Parallel I/O

6230 – Spring (1/31/2023)

Jong Choi
Discrete Algorithm Group
Mathematics in Computation Section
Oak Ridge National Laboratory (ORNL)

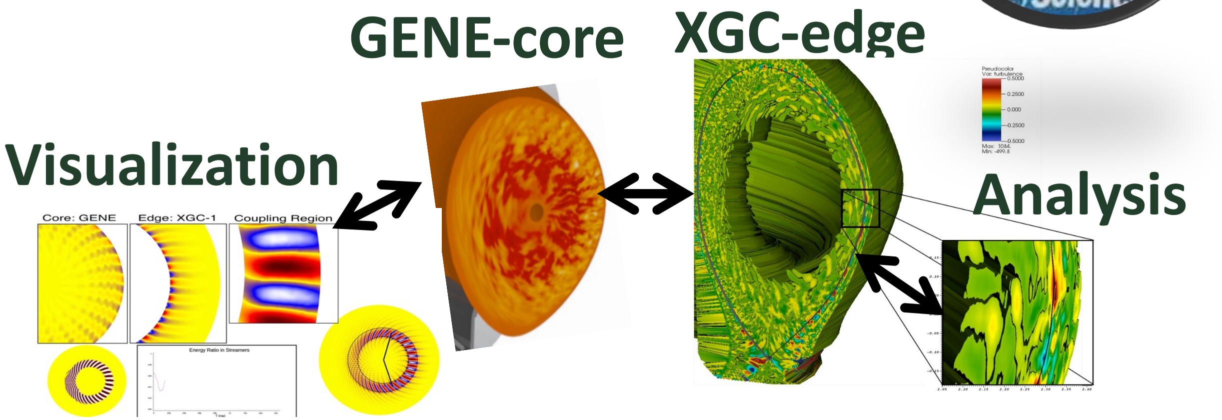
Special Thanks

- Ramakrishnan Kannan, ORNL
- Scott Klasky, Norbert Podhorszki, ADIOS Team, ORNL
- CS Chang, Princeton Plasma Physics Laboratory (PPPL)

About me

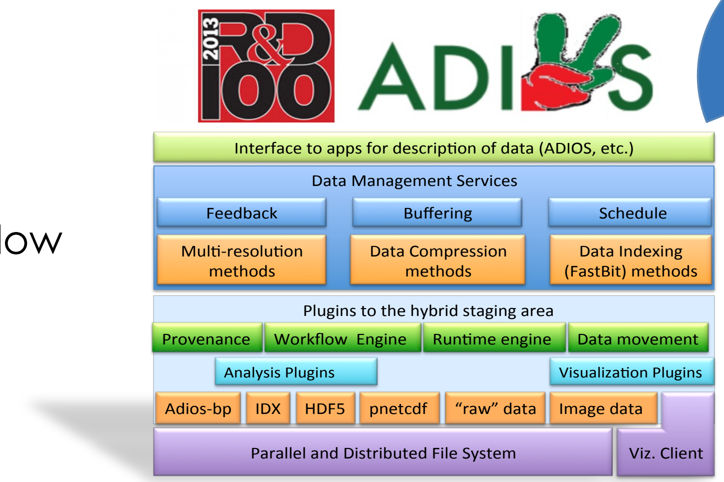
choij@ornl.gov

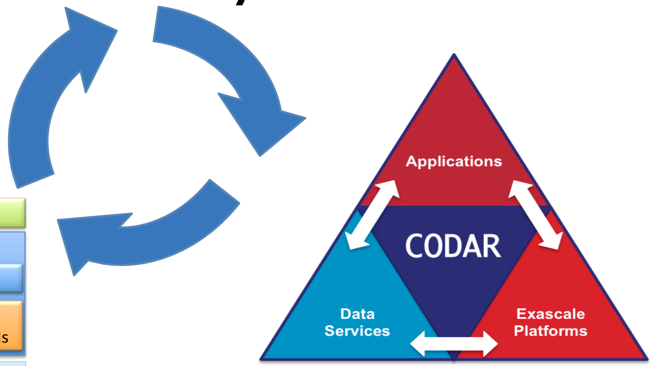
Develop tools for composability of complex, coupled workflows consisting of independently running simulation and analysis applications



WDM Coupled In Situ Analysis Workflow

- Challenges
  - Big data and performance challenge
  - Supporting In situ/online analysis
  - Managing complex workflow
- Impact
  - Big data analysis
  - Whole device modeling with coupled workflow
  - CODAR co-design study





CODAR Savanna/Cheetah Workflow Orchestrator

Outline

- Introduction
- Parallel filesystem
  - Lustre
  - GPFS
- Parallel I/O
- High-level I/O libraries
  - Adios
  - HDF5
- I/O Performance measurement tools
- Hands-on demonstration

MPI

OPEN MPI

ADIOS

HDF

Score-P
Scalable performance measurement infrastructure for parallel codes

TAU Performance System®

Introduction

Department of Energy (DOE) Roadmap to Exascale Systems

An impressive, productive lineup of *accelerated node* systems supporting DOE's mission

(Exa-scale: 10^{18})

Pre-Exascale Systems [Aggregate Linpack (Rmax) = 323 PF!]

First U.S. Exascale Systems

2012	2016	2018	2020	2021-2023

2012

ORNL
Cray/AMD/NVIDIA

ANL
IBM BG/Q

LLNL
IBM BG/Q

2016

ANL
Cray/Intel KNL

LBNL
Cray/Intel Xeon/KNL

LANL/SNL
Cray/Intel Xeon/KNL

2018

ORNL
IBM/NVIDIA

LLNL
IBM/NVIDIA

2020

LBNL
Announced: Oct 30, 2018
Expected delivery: late 2020
Cray / CPU: AMD Milan, GPU: Nvidia A100

CROSSROADS

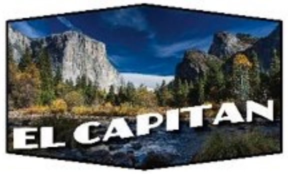
LANL/SNL
Announced: Sep 30, 2020
Expected delivery: 2022
Cray / CPU: Intel Sapphire Rapids, GPU: "Do Not Want"

2021-2023

FRONTIER

ORNL
Announced: May 7, 2019
Expected delivery: 2021
Cray / CPU: AMD Milan-HPC, GPU: AMD Radeon Instinct MI200

ANL
Announced: Mar 18, 2019
Expected delivery: 2021
Cray / CPU: Intel Sapphire Rapids, GPU: Intel Ponte Vecchio



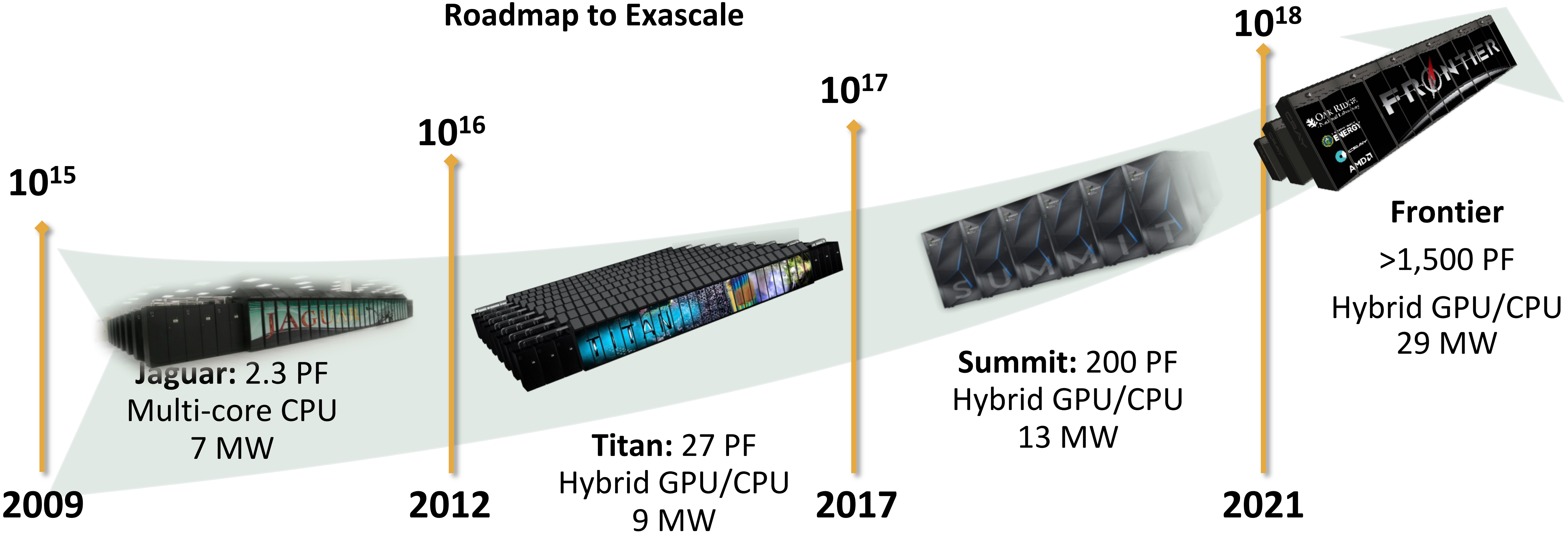
LLNL
Announced: Mar 5, 2020
Expected delivery: 2023
Cray / CPU: AMD Genoa, GPU: AMD Radeon Instinct

ECP EXASCALE COMPUTING PROJECT

Oak Ridge Leadership Computing Facility (OLCF)

Mission: Providing world-class computational resources and specialized services for the most computationally intensive global challenges	**Vision**: Deliver transforming discoveries in energy technologies, materials, biology, environment, health, etc.

Roadmap to Exascale



TOP500 The List.

November 2022

The 60th TOP500 List was published Nov. 15, 2022 in Dallas, TX.

https://www.top500.org/lists/top500/

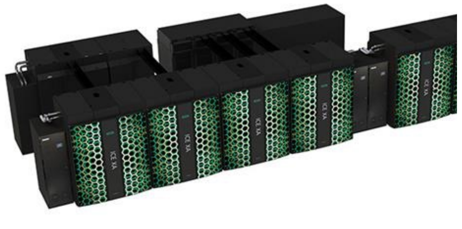
Rank	System	Cores	Rmax (PFlop/s)	Rpeak (PFlop/s)	Power (kW)
1	**Frontier** - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, HPE DOE/SC/Oak Ridge National Laboratory United States	8,730,112	1,102.00	1,685.65	21,100
2	**Supercomputer Fugaku** - Supercomputer Fugaku, A64FX 48C 2.2GHz, Tofu interconnect D, Fujitsu RIKEN Center for Computational Science Japan	7,630,848	442.01	537.21	29,899
3	**LUMI** - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, HPE EuroHPC/CSC Finland	2,220,288	309.10	428.70	6,016
4	**Leonardo** - BullSequana XH2000, Xeon Platinum 8358 32C 2.6GHz, NVIDIA A100 SXM4 64 GB, Quad-rail NVIDIA HDR100 Infiniband, Atos EuroHPC/CINECA Italy	1,463,616	174.70	255.75	5,610
5	**Summit** - IBM Power System AC922, IBM POWER9 22C 3.07GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband, IBM DOE/SC/Oak Ridge National Laboratory United States	2,414,592	148.60	200.79	10,096

Comparison of Titan, Summit, and Frontier Systems

Specs	Titan	Summit	Frontier
Peak	27 PF	200 PF	>1.5 EF
# cabinets	200	256	> 100
Node	1 AMD Opteron 1 NVIDIA K20X Kepler GPU	2 IBM POWER9™ CPUs 6 NVIDIA Volta GPUs	1 HPC and AI Optimized AMD EPYC 4 AMD Radeon Instinct GPU
On-node interconnect	PCI Gen2 No coherence across the node	NVIDIA NVLINK Coherent memory across a node	AMD Infinity Fabric Coherent memory across the node
System Interconnect	Cray Gemini network 6.4 GB/s	Mellanox Dual-port EDR IB network 25 GB/s	Cray four-port Slingshot network 100 GB/s
Topology	3D Torus	Non-blocking Fat Tree	Dragonfly
Storage	32 PB, 1 TB/s, Lustre Filesystem	250 PB, 2.5 TB/s, IBM Spectrum Scale™ with GPFS™	2-4x performance and capacity of Summit's I/O subsystem.
NVMe	No	Yes	Yes

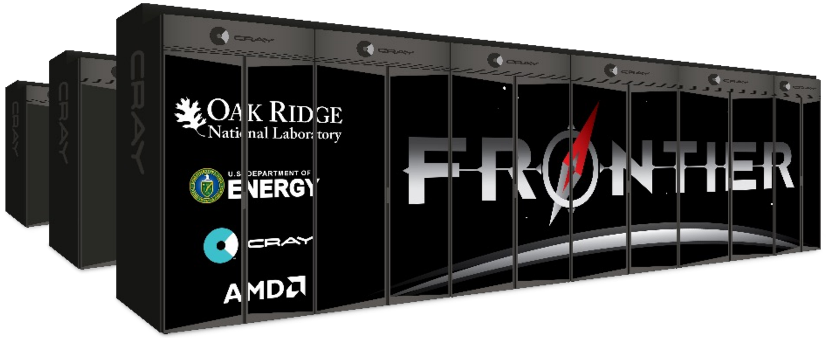
- from J. Wells (OLCF)

Different resources, different storage and Burst Buffer



	Summit ORNL	Theta ANL	Cori NERSC	Tsubame3 Tokyo Tech
CPU	IBM Power9	Intel KNL	Intel KNL and Haswell	2x Intel Xeon (2.4 GHz, 14 core)
Memory	512 GB DDR4 + 16 GB HBM2	192 GB DDR4 + 16 GB MCDRAM	KNL: 96 GB DDR4 + 16 GB MCDRAM Haswell: 128 GB DDR4	256 GB
GPU	6x NVIDIA Tesla V100 (Volta)	N/A	N/A	4x NVIDIA Tesla P100 (Pascal)
Interconnect	Fat-tree	Aries Dragonfly	Aries Dragonfly	Fat-tree
File System	GPFS	Lustre	Lustre/GPFS	Lustre
Burst buffer/SSD	Local NVMe	Local SSD	Shared SSDs	SSDs with BeeGFS share file system

