SNIA Global Education

Tutorial

Storage Lessons from HPC: Extreme Scale Computing Driving High Performance Data Storage

Gary Grider

HPC Division Leader

LANL/US DOE

Los Alamos NATIONAL LABORATORY EST.1943

NNSA

About the Instructor

SNIA Global Education

- Gary Grider is the Leader of the High Performance Computing (HPC) Division at Los Alamos National Laboratory. As Division Leader, Gary is responsible for all aspects of High Performance Computing technologies and deployment at Los Alamos. Additionally, Gary is responsible for managing the R&D portfolio for keeping the new technology pipeline full to provide solutions to problems in the Lab's HPC environment, through funding of university and industry partners.
- Gary is also the US Department of Energy Exascale Storage, IO, and Data Management National Co-Coordinator. In this role, Gary helps managed the US government investments in Data Management, Mass Storage, and IO. Gary has 30 active patents/applications in the data storage area and has been working in HPC and HPC related storage since 1984.

SNIA Legal Notice

SNIA Global Education

- The material contained in this tutorial is copyrighted by the SNIA unless otherwise noted.
- Member companies and individual members may use this material in presentations and literature under the following conditions:
  - Any slide or slides used must be reproduced in their entirety without modification
  - The SNIA must be acknowledged as the source of any material used in the body of any document containing material from these presentations.
- This presentation is a project of the SNIA Education Committee.
- Neither the author nor the presenter is an attorney and nothing in this presentation is intended to be, or should be construed as legal advice or an opinion of counsel. If you need legal advice or a legal opinion please contact your attorney.
- The information presented herein represents the author's personal opinion and current understanding of the relevant issues involved. The author, the presenter, and the SNIA do not assume any responsibility or liability for damages arising out of any reliance on or use of this information.

  NO WARRANTIES, EXPRESS OR IMPLIED. USE AT YOUR OWN RISK.

