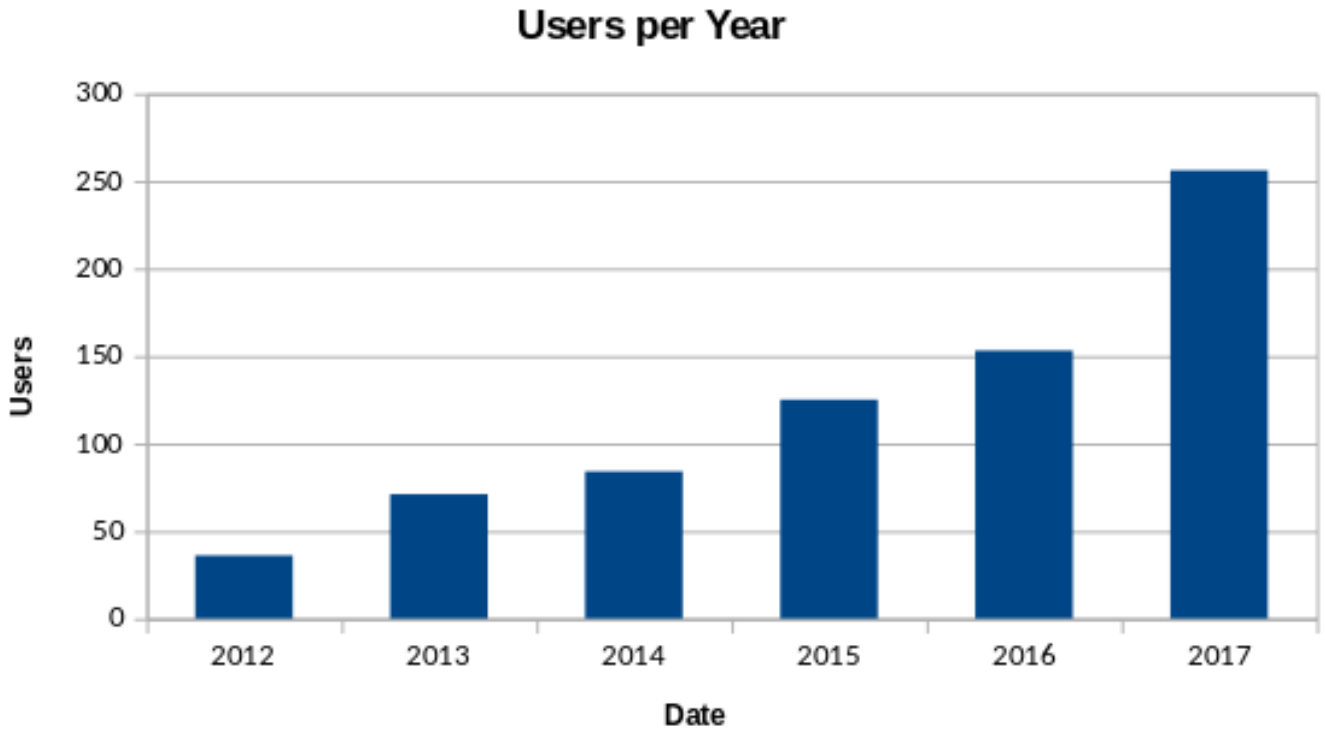
CRI Infrastructure and New Parallel Storage

