

A High Throughput Massive I/O Storage Hierarchy for PETA-scale High-end Architectures

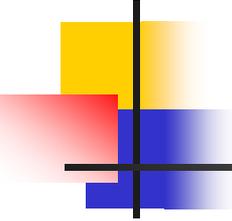
Brian Lucas (lucasb@udel.edu)

PI: Dr. Guang R. Gao

CAPSL Research Group

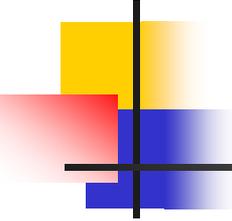
University of Delaware





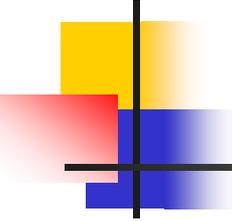
Problem

- Widening Performance Gap between Computing and I/O
 - HEC chips reach up to 100 GFLOP
 - I/O & storage performance needed to sustain calculations
- Modern High-end Computer (HEC) Architectures scale >10,000 nodes
- Nodes scale up in number of cores
 - Hundreds of thousands of total cores



FlashNode

- Target: Cellular Architecture
 - IBM Cyclops-64 (C64)
- Add a Flash SSD (Solid State Drive) to each node
 - Flash technology is rapidly developing
 - Use as File cache and Memory Extension
- I/O service threads at each node

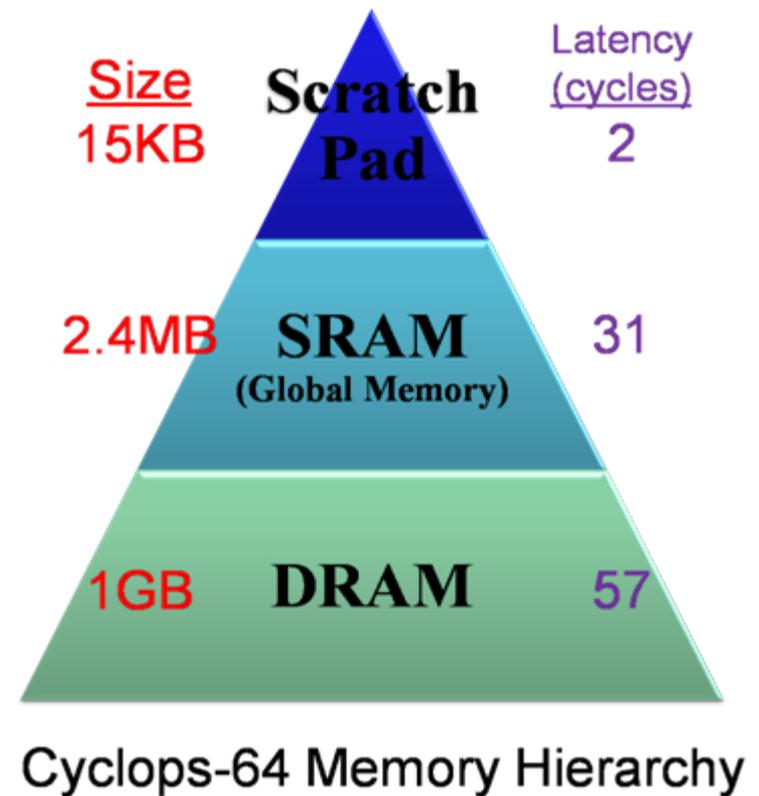


Goals

- Low Latency
 - Keep threads computing
- High IOPS
 - Many threads = many requests
- High Burst Bandwidth
 - Meet the burst nature of node I/O
- Effectively Utilize Fileserver bandwidth
- Scalability