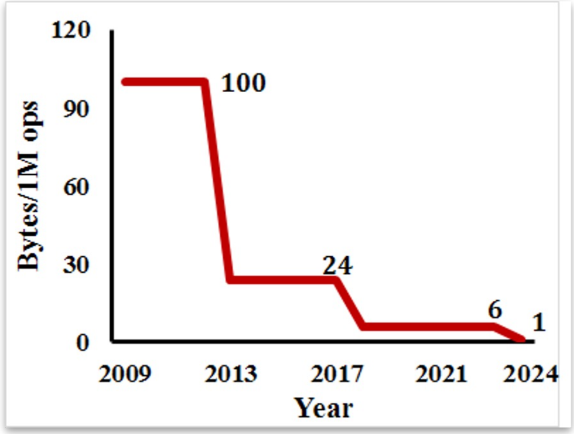
Frontier Overview



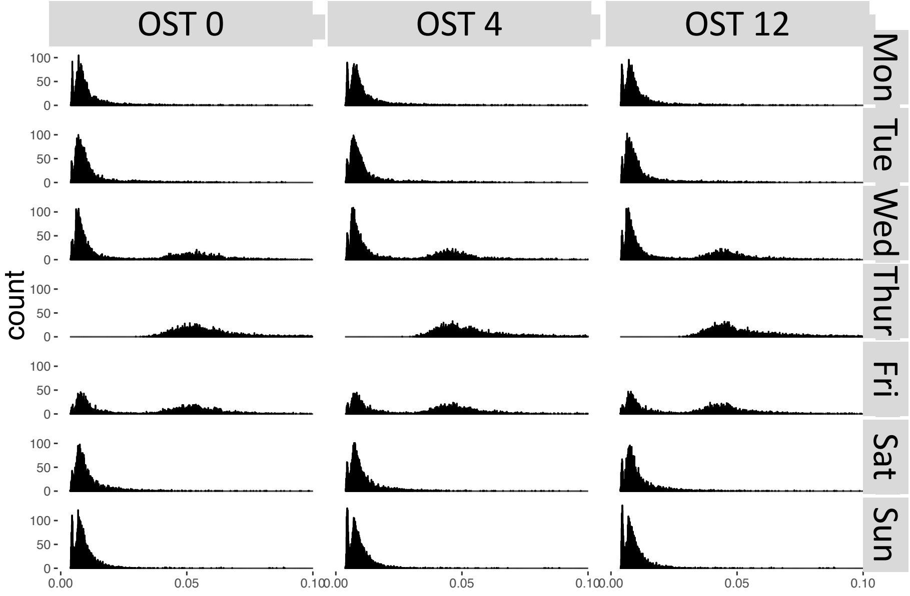
- Partnership between ORNL, Cray, and AMD
- **Peak Performance greater than 1.5 EF**
- Composed of more than 100 Cray Shasta cabinets
  - Connected by **Slingshot™ interconnect** with adaptive routing, congestion control, and quality of service
- Node Architecture:
  - An **AMD EPYC™ processor** and four **Radeon Instinct™ GPU** accelerators built for exascale computing
  - Fully connected with high speed AMD Infinity Fabric links
  - Coherent memory across the node
  - 100 GB/s injection bandwidth
  - Near-node NVM storage
- Researchers will harness Frontier to advance science in such applications as systems biology, materials science, energy production, additive manufacturing and health data science.

https://www.olcf.ornl.gov/ from J. Wells (OLCF)

I/O on leadership HPC machines is challenging



- I/O & Storage Bandwidth are not keeping up with FLOPS and Memory
- The Storage hierarchy is getting more complex
  - The usage of non-volatile memory will further deepen the storage hierarchy
- The scale of storage and I/O subsystems has increased significantly
- The increase of system scale and complexity along with decrease in overall Storage BW/FLOP exacerbates the variability of I/O performance in HPC environments



Histograms of latencies of 1MB writes to different OSTs on different days

B. Xie, J. Chase, D. Dillow, O. Drokin, S. Klasky, S. Oral, and N. Podhorszki. Characterizing output bottlenecks in a supercomputer. In High Performance Computing, Networking, Storage and Analysis (SC), 2012.

L. Wan, M. Wolf, F. Wang, J. Y. Choi, G. Ostrouchov, S. Klasky, Analysis and Modeling of the End-to-End I/O Performance on OLCF's Titan Supercomputer in High Performance Computing and Communications; IEEE 15th International Conference on Smart City; IEEE 3rd International Conference on Data Science and Systems (HPCC/SmartCity/DSS), 2017 IEEE 19th International Conference on, IEEE, pp. 1–9, best paper nominee.

(ADIOS tutorial, ORNL)

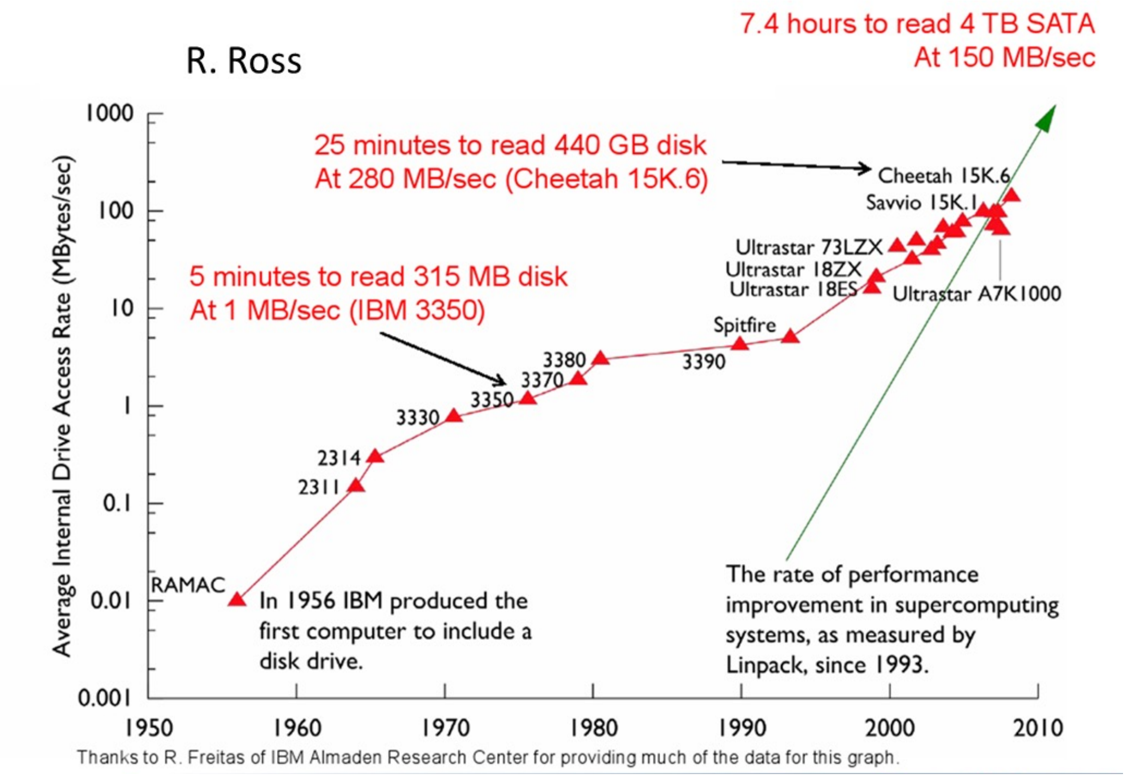
The Data Problem

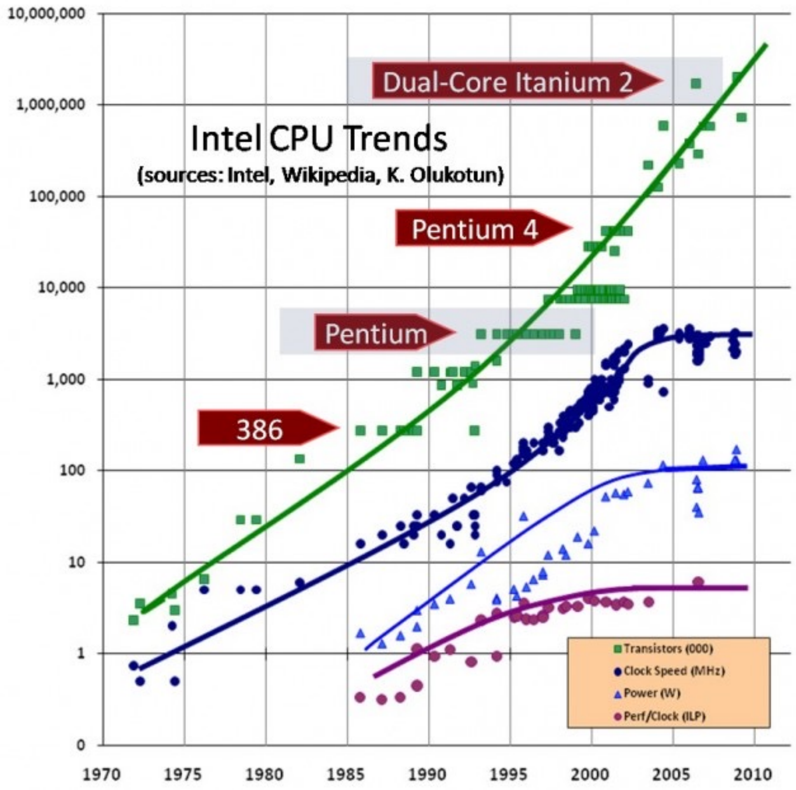
- **Push** from Storage and Network technology, not keeping pace with the growing data demand
  - Current Storage technologies for HPC
  - New storage technologies are giving new opportunities for Storage and I/O
  - Growth of new storage tiers
  - New types of User-Defined Storage for new user-defined tiers
  - **Common Parallel File Systems**
    - Lustre
    - GPFS
    - Burst Buffer File Systems

- **Pull from** Applications
  - HPC Simulations – traditional
  - HPC Simulations – new I/O patterns
  - Experiments – streaming data
  - Observations
- **The need for self-describing data**
  - On line processing
  - Off line processing
  - Data Life-cycle

Disk Transfer Rates over time compared to processor speeds

- Both processor speed and disk speeds have mostly been increasing due to **concurrency**
- **Accelerators** (NVRAM) have been used to speed up the processing power/disk speeds





Parallel filesystem

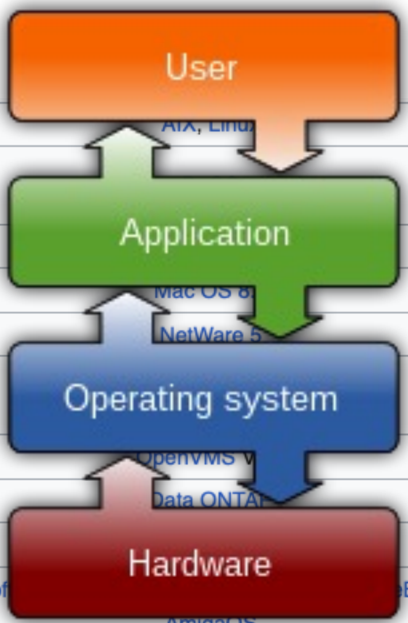
We need a Parallel File System

- High Performance Computing & Big Data requirements have outgrown the capabilities of any single host
  - (data set sizes) > (drive capacities)
  - Single server bandwidth is not sufficient to support access to all data from thousands of clients
- Need a parallel file system that can:
  - Scale capacity/bandwidth
  - Support large numbers of clients
- Lustre and GPFS are popular choices to meet these needs

List of File Systems

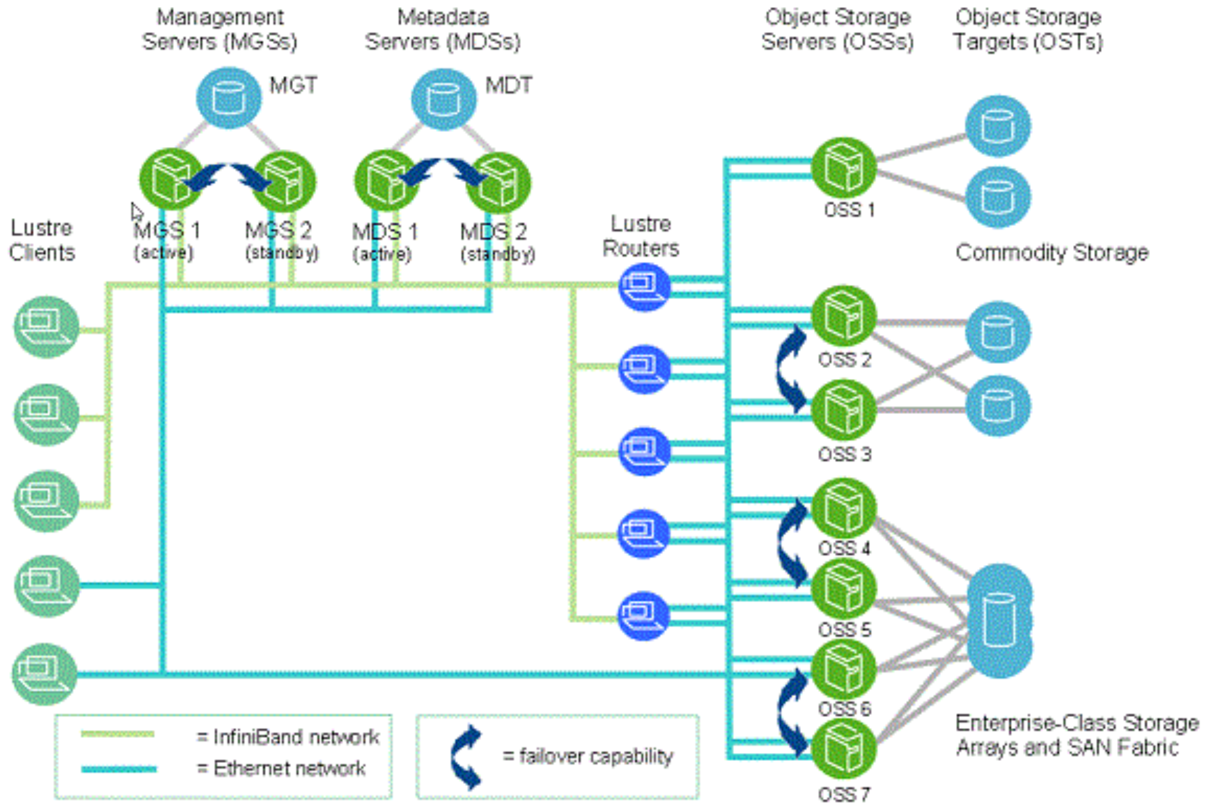
File system	Creator	Year of introduction	Original operating system
DECtape	DEC	1964	PDP-6 Monitor
OS/3x0 FS	IBM	1964	OS/360
Level-D	DEC	1968	TOPS-10
George 3	ICT (later ICL)	1968	George 3
Version 6 Unix file system (V6FS)	Bell Labs	1972	Version 6 Unix
RT-11 file system	DEC	1973	RT-11
Disk Operating System (GEC DOS)	GEC	1973	Core Operating System
CP/M file system	Digital Research (Gary Kildall)	1974	CP/M[1][2]
ODS-1	DEC	1975	RSX-11
GEC DOS filing system extended	GEC	1977	OS4000
FAT (8-bit)	Microsoft (Marc McDonald) for NCR	1977	Microsoft Standalone Disk BASIC-80 (later Micro Standalone Disk BASIC-86)
DOS 3.x	Apple	1978	Apple DOS
UCSD p-System	UCSD	1978	UCSD p-System
CBM DOS	Commodore	1978	Commodore BASIC
Atari DOS	Atari	1979	Atari 8-bit
Version 7 Unix file system (V7FS)	Bell Labs	1979	Version 7 Unix
ODS-2	DEC	1979	OpenVMS
FAT12	Seattle Computer Products (Tim Paterson)	1980	QDOS/86-DOS (later IBM PC DOS 1.0)
ProDOS	Apple	1980	Apple SOS (later ProDOS 8)
DFS	Acorn Computers Ltd	1982	Acorn BBC Micro MOS
ADFS	Acorn Computers Ltd	1983	Acorn Electron (later Arthur/RISC OS)
FFS	Kirk McKusick	1983	4.2BSD
FAT16	IBM, Microsoft	1984	PC DOS 3.0, MS-DOS 3.0
MFS	Apple	1984	System 1
Elektronika BK tape format	NPO "Scientific centre" (now Sitronics)	1985	Vilnius Basic, BK monitor program
HFS	Apple	1985	System 2.1
Amiga OFS[1]	Metacomco for Commodore	1985	Amiga OS
GEMDOS	Digital Research	1985	Atari TOS
NWFS	Novell	1985	NetWare 286
High Sierra	Ecma International	1986	MSCDEX for MS-DOS 3.1/3.2[3]
FAT16B	Compaq	1987	Compaq MS-DOS 3.31

