thread 1

Core 7
- Thread 4
- Thread 3
- Thread 2
- Thread 1

Core 6
- Thread 4
- Thread 3
- Thread 2
- Thread 1

Core 5
- Thread 4
- Thread 3
- Thread 2
- Thread 1

Core 4
- Thread 4
- Thread 3
- Thread 2
- Thread 1

Core 3
- Thread 4
- Thread 3
- Thread 2
- Thread 1

Core 2
- Thread 4
- Thread 3
- Thread 2
- Thread 1

Core 1
- Thread 4
- Thread 3
- Thread 2
- Thread 1

Time

Memory Latency
Compute
Servers

• Tons of threads available
  > Multi-threaded is also multi-process

• Examples
  > J2EE based application server
    > Already scales to tens of threads today
  > Large database
    > Easily scales to tens of threads today
  > ECAD
    > 100k jobs per day in our server ranch
  > Searches
    > Billions of threads/processes
Single Application

• MAJC started in 1995
  > Dual core, running Java, lots of “compute”
  > Multithreaded program “will be there”
  > Not so on the desktop -> speculative multithreading (Space Time Computing)

• Current simulator for high-end Sparc CMT which has tens of threads
  > Written in Java
  > Cores are instantiated
  > Not multithreaded yet!
Opportunities

• Communication latency is much better
  > From 300-500 cycles (off-chip), to single digit (in core) to 20-25 (on-chip)
  > Economics of parallelizing, amortizing thread creation are greatly improved
  > Scalability is much better
  > Software bottlenecks are exposed!
Opportunities

• Synchronization
  > Lots of work on transactional memory, lock elision, etc.
  > Great for
    > JVM
    > Thread-safe (too safe) libraries
    > Kernel
    > Multi-threaded programs
  > Also great for thread-level speculation
    > Space Time Computing relied on tight atomics between on-chip cores...
Conclusion

• We have to help programmers
  > Language
  > Transactional memory
  > Automatic parallelization
  > Speculative multithreading