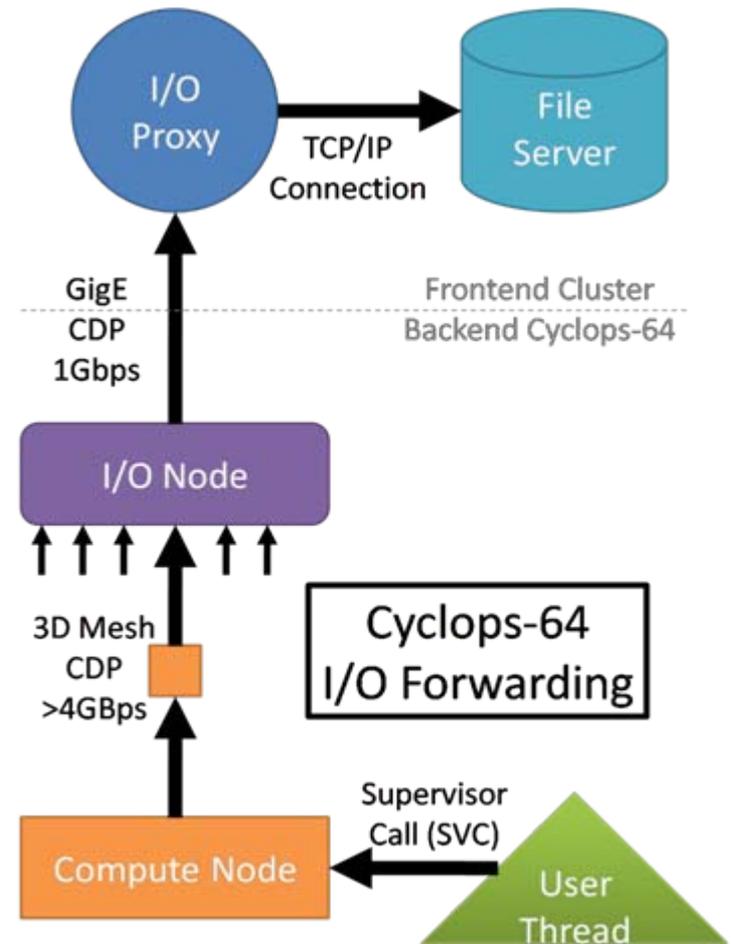
C64 Overview

- Each Compute Node
 - 1 C64 chip
 - 1 GB DRAM
- 24X24X24 3D Mesh
- 160 Thread Units/chip
 - Nonpreemptive threads
- Memory Hierarchy visible
 - No Data Cache

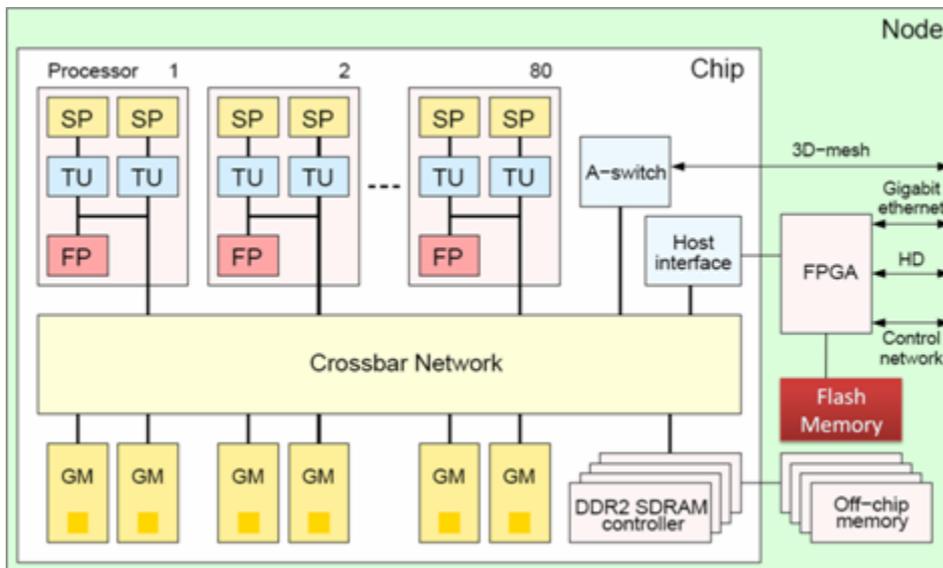


C64 I/O Overview

- POSIX API
 - Supervisor calls
- I/O Forwarding
 - Backend I/O nodes
 - Frontend cluster I/O Proxies
- GigE connects Frontend & Backend
- On-chip A-switch connects nodes