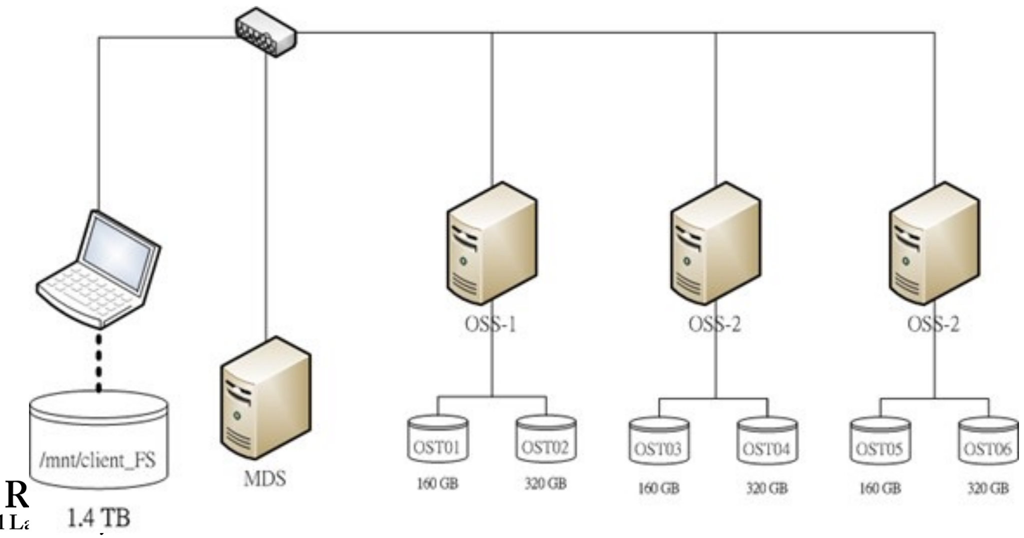
GPFS	IBM	1996	AIX, Linu
Be File System	Be Inc. (D. Giampaolo, Cyril Meurillon)	1996	
Minix V2 FS	Andrew S. Tanenbaum	1997	
HFS Plus	Apple	1998	Mac OS 8
NSS	Novell	1998	NetWare 5
PolyServe File System (PSFS)	PolyServe	1998	
ODS-5	DEC	1998	OpenVMS V
WAFL	NetApp	1998	Data ONTAP
ext3	Stephen Tweedie	1999	
ISO 9660:1999	Ecma International, ISO	1999	Microsof … BSD, … AmigaOS
JFS	IBM	1999	OS/2 Warp Server for e-business
GFS	Sistina (Red Hat)	2000	Linux
ReiserFS	Namesys	2001	Linux
zFS	IBM	2001	z/OS (backported to OS/390)
FATX	Microsoft	2002	Xbox
UFS2	Kirk McKusick	2002	FreeBSD 5.0
OCFS	Oracle Corporation	2002	Linux
SquashFS	Phillip Lougher, Robert Lougher	2002	Linux
VMFS2	VMware	2002	VMware ESX Server 2.0
Lustre	Cluster File Systems[5]	2002	Linux
Fossil	Bell Labs	2003	Plan 9 version 4
Google File System	Google	2003	Linux
ZFS	Sun Microsystems	2004	Solaris
Reiser4	Namesys	2004	Linux
Non-Volatile File System	Palm, Inc.	2004	Palm OS Garnet
BeeGFS	Fraunhofer/ ThinkParQ	2005	Linux
GlusterFS	Gluster Inc.	2005	Linux



Lustre

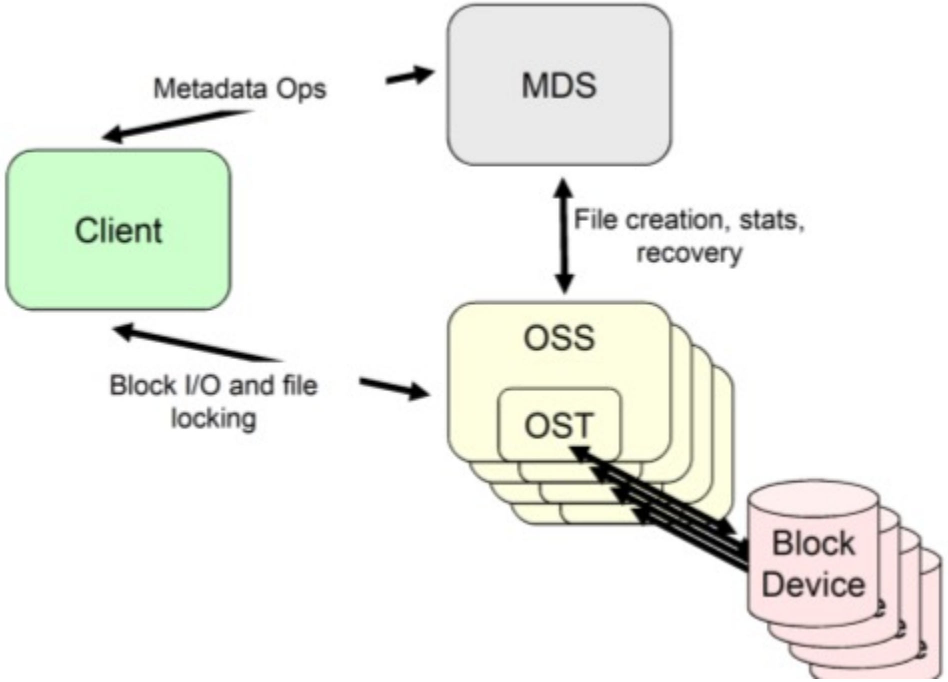
- Distributed file system
- Hierarchical management
- Concurrency from multiple OSTs
- Meta data management with multiple MDTs

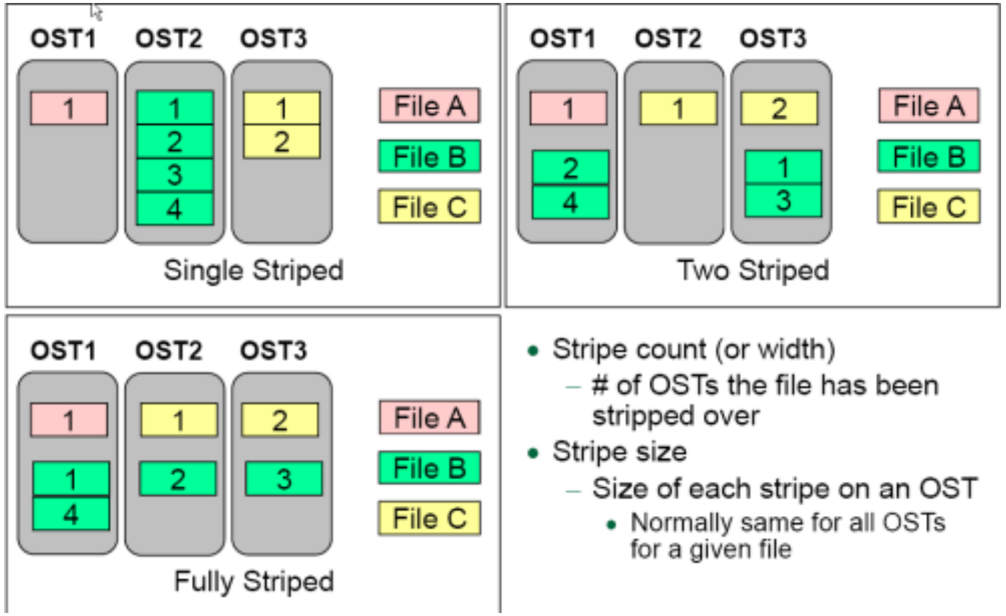




Lustre Components

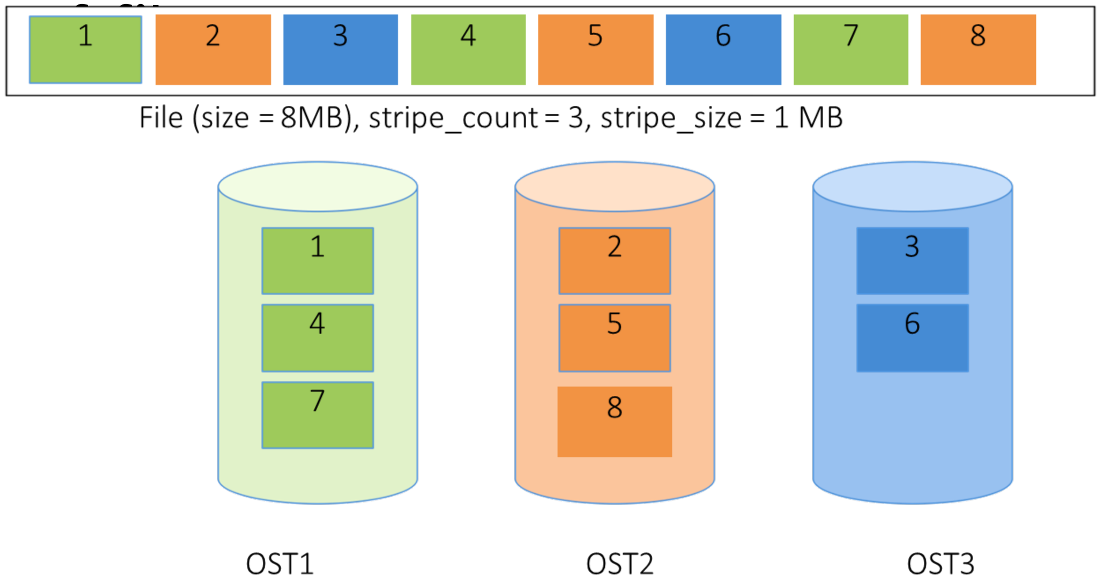
- Lustre consists of four major components
  - MetaData Server (MDS)
  - Object Storage Servers (OSSs)
  - Object Storage Targets (OSTs)
  - Clients
- MDS: track meta data (eg., name, location)
- OST: back-end storage for file object data
- Performance: Striping, alignment, placement

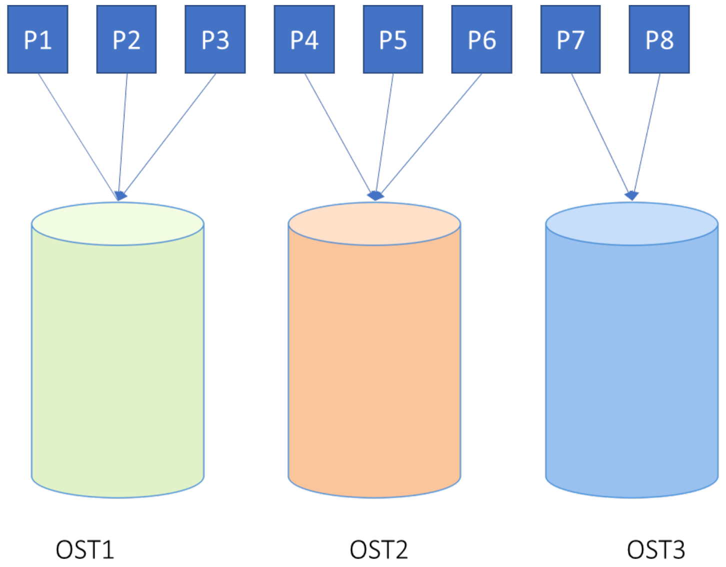


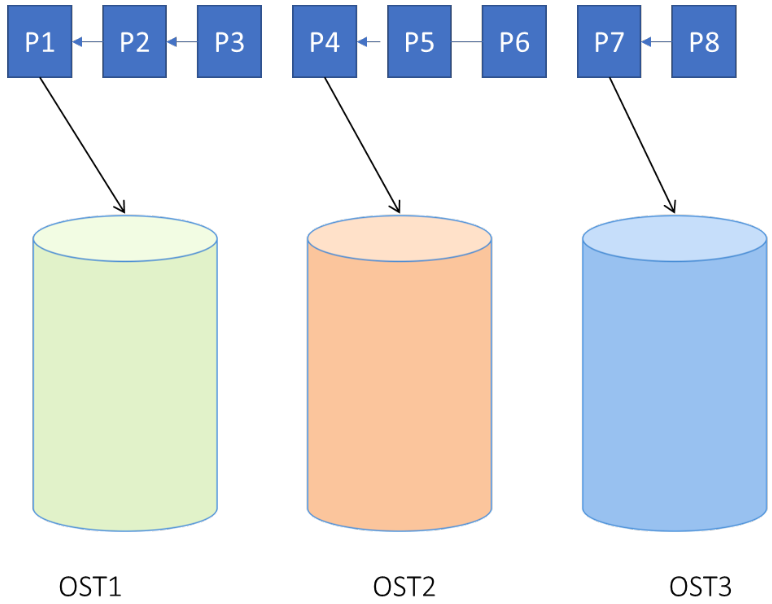


Multiple methods to obtain high performance for Lustre

- Stripe across the disks using one write statement, using ioctls for the stripe_count and stripe_size
- Write multiple files
- Aggregate data to the “best” number