Eight Decades of Production Weapons Computing to Keep the Nation Safe

Maniac

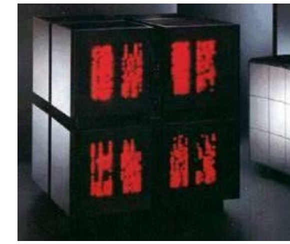
IBM Stretch

CDC

Cray 1

Cray X/Y

CM-2



CM-5

SGI Blue Mountain

DEC/HP Q

IBM Cell Roadrunner

Cray XE Cielo

Cray Intel KNL Trinity

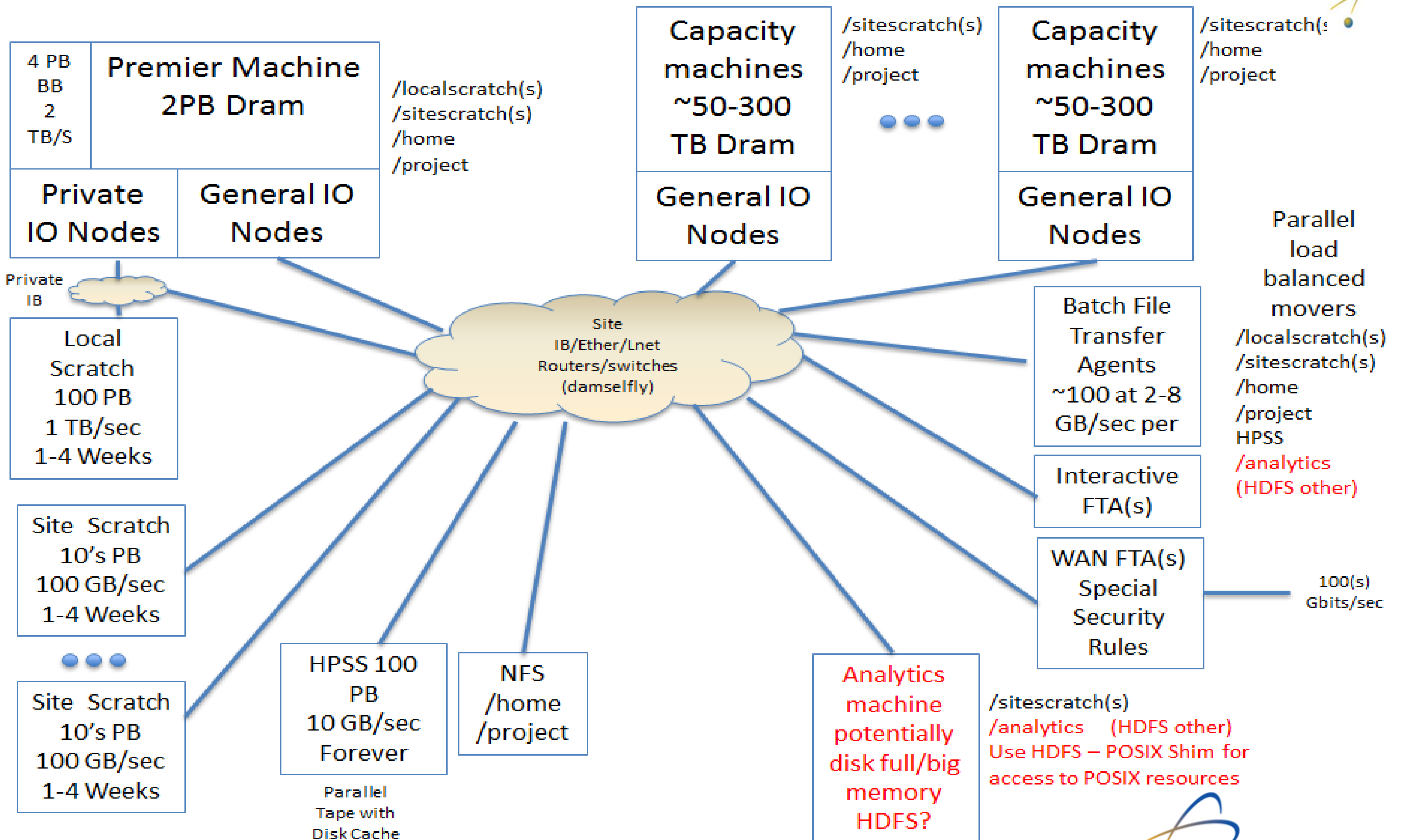
Ising DWave

Cross Roads

CROSSROADS
An APEX Collaboration

Los Alamos
NATIONAL LABORATORY
EST. 1943

Simple View of our Computing Environment



Large Machines and Infrastructure

Trinity

- Haswell and KNL
- 20,000 Nodes
- Few Million Cores
- 2 PByte DRAM
- 4 PByte NAND Burst Buffer ~ 4 Tbyte/sec
- 100 Pbyte Scratch PMR Disk File system ~1.2 Tbyte/sec
- 30PByte/year Sitewide SMR Disk Campaign Store ~ 1 Gbyte/sec/Pbyte (30 Gbyte/sec currently)
- 60 PByte Sitewide Parallel Tape Archive ~ 3 Gbyte/sec

Pipes for Trinity Cooling

- 30-60MW
- Single machines in the 10k nodes and > 18 MW
- Single jobs that run across 1M cores for months
- Soccer fields of gear in 3 buildings
- 20 Semi's of gear this summer alone

HPC Driving Industry

Flash Burst Buffers

DDN STORAGE

INFINITE MEMORY ENGINE™

IBM® GPFS

EMC²

HPSS High Performance Storage System

THE SUPERCOMPUTER COMPANY *Data Warp*

Parallel File Systems

lustre®

panasas®

pNFS

IBM GPFS

ceph

Parallel Archives

HPSS High Performance Storage System

UniTree software

DataTree

CFS

Other

HDF

Tokutek

SNIA. Tutorial

Ceph begins

UNIVERSITY OF CALIFORNIA, SANTA CRUZ

BERKELEY • DAVIS • IRVINE • LOS ANGELES • RIVERSIDE • SAN DIEGO • SAN FRANCISCO SANTA BARBARA • SANTA

DEPARTMENT OF COMPUTER SCIENCE SANTA CRUZ, CALIFORNIA 95064

April 8, 2001

Lawrence Livermore National Laboratory
Attention: Barbara Larson, L-550
P.O. Box 808
Livermore, CA 94551

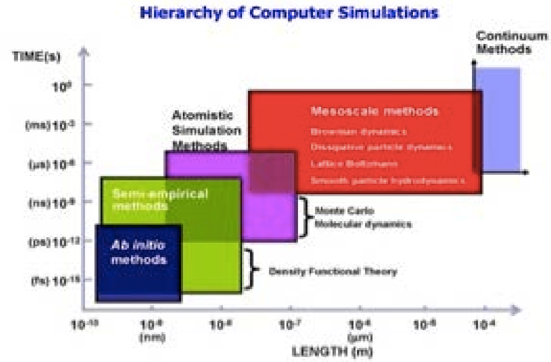
Dear Ms. Larson,

We are pleased to submit this white paper to the ASCI ASAP Request for Expressions of Interest in Level 2 Strategic Investigations. We are responding to the *Scalable and Parallel I/O and File Systems* area of interest.

We propose to improve both file system performance and functionality by building a storage system from object-based storage devices (OBSDs) connected by high-speed networks. The key advantage of OBSDs in a high-performance environment is the ability to delegate low-level block allocation and synchronization for a given segment of data to the device on which it is stored, leaving the file system to decide only on which OBSD a given segment should be placed. Since this decision is quite simple and allows massive parallelism, each OBSD need only manage concurrency locally, allowing a file system built from thousands of OBSDs to achieve massively parallel data transfers. Additionally, OBSDs can each manage their own storage consistency, removing the need to run a system-wide consistency check that could take days on a petabyte-scale traditional file system.

