A Case for User-Level Interrupts

Mike Parker
School of Computing, University of Utah
map@cs.utah.edu
http://www.cs.utah.edu/~map
Introduction

• Motivation
  - I/O bandwidths and frequencies growing
  - Interrupts used to be infrequent (legacy)
  - Interrupts (OS) cause significant cache misses
  - Misses scale at DRAM latencies
  - Gets worse over time (memory gap)

• Deliver interrupt directly to user process
  - Avoid OS overhead
  - Signals in Unix appear as interrupt to user process
  - Let HW do similar things w/o OS help
Goals

• Keep OS overhead minimal
  - Avoid involving OS
  - Lazy involvement otherwise

• Keep general-purpose architecture
  - Avoid a custom “I/O” processor
  - Keep in mind current and near future architectures

• Keep general-purpose OS
  - Arbitrary user-level interrupt handlers
  - Support existing programming models
  - Maintain Unix-like protection
Interrupt Mechanisms

• Superscalar processors
  - “Asynchronous” branch
  - Legacy style interrupt, deliver directly to destination process
  - Policies control “if”
  - Notify OS (in lazy fashion) if not delivered

• Multithreaded processors
  - Asynchronous branch
  - Modify runability of threads
  - Schedule a pre-existing or new thread
Related Work

• **User-level I/O**
  - User-level communication (U-Net, M-Machine, etc.)
  - User-level control (CSB, ...)
  - ...

• **OS**
  - Too much to mention

• **Interrupts**
  - Interrupt coalescing
  - User-mode interrupts (FLIP - Henry)
Questions?

http://www.cs.utah.edu/~map

map@cs.utah.edu
Anatomy of an Interrupt

- Sun Ultra 1, Solaris 2.5.1
  - 119 µs interrupt latency (~17500 cycles @ 147 MHz)
  - 380 L2-cache misses @ 270 ns / miss
  - 103 µs or 87% in cache misses
Simulator

• Extending L-RSIM (RSIM based)
  - Accurate cache, memory bus, MMC, I/O bus, and device models
  - Runs extensive BSD-based kernel
  - Unmodified Solaris binaries

• Look at tomorrow’s architecture
  - 2-4 GHz (MIPS style)
  - 32k - 128k L1
  - 4M - 16M L2
  - I/O Infiniband/U-Net-ish