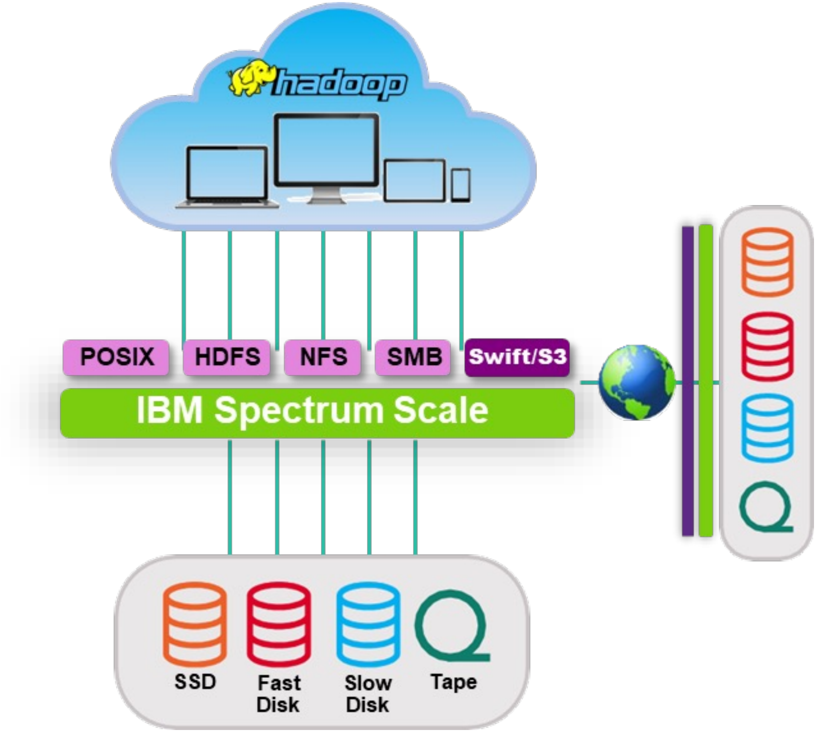


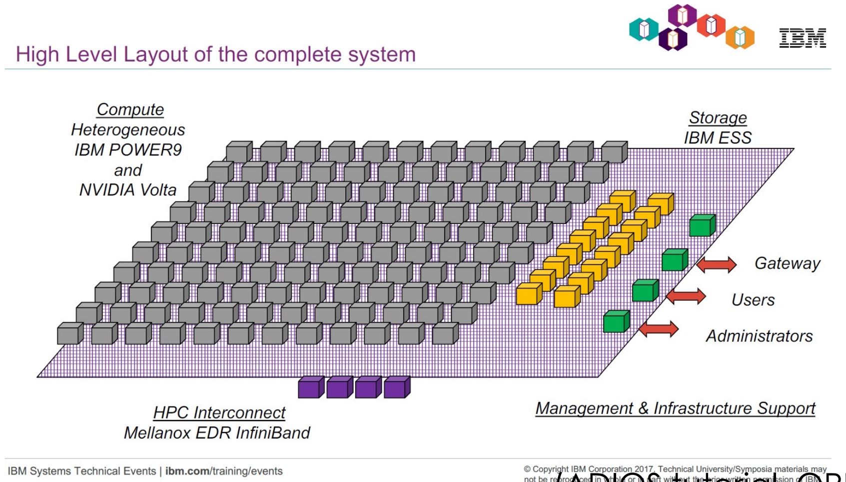
Optimize concurrency by writing to more OSTS: avoid locking when possible

(ADIOS tutorial, ORNL)

General Parallel File System – IBM Spectrum Scale

- 1993: Started as "Tiger Shark" research project at IBM Research Almaden as high-performance filesystem for accessing and processing multimedia data
- Next 20 years: Grew up as General Parallel File System (GPFS) to power the world's largest supercomputers
- Since 2014: Transforming to IBM Spectrum Scale to support new workloads which need to process huge amounts of unstructured data
- Automatic de-staging of cold data to on premise or off premise object storage
- Exchange of data between Spectrum Scale clusters via object storage in the cloud





Difference between GPFS and Lustre → Lustre allows users to set stripe size, stripe count, and even determine which OSTs (GPFS does NOT allow this)

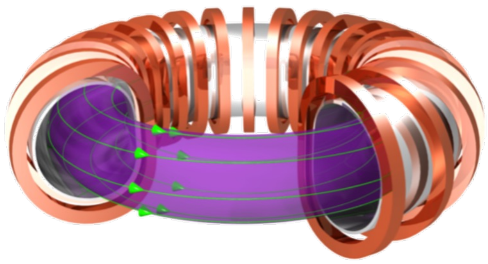
(ADIOS tutorial, ORNL)

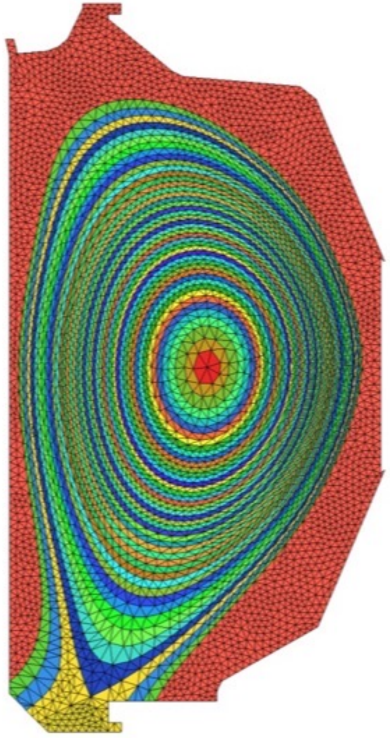
Parallel I/O

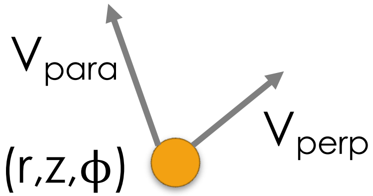
Parallel Application Example: XGC Fusion Simulation

- Gyrokinetic particle-in-cell (PIC) code
- Mesh data
  - High-resolution unstructured mesh
- Particles data
  - 5D particle information
  - Aggregated particle information
  - Representing particle distribution per mesh node
  - Histogram over 2D velocity space (a vertical and perpendicular space)
- Challenge
  - Large scale (e.g., 4096x6 processes on Summit)
  - Large volume of data (e.g., ~GBs per step)
  - Long runs for production scale

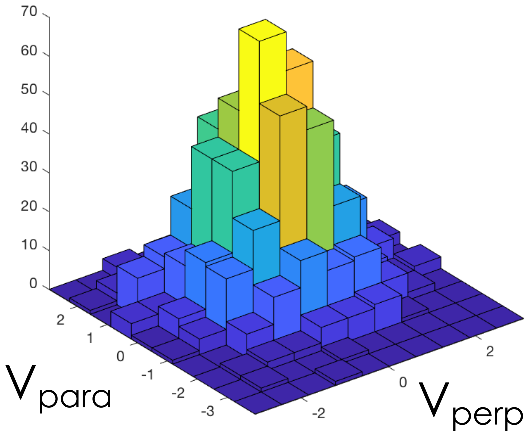
HBPS
High-fidelity
Boundary
Plasma Simulation





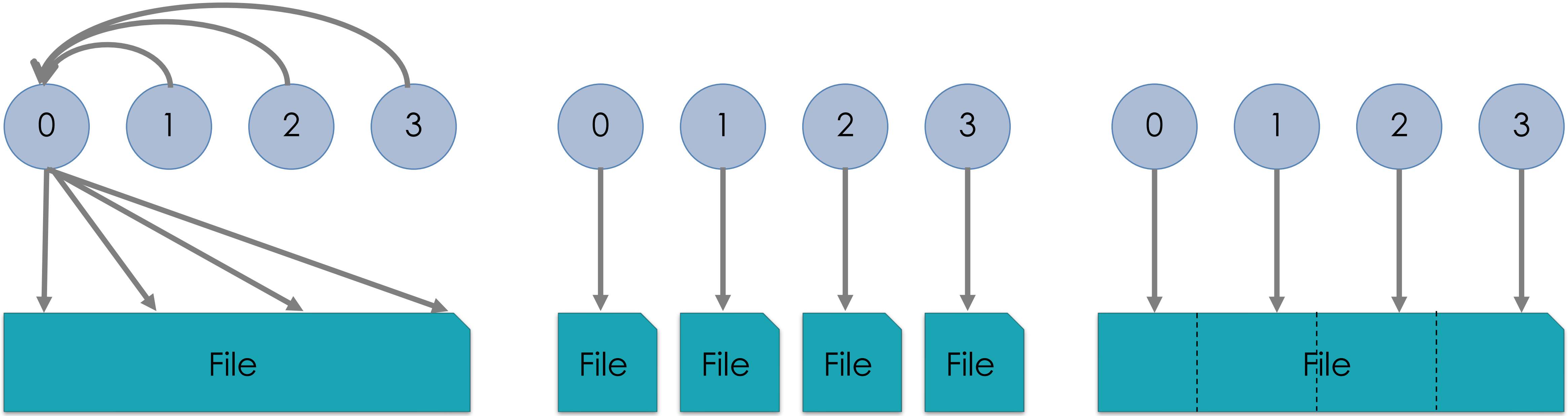


Particle



HPC I/O Patterns

Non-parallel I/O | Parallel Multi-file I/O | Parallel Single-file I/O



Non-parallel I/O

- Serial performance
- Scaling issue
- Memory limit

Parallel Multi-file I/O

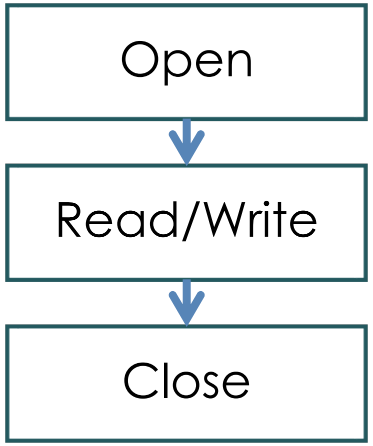
- Metadata issue
- Management issue

Parallel Single-file I/O

- User-friendly
- Sync/lock overhead

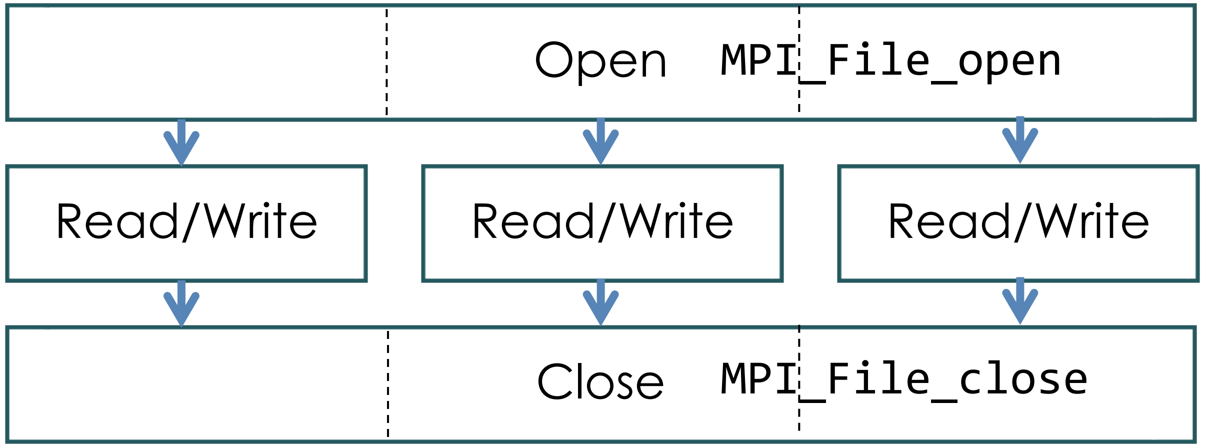
What is MPI I/O?