HPC Simulation Background

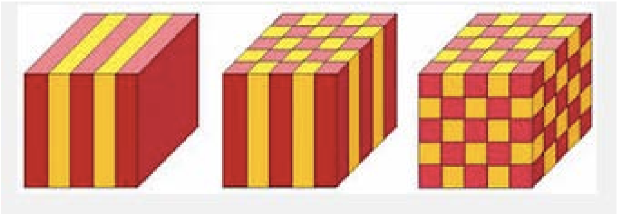
Link scales

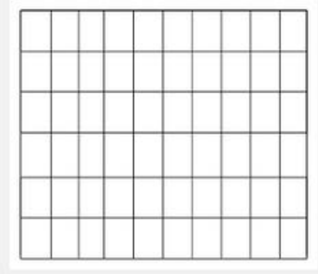


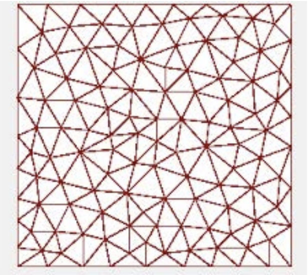
http://eng-cs.syr.edu/research/mathematical-and-numerical-analysis

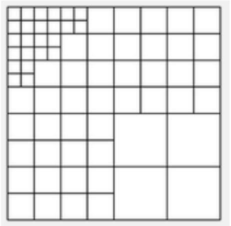
Meshes

1D 2D 3D Structured Unstructured Resolutions

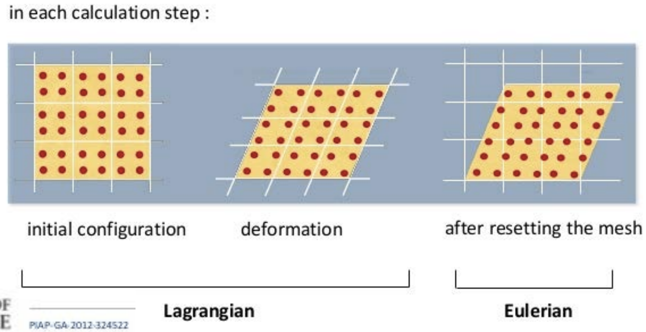


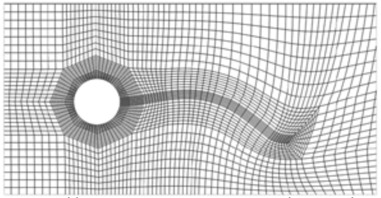






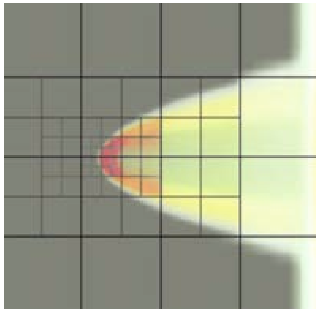
Methods Lagrangian Eulerian ALE AMR





http://media.archnumsoft.org/10305/

ALE



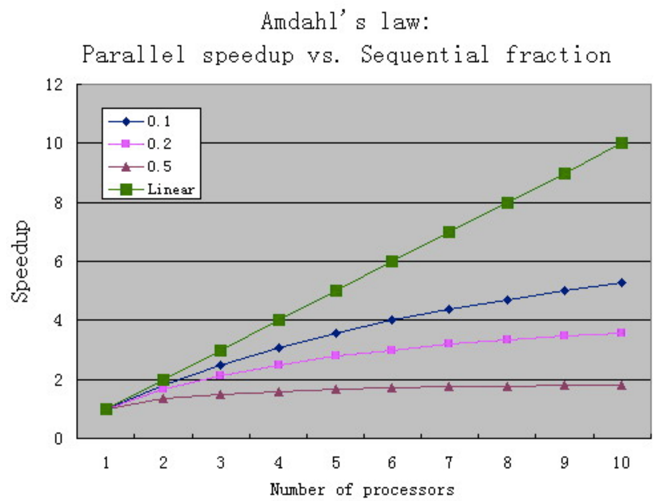
http://web.cs.ucdavis.edu/~ma/VolVis/amr_mesh.jpg

Eulerian AMR

Scaling and Programming Models

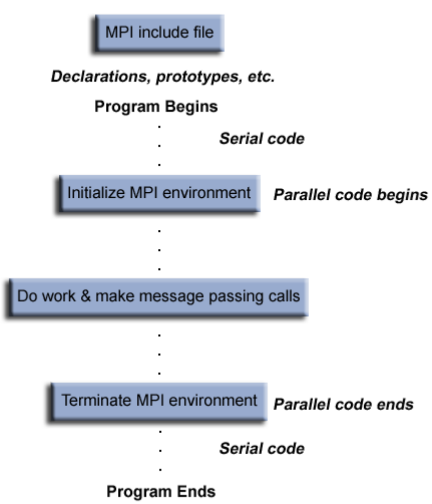
- The first is *strong scaling*, which is defined as how the solution time varies with the number of processors for a fixed *total* problem size.
- The second is *weak scaling*, which is defined as how the solution time varies with the number of processors for a fixed problem size *per processor*.