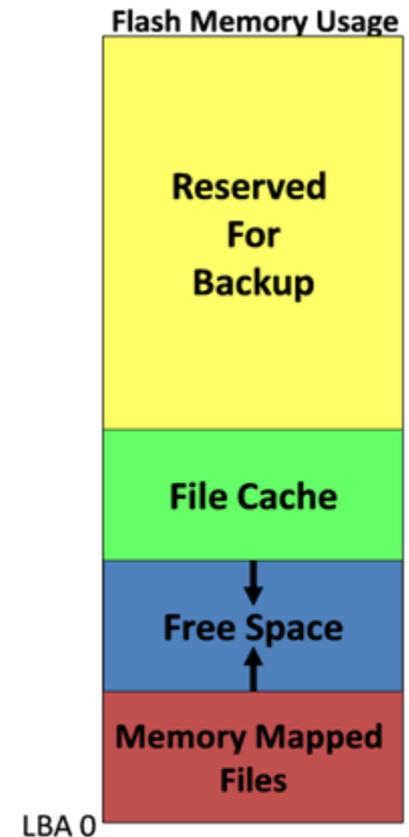
SSD Attached Node

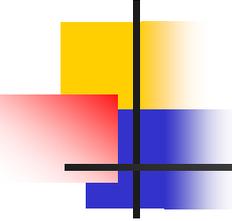


- Attach Flash SSD to the Node's FPGA
 - Host interface connection to chip
- FPGA configured as SSD driver
- 1 or more Dedicated Service Agent (DSA) I/O threads

Roles of Flash and DSA

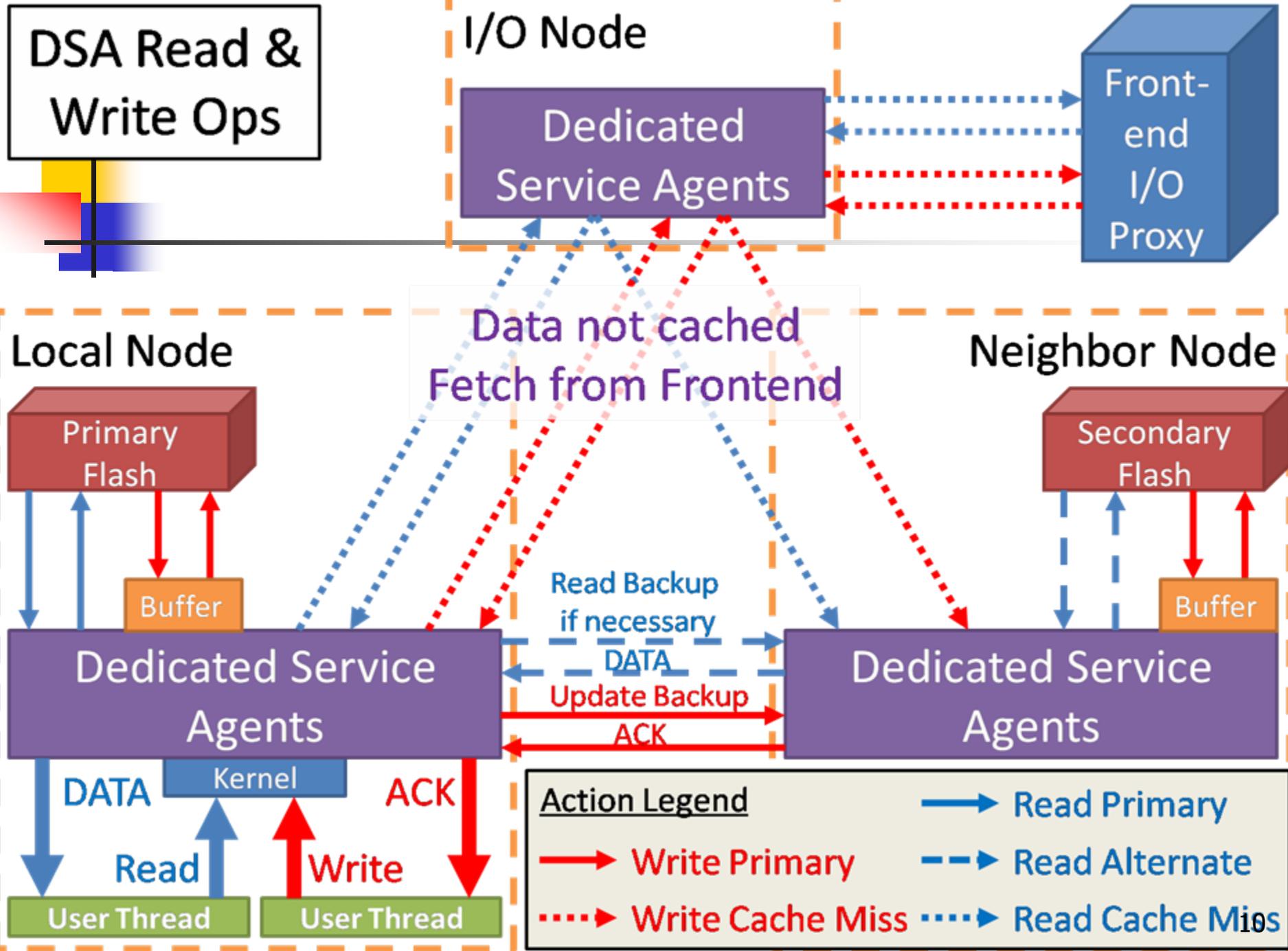
- Flash:
 - File cache for local node
 - Memory mapped files
 - Mirrored backup of neighbor
- DSA:
 - Service local requests
 - Manage flash usage (above)
 - Monitor network & flash traffic
 - I/O Node DSA: handle frontend & internode communication





