The CORFU Hardware Platform

Michael Wei, Mahesh Balakrishnan, John Davis, Dahlia Malkhi, Vijayan Prabhakaran, Ted Wobber
The I/O Story

Processors

Main Memory

Storage
The I/O Story

Disk Capacity
- 2011: 2 TB
- 1980s: 30 MB

Transfer Rate
- 2011: 150 MB/s
- 1980s: 2 MB/s

Latency
- 2011: 10 ms
- 1980s: 20 ms

Capacity/Bandwidth
- Large Transfers
  - 2011: 5,000s
  - 1980s: 15s
  - 2011 vs. 1980s: 300x Worse

- Small Transfers
  - 2011: 58 days
  - 1980s: 600s
  - 2011 vs. 1980s: 8,000x Worse
The I/O Story

- NAND Flash
  - 512 Gb
  - ~50 MB/s
  - 1,280 s @ 4k

- Phase Change
  - 128 Mb
  - ~50 MB/s
  - 0.32 s @ 1 byte

- STT-RAM
  - 4 Mb
  - ~200 MB/s
  - 2.5ms @ 1 byte
PCIe 3.0: 16 GB/s
iSCSI: 10 Gb/s (on 10GigE)

~320GB
$7,000
$21/GB

500GB-10TB
$10,000+
$20/GB

2 TB
$88,000
$44/GB

- Bottleneck
- Single Point of Failure
- Difficult to Scale
- Power-Inefficient
- Expensive
The CORFU Architecture

- No Bottlenecks
- Fault Tolerant
- Highly Scalable
- Low Power (10W /unit)
- Cheap (@ Cost of Flash)

Previously known as Falcon

Cluster of raw flash units
$4/GB
Outline

• The I/O Story
• CORFU Overview
• Hardware Platform
• Conclusion
Traditional Storage

- Ethernet
- Network Card
- PCIe
- Server
- SATA, SAS, PCIe
- FTL
- Flash (DATA)
- Resource Sharing
- Consistency
- Processing
- Load Balancing
- Flash Management
- Wear Leveling
- Garbage Collection
- Striping, ECC
The CORFU Architecture
The CORFU Architecture

Application

Resource Sharing
Consistency
Processing
Load Balancing
Flash Management
Decisions
Striping

Client Library

Shared Log

DATA

ECC
Flash Management
Garbage Collection
Wear Leveling
Logical to Physical Map
The CORFU Architecture

Application → Client Library → Shared Log

- Resource Sharing
- Consistency Processing
- Load Balancing
- Flash Management
- Decisions Striping

DATA → ECC
- Flash Management
- Garbage Collection
- Wear Leveling
- Logical to Physical Map
The CORFU Architecture

DATA
ECC
Flash Management
Garbage Collection
Wear Leveling
Logical to Physical Map

Write-Once, ∞ Address
Read
Trim
Management

Ethernet
FPGA
SATA
NAND
Outline

• The I/O Story
• CORFU Overview
• Hardware Platform
• Conclusion
The CORFU Hardware Platform

- 2 Prototype Systems
  - XUPv5
    - Virtex5 XC5VLX110T
    - 2 GB DDR2 RAM
    - 2x SATA 2.0
  - BEE3
    - Virtex5 XC5VLX155T x4
    - 8 GB DDR2 RAM
    - 8x SATA 2.0
    - 32/64 GB Flash DIMM
BeeHive Architecture

- Ring of simple RISC softcores (100MHz)
- Non-coherent caches, message-passing preferred
- GCC toolchain
- Specialized cores for Ethernet and memory
Extending BeeHive
• Add features by adding specialized cores
  • NAND Core
  • SATA Core
Hardware Architecture

- Using a traditional “Microprocessor” programming model was the wrong thing for the Beehive architecture.
Hardware Architecture

Message Passing API
- Request / Response “RPC”
- Interfaces for core “types”
  - Storage Core
  - Communications Core
  - Control Core
  - Metadata Core
  - Read Core
  - Write Core
• Upgrade of original BeeHive Ethernet Core
• Jumbo Frame support added, IP acceleration planned
Inbound packet handling

- Step 1: Packet from upper layers comes in from Ethernet PHY
Step 2: Communications core puts packet into memory using DMA
Inbound packet handling

• Step 3: Communications core sends a response message to the message processing core with the memory address of the packet
Message Processing Core

- Processes messages from the upper layer and dispatches
- Manages and constructs reply buffers
Step 4: Read Ethernet packet from memory and process
• Step 5: Dispatch a message to the write core
  Message contains memory address of buffer and logical address
• Step 6: Construct a reply buffer to the client while waiting
Write Core

- Step 7: Pick a physical address off the free-list, send the physical address and logical address to the metadata core
• Step 8: Metadata core checks its hash table which translates logical addresses to physical addresses.
  Cuckoo Hash – (with Udi Wieder) (GC is also done here)
• Write: Return ok if not written
  Read: Return physical address, if mapped
Storage Core

- Step 9: Pass logical and physical address to storage core, with the memory address of the data buffer
Step 10: Write to the underlying storage device, performing a DMA into the memory buffer given
- Step 11: Return completion to the Read/Write Core
- Step 12: Return completion to the Message Processing Core
- Step 13: Send Ethernet core a request to return packet
Performance and Power

- 800 Mbit/s on 1 Gb/s line, client saturated (100% CPU)
- XUPv5: 15 W (Lots of unneeded peripherals)
- BEE3: 12.5W / FPGA with 80 GB SSD and 32 GB FDIMM, 8 GB DDR2 RAM
- Server implementation: 260W Idle
- Power is relatively static, even with load
Conclusion

- Slow cores and lack of cache coherency provide problems
- Slow cores imply lower power (100MHz clock)
- Multiple cores allow for logical separation of tasks
  - Special-purpose metadata core does not need locking
- Easy to multiply cores in the system for performance
  - Removed bottleneck by simply duplicating a message processing core and multiplexing between two
- Easy to add hardware accelerators
  - Adding an LFSR and IP accelerator to perform checksums is a matter of a few lines of Verilog
- Can perform extra processing (cores are relatively cheap).
- Built an implementation of the CORFU Flash unit at very fast speeds using commodity hardware
Special Thanks:
Mihir Nanavati
The CORFU Project
Chuck Thacker
Andrew Birrell
Tom Rodeheffer
Roy Levin
Michael Schroder
Lori Blonn
Mihai Budiu
Everyone else @ MSR
Hardware Architecture