http://media.archnumsoft.org/10305/

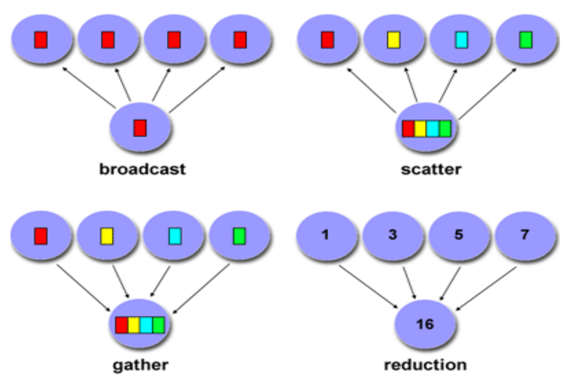


Scaling

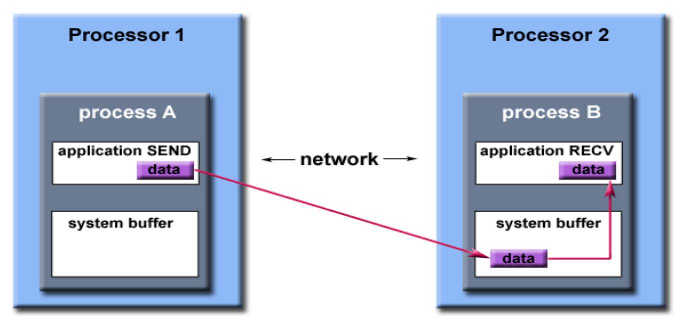
Process Parallel Dominates the Past



Flow



Collectives

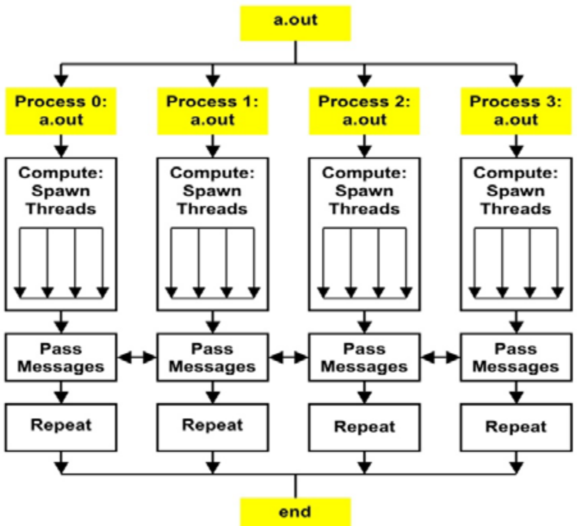


Path of a message buffered at the receiving process

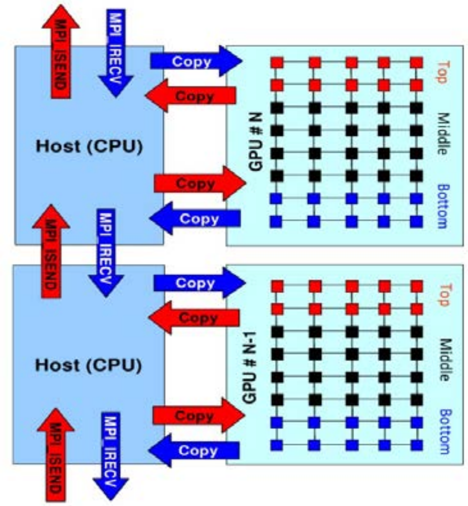
Point to Point

Current Programming Model Directions Combining Process Parallel, Data Parallel, Async Tasking, and Threading

