Exploring the Future of Out-Of-Core Computing with Compute-Local Non-Volatile Memory

Myoungsoo Jung [1] **Ellis H. Wilson III** [2] Wonil Choi [1,2] John Shalf [3,4] Hasan Metin Aktulga [3] Chao Yang [3] Erik Saule [5] Umit V. Catalyurek [5,6] Mahmut Kandemir [2]

[1]Department of Electrical Engineering, The University of Texas at Dallas

[2]Department of Computer Science and Engineering, The Pennsylvania State University

[3]Computational Research Division, Lawrence Berkeley National Laboratory

[4]National Energy Research Scientific Computing Center, Lawrence Berkeley National Laboratory

[5]Biomedical Informatics, The Ohio State University

[6]Electrical and Computer Engineering, The Ohio State University

November 20th, 2013

Before We Begin: Get the Slides and Paper

Slides and Paper are Available At:

www.ellisv3.com

1. Overview of OoC Computing and Motivations
   - OoC Computing in Today's HPC Environment
   - Current Approaches to Acceleration in HPC
   - Motivating a Move to Compute-Local NVM
2. Advancing OoC Computing via Holistic System Analysis
   - System Organization and a Software Management Framework
   - File System Analysis: Traditional versus a Unified File System
   - NVM Device Architecture: Uncovering Hidden Bottlenecks
3. Evaluation and Analysis of Our Proposed Solutions
   - Experimental Configuration and Tracing Methodology
   - Results of Holistic System Improvement for OoC Computing
   - Major Take-Aways and Conclusion

What's an OoC?

Definition of Out-Of-Core (OoC) Computation:
Computation requiring constant or near-constant use of datasets, which are impossible to fit entirely in-memory for a single host.

Exemplary OoC Application

Predicting Properties of Light Atomic Nuclei

- Performs high-accuracy calculations of nuclear structures via the Configuration Interaction (CI) method
- CI method utilizes the nuclear many-body Hamiltonian, $\hat{H}$, which is sparse, so a parallel iterative eigensolver is used
- $\hat{H}$ can be absolutely massive, and requires much more time to compute than any single eigensolver iteration
- Result is preprocessing and storing $\hat{H}$ for repeated use

Current OoC Solution: Shared Memory

Current Solution:

- Dealt with by splitting dataset across numerous nodes' memories and sharing the memory space.

Pitfalls:

- DRAM is extremely costly and power inefficient
- Capacity constrained DRAM limits scale of experiments
- Application dataset sizes are growing faster than DRAM capacity is scaling
- Expensive networking (e.g., top-tier Infiniband) is required to facilitate such demanding data movement

Acceleration: From Compute to Storage

HPC is currently witness to a sea-change in computation:

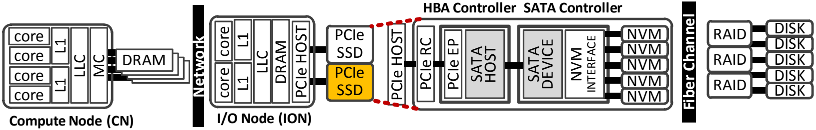
- No longer simply General Purpose CPUs
- GPGPUs and co-processors are seeing increasingly serious use in numerous Top500 machines

Storage in HPC is beginning to follow suit:

- Traditional magnetic disk is often too slow, even at scale
- Flash-cache accelerated NAS/SAN was first to assist
- Natural Extension: Recent works have explored flash on I/O Node (ION) for OoC acceleration

ION-Local Acceleration for OoC Computation

Architecture For ION-Local NVM Acceleration:

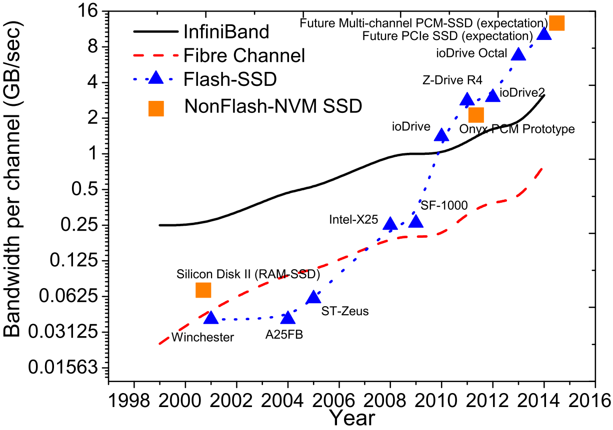


Caveat:

- Data movement from ION to compute still required

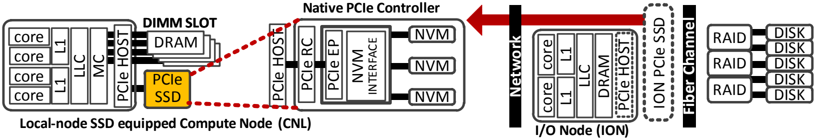
Problem: NVM Bandwidth is Out-Pacing the Network

Bandwidth Trend: High-Performance Network vs. SSDs



Retrain Your Brain: Flash is Memory, not Storage

"We must begin to envision and find ways to implement NVM as a form of compute-local, large but slow memory, rather than client-remote, small but fast disk."



Our Contributions

1. Design OoC HPC architecture with co-located NVM storage and compute
2. Demonstrate that traditional file systems are not well-tuned for the massively parallel architecture within modern SSDs
3. Propose new Unified File System (UFS)
4. Expose overheads implicit in modern SSD architecture
5. Present necessary protocol/interface fixes for near-optimal performance
6. Provide comparative evaluations for all suggested improvements using real OoC workloads

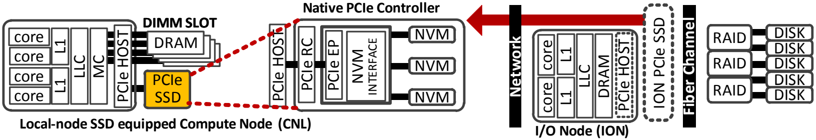
Future System Design Requires a Holistic Approach

Full exploration of potential future OoC systems requires a holistic approach to system analysis and redesign:

- Hardware organization
- Software framework and applications
- File systems
- Device protocol
- Device architecture and interfaces

Co-locating Compute and NVM: Considerations

Another look at our architecture:



Considerations:

- **Cost:** SSDs aren't cheap, but prices are dropping and bandwidth/capacity is consistently rising
- As SSDs out-pace network, it becomes increasingly expensive to keep them off the compute node
- **Tradition:** Typical separation of compute and storage for management reasons
- Administration of coupled architectures has been recently proven quite doable (e.g., Hadoop, Mesos)

Our Data Management Framework

We enable application-managed data staging via:

- **DOoC** - Distributed data storage and scheduler with OoC capabilities via out-of-core linear algebra framework (LAF)
- **DataCutter** - A middleware that abstracts dataflows via the concepts of filters and streams

All together, this works much in the way OpenMP does – directives and routines in the application code enable automated data storage management

Traditional File Systems

The Good Ol' (Magnetic) Bits Club

- Most filesystems, even modern ones, are built on a foundation of assumptions for spinning magnetic disk
- This prevents full utilization of the massively parallel architectures in modern SSDs due to:
  1. Small block sizes (512B to 4KB)
  2. Low coalescing limits
  3. Metadata/journaling contention

The Unified File System

Enabling Full Parallelism in SSDs:

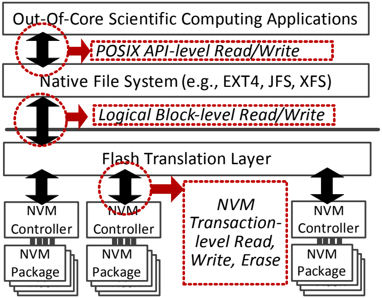
- Fixes the woes of existing file systems and an untuned block device layer
- Provides near-direct access to SSD by punching straight through the file system, block layer, and FTL
- FTL and file system duties become more tightly integrated in the host

Dubious? Fusion-IO already employs a lesser variation on this

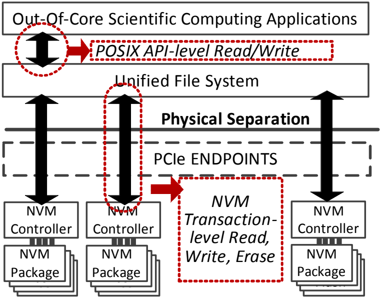
Exemplary Request Comparison

Consider the path of a request between the following:

Traditional File System:



Our Unified File System (UFS):