Mike Jarsulic

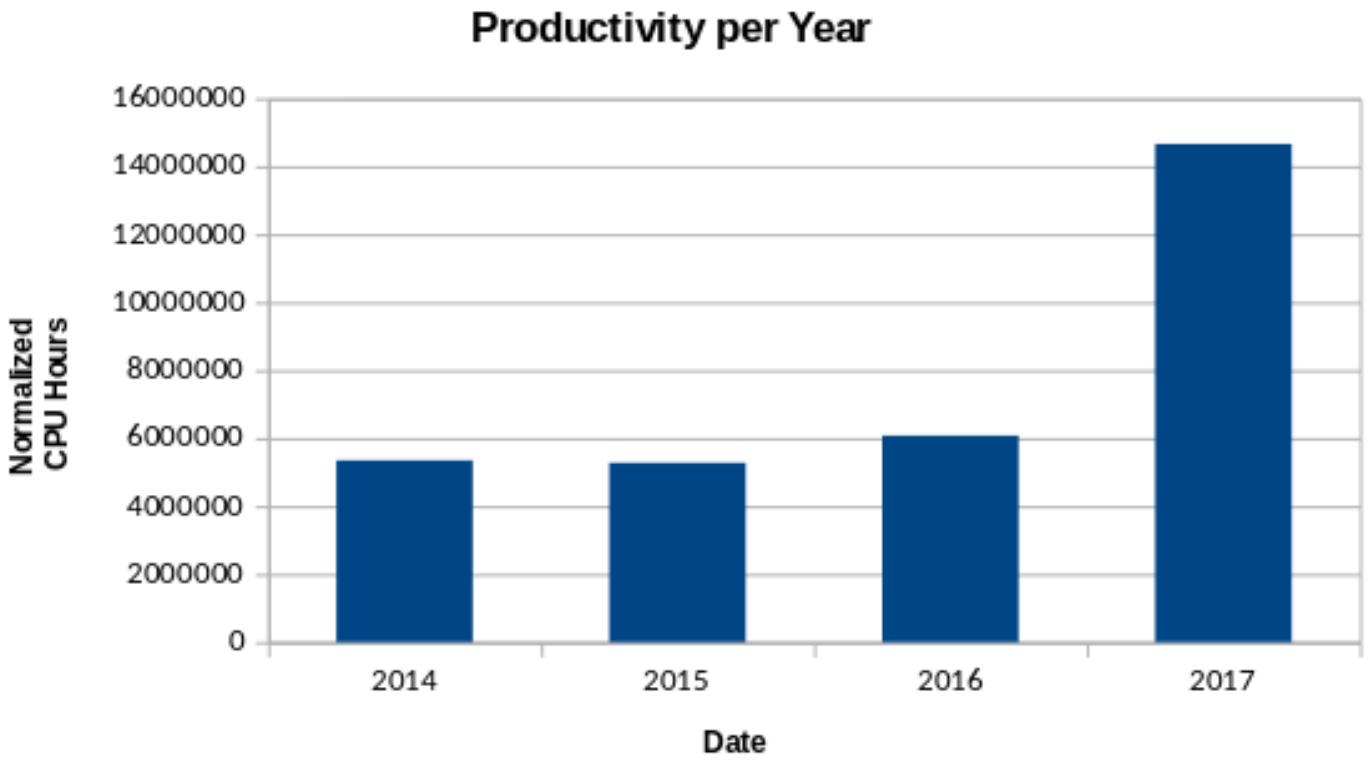
Olumide Kehinde

HPC Overview

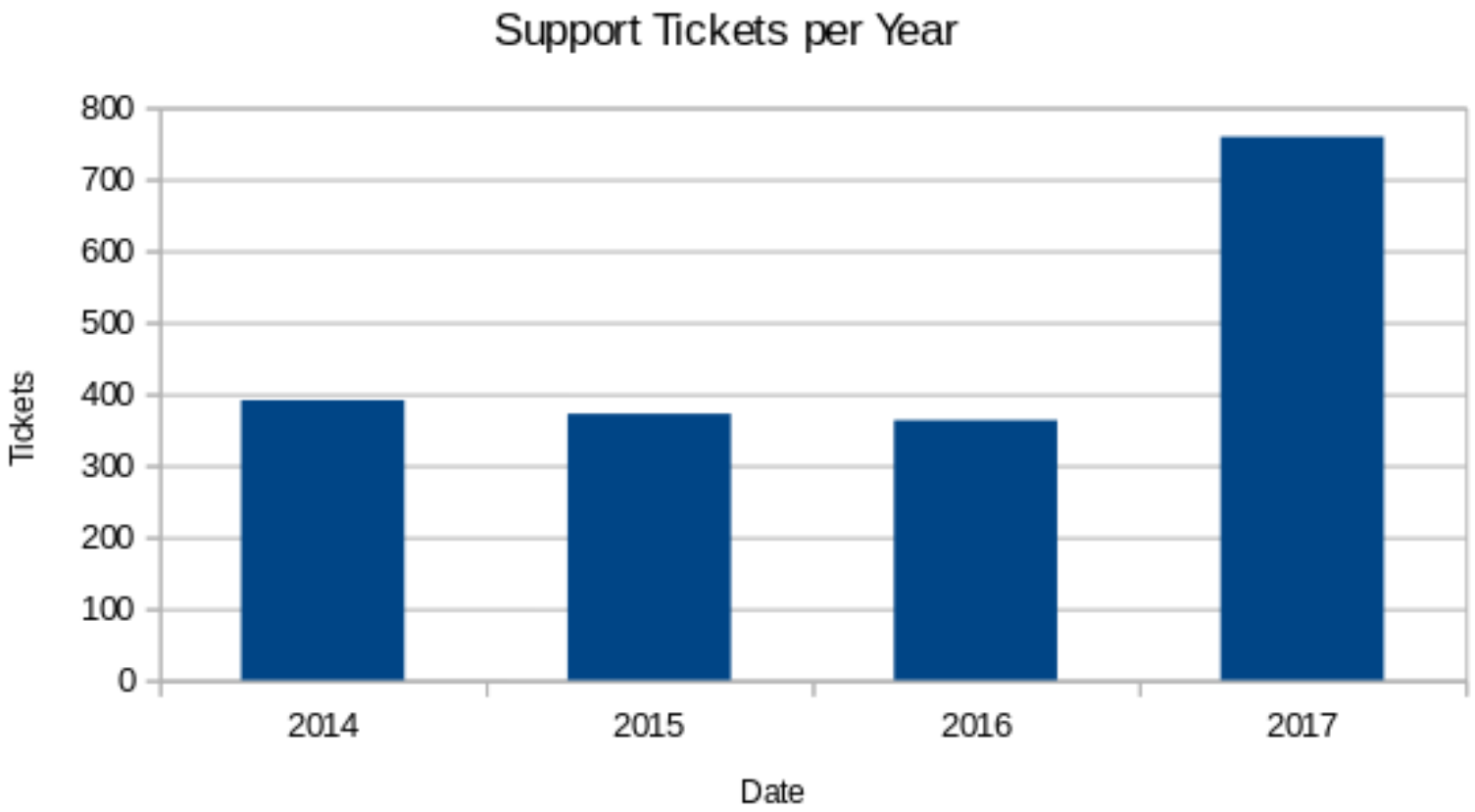
HPC Metrics – Good News



HPC Metrics – Good News



HPC Metrics – Bad News



Upcoming HPC Work

HPC Survey

New FY20 Cluster (Andrus)

