Operating System Support for Performance Monitoring

Witawas Srisa-an

Chapter: not in the book

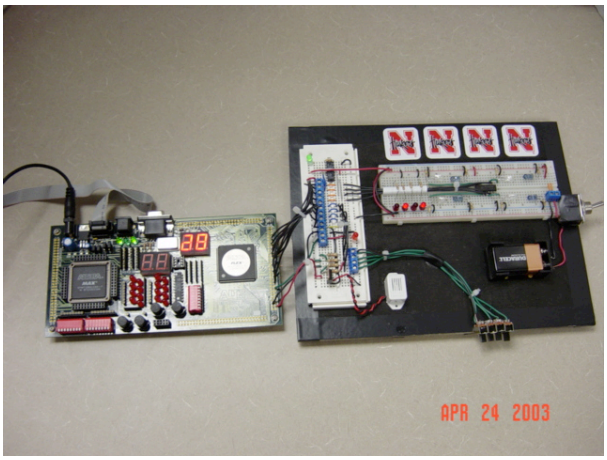
Imagine …

- Tom is deploying a new server program for his client. During the field test, he notices that its performance is below his expectation. How would he identify the main causes?
- A company is planning to upgrade the server systems. You are asked to analyze which systems and which configurations will perform best when executing its main point-of-sale application. How would you perform the analysis?

Imagine …

- You are deploying an embedded computer system that will run only 3 applications simultaneously. You are asked to hand tuned these three applications to minimize:
  - Instruction count
  - Execution time in cycles
  - L1 data cache misses
  - Pipeline stalls, etc.
- You want to compare the performance of your very own *malloc* function to existing techniques. How would you conduct the comparison?

Remember?



Outline

- What are Model Specific Registers (MSRs)
- Why are they there?
- How do we use them in the OS context?

MSRs

- Programmable registers to satisfy performance cognizant consumers
  - A small number (2 in P1, 4 in Athlon, 8 in P4)
  - Can count several on-going events
  - Each counter has a control register
    - Set interrupts on overflow
    - Cycle detection versus event detection
    - Work in various privilege levels
    - Set the event(s) to monitor

MSRs

- Currently supported by many architectures
  - AMD Athlon, Opteron
  - HP (DEC or Compaq) Alpha
  - Intel Itanium, Pentium
  - Sun UltraSparc II
  - Many more

MSRs

- Intel
  - P1: RDTSC and 2 MSRs
  - P4: RDTSC and 9 registers to monitor 48 events
- AMD
  - Athlon & Opteron: RDTSC and 4 registers to monitor 58+ events

MSRs

- L1 Misses
- Load-Store Buffer Full
- Dispatched Floating Point Unit operations
- Cycle counter
- many many more events

MSRs

- RDTSC (in X86)
  - A special 64 bit register
  - Stored the cycle counter value in eax and ecx
  - Start from zero when the machine is booted
    - For a 3GHz, takes about 195 years to roll over

MSRs

Sounds great! What's the catch?

- Not well documented
- No APIs support
- Change frequently
- Last thing on the (long) to-do list of the hardware engineers

No More Wall Clock Time?

- Yes, cycle counting is more accurate
- Yes, monitoring micro-architecture events give us more insights
- No, the information is still system wide.
  - Getting L1 misses in a system with 105 processes. What does that tell you about each process?

Performance Monitoring in the OS Context

- The focus of Programming Assignment 2
  - To obtain events information per process
    - Simplified by using RDTSC
    - Create new system calls to provide per process information

What do we need from the OS?

OS support