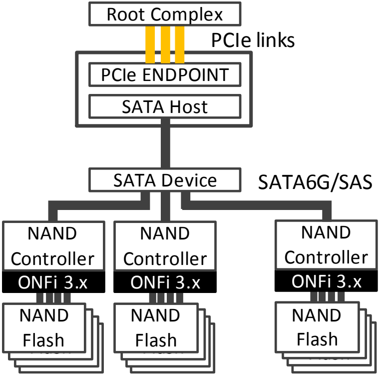


A Silent (Performance) Killer: Bridged PCIe Flash

First device hurdle discovered:

Bridged SATAe-based PCIe:

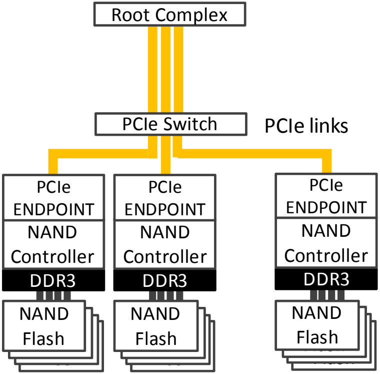
- Many “PCIe” SSDs are simply flash chips with SATAe interfaces
- An internal transcode from SATA to PCIe (and back) occurs
- Biggest issue: SATA uses a 8/10b encoding (25% overhead), whereas PCIe 3.0 uses a 128/130b (1.5% overhead) encoding



Correcting Performance with Native PCIe 3.0

Native PCIe 3.0:

- Native PCIe links to controller
- Achieves low overhead of 1.5% bits to assure DC-balance and bounded disparity
- We compare native PCIe 3.0 against PCIe 2.0, which uses 8/10b encoding, in evaluation



Lane Width and Interface Frequency Bottlenecks

Second and third device hurdles discovered:

PCIe lane-widths:

- Post-conversion overheads, current PCIe 2.0 SSDs only provide four lanes at 2GBps
- Well under maximum possible throughput potential of flash chips
- We explore future expanded-lane architectures with 8 and 16 lanes

NVM interface frequencies:

- Even cutting-edge protocols such as ONFi3 leave NVM bandwidth behind
- Only reaches equivalent of DDR2 @ 200MHz
- Experimenting with next-generation speeds such as DDR3 @ 1600 will unthrottle NVM

Experimental Setup

High-fidelity trace-based NVM SSD simulation

- NANDFlashSim
- Already supported SLC and MLC flash, extended for TLC and PCM
- Enabled queueing optimizations as described in prior work

NVM Architecture Considered:

- SSDs filled with four NVM types: SLC, MLC, TLC, and PCM
- 8 internal channels
- 64 NVM packages
- 128 NVM dies (2/package)

Real OoC Application Tracing Methodology

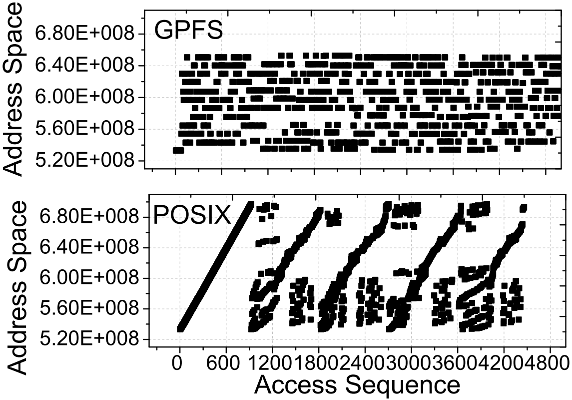
OoC Physics Tracing for Simulation

- Traced the OoC physics application mentioned earlier at scale on the LBNL Carver Cluster

Trace points:

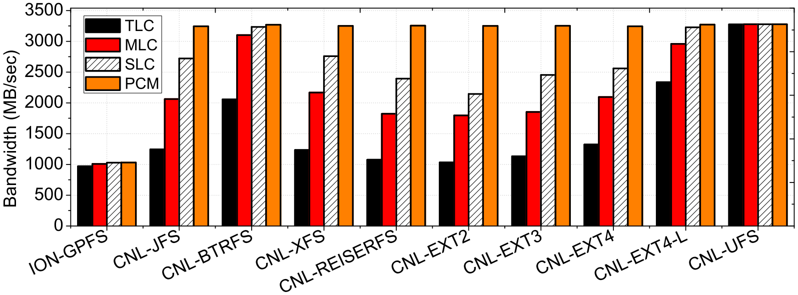
- At the ION-local SSDs (under GPFS)
- At each compute node (at POSIX level)
- Rerun and retraced with a variety of file systems (ext2, ext3, ext4, tuned ext4, JFS, BTRFS, and XFS) (at the block level)

