A high performance commodity I/O controller for storage virtualization

FORTH Institute of Computer Science

M. Fountoulakis, M. Marazakis, M. Flouris and A. Bilas

{mfundul,maraz,flouris,bilas}@ics.forth.gr

I/O controller for storage virtualization

Motivation

- Increasing requirements for capacity and performance
- Today storage systems offer simple virtualization only
  - Volume management, redundancy
- Applications today require more advanced virtualization
  - Protection from user errors
    - Transparent Versioning, ease of use at controller level
  - Protection from bit errors
    - Error detection codes, persistent in storage
- Guarantees on longevity of data

Challenges

- Applications become more demanding
- Efficient utilization of controller resources
  - Actual performance is inferior to aggregate performance of the hardware components
  - Controller CPU the most valuable resource
- Complexity to program in an efficient way
  - Adding or removing functionality is costly
- Controller must be transparent to the host
  - Host should only see physical disks
  - Controller must provide virtual functions

Proposed solution

- Design and implementation of a high performance, extensible storage controller
  - Capable of GB/s I/O performance
- Firmware running on the controller
  - Structured in a modular framework
  - Extensive scheduling of tasks to efficiently use scarce controller resources
- Host-controller protocol
  - Large number of outstanding I/Os
  - DMA vs PIO, throughput vs latency, trade-offs

Design Outline

I/O Controller: Intel IOP348 I/O CPU

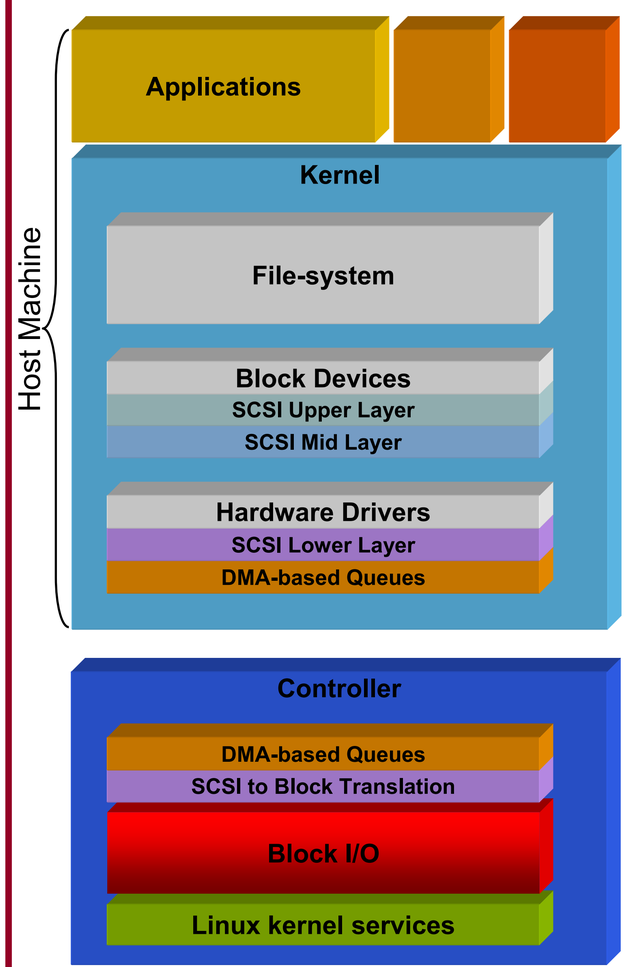
- 2 XScale 1.2 GHz cores
  - Application/Transport processor
- 1 GB DDR-SDRAM
- 3 Application DMA channels (ADMA)
  - XOR, CRC-32C calculations
- 8 SAS/SATA ports
- PCI Express x8 slot
- Address Translation Unit (ATU)
  - Contains the Messaging Unit (MU)

Host-controller protocol

- Host driver sends SCSI commands to controller, receives I/O completions
- Circular FIFO queues used for SCSI commands & completions
- Commands can be transferred by using either PIO or DMA
  - DMA requires the controller to poll host CPU for available commands
  - Host PIO has lower latency than DMA and reduces work in controller
- Completion FIFO mirrored between controller, host memory
  - Completions transferred via DMA or via PIO
  - Piggy-backed to actual data transfers when using DMA