POSIX File Caching

- Transparent replacement for current I/O Forwarding model
- All data in at least 1 of 3 places
 - Primary Flash on local node
 - Secondary Flash on neighbor node
 - Frontend file server
- DSA services all requests from a queue
- DSA uses nonblocking I/O to Flash, I/O Node, and neighbor node



DSA Read & Write Ops

I/O Node

Dedicated Service Agents

Front-end I/O Proxy

Local Node

Primary Flash

Buffer

Dedicated Service Agents

Kernel

Read

Write

User Thread

User Thread

Neighbor Node

Secondary Flash

Buffer

Dedicated Service Agents

Data not cached
Fetch from Frontend

Read Backup if necessary

DATA

Update Backup

ACK

Action Legend

Read Primary

Write Primary

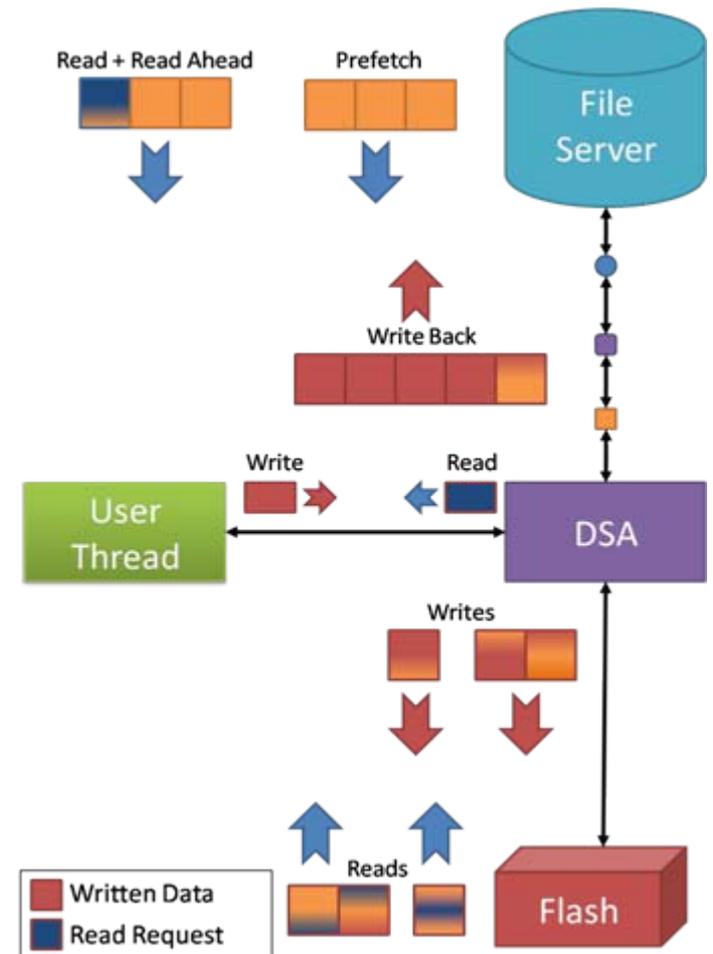
Read Alternate

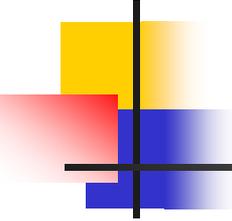
Write Cache Miss

Read Cache Miss