Access Pattern Considerations: GPFS vs POSIX



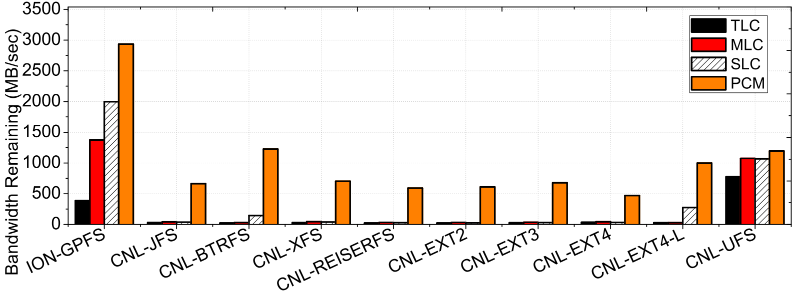
Take-Away: GPFS striping creates access patterns that fail to leverage full bandwidth of flash – ability to issue POSIX access patterns directly to flash would be ideal

Architecture and File System Results: Bandwidth Achieved



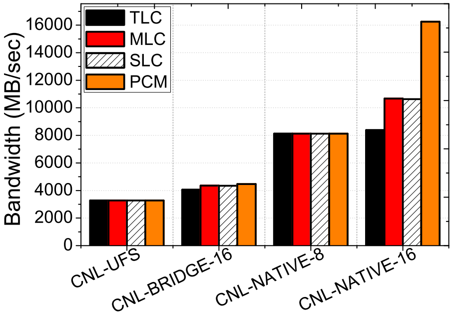
Take-Aways: 1) ION-local is harshly limited by network. 2) CN-local varies extensively with behaviors of underlying file systems. 3) UFS reaches architectural bottlenecks.

Architecture and File System Results: What Remains?



Take-Aways: 1) ION-local and UFS leaves considerable bandwidth untapped. 2) Traditional file systems workloads are flash-limited due to workloads.

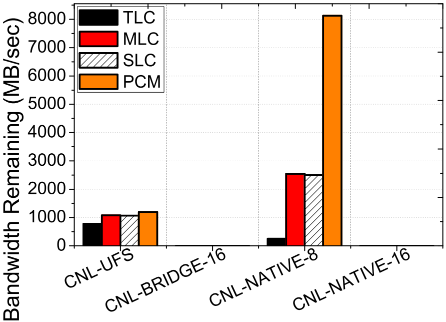
Device Improvement: Bandwidth Achieved



Take-Aways:

1. *Even with 4X more interface bandwidth, only marginal improvements for bridged architecture*
2. *Native PCIe with improved frequencies delivers superior performance*
3. *NVM is finally the real bottleneck in last architecture*

Device Improvement: What Remains?



Take-Aways:

1. *Despite low performance, bridged architecture incurs so much overhead nothing is left behind*
2. *Move to native opens up throttle, but gets bottlenecked on only 8 channels*

Main Evaluation Take-Aways

- **Move it Local**: Keeping NVM remote (or using shared memory) is an increasingly costly decision for future systems
- **Holistic Eye**: Achieving full performance for NVM SSDs requires holistic approach to system analysis
- **File Systems Matter**: Which file system is employed plays a huge role in fully leveraging SSDs
- **Unthrottled SSDs**: Future SSD architecture has to expand lanes and increase frequencies to fully unthrottle NVM storage

Conclusions

Conclusions:

- Have to think of NVM SSDs more as nearby, slow memory, than distant, fast storage.
- Demonstrate *108%* improvements just by moving it nearby
- Another *52%* and *250%* in improvements can be realized by tuning file systems and SSD architecture properly
- Overall, comparing the original ION-local, untuned SSD architecture, to our last, CN-local fully-unthrottled SSD architecture, we are able to unlock **16X bandwidth for our OoC application, without ever changing the underlying NVM chips**

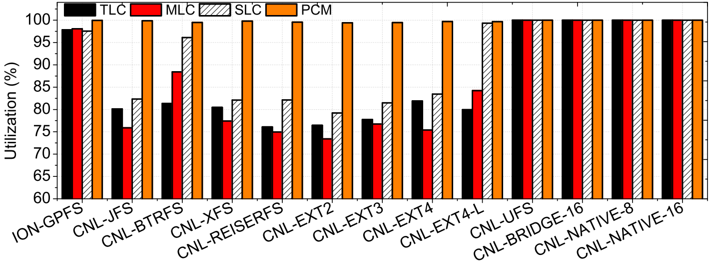
Questions?