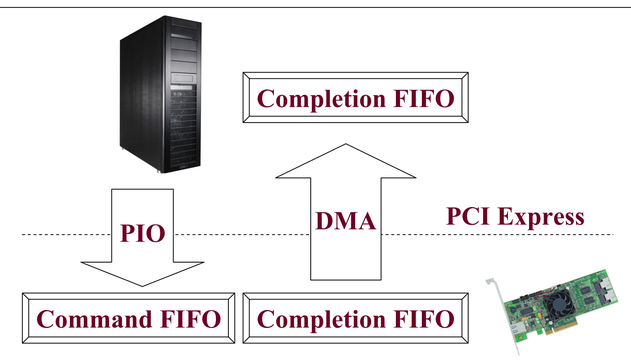
SCSI to block I/O command translation

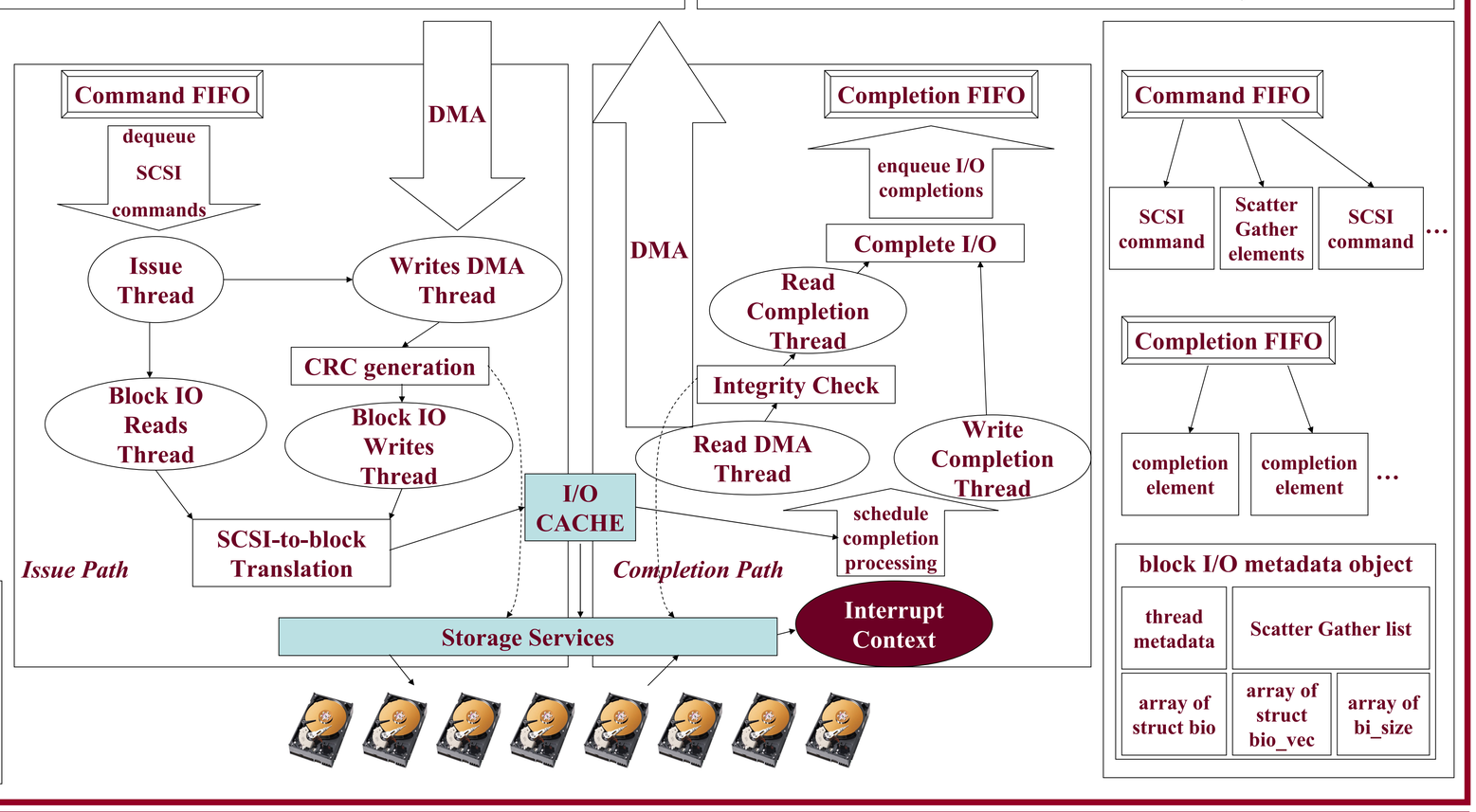
- Commands received are translated from SCSI to Linux block-layer I/O
- Takes advantage of the embedded Linux and the Violin extensible framework
- All operations take place in kernel thread context as opposed to interrupt context
  - Allows context switches when waiting for software or hardware resources to become available
  - Simplifies explicit management of the I/O stack



