

Tera Hardware- Software Cooperation

Gail Alverson, Preston Briggs, Susan Coatney, Simon Kahan,
Richard Korry

presented by Gary Jackson

Basic Idea

- Goals:
 - Easy Programming
 - Fast Execution
- Implementation:
 - Uniform memory access
 - Cheap multithreading
 - Software to exploit the features

Important Features: Execution

- A “stream” is a hardware context for a thread
 - Program Counter
 - Register file of 32 registers
- As many as 128 streams per processor

Important Features: Execution

- Processor issues one instruction word per cycle
- Memory accesses have latency
- But the processor can execute an instruction from another stream immediately

Important Features: Execution

- Very Large Instruction Word (VLIW)
- Three instructions per word
- Lookahead value for the next time that the result of the operation is needed
- Atomic fetch-add instruction

Important Features: Memory

- Uniform access
 - No cache, no locality
 - High latency (150+ cycles)
- Four state bits for synchronization and other important features
- Multiple hardware protection domains per processor

Compiler

- Current compiler technology drives the architecture, not the other way around
- What does the compiler need to do?
 - Exploit parallelism: loops and pipelining
 - Reduce work: traditional optimization

Compiler

- What doesn't it need to do?
 - Distribute data
 - Manage cache
 - Branch prediction
 - Structural hazards
 - Like the delay slot following a branch in many RISC architectures ...

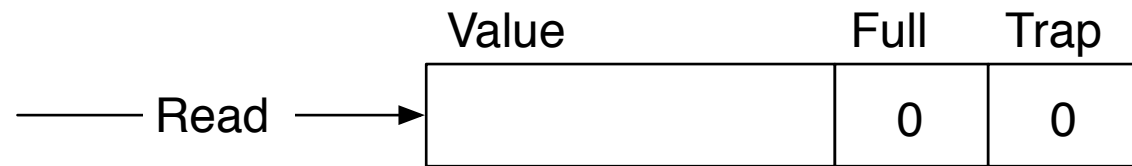
Runtime

- Many typical OS features are handled at the user level with the Runtime
- For instance, traps are handled at user level

Lightweight Synchronization

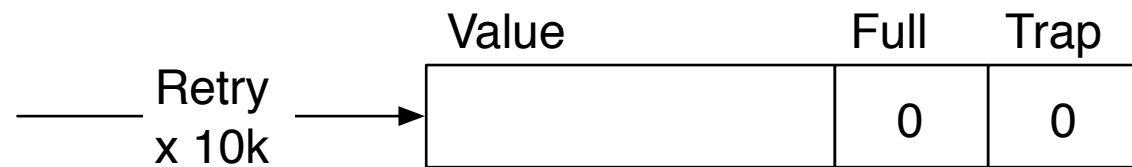
- Uses hardware flags in the memory to manage synchronization on a word-by-word basis
- Example: synchronized memory access

Lightweight Synchronization



Read can't
complete since the
full-empty bit is
unset

Lightweight Synchronization



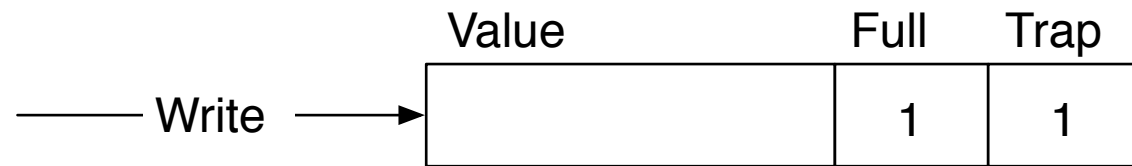
Read automatically
retries some fixed
number of times

Lightweight Synchronization

Value	Full	Trap
	0	1

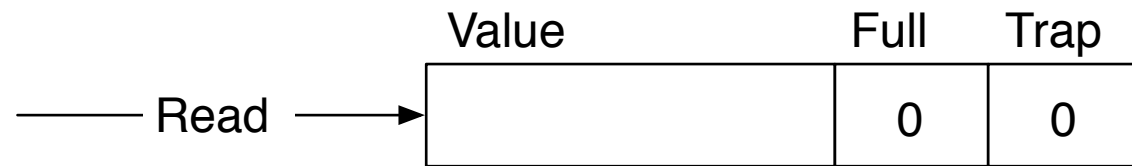
The retry count is exceeded, and the trap bit is set for when the new value is written

Lightweight Synchronization



Write sets the full-empty bit and fires the trap since the trap bit is set

Lightweight Synchronization



Read completes and resets the full-empty bit. If this was the only pending read, it will also reset the trap bit.

Stream Management

- Daemons
 - Auxiliary streams that service the program
 - Growth, for allocation and management, as well as debugger and profiler daemons
- Swapping (stopping and starting a process)
 - All streams store and reload their own state

Dynamic Allocation

- Growth daemon manages dynamic allocation of streams
- Created when the queue of future computations is accessed, since a trap is fired (the second trap bit is set)
- Slows its own execution by masking all traps and accessing empty memory in a way that would block

Operating System

- Microkernel
- No Hardware Interrupts
 - Privilege change done by jumping to fixed entry points instead of causing a system call interrupt
- OS uses daemons for performing services as well

Operating System

- Multiple Protection Domains
 - Roughly correspond to processes
- Multiple schedulers
 - Interactive: one processor, many domains
 - Batch: many processors, few domains

Debugger

- Implemented through the debugging daemon, or “nub”, running in the subject process
- Communication from debugger to nub is through IPC
- Expression evaluation happens in the nub
- The nub attempts to protect itself with a separate malloc pool

Breakpoints

- Usual breakpoint method might cause streams to miss a breakpoint while an instruction is rewritten
- Instead, the break instruction causes a trap, and instructions in an “out-of-line” buffer are executed before jumping back to the instruction after the break

Watches

- Another feature implemented on the second trap bit ...
- This means watches are efficient on the MTA
 - As opposed to setting a breakpoint after every instruction and comparing values
- Useful debugging tool on the MTA

Summary

- Software driving the implementation
 - Ease of use
 - Compiler technology
- Unique features
 - Fast thread switching
 - Tagged memory
 - Uniform memory multiprocessor

Where are they now?

- SGI bought Cray Research in 1996
- SGI created a separate Cray Research unit in 1999
- Tera bought Cray Research in 2000 and renamed itself to ... Cray
- Burton Smith (Tera founder) is working on the Cascade initiative for DARPA, says that multi-threading is a part of Cascade

Questions

- What is the difference between a thread and a stream?
- A thread is a task. A stream is the hardware support for a single thread on the processor. There can be more threads than streams.

Questions

- What about virtual memory?
- If I had to guess, I'd say virtual memory is probably implemented through the Protection Domain abstraction. I haven't seen anything to conform or elaborate on this, though.

Questions

- What is a future?
- A future is a computation and a place marked to store that computation. Future reads block if the value has not been computed yet. Using futures precipitates the creation of the Growth daemon.

Resources

- Cray.com for some history
- www.cs.ucsd.edu/users/carter/Papers/siam99.ps
- Very short