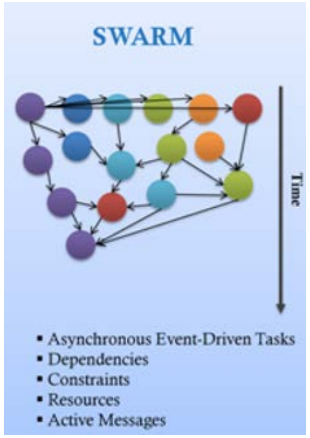
MPI+ Threads

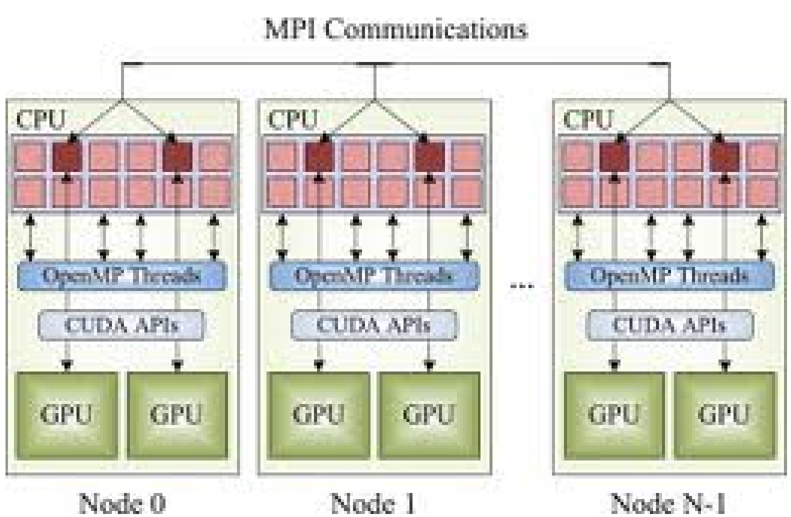


MPI+Data Parallel



Async Tasking



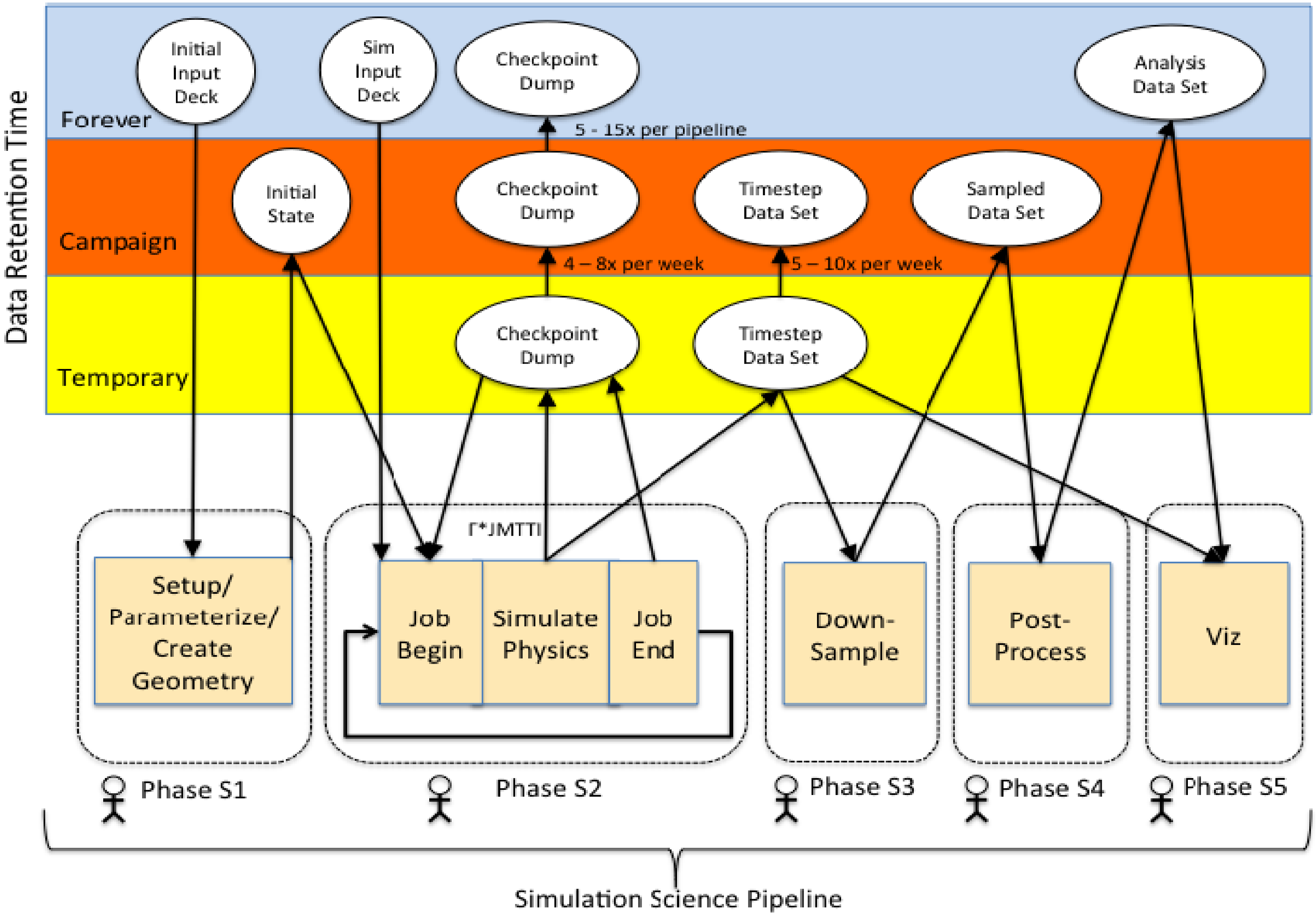


MPI+ Threads+ Data Parallel

Domain Specific Languages

Others: OpenShmem, CAF, Chapel, X10 are possible as well

Workflow Taxonomy from APEX Procurement