Deep Learning Platform

Visualization

Training

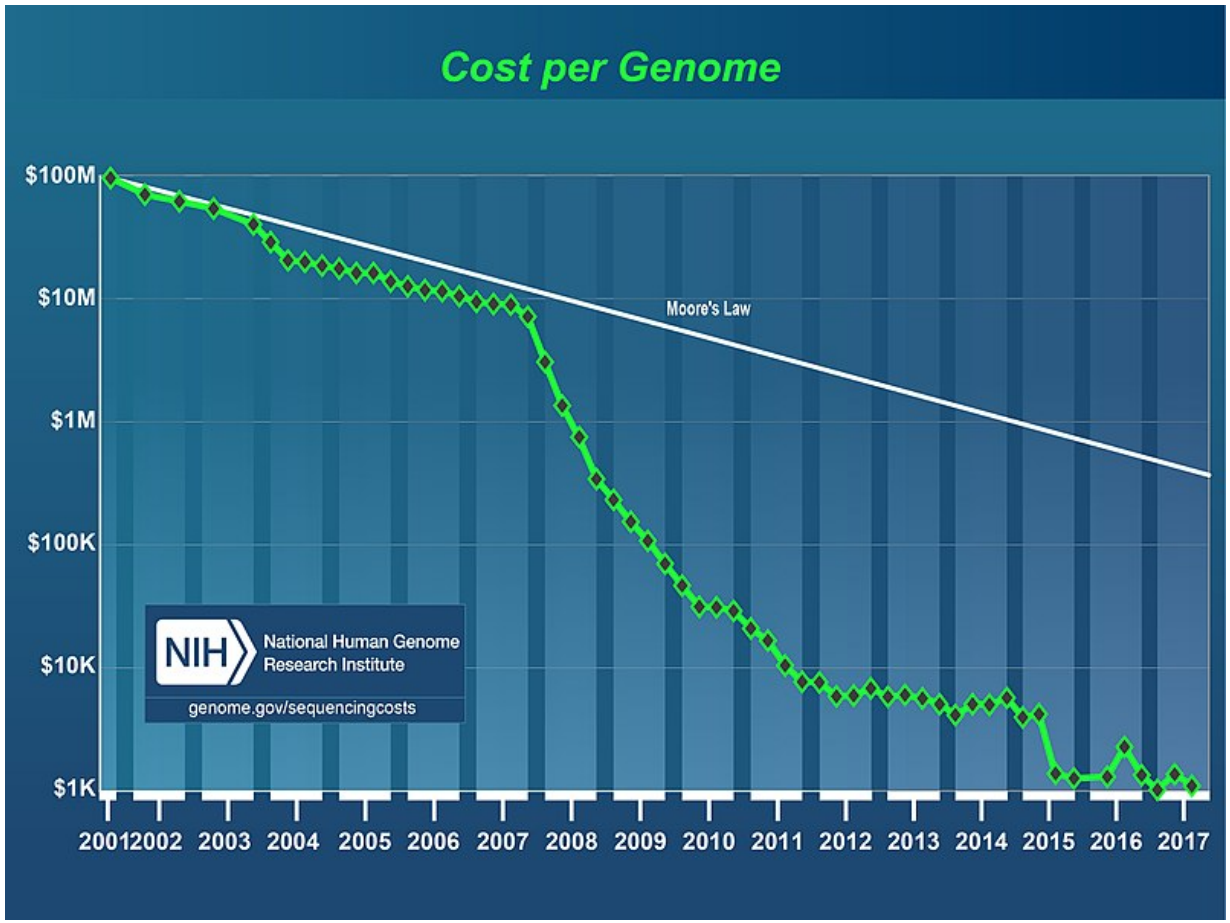
Containers (either Singularity of Shifter)

Jupyterhub

Performance Profiling

Introduction

Obligatory Cost per Genome Slide



Scientific Computing

Use of computers as a tool for gaining a better understanding of the real world

- Complements theory and experimentation
- Techniques:
  - Modeling
  - Simulation
  - Data Analysis
  - Visualization
  - AI

Scientific Computing Problems

Problems are increasingly computationally expensive

- Thus, we need large parallel machines to perform the necessary calculations
- It's critical to leverage parallelism at all phases of an analysis
- This includes I/O operations

Data access is a huge challenge

- In the BSD, we are particularly bad at this
- Using parallelism to obtain performance
- Finding usable, efficient, portable interfaces
- Understanding and tuning I/O

Data Volumes at the CRI

Department	Vol. Size (TB)
Human Genetics	192
CRI Bioinformatics Core	184
Medicine	174
Organismal Biology and Anatomy	166
Ecology and Evolution	151
Pathology	100
Pediatrics	54
Surgery	37
Ben May Cancer Research	23
Public Health Sciences	15