- I/O interface specification for parallel MPI applications
- Parallel I/O part for MPI
- MPI IO provides
  - Parallel I/O operations
  - Enable to use efficiently parallel file systems
  - Independent/collective I/Os
- Low-level interface
- At the application level, users may want to use of a more abstract library




```
fd = open("foo.txt", O_WRONLY
| O_CREAT, 0644);
write(fd, buf, len);
close(fd)
```

POSIX I/O



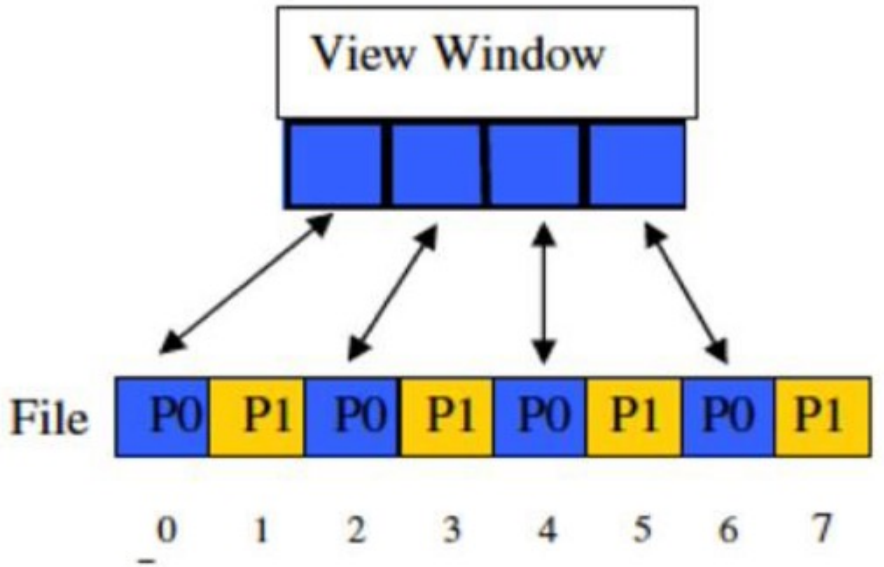
MPI I/O

Main features

- Independent/Collective I/Os
- File View

- MPI_File_read_shared
- MPI_File_write_shared
- MPI_File_read_all
- MPI_File_write_all
- MPI_File_read_at_all
- MPI_File_write_at_all

MPI File set view



(Philip Dickens, U of Maine)

(Code is available at https://github.com/jychoi-hpc/mpiio-example)

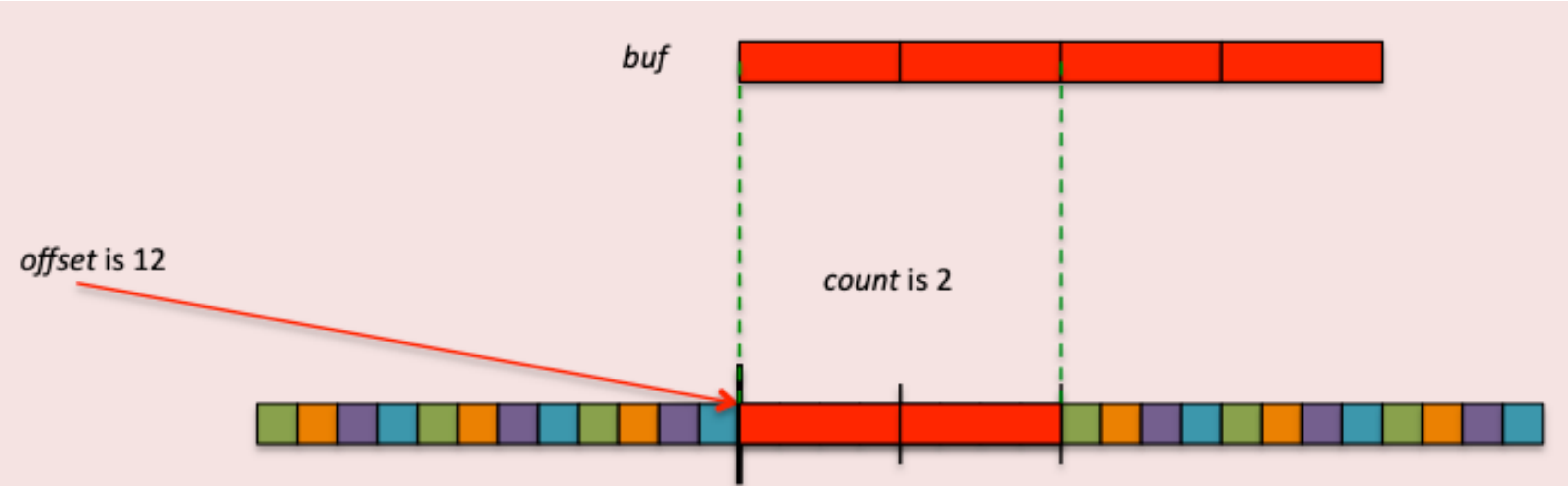
Independent Write

```
// Open a file and shared by all
   MPI_File_open(MPI_COMM_WORLD, "out.bin", MPI_MODE_CREATE | MPI_MODE_RDWR, MPI_INFO_NULL, &fh);

   for (i = 0; i < LEN; i++)
       buf[i] = rank;

   // Independent
   for (k = 0; k < nprocs; k++)
   {
       if (rank == k)
       {
           printf ("rank %d writes\n", rank);
           MPI_File_write_at(fh, rank, buf, 1, MPI_INT, &status);
       }
       MPI_File_sync(fh);
       MPI_Barrier(MPI_COMM_WORLD);
   }

   // Close the file
   MPI_File_sync(fh);
   MPI_File_close(&fh);
```

(Code is available at https://github.com/jychoi-hpc/mpiio-example)

Collective Write

```
// Open a file and shared by all
MPI_File_open(MPI_COMM_WORLD, "out.bin", MPI_MODE_CREATE | MPI_MODE_RDWR, MPI_INFO_NULL, &fh);

for (i = 0; i < LEN; i++)
    buf[i] = rank + 1;

// Set view
offset = (MPI_Offset)rank * 2 * sizeof(int);
MPI_File_set_view(fh, offset, MPI_INT, MPI_INT, "native", MPI_INFO_NULL);
MPI_File_write_all(fh, buf, 2, MPI_INT, &status);

// Close the file
MPI_File_close(&fh);

MPI_Finalize();
return 0;
```



(MPI-IO tutorial, CSCS)

High-level I/O libraries

I/O and Storage Stack

- We encourage the use of self-describing, binary, portable I/O formats
- We encourage users to push the envelope of the I/O Middleware (ADIOS, HDF5, etc.)
- Abstractions should not "force" implementations
  - Use data in streams or files
  - Write data according to the matching of the I/O(??) from the application(s) and the storage layers

Application

DRAM, local, remote

Self Describing Parallel I/O — **ADIOS , HDF5, pnetcdf** maps variables to data output in a file and/or stream

Lower Level I/O — **MPI-IO, POSIX** Places the data to the storage system, often optimizing the data from the I/O path to the storage system

SSD/NVRAM — **Burst Buffer optimizations**

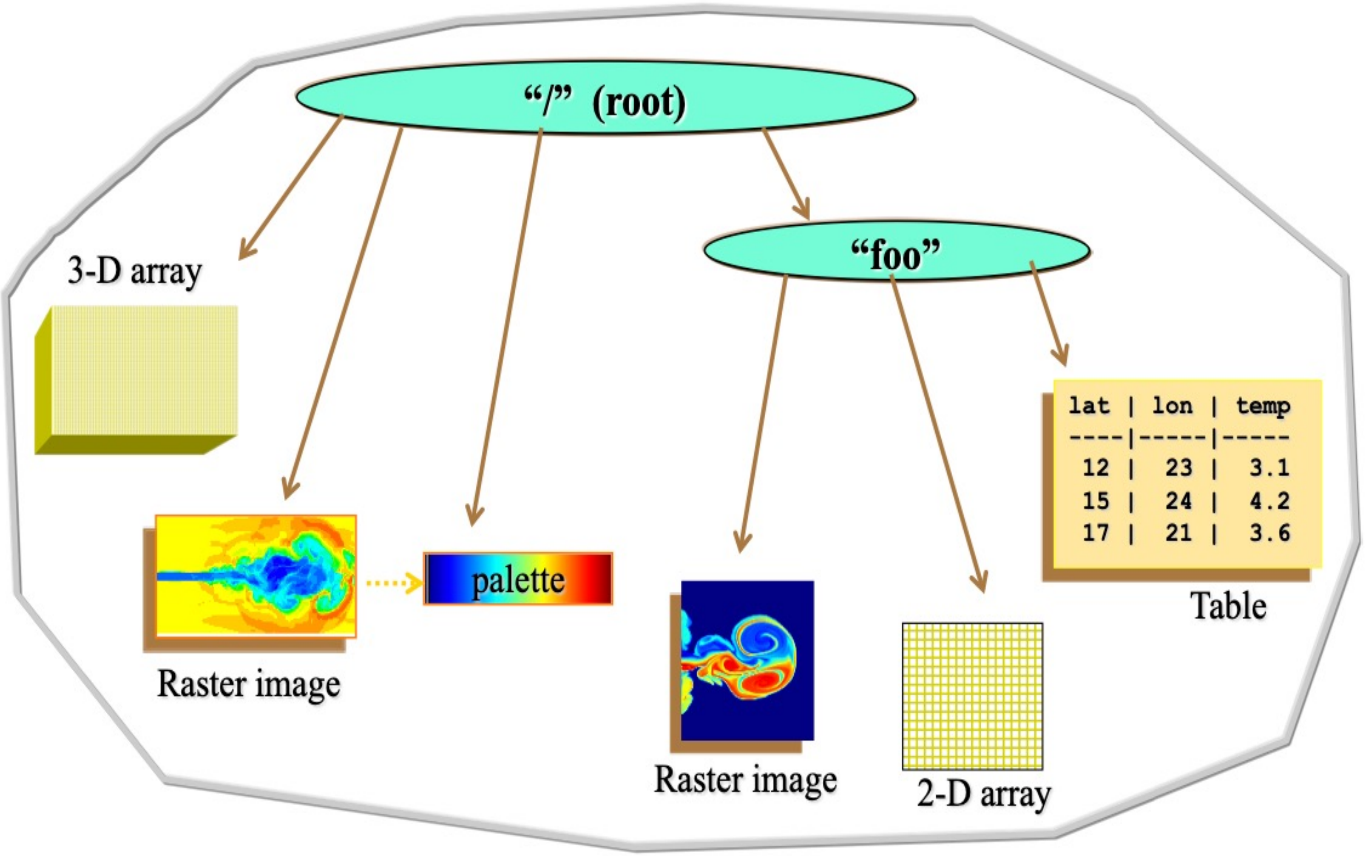
Parallel File System — **GPFS, Lustre, …** maintains logical space and provides efficient access to data

Cold Storage — **TAPE, HPSS, …**

(ADIOS tutorial, ORNL)

HDF5 and ADIOS common features

- Container structure to manage data collection
- Various data object
- Meta data
- Portable file format
- Multi-platform and binding
- Data compression
- Tools and services



(HDF5 tutorial, hdf5group.org)

ADIOS: High-Performance Publisher/Subscriber I/O framework

Vision

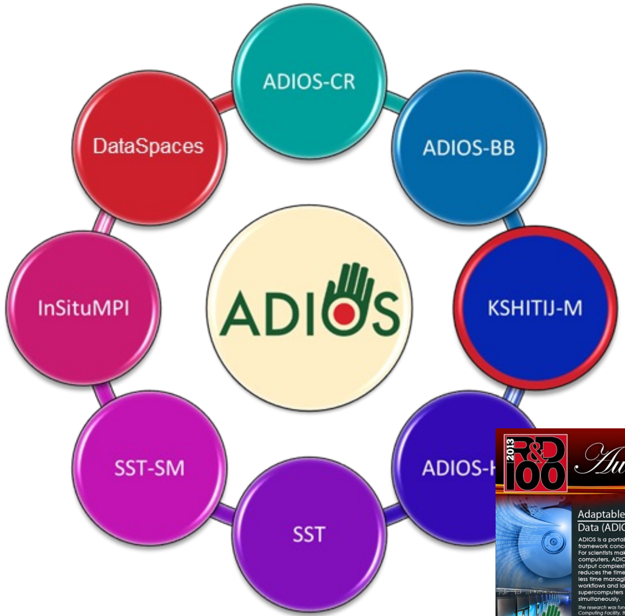
Create a high performance I/O abstraction to allow for on-line/off-line memory/file data subscription service

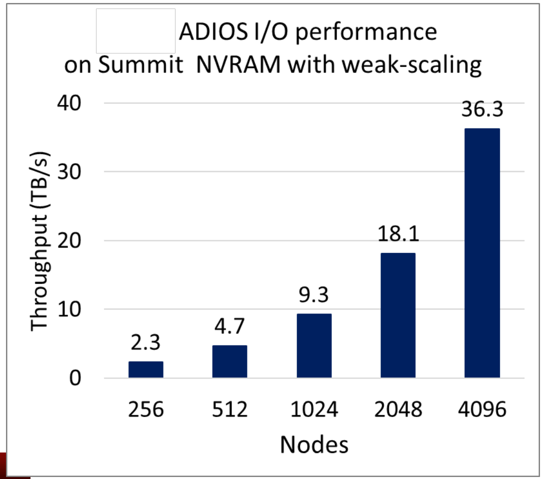
Create a sustainable solution to work with multi-tier storage and memory systems

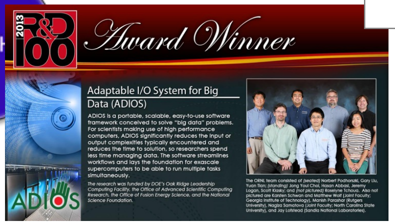
Research Details

- Declarative, publish/subscribe API is separated from the I/O strategy and use of multi-tier storage
- Multiple implementations (engines) provide functionality and performance in different use cases
- Rigorous testing ensures portability
- Data reduction techniques are incorporated to decrease storage cost