A Simulation Pipeline



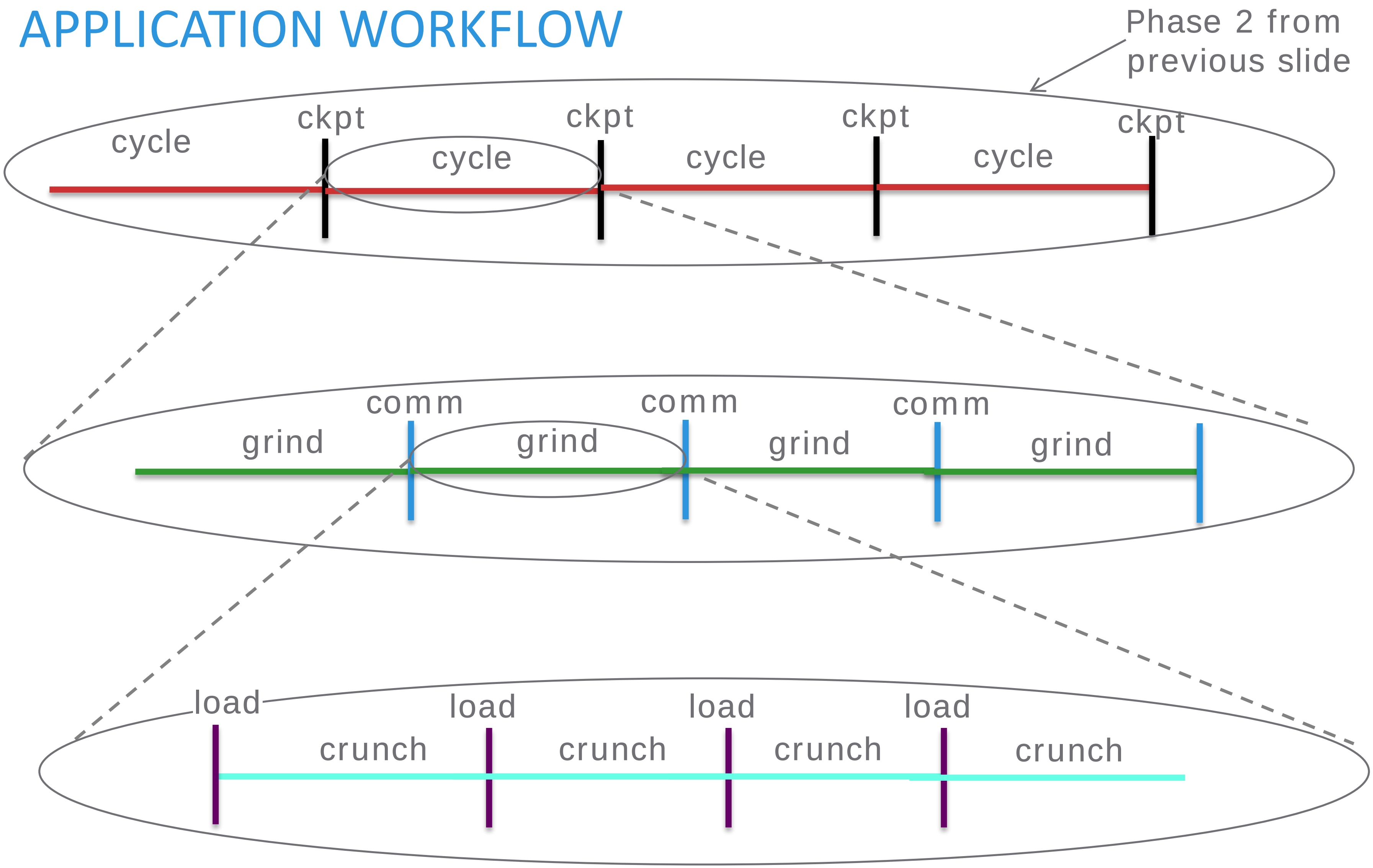
APEX Workflows

LANL, NERSC, SNL

SAND2015-10342 O
LA-UR-15-29113

Figure 1: An example of an APEX simulation science workflow.

APPLICATION WORKFLOW



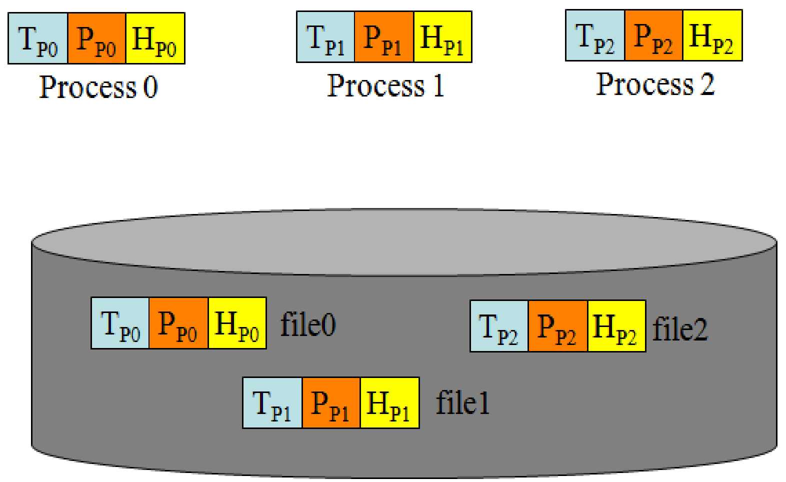
Key observation: a grind (for strong-scaling apps) traverses all of memory.