Department	Vol. Size (TB
Biochemistry/Molecular Biology	15
Molecular Genetics/Cell Biology	14
Neurobiology	14
Psychiatry	11
Center for Health and Social Sciences	8
Family Medicine	6
Orthopaedic Surgery and Rehabilitation Medicine	6
Radiology	5
Other	15

I/O vs Application Data Complexity

I/O Systems have very simple data models

- Tree-based hierarchy of containers
- Some containers have streams of bytes (files)
- Others hold collections of containers (directories)

Applications have models appropriate to the domain

- Multidimensional typed arrays, variable length records
- Headers, attributes on data

Challenges in Application I/O

Storing data in portable formats

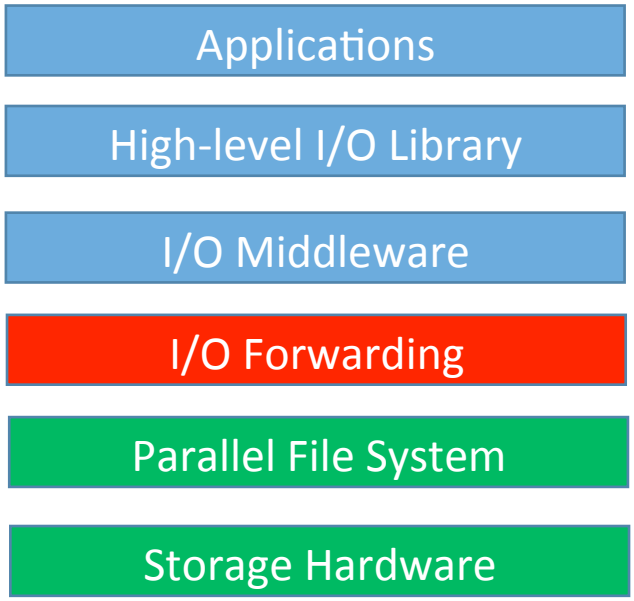
Interacting with storage through convenient abstractions

Limiting the number of files that must be managed

Leveraging aggregate I/O bandwidth of clients

Avoiding unnecessary post-processing

I/O Model for Scientific Computing



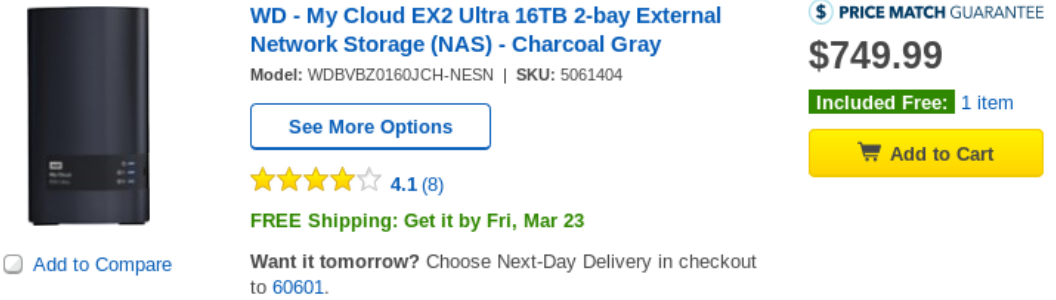
I/O Hardware

Magentic Tape

MicroSD cards

Solid State Drives

Magnetic Hard Drives



Parallel File System

Manage Storage Hardware

- Present a single view of multiple components
- Stripe files for performance

In the I/O Software Stack

- Focus on concurrent, independent access
- Publish an interface that middleware can use effectively

Examples: OneFS, GlusterFS, GPFS, Lustre

I/O Middleware

Match the programming model (e.g., POSIX, MPI)

Facilitate concurrent access by groups of processes

- Collective I/O
- Atomicity Rules

Expose a generic interface

- Good building block for high-level I/O libraries

Efficiently map middleware operations to PFS operations

Examples: POSIX, MPI I/O

High- Level I/O Libraries

Match the storage abstraction domain

- Multidimensional datasets
- Typed variables
- Attributes

Provide self-describing, structured files

Map to the middleware interface to encourage collective I/O

Implement optimizations that the middleware cannot

- Caching attributes of variables
- Chunking of datasets

Examples: HDF5, NetCDF, ADIOS

So far...