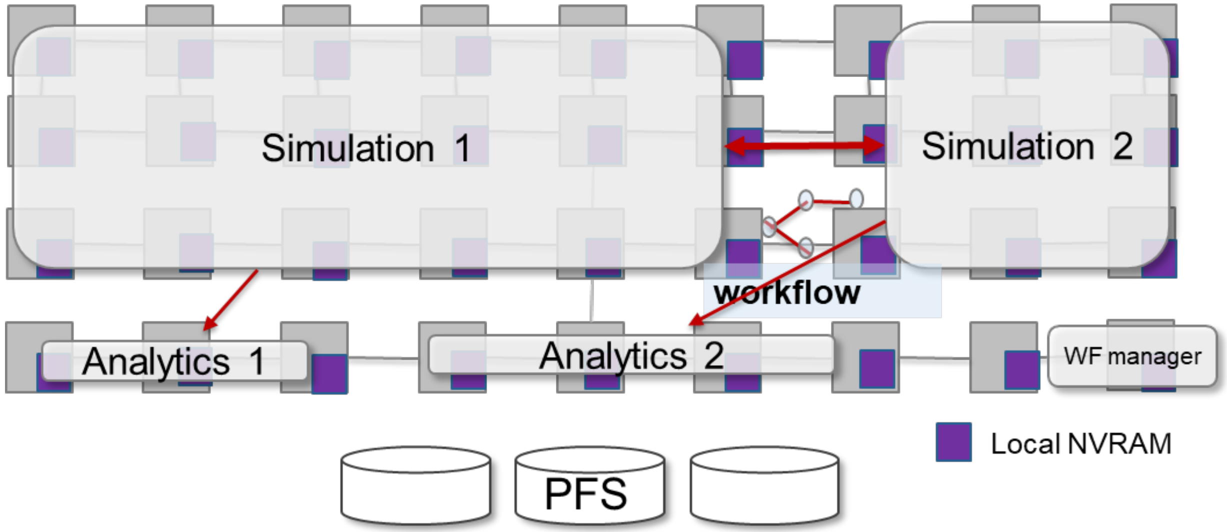
https://github.com/ornladios/ADIOS2



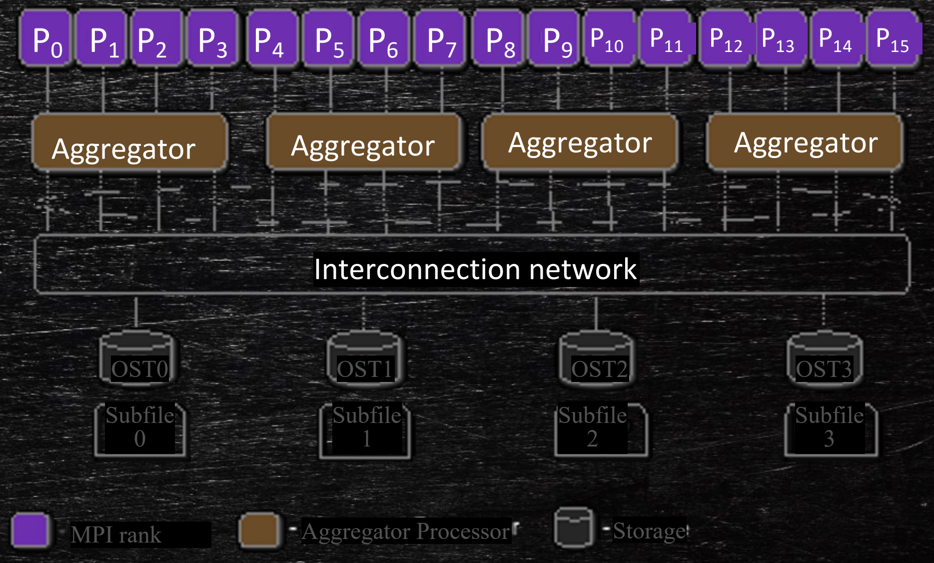




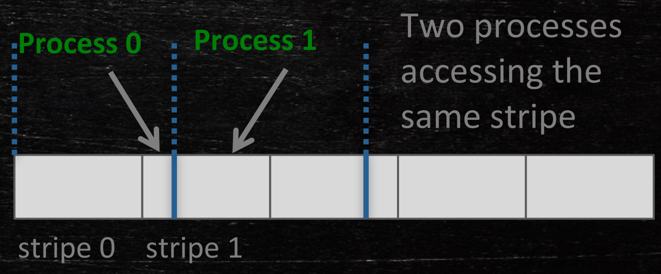
Writing particles on Summit's local NVRAM: 18.3 TB in 0.5 seconds on 4K nodes



Optimizations for a parallel file system



- Avoid latency (of small writes): Buffer data for large bursts – use a type of self-describing log file format
- Avoid accessing a file system target from many processes at once
  - Aggregate to a small number of actual writers:
  - Avoid lock contention
  - Striping correctly & writing to subfiles
- Avoid global communication
- Topology-aware data movement that takes advantage of topology
  - Find the closest I/O node to each writer
  - Minimize data movement across racks/mid-planes



Application	Nodes/GPUs	Data Size/step	I/O speed
SPECFEM3D	3200/19200	250 TB	~2 TB/sec
GTC	512/3072	2.6 TB	~2 TB/sec
XGC	512/3072	64 TB	1.2 TB/sec
LAMMPS	512/3072	457 GB	1 TB/sec

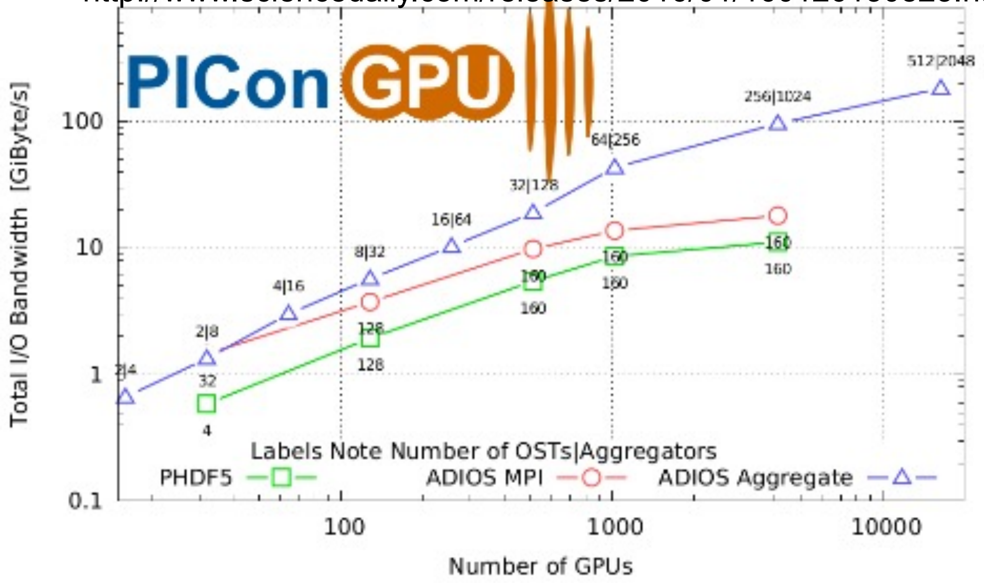
Liu, Q., Klasky, S., et al. "Hello ADIOS: the challenges and lessons of developing leadership class I/O frameworks." Concurrency and Computation: Practice and Experience 26.7 (2014): 1453-1473.

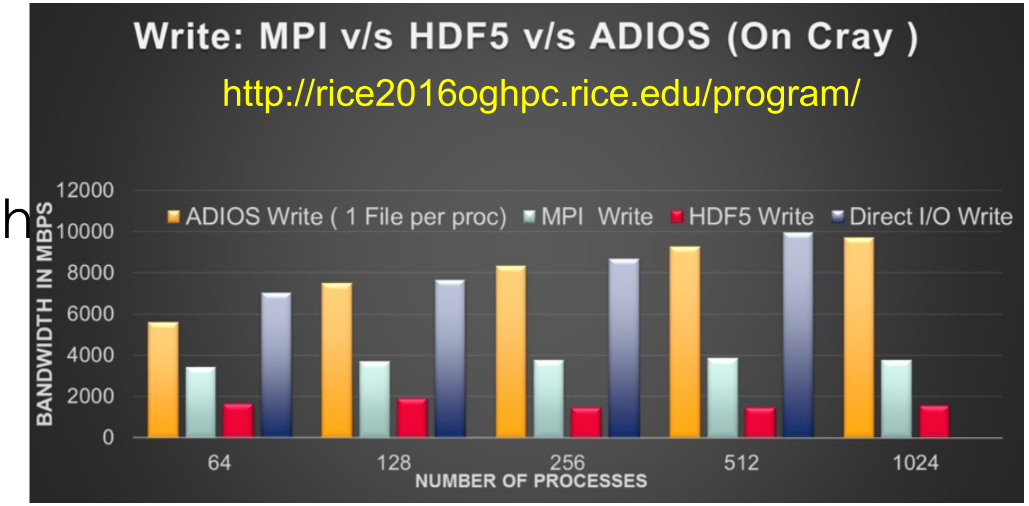
klasky@ornl.gov, scottk007@gmail.com

Impact to some LCF applications

- Accelerators – PIConGPU
  - M. Bussmann, et al. - HZDR
  - Study laser-driven acceleration of ion beams and its use for therapy of cancer
  - Computational laboratory for real-time processing for optimizing parameters of the laser
  - Over 184 GB/s on 16K nodes on Titan
  - 80 TB / output step
- Seismic Imaging – RTM by Total Inc.
  - Pierre-Yves Aquilanti, TOTAL E&P in context of a CRADA
  - TBs as inputs, outputs PBs of results along with intermediate data
  - Company conducted comparison tests among several I/O solutions. ADIOS is their choice for other codes: FWI, Kirchoff

http://www.sciencedaily.com/releases/2016/01/160126130823.htm



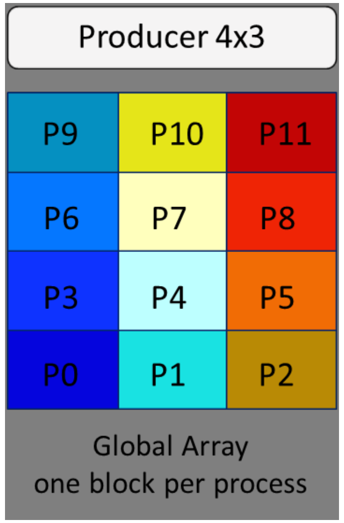


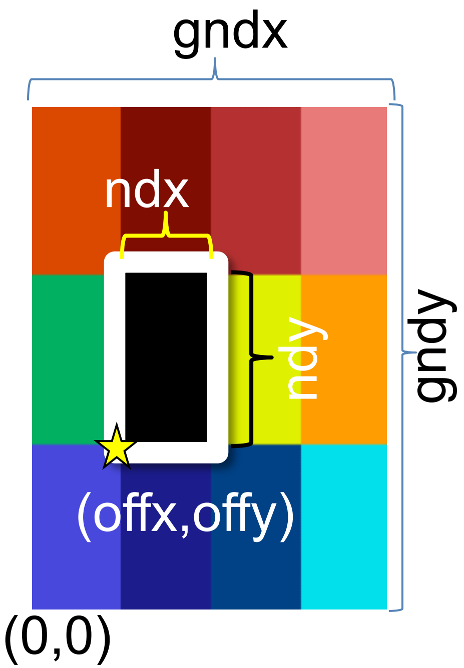
ADIOS Approach: “How”

- I/O calls are of declarative nature in ADIOS
  - which process writes/reads what
    - add a local array into a global space (virtually)
  - EndStep() indicates that the user is done declaring all pieces that go into the particular dataset in that output step or what pieces each process gets
- I/O strategy is separated from the user code
  - aggregation, number of sub-files, target file-system hacks, and final file format not expressed at the code level
- This allows users
  - to choose the best method available on a system without modifying the source code
- This allows developers
  - to create a new method that’s immediately available to applications
  - to push data to other applications, remote systems or cloud storage instead of a local filesystem

ADIOS Variable

```
adios2::IO io = adios.DeclareIO("Analysis_Data");

if (!io.InConfigFile()) {

      io.SetEngine("BPFile");

}

adios2::Variable<double> varT = io.DefineVariable<double>
(
    "Temperature",          // name in output/input
    {gndx,gndy,gndz},       // Global dimensions (3D here)
    {offx, offy, offz},     // starting offsets in global space
    {nx,ny,nz}              // local size
);

io.DefineAttribute <std::string>("unit", "C", "Temperature");
```





Shape = {gndx, gndy}

Start = {offx, offy}

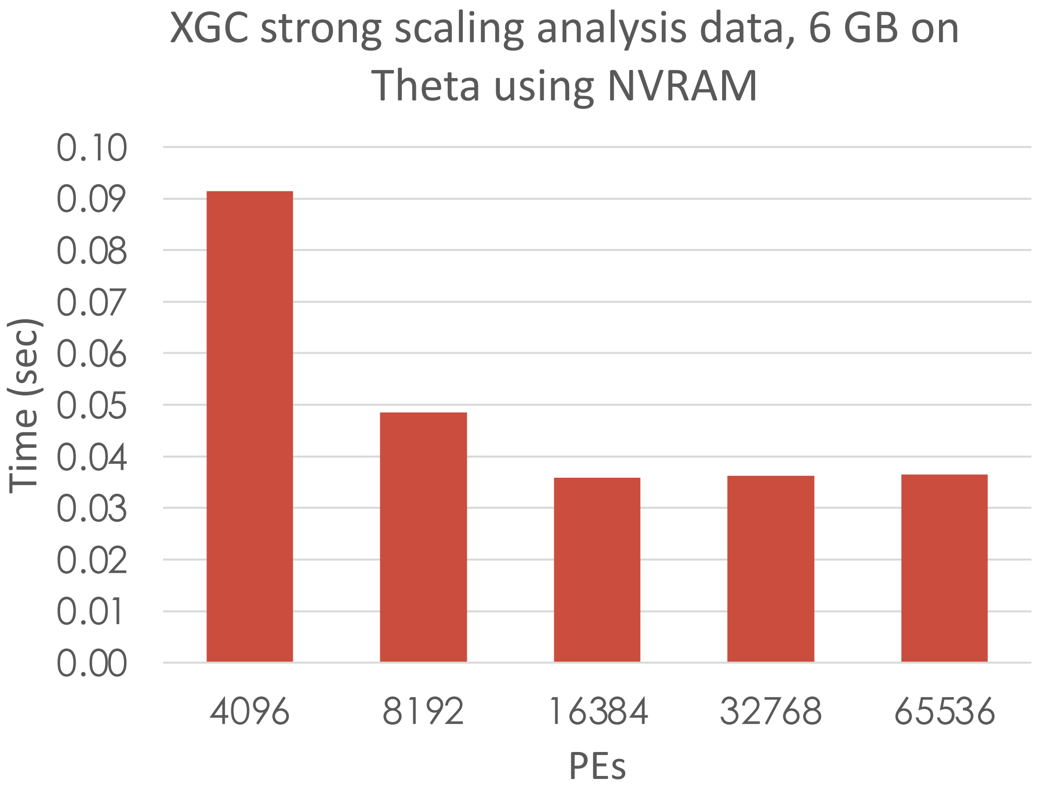
Count = {ndx, ndy}

```
double   Temperature         10*{20, 30, 40} = 8.86367e-07 / 200
string   Temperature/unit    attr   = "C"
```