I/O cache design:

- Replacement policies
  - LRU/direct mapped
- Hit under miss
- Write back, write through
  - Write-allocate, Write-no-allocate
  - Depends on reliability needs





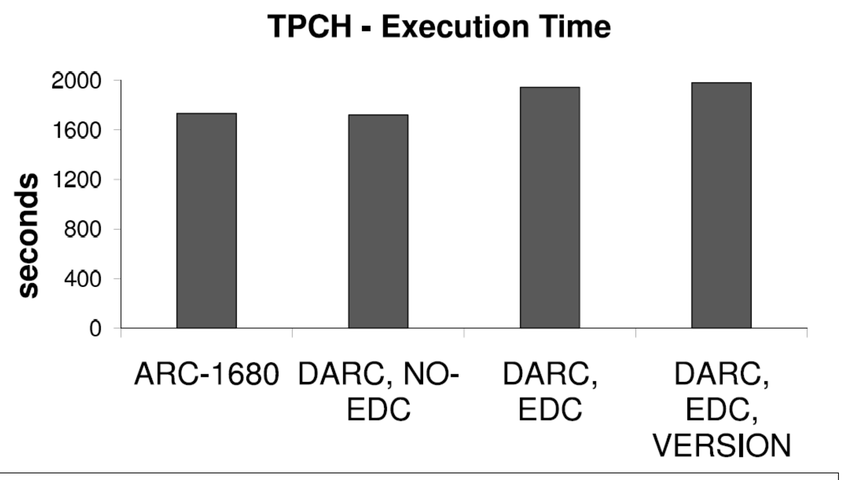
Open problem

- Transform design to incorporate an I/O cache
- Cache policy CPU utilization and hit-ratio tradeoffs
- I/O patter prediction, cache pre-fetching and flushing

Preliminary Experimental Results

Platform

- Host machine - MS Windows 2003 Server
  - 2 4-core Intel Xeon CPUs timed at 2 GHz
  - Tyan S5397 motherboard, 4 GB RAM
- Compare our base design, without cache, with the commercial I/O controller Areca ARC-1680, also based on IOP348
- 8 Seagate Cheetah SAS drives at 15k RPM
- TPCH benchmark (over RAID-10)[1]
  - Queries 1, 3, 5, 6, 7, 8, 11, 12, 14, 19
  - 4 GB dataset size, MySQL Server v5.1, NTFS
  - EDC: error correction & detection layer
  - VERSION: captures version every 60 seconds, purge of previours versions



[1]Markos Fountoulakis, Manolis Marazakis, Michail D. Flouris and Angelos Bilas: "Design and Evaluation of an I/O Controller for Data Protection", In: Proceedings of the 3rd Annual Haifa Experimental Systems Conference SYSTOR), 2010 (to appear).

Conclusions

- Our controller performs comparably to a commercial controller at realistic workloads
- 100% cache hit ratio synthetic workload
  - 1.6 GB/s for reads (ADMA max 1.6GB/s)
  - 1.1 GB/s for writes (ADMA max 1.4GB/s)
  - PIO affects DMA performance
- Maximum sequential I/O throughput for Seagate drives without cache
  - 1 GB/s for reads
  - 800 MB/s for writes

We thankfully acknowledge the support of the European Commission under the 6th and 7th Framework Programs through the STREAM (FP7-ICT-216181), HiPEAC (NoE-004408), and HiPEAC2 (FP7-ICT-217068) projects.

Computer Architecture & VLSI Systems Laboratory (CARV)