

Low Overhead Interrupt Handling with SMT

CS252 Spring 2005 – Final Presentation

Greg Gibeling
Andrew Schultz

Problem

- Traditional OS interrupt handling
 - High latency context switch
 - Excessive data copies
- High bandwidth devices I/O architecture
 - Ethernet speed increasing faster than processor
 - 1 Gb/s Ethernet faster than a standard architecture
 - 10 Gb/s Ethernet becoming common
- Existing Attempted Solutions
 - Interrupt coalescing – poor latency
 - TOEs – not well supported, application specific
 - Smart NICs – typically for special purpose (MPI)
 - Polling – Unreliable, machine dependant

Observations

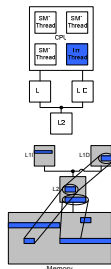
- Trend towards many hardware contexts
 - Intel now has HT standard in their cores
 - Multi-core chips starting to ship
 - Multiprocessor SOC's
- Many contexts
 - Easier to partition
 - How do we take advantage?
 - Devoting one to interrupts wont reduce processing
- Programming inertia slows changes
 - Programmers use certain interfaces
 - Solutions should not require massive changes
 - Major changes take forever to be adopted
 - TOEs
 - Have been around for a while
 - Still don't have wide-deployment

Solution (1)

- SMT Hot Thread for Interrupts
 - Accelerate Interrupt Handling
 - Remove context switch overhead
 - Pin cache contents
 - Allow OS control
 - Increase interrupt handler flexibility
 - Reduce hardware dependence (Smart NIC)
 - Use programmer intuition about program structure
 - Very high level cache prefetch hints
 - Zero Copy (future work)
 - Data is placed directly in user space
 - Does not require a large physical memory block

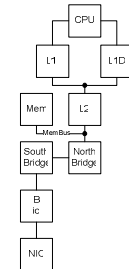
Solution (2)

- SMT Hot Thread
 - Reserved
 - One of many threads
- Cache Pinning
 - Based on LRU
 - N-way associative
 - Limited set pinning
 - L1 Instruction
 - Trace pinning
 - L2 Combined
 - Range Pinning



Experimental Setup

- M5 Simulator
 - Multiple full system simulation
 - Event driven memory system
 - Cycle accurate in-order or OoO
- Our Parameters
 - Current Pentium4 like system
 - Alpha ISA (EV6 211264)
 - Simple CPU model
 - Multiple contexts to simulate SMT
 - Full Memory Model



Benchmarks

- Netperf
 - Basic UDP traffic flood
 - Simulates high network load, low CPU load
- Netperf (w/NAT)
 - Insert a NAT box between server and client
 - Simulates high network load, high CPU load

5/10/2005 CS252 Final Presentation 7

Instrumentation

- Simple in-order SMT model
 - Lower potential pressure on memory system than aggressive out-of-order model, but sufficient for measuring functional memory access
- Instrumented Linux kernel
 - Create "bins" to measure statistics in different parts of the OS (kernel, user, interrupt, tasklets)
 - Capture SMT thread to service interrupts and run NIC related tasklets
- Created "cache pinning" policy
 - Replaces basic LRU replacement policy
 - Pseudo-instructions explicitly pin data structures or turn on trace pinning (in I or D cache)

5/10/2005 CS252 Final Presentation 8

Instrumentation

- Metrics
 - Cache Misses (L2/I/D)
 - M5 allows binning, in order to separate misses during the interrupt handler from other code
 - Measure impact of cache pinning
 - NetPerf bandwidth
 - Application level performance
 - "What can this accomplish?"