Prefetching & Writeback

- Read ahead multiple blocks from frontend
- Opportunistically prefetch frontend
- Lazy write back
- Transfer whole blocks from Flash
 - Combine requests in block if possible

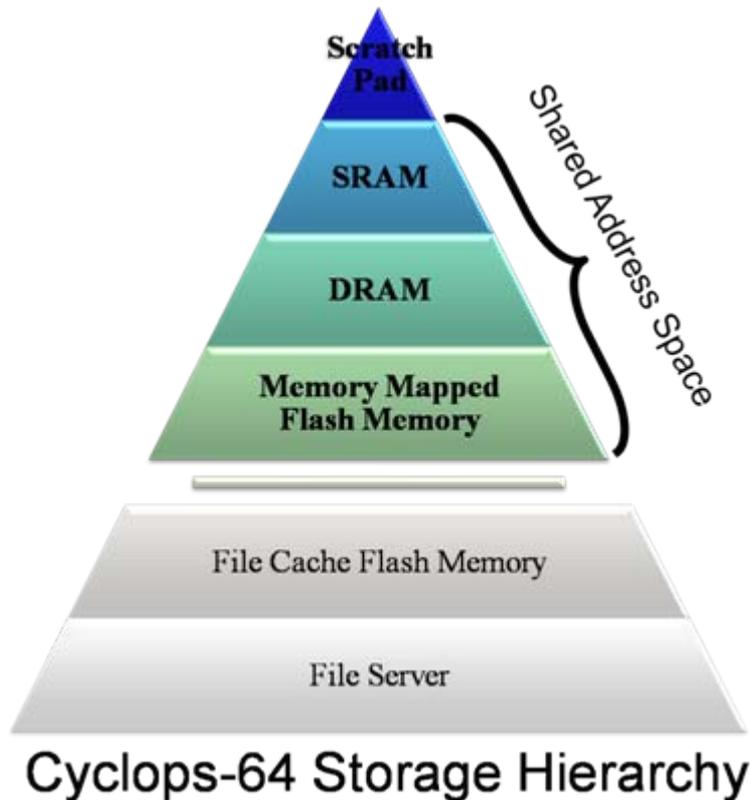




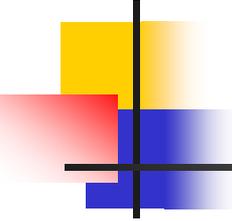
Collective I/O

- Use collective calls & file views similar to MPI-IO
 - No MPI Implementation for C64
- Maintain same interface to frontend
- Communicator hierarchy
 - Threads on nodes
 - Nodes on same I/O node
 - Far nodes (on other I/O nodes)
- I/O nodes manage locking, block movement
- Utilize SHMEM to share between nodes

Memory Extension

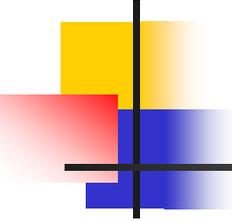


- Reserved shared address space & Flash memory
- Use `mmap()` to map files into this space
- MM files into sequential blocks in MM Flash segment