Engine object

- To perform the IO

```
adios2::Engine writer =
io.Open("analysis.bp",
        adios2::Mode::Write);


writer.BeginStep()

   writer.Put(varT, T.data());

writer.EndStep()


writer.Close()
```



Put API explained

engine.Put(varT, T.data())

- Equivalent to “write”

engine.Put(varT, T.data(), adios2::Mode:Deferred)

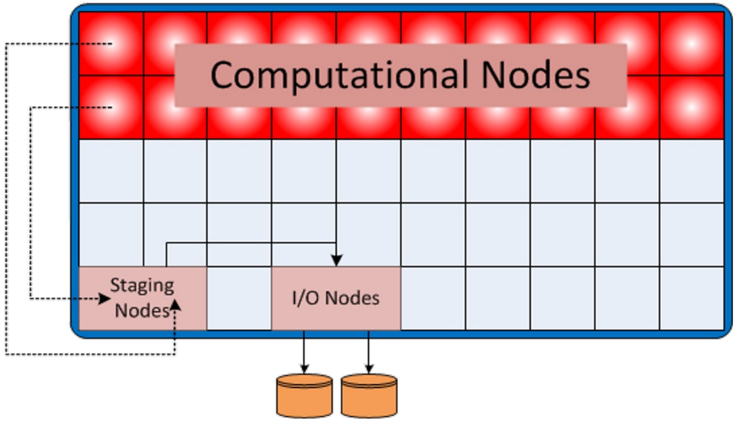
- This does NOT do the I/O (to disk, stream, etc.) once put return.
- you can only reuse the data pointer after calling **engine.EndStep()**

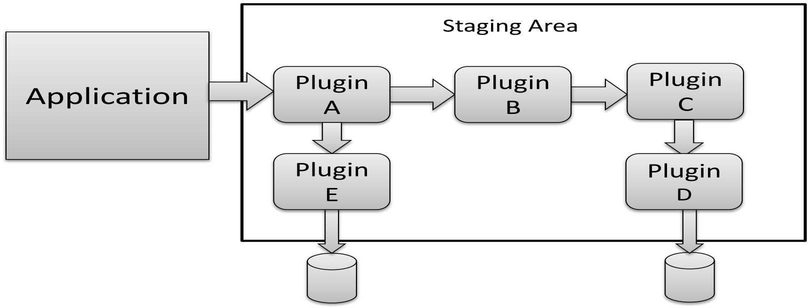
engine.Put(varT, T.data(), adios2::Mode:Sync)

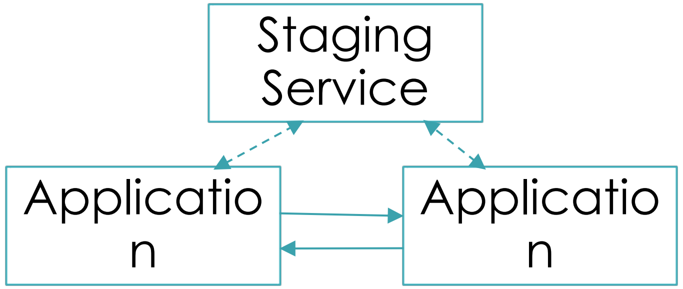
- This makes sure data is flushed or buffered before put returns
- Get() works the same way
- The default mode is deferred
- Disk I/O:
  - Put only flushes to disk if the buffer is full, otherwise flushed in EndStep()
  - No difference in performance between using sync and deferred Put

Coupling with staging

- Move data directly to remote memory in a "staging" area
  - a.k.a in situ, online, concurrent processing
- Decouple application performance from storage performance
- Enhance data services by providing an intermediate common area (staging) that reduces file system access costs
- Address performance/variability issue
- Components:
  - Asynchronous I/O buffers from Applications
  - Services provided as plugins:
  - Analytics & Visualization
  - Data Reduction







Oak Ridge National Laboratory

Key ideas for good performance of ADIOS for large writes

- Avoid latency (of small writes)
  - Buffer data for large bursts
- Avoid global communication
  - ADIOS has that for metadata only, which can even be postponed for post-processing
- Later: Topology-aware data movement that takes advantage of topology
  - Find the closest I/O node to each writer
  - Minimize data movement across racks/mid-planes (on Bluegene/Q)

ADIOS-BP stream/file format

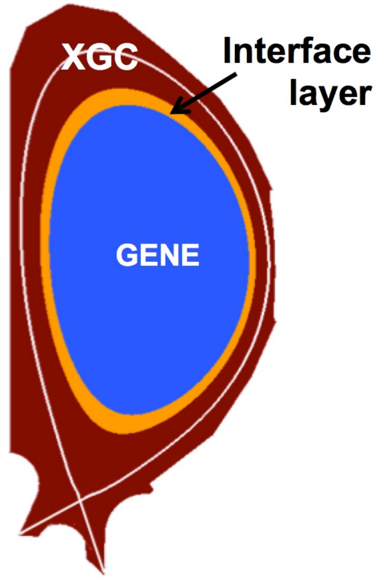
- Self-describing data format
- Allows data from each node to be written independently with each other with metadata
- Ability to create a separate metadata file when "sub-files" are generated
- Allows variables to be individually compressed
- Has a schema to introspect the information, on each process
- Format is for "data-in-motion" and "data-at-rest"

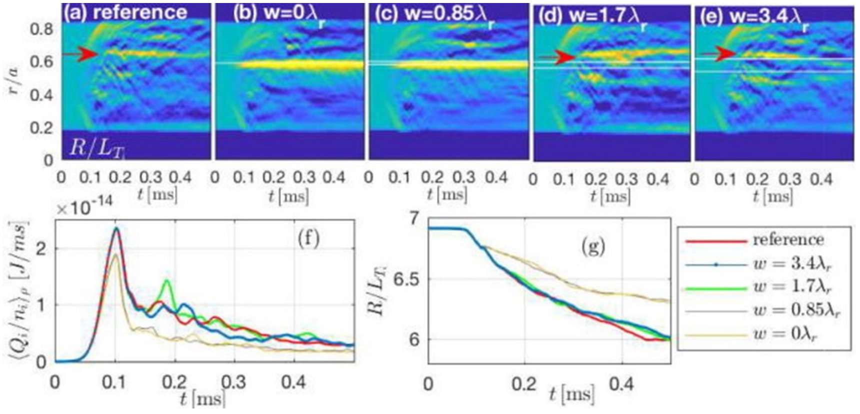
PI: Amitava Bhattacharjee, PPPL,
C.S. Chang, PPPL

2.2.2.05 ADSE12-WDMApp: High-Fidelity Whole Device Modeling of Magnetically Confined Fusion Plasmas

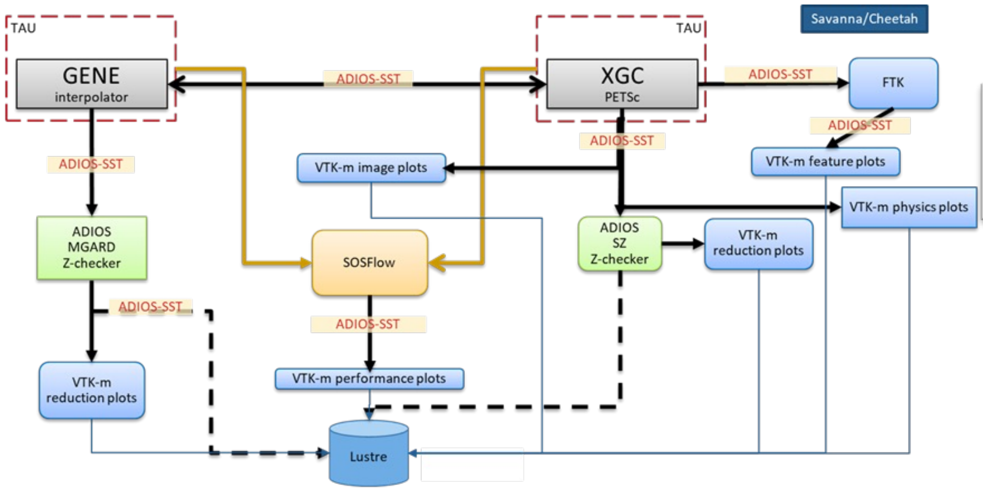
- Different physics solved in different physical regions of detector (spatial coupling)
- Core simulation: **GENE**
  Edge simulation: **XGC**
  Separate teams, **separate codes**
- Recently demonstrated first-ever successful kinetic coupling of this kind
- Data Generated by one coupled simulation is predicted to be > 10 PB/day o

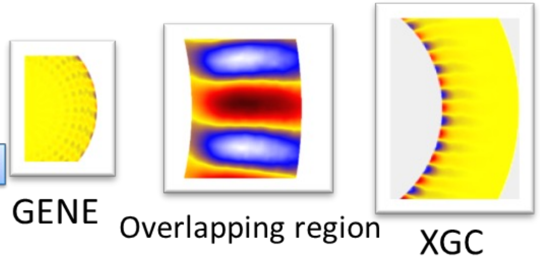
Core-edge coupling





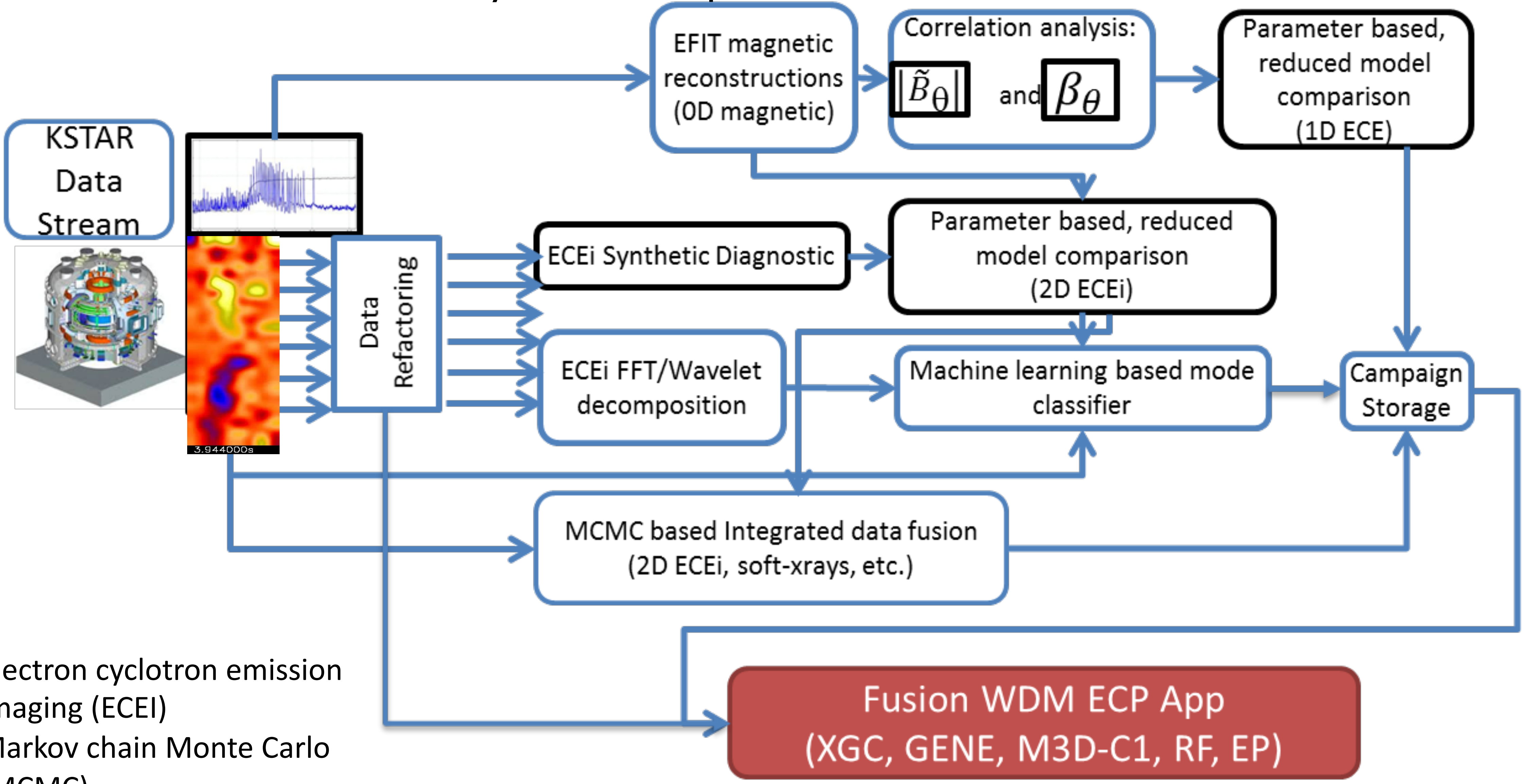
J. Dominski; S. Ku; C.-S. Chang; J. Choi; E. Suchyta; S. Parker; S. Klasky; A. Bhattacharjee; *Physics of Plasmas* **2018,** 25, DOI: 10.1063/1.5044707





New I/O Pattern: emerging from running complex in situ workflows

Near Real Time Analysis of Experimental Data



Electron cyclotron emission imaging (ECEI)
Markov chain Monte Carlo (MCMC)

New I/O Pattern: Streaming, reducing data from experiments to HPC with ML

I/O Performance measurement tools

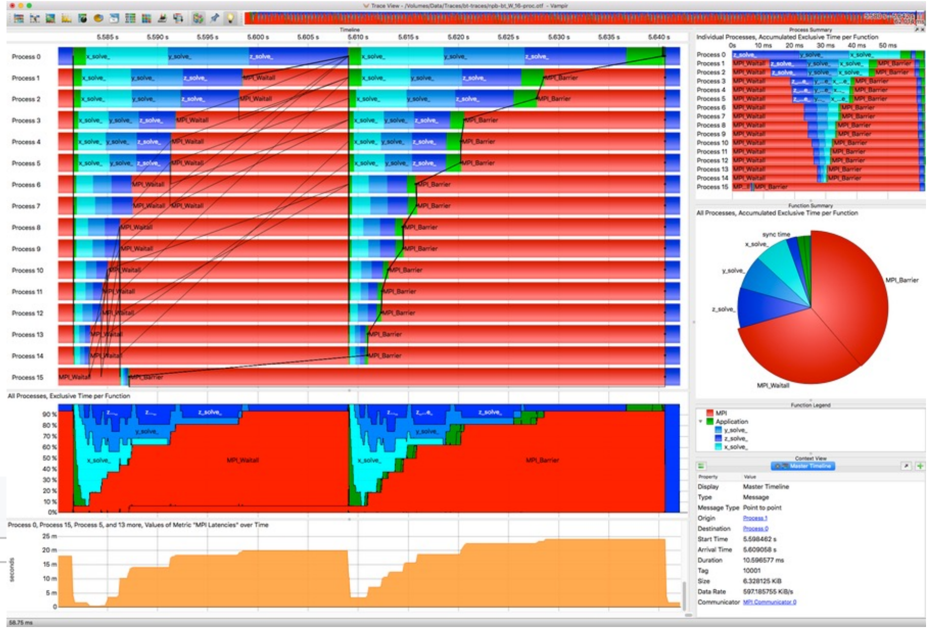
HPC I/O Tools

PAPI
The Performance Application Programming Interface

ICL UT

Score-P
Scalable performance measurement infrastructure for parallel codes

TAU Performance System®



darshan-hpc/
darshan

Darshan I/O characterization tool

16 Contributors | 185 Issues | 3 Discussions | 40 Stars | 18 Forks

hpc / ior Public

HPC IO Benchmark Repository build passing

This repository contains the IOR and mdtest parallel I/O benchmarks. The official IOR/mdtest documentati
be found in the `docs/` subdirectory or on Read the Docs.