Scientists (i.e., You) have basic goals when interacting with storage

- Never having to worry about running out of space, losing data, backups, etc. (obviously)
- Keep productivity high (meaningful interfaces)
- Keep efficiency high (extracting high performance from hardware)

Application programmers in the life sciences have failed you

- This is largely due to reliance on the POSIX API, which is poorly designed for scientific computing

There is software available that address these goals

- Provide meaningful interfaces with common abstractions
- Interact with the files system in the most efficient way possible

Storage Hardware Rant

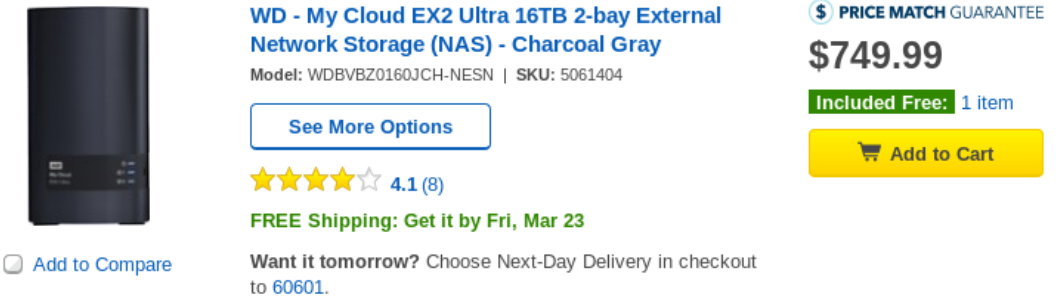
I/O Hardware

Magentic Tape

MicroSD cards

Solid State Drives

Magnetic Hard Drives



Storage Device vs. Storage System

We've already discussed storage devices such as magnetic hard drives, solid state drives, etc.

Storage System

- Software to aggregate many devices for performance
- Software to handle device failures (RAID, erasure coding)
- Software to handle hardware failure (server failover)
- Software to handle software failure (write ahead logging, atomicity)
- APIs to layer structure over raw storage

Base-10 vs. Base-2 Units

nit	Base-10	Base-2	% Difference
KB	10^3 = 1,000	2^10 = 1,024	2.5%
MB	10^6 = 1,000,000	2^20 = 1,048,576	5%
GB	10^9 = 1,000,000,000	2^30 = 1,073,741,824	7.5%
TB	10^12 = 1,000,000,000,000	2^40 = 1,099,511,627,776	10%
PB	10^15 = 1,000,000,000,000,000	2^50 = 1,125,899,906,842,624	12.5%

Storage vendors are always looking to rip you off!!!

Any storage you buy will be sold to you in Base-10 units

Computer scientists often think in Base-2 units

Storage Latency

Technology	Latency	Size (example)
L1 CPU Cache	4 cycles (~1 nsec)	32 KB
L2 CPU Cache	10 cycles (3 nsec)	256 KB
LLC CPU Cache	40 cycles (13 nsec)	1 MB
DRAM	240 cycles (80 nsec)	16 GB
NVRAM	1200 cycles (400 nsec)	128 GB
RDMA Read	**6000 cycles (2 µsec)**	**16 GB**
RDMA Write	**120,000 cycles (40 µsec)**	**16 GB**
SSD Read	**150,000 cycles (50 µsec)**	**128 GB**
SSD Write	**1,500,000 cycles (500 µsec)**	**128 GB**
HDD Write (track cache)	**1,500,000 cycles (500 µsec)**	**4 TB**
HDD Read	**15,000,000 cycles (5 msec)**	**4 TB**
Tape Access	150,000,000,000 cycles (50 sec)	6 TB