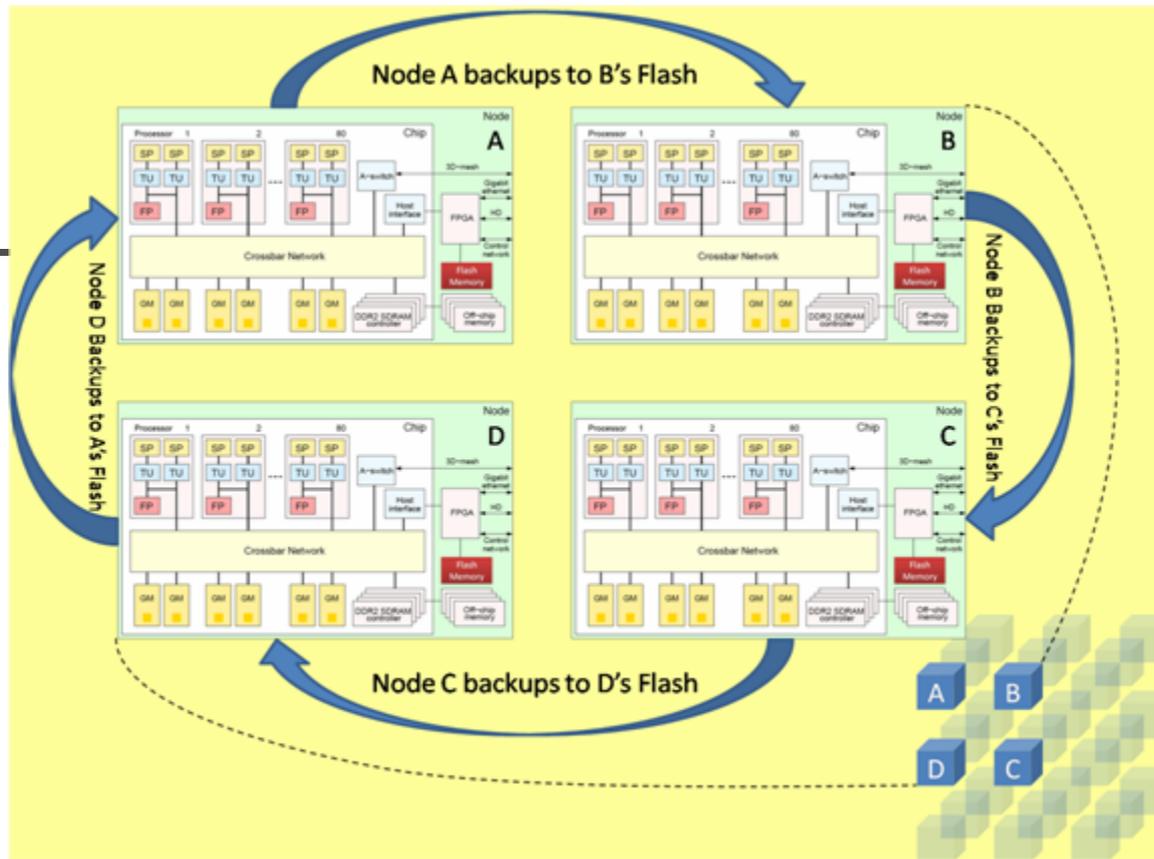
Memory Extension (cont..)

- FPGA services loads and stores from user threads
 - FPGA translates address to LBA
- Copies of all instructions sent to DSA
 - DSA send stores to backup Flash
- Unmapped files go back into cache
 - Start writing dirty blocks back to file server



Reliability, Availability, & Serviceability (RAS) Model

- Every node mirrors its data to specific neighbor
 - Nodes do not use backup data for local requests
- DSA monitor their SSD's health
- When a SSD fails backup becomes primary
 - Increase write back rate to file server
 - Self-Heal when SSD repaired or replaced