Not Just Computation

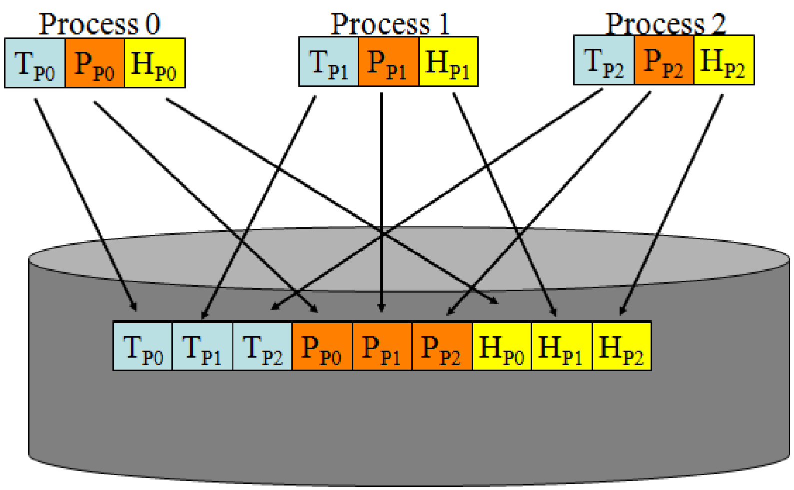
failure

HPC IO Patterns

N-to-N



N-to-1 Strided

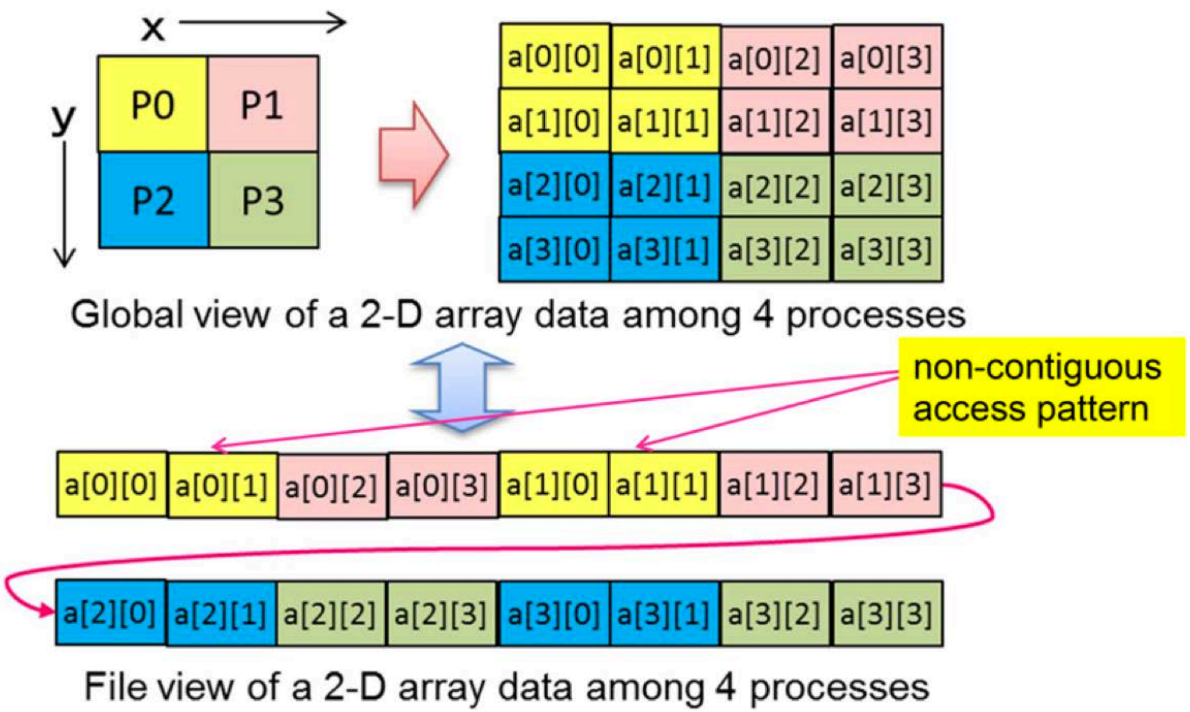


- Million files inserted into a single directory at the same time
- Millions of writers into the same file at the same time
- Jobs from 1 core to N-Million cores
- Files from 0 bytes to N-Pbytes
- Workflows from hours to a year (yes a year on a million cores using a PB DRAM)