Hands-on demonstration

Lustre Commands

$ lfs mdts

$ lfs osts

```
jyc@login11 restart_dir $ lfs mdts
MDTS:
0: scratch-MDT0000_UUID ACTIVE
1: scratch-MDT0001_UUID ACTIVE
2: scratch-MDT0002_UUID ACTIVE
3: scratch-MDT0003_UUID ACTIVE
4: scratch-MDT0004_UUID ACTIVE
5: scratch-MDT0005_UUID ACTIVE
6: scratch-MDT0006_UUID ACTIVE
7: scratch-MDT0007_UUID ACTIVE
8: scratch-MDT0008_UUID ACTIVE
9: scratch-MDT0009_UUID ACTIVE
10: scratch-MDT000a_UUID ACTIVE
11: scratch-MDT000b_UUID ACTIVE
12: scratch-MDT000c_UUID ACTIVE
13: scratch-MDT000d_UUID ACTIVE
14: scratch-MDT000e_UUID ACTIVE
15: scratch-MDT000f_UUID ACTIVE
```

```
jyc@login11 restart_dir $ lfs osts
OBDS:
0: scratch-OST0000_UUID ACTIVE
1: scratch-OST0001_UUID ACTIVE
2: scratch-OST0002_UUID ACTIVE
3: scratch-OST0003_UUID ACTIVE
4: scratch-OST0004_UUID ACTIVE
5: scratch-OST0005_UUID ACTIVE
6: scratch-OST0006_UUID ACTIVE
7: scratch-OST0007_UUID ACTIVE
8: scratch-OST0008_UUID ACTIVE
9: scratch-OST0009_UUID ACTIVE
10: scratch-OST000a UUID ACTIVE
266: scratch-OST010a_UUID ACTIVE
267: scratch-OST010b_UUID ACTIVE
268: scratch-OST010c_UUID ACTIVE
269: scratch-OST010d_UUID ACTIVE
270: scratch-OST010e_UUID ACTIVE
271: scratch-OST010f_UUID ACTIVE
272: scratch-OST0110_UUID ACTIVE
273: scratch-OST0111_UUID ACTIVE
```

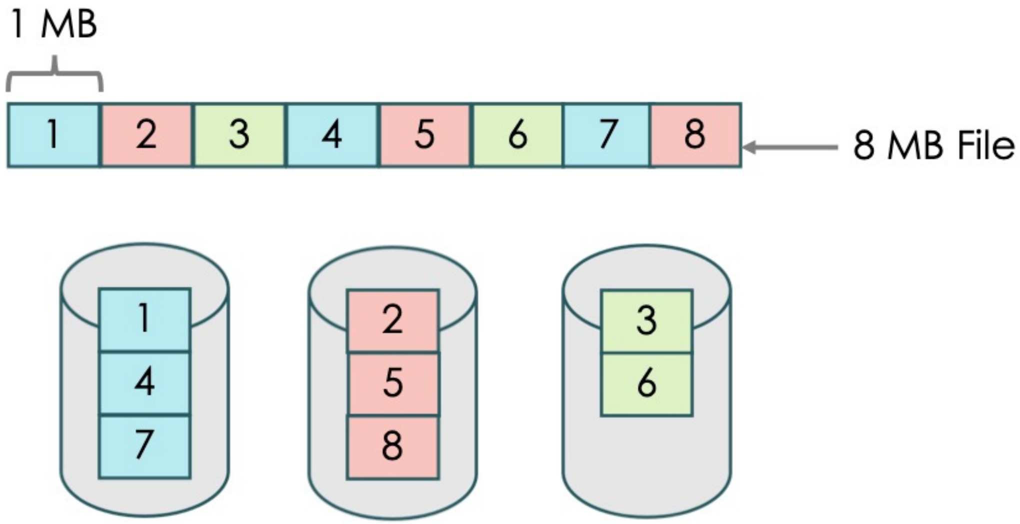
$ lfs du

MDTs

OSTs

```
jyc@login11 restart_dir $ lfs df -h
UUID                       bytes        Used   Available Use% Mounted on
scratch-MDT0000_UUID       71.9T       84.8G       71.0T   1% /pscratch[MDT:0]
scratch-MDT0001_UUID       71.9T       27.6G       71.1T   1% /pscratch[MDT:1]
scratch-MDT0002_UUID       71.9T        9.3G       71.1T   1% /pscratch[MDT:2]
scratch-MDT0003_UUID       71.9T       18.8G       71.1T   1% /pscratch[MDT:3]
scratch-MDT0004_UUID       71.9T        7.1G       71.1T   1% /pscratch[MDT:4]
scratch-MDT0005_UUID       71.9T       14.1G       71.1T   1% /pscratch[MDT:5]
scratch-MDT0006_UUID       71.9T       31.4G       71.1T   1% /pscratch[MDT:6]
scratch-MDT0007_UUID       71.9T       21.3G       71.1T   1% /pscratch[MDT:7]
scratch-MDT0008_UUID       71.9T        5.3G       71.1T   1% /pscratch[MDT:8]
scratch-MDT0009_UUID       71.9T       17.5G       71.1T   1% /pscratch[MDT:9]
scratch-MDT000a_UUID       71.9T       10.5G       71.1T   1% /pscratch[MDT:10]
scratch-MDT000b_UUID       71.9T       11.7G       71.1T   1% /pscratch[MDT:11]
scratch-MDT000c_UUID       71.9T       14.7G       71.1T   1% /pscratch[MDT:12]
scratch-MDT000d_UUID       71.9T       17.6G       71.1T   1% /pscratch[MDT:13]
scratch-MDT000e_UUID       71.9T        5.7G       71.1T   1% /pscratch[MDT:14]
scratch-MDT000f_UUID       71.9T       21.8G       71.1T   1% /pscratch[MDT:15]
scratch-OST0000_UUID      121.3T       83.5T       36.5T  70% /pscratch[OST:0]
scratch-OST0001_UUID      121.3T       89.7T       30.3T  75% /pscratch[OST:1]
scratch-OST0002_UUID      121.3T       90.0T       30.0T  75% /pscratch[OST:2]
scratch-OST0003_UUID      121.3T       96.4T       23.6T  81% /pscratch[OST:3]
scratch-OST0004_UUID      121.3T       89.5T       30.5T  75% /pscratch[OST:4]
scratch-OST0005_UUID      121.3T       90.6T       29.5T  76% /pscratch[OST:5]
scratch-OST0006_UUID      121.3T       88.8T       31.3T  74% /pscratch[OST:6]
scratch-OST0007_UUID      121.3T       89.3T       30.8T  75% /pscratch[OST:7]
scratch-OST0008_UUID      121.3T       88.3T       31.7T  74% /pscratch[OST:8]
scratch-OST0009_UUID      121.3T       89.9T       30.1T  75% /pscratch[OST:9]
scratch-OST000a_UUID      121.3T       82.5T       37.5T  69% /pscratch[OST:10]
scratch-OST000b_UUID      121.3T       89.1T       30.9T  75% /pscratch[OST:11]
scratch-OST000c_UUID      121.3T       89.2T       30.8T  75% /pscratch[OST:12]
```

$ lfs getstripe



(Rick Mohr, ORNL)

stripe_count = 3
stripe_size = 1 MB

```
jyc@login11 restart_dir $ lfs getstripe xgc.restart.00020.bp/data.*
xgc.restart.00020.bp/data.0
lmm_stripe_count:  1
lmm_stripe_size:   1048576
lmm_pattern:       raid0
lmm_layout_gen:    0
lmm_stripe_offset: 109
        obdidx           objid           objid           group
           109         2225533        0x21f57d    0x27c0000407

xgc.restart.00020.bp/data.1
lmm_stripe_count:  1
lmm_stripe_size:   1048576
lmm_pattern:       raid0
lmm_layout_gen:    0
lmm_stripe_offset: 222
        obdidx           objid           objid           group
           222         2369928        0x242988    0x4400000418

xgc.restart.00020.bp/data.10
lmm_stripe_count:  1
lmm_stripe_size:   1048576
lmm_pattern:       raid0
lmm_layout_gen:    0
lmm_stripe_offset: 103
        obdidx           objid           objid           group
           103         2223077        0x21ebe5    0x2640000424
```

Backup

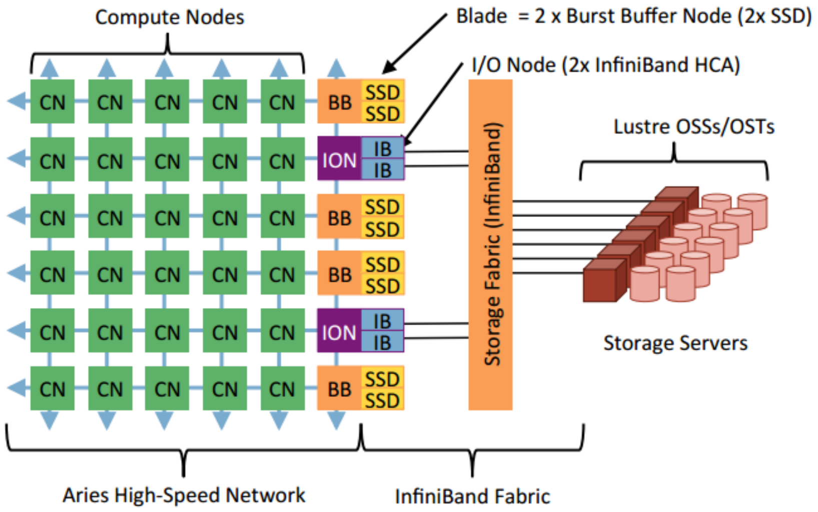
Push: The widening compute-data gap on multi-tier storage

- Filesystem/network bandwidth falls behind CPU/memory: Fewer bytes/operation -Need efficient I/O
- Filesystem has an additional layer (Burst Buffer) which is different on all of the LCFs/NERSC- Need new functionality to write/read to all storage layers

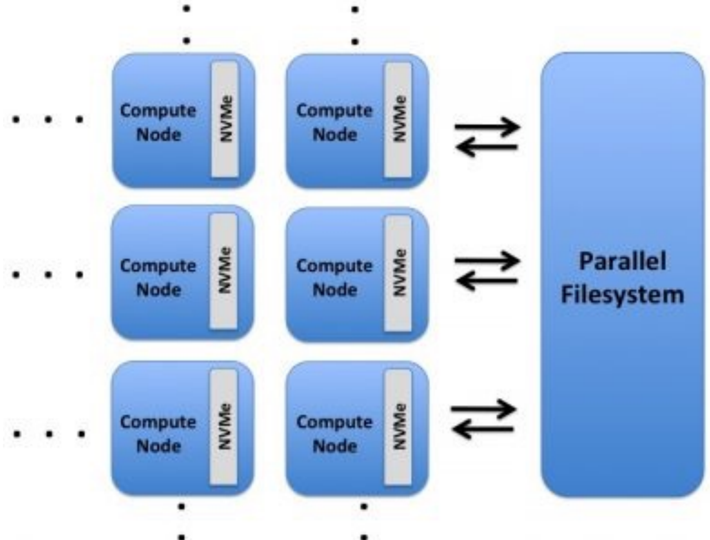
Feature	Titan	Summit
Peak Flops	27 PF	200 PF
Application Performance	Baseline	5-10x Titan
Number of Nodes	18,688	~4,600
Node performance	1.4 TF	> 40 TF
Memory per Node	32 GB DDR3 + 6 GB GDDR5	512 GB DDR4 + 96 GB HBM
NV memory per Node	0	1600 GB
Total System Memory	710 TB (600 TB DDR3 + 110 TB GDDR5))	14X 10 PB (2.3 PB DDR4 + 0.4 PB HBM + 7.4 PB NVRAM)
System Interconnect (node injection bandwidth)	Gemini (6.4 GB/s)	Dual Rail EDR-IB (23 GB/s)
Interconnect Topology	3D Torus	Non-blocking Fat Tree
Processors per node	1 AMD Opteron™ 1 NVIDIA Kepler™	2.5X 2 IBM POWER9™ 6 NVIDIA Volta™

I/O optimizations for next generation HPCs

	Summit	Theta	Cori2
System Peak (PF)	150 PF	> 8.5	1.9
Processors	2 POWER9	KNL	KNL
GPU	6 NVIDIA Volta		
Cores Per Node	12 or 24	64	68
HBM		16 GB	16 GB
DDR4	512 GB	192 GB	96 GB
BurstBuffer / SSD/NVMe	800 GB per node	128 GB per node	1.8 PB total
Filesystem	GPFS	Lustre	Lustre
Interconnect	Dual Rail EDR-IB	Cray Aries Dragonfly	Cray Aries Dragonfly



Cori Burst Buffer (out-of-node SSDs)



Summit/Theta (in-node SSDs)

Vision: building scientific collaborative applications