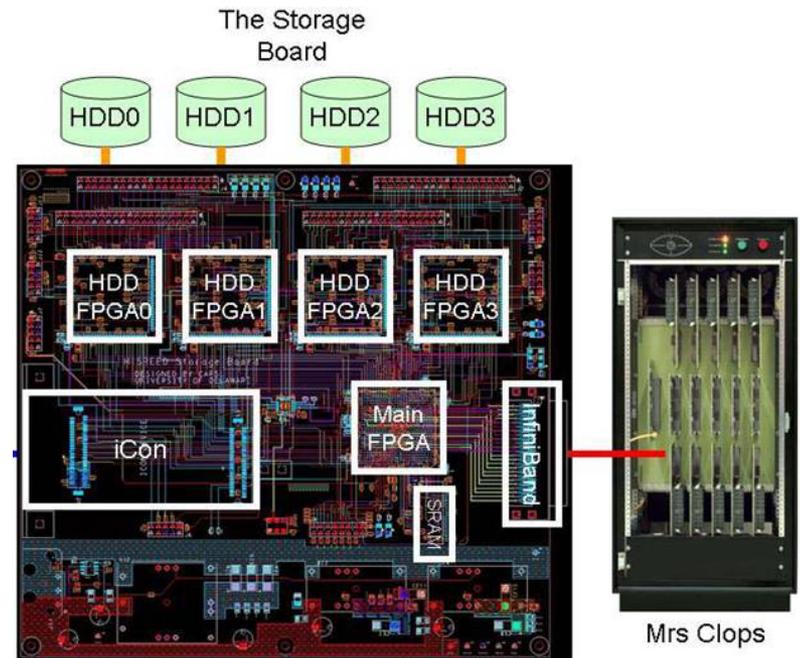


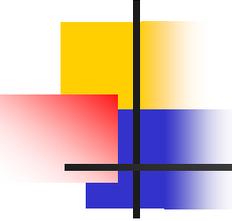
Backup Pattern

Simple 4 node pattern distributes the traffic over 4 links
 2D leaves the 3rd dimension (in/out screen) for path to I/O node
 Recover from SDD failures by non-adjacent nodes
 Baseline for comparison to more complex patterns

Experimental Testbed

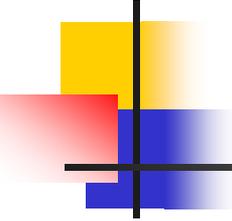
- FAST C64 simulator
 - Use FUSE to simulate DSA behavior
 - Rewrite FAST to included Flash
- Mrs. Clops Emulator w/Storage Extension
 - Main FPGA on board = C64 node FPGA





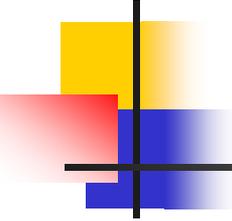
Benchmarks

- Interleaved Or Random (IOR)
 - Currently porting IOR to Cyclops at ORNL
 - Synthetic benchmark, no calculations to slow simulator
 - Options for many different workloads
 - Reports times, bandwidth, IOPS
- Additional micro benchmarks to be developed
 - Needed to test Memory mapping



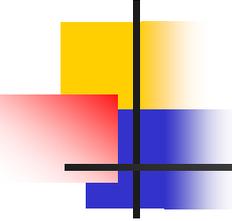
Acknowledgements

- Funding/Support: NSF-0702244, ORNL
- ORNL: Steve Poole, Steve Hodson
- ETI: Dr Juan del Cuvillo, Rishi Khan
- UD: Dr. Guang Gao, Dr. Mihailo Kaplarevic, Dr. Ziang Hu, Dr. Ioannis Venetis, Dr. Haiping Wu
- UD: Yuhei Hayashi, Dimitrij Krepis, Kelly Livingston, Fei Chen

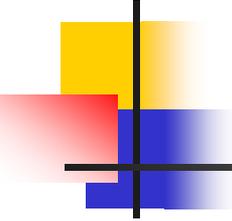


Thank you

- Questions?
- Comments, etc..

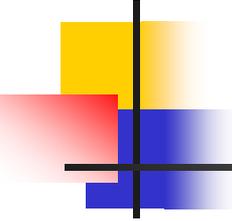


Additional slides



Bitmicro E-Disk Altima SATA 2.5" Flash SSD Series Specs

- Up to 300MB/s burst, 100MB/s
- Up to 20,000 Random IOPS
- 30 to 100 μ sec access time
- Up to 64 GB at 9.5mm
- Up to 64MB fully associative cache
 - Powerguard® writes dirty cache on power loss
- ECC up to 9 bit errors per block
- Write Endurance 657 years at 100GB/day



Write Endurance (Max Write)

- Bitmicro 16GB: 657 years at 100GB/day
- Max Daily load: $100\text{MB/s} \times 86400\text{s/day} \times 1\text{GB}/1024\text{MB} = 8437.5\text{GB/day}$
- Adjusted life: $657\text{ years} \times 100/8437.5 = 7.786\text{ years}$
- With good wear leveling, write endurance scales linearly with total size