Bandwidth Hierarchy

Device 1	Bus	Device 2	Bandwidth
CPU	QPI	CPU	64 GB/sec
CPU	QPI	Memory (3 channel)	30 GB/sec
CPU	QPI	Memory (4 channel)	64 GB/sec
CPU	PCIe	NIC	24 GB/sec
CPU	**PCIe**	**SSD**	**2 GB/sec**
CPU	USB2	Hard Drive	35 MB/sec
CPU	**USB3**	**Hard Drive**	**400 MB/sec**
NIC	**GbE**	**Storage**	**125 MB/s**
NIC	**10 GbE**	**Storage**	**1.25 GB/s**
NIC	**FDR IB**	**Storage**	**7 GB/s**
NIC	EDR IB	Storage	12.5 GB/s

SSD Characteristics

Controller

FLASH

DRAM

Interface Bus

- PCIe
- SAS
- SATA

Flash Characteristics

Non-volatile

- Each bit is stored in a floating gate that holds value without power
- Electrons can leak, so life and write count is limited

Page-oriented

Flash Translation Layer (FTL)

- Allows wear leveling
- Requires garbage collection

Performance

- Fast reads (no seeks)
- Slower writes
- Slow erase cycles
- Background tasks (e.g., garbage collection) cause interference

FLASH Performance

SLC – Single Level Cell

- 10^5 to 10^6 write cycles per page
- Fastest, longest life

MLC – Multi Level Cell

- 10^4 write cycles per page
- Denser and cheaper, but slower and less reliable

TLC – Triple Level Cell

- 500 write cycles per page
- Cheapest, slowest writes

FLASH Translation Layer (FTL)

Level of indirection

- Allows controller to write to any free page
- Page write may trigger background copies and erases

Wear leveling is critical

- Different pages will wear out at different times depending on how often each page is written
- Pages in an Erase Block are garbage collected together

Over provisioning

- 960 GB device is physically 1024 GB to support wear leveling

SSD Final Thoughts

Large I/O Queue Depth exploit device parallelism

- High IOPs from doing many async ops concurrently

Read-only or write-only yields the best performance

- Mixed workloads increase device-level conflicts

Write < 1 page cause premature wear out

Not all devices are power-fail safe

- Consumer grade drives lack DRAM buffers
- Will pay ~2x as much for power-fail safe

Storage Comparison

Description	USB HDD	SSD	2 Bay NAS	CRI Storage
Type	Storage Device	Storage Device	Storage System	Parallel File System
Aggregate multiple devices			x	x
Device Redundancy			x	x
Hardware Redundancy				x
Software Redundancy				x
High-level APIs			x	x
Max Latency (Read)	5 msec	50 µsec	5 msec	2 µsec
Max Latency (Write)	500 µsec	500 µsec	500 µsec	40 µsec
Limiting Bandwidth	400 MB/s	2 GB/s	125 MB/s	7 GB/s
Tape Backup / Recovery				x
CRI Support				x

Parallel File System

Isilon Cluster (bulkstorage)

Capacity: 1.8 PB

Interconnect: 10 Gbps Ethernet

Parallel File System: OneFS

5 X-Series Nodes, 10 NL Nodes

Protocols: NFSv3, CIFS

Used for lab shares and home directories

Will be decommissioned

Gardner Scratch Space

Capacity: 175 TB

Interconnect: 56 Gpbs Infiniband

Parallel File System: GlusterFS

Protocols: Gluster Native

Used for /scratch

Plans to repurpose as archive space

CRI FY18 Storage Project Goals

Decrease maintenance costs

Decrease cost of future capital requests for expansion

Avoid significantly increasing number of FTEs to manage storage

Increase throughput and IOPs

Increase efficiency of parallel workloads

Utilize high-speed interconnect

Maintain the same level of system stability, compliance, and data integrity

New Storage (pfs)

Capacity: 3.9 PB (in vendor speak)