Backup Slides

– Begin Backup Slides –

Digging Deeper: Channel Utilization

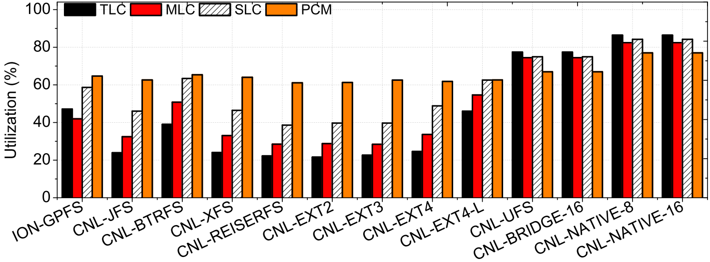
Channel Utilization = Average Percent of Channels Kept Busy



Take-Away: GPFS striping results in high utilization, but low performance

Digging Deeper: Package Utilization

Package Utilization = Average Percent of Packages Serving Requests



Take-Away: Even small percentage increases in package utilization can mean large increases in bandwidth

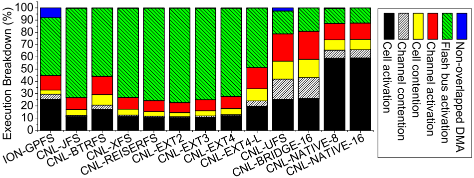
Digging Deeper: Operation Breakdown Definitions

Six Major Categories of Operations Possible in SSD

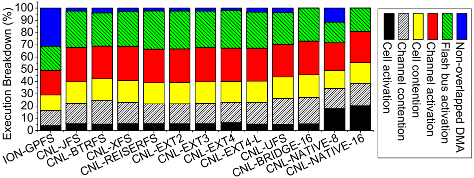
- *Non-Overlapped DMA:* Data movement between SSD and the host, including thin interface (SAS), PCIe bus, and network.
- *Flash-Bus Activation:* Data movement between registers (or SRAM) in NVM packages and the main channel.
- *Channel-Bus Activation:* Data movement on the data bus shared by NVM packages.
- *Cell Contention:* Waiting on an NVM package already busy serving another request.
- *Channel Contention:* Waiting on a channel already busy serving another request.
- *Cell Activation:* Performing a read, write, or erase operation on an NVM cell, including time spent moving data between internal registers (or SRAM) and the cell array.

Digging Deeper: Operation Breakdown

TLC Operation Breakdown:



PCM Operation Breakdown:



Take-Aways:

1. *ION-local spends significant time in non-overlapped DMA due to the network*
2. *UFS relieves internal bus activities obvious in traditional file systems*
3. *Cell activation increases dramatically towards later architectures*

Digging Deeper: Parallelism Classification

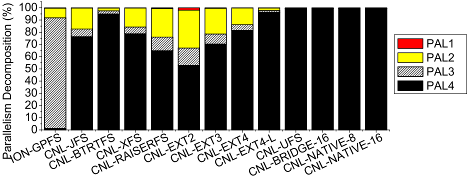
Four Stages of Parallelism

- *PAL1:* System-level parallelism via solely channel striping and channel pipelining.
- *PAL2:* Die (Bank) interleaving on top of PAL1.
- *PAL3:* Multi-plane mode operation on top of PAL1.
- *PAL4:* All previous levels above.

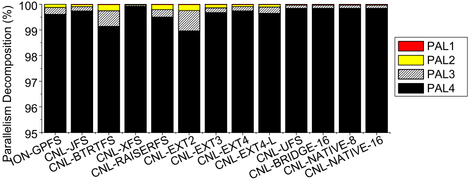
Digging Deeper: Parallelism Breakdown

Difference: Perspective of a request – not perspective of hardware

TLC Parallelism Breakdown:



PCM Parallelism Breakdown:



Take-Aways:

1. *ION-local has difficulty reaching full parallelism*
2. *UFS-based workloads reach high PAL4 levels*