5/10/2005 CS252 Final Presentation 9

NETPERF Performance (1)

5/10/2005 CS252 Final Presentation 10

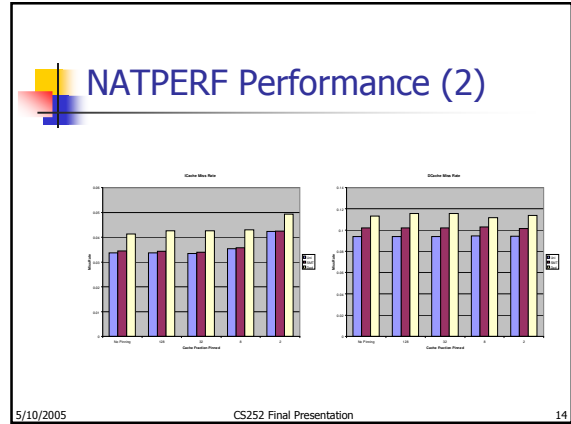
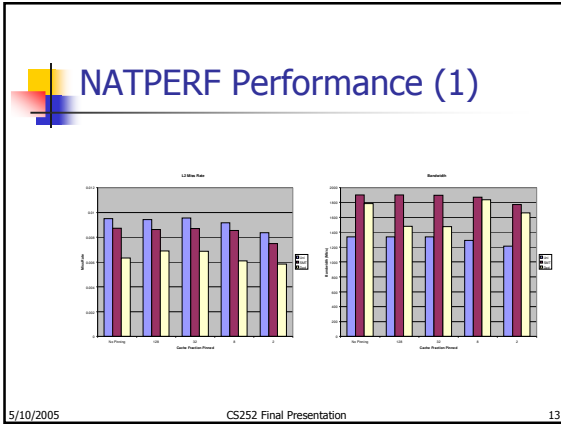
NETPERF Performance (2)

5/10/2005 CS252 Final Presentation 11

NETPERF Results

- Bandwidth increase of 30%
 - Dedicated SMT thread successful
- Cache performance is even
 - Cache Pinning
 - L2 - No Impact
 - ICache - Performance Hit
 - DCache - Performance Hit
 - This is an IO bound benchmark

5/10/2005 CS252 Final Presentation 12



- ## NATPERF Results
- Bandwidth decrease
 - All processing for NAT is handled in kernel
 - Dedicated thread is doing all the work
 - Cache performance improvement
 - Cache Pinning
 - L2 – 30% improvement at 1/2 pinned
 - Note that this corresponds to bandwidth peak
 - ICache - Performance Hit
 - DCache - Performance Hit
 - Processing dependent benchmark
- 5/10/2005 CS252 Final Presentation 15

- ## Conclusions
- SMT Hot Thread
 - Definite success
 - Requires a multithreaded application
 - 130% speed increase on NETPERF
 - Cache Pinning
 - Qualified Success
 - Helpful at the L2 level for NATPERF
 - Seemingly detrimental at L1
 - Deserves more testing
- 5/10/2005 CS252 Final Presentation 16

- ## Future Work (1)
- Processing Intensive Benchmark
 - NETPERF
 - Excessively simple
 - Doesn't include processing overhead
 - NATPERF
 - Single threaded
 - Doesn't fully test an SMT system
 - SPECWeb99
 - Died horribly on the M5 system
 - This might provide a better benchmark
- 5/10/2005 CS252 Final Presentation 17

- ## Future Work (2)
- Detailed profiling of memory usage
 - Determine optimal data structures/traces to pin
 - Examine performance with OOO CPU model
 - Ready, but would require massive sim time
 - Extend basic architecture
 - Implement zero-copy protocol
 - Preferably consistent with traditional interfaces
 - Comparisons
 - Click – Can this match polling performance
 - IDS/Package Capture – Can this handle line speed
- 5/10/2005 CS252 Final Presentation 18



Related Work

- Hardware
 - EMP – True zero copy on NIC
 - SMT exception handling – Done for TLB misses
 - On-chip NICs – Tighter coupling between NIC and CPU
 - TOEs and accelerators – Shift processing to special devices
- Software (OS)
 - U-Net – ADU + fewer copies, more application control
 - Click – Change method from interrupt to polling
 - Interrupt coalescing – Add latency, reduce thrashing
- Commercial Products
 - There are commercial products approaching this solution
 - Network Analyzers
 - IDS Systems