Infiniband FDR/EDR: 56 Gbps/100 Gbps

Parallel File System: GPFS

Protocols: Native (Verbs/RDMA), NFSv3, CIFS

Used for lab shares, home directories, scratch space

5 Year Cost Projection

Isilon

- Maintenance: $1.1 million (operating)
- Expansion to 3.9 PB: $700,000
- Expansion to 7.8 PB: $1.9 million
- Total: $3.7 million

GPFS

- Maintenance: $0 (operating)
- Expansion to 3.9 PB: $0
- Expansion to 7.8 PB: $550,000
- Total: $550,000

Parallel Workloads

Isilon

- Uses NFS (not POSIX compliant)
- Cache Coherency
  - Unpredictable when data written by one client will be accessible to others
  - Metadata operations are also inconsistent
  - MPI-IO over NFS: All processes have to perform the same I/O operations

Benchmarks

IOR Benchmark

16 nodes, 448 cores

1 file per process

All values in MB/s

API	Operation	Isilon	Scratch	GPFS Data	GPFS Scratch
POSIX	WRITE	864	848	14537	20719
POSIX	READ	880	2776	24415	32978
MPI IO	WRITE	80	1104	17132	21228
MPI IO	READ	72	2960	24439	32922

Benchmarks

IOR Benchmark

16 nodes, 448 cores

Single shared file

All values in MB/s

API	Operation	Isilon	Scratch	GPFS Data	GPFS Scratch
POSIX	WRITE	98	896	14309	20230
POSIX	READ	115	952	22934	29911
MPI IO	WRITE	12	520	16979	21770
MPI IO	READ	14	880	22842	30508

Benchmarks – Metadata Ops

MDTEST Benchmark

16 nodes, 448 cores

All values in ops/sec

Operation	Isilon	Scratch	GPFS Data	GPFS Scratch
Creation	3188	884	19795	19708
Stat	80940	5684	4577691	4854983
Read	6474	6176	2750277	2330840
Removal	210	209	3063	3065

Downsides to GPFS

Performance issues with small files

Block size

- Isilon: 8KB
- Lab Shares: 4MB (128KB)
- Scratch Space: 16MB (512KB)

Expansion

Data Migration

Data Migration

How will data be transferred to the new storage?

- We will use AFM (Active File Management) to transfer data from the Isilon cluster to the new Spectrum Scale cluster
- Data transfer can be asynchronous or synchronous
- Data will be transferred in independent writer (IW) mode

Benefits of using AFM to transfer data

Multiple gateway nodes can be used for AFM processing

If a gateway fails, GPFS automatically moves AFM jobs to another gateway node

Data is kept in sync

User and group permissions are preserved

Data transfers very quickly

Metadata can be prefetched without copying actual data

We can fail-back at any time

Migration methodology

Mount current shares as NFS exports on the gateway node

Create independent-writer mode filesets

(Optional) Prefetch the metadata into the cache for large shares

Data is copied into the cache filesets using prefetch

Old and new shares will remain synced

Perform a phased cut-over (To allow for adequate support time)

Data Migration Priority

- Scratch Space
- Home Directories
- Applications
- Lab Shares
- Others

- Notifications will be sent in advance

During Migration

You can continue to access shares normally

No noticeable performance degradation

There will be no service downtime until cutover

The CRI will update you on progress throughout the migration

What will change?

"*newstorage-name.cri.uchicago.edu*" will replace "*bulkstorage.uchicago.edu*"

Path to share will change slightly:

Old: /group/<share-name>

New: /gpfs/<share-name>

Path to share in your job scripts will need to be updated

GPFS native mounts will replace NFS mounts on servers

Share size will change due to new block size (4MB)

Instructions for accessing to new storage

What won’t change?

SMB/CIFS access (Mac and Windows)

Share name will remain the same

Permissions will remain the same

Group membership will not change

UID and GIDs will remain the same

BSDAD access

Getting Support

Email storage@rt.cri.uchicago.edu for storage related issues

Email hpc@rt.cri.uchicago.edu for HPC related issues

Go to cri.uchicago.edu, click "Technical Help"

Fill and submit support form

Questions?