CPU Shielding:
Investigating Real-Time Guarantees Via Resource Partitioning

John Scott Tillman
jstillma@ncsu.edu

CSC714 Real-Time Computer Systems
North Carolina State University
Instructor: Dr. Frank Mueller

MOTIVATION

Recent years have seen an explosion in the parallelism available in COTS systems. Dual core technologies have become commonplace in the typical desktop computing system. This popularity alongside ever decreasing manufacturing costs has rapidly lowered the costs associated with these technologies. This type of single chip parallelism will only become more pervasive as techniques are discovered to ease the challenge of developing for parallel architectures.

The trend in computing systems is towards larger, more complex, more highly integrated systems. From aircraft fly-by-wire, to ever more complex medical imaging and life support, these systems bring together many highly diverse requirements. Only some of these requirements have real-time characteristics (whether hard or soft), while others can be considered improvements to quality of service. The ability to leverage existing applications for and developer knowledge of 'standard' operating systems greatly reduces the costs of real-time system development.

Therefore providing real-time operation alongside today's highly complex operating systems is an idea unlikely to go away. The idea of *CPU shielding* has been put forward as a method for co-hosting a standard operating system while still providing for real-time guarantees. In CPU shielding a subset of a system's CPUs are isolated from all standard processing, and can be dedicated exclusively to real-time tasks. This isolation allows the performance and behavior of the real-time subsystem to be evaluated in (relative) isolation, with minimal, and predictable interference from the non-real-time portions of the system.

CPU shielding has received relatively little attention as a means to achieve hard real-time performance while still maintaining interoperation with standard non-real-time subsystems.

OBJECTIVES

This project seeks to investigate the feasibility and limitations of using CPU shielding to allow hard real-time operation in commercial, off-the-shelf (COTS) systems. This will be done by bounding and verifying worst case interrupt response times (CPU contention), worst case bus reaction times (bus contention), and worst case slowdown associated with additional cache misses (cache contention). There are existing documents which discuss various models of these delays. The goal is to verify the models/predictions and evaluate the predictability that can be achieved using this co-hosting method.

RESOURCES

The target system for this project is an AMD Athlon X2 based system, available in NCSU's operating systems lab. The current Linux install will be replaced as needed to create a test bed for verifying and measuring the system behavior under CPU shielding. Measurements will be taken, when possible using the built-in facilities of the processor

and platform. Where needed external timing measurements will be performed using a logic analyzer. Depending on time constraints some additional PCI based hardware may be installed to increase bus loads for dynamic testing.

TIMELINE

March 16th	Project proposal submission
March 22nd	Replicate system setup from [1]. Evaluate literature predictions of interference from known sources (PCI, Interrupt and Cache induced)
March 29th	Design/replicate experiments to test latency from known sources
April 2nd	Initial Project Status Report
April 5th	Gather and evaluate initial test results. Identify and categorize unknown latency factors.
April 19th	Demonstrate mixed (real-time and non-real-time) mode operation at predicted highest frequency.
April 21st	Final Project Status Report

RISKS

There are quite a few sources of uncertainty in the timeline provided. Very little exists to show how CPU shielding should be setup. This may lead to difficulties in configuring a baseline system with which to perform the later evaluations. Once the CPU shielding can be performed there is a high likelihood of unknown sources of latency. There is also some question as to how to go about measuring some latency periods given the time and equipment available. These factors may limit the practicality of using this co-hosting method.

WEBSITE

http://www4.ncsu.edu/~jstillma/csc714/

REFERENCE WORKS

[1] S. Brosky. *Shielded CPUs: real-time performance in standard Linux*. Linux Journal, May 2004, pg 121

[2] M. Caccamo. *Toward the Predictable Integration of Real-Time COTS Based Systems*.

[3] R. Pellizzoni, B. Bui, M. Caccamo, L. Sha. *Coscheduling of CPU and I/O Transactions in COTS-based Embedded Systems*. Real Time Systems Symposium, 2008

[4] T. Huang, J. Lui, J. Chung. *Allowing cycle-stealing direct memory access I/O concurrent with hard-real-time programs*. International Conference on Parallel and Distributed Systems, Tokyo, 1996.

[5] S. Schönberg. *Impact of PCI-bus load on applications in a PC architecture*. Proceedings of the 24th IEEE international Real-Time Systems Symposium, Cancun, Mexico, December 2003