A Simple collective 2D layout as an example

Collective I/O for 2-Dimensional Data

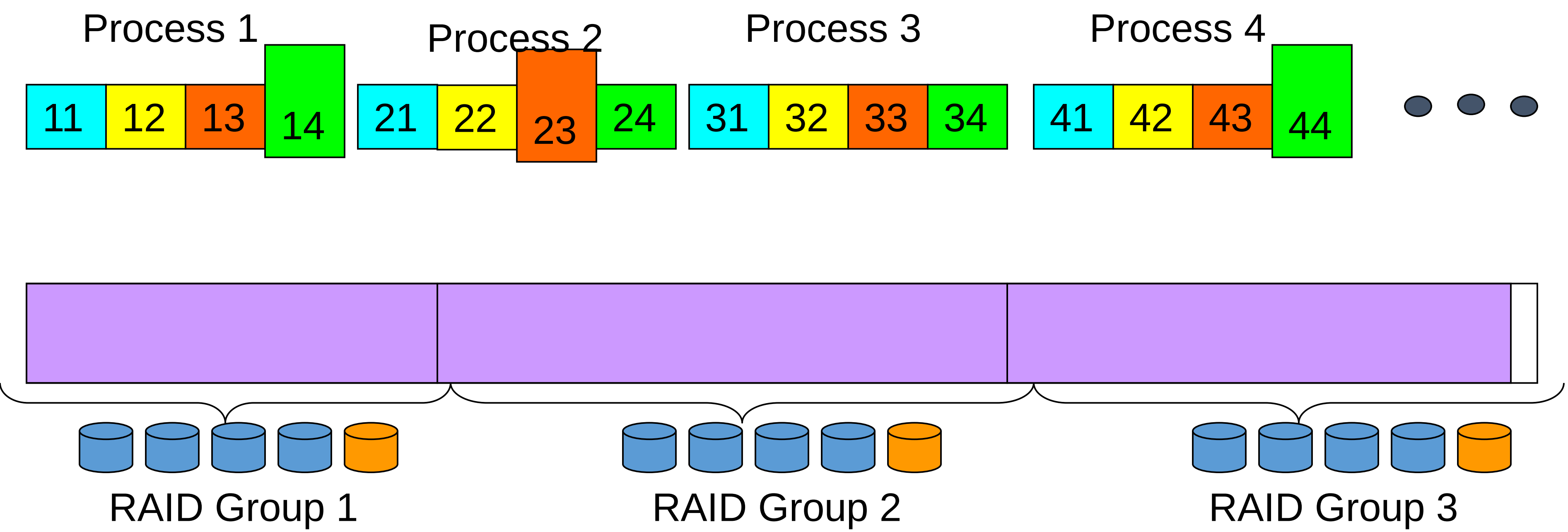
- 2-Dimensional data accesses by 4 processes

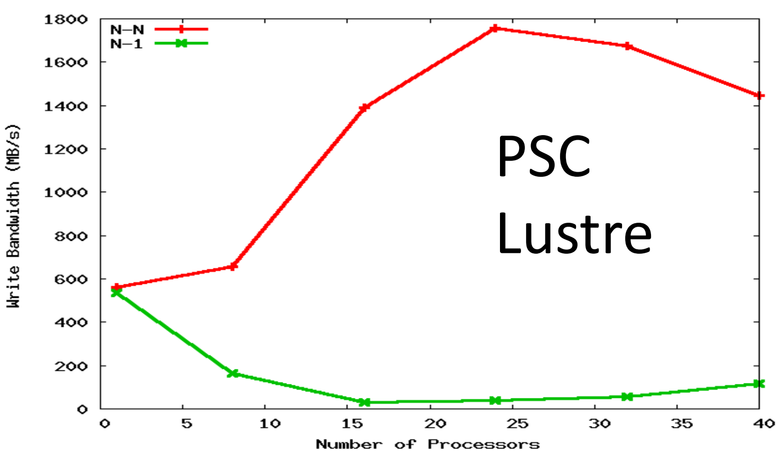


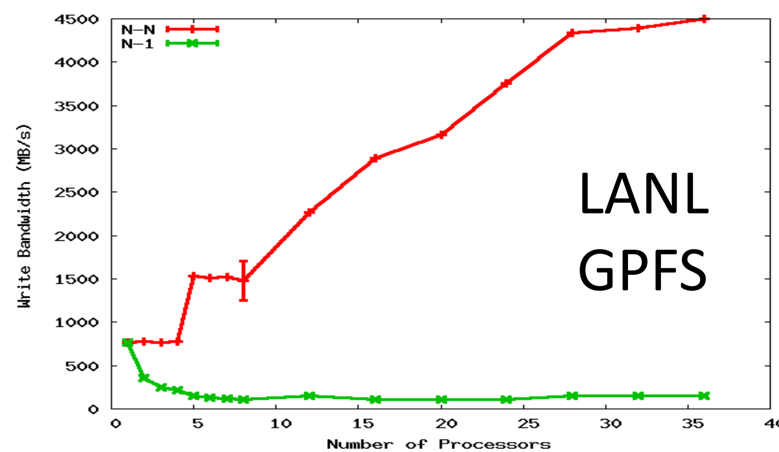
Apps can map their 1,2,3,N dimensional data onto files in any way they think will help them

This leads to formatting middleware like HDF5, NetCDF, MPI-IO

Why N->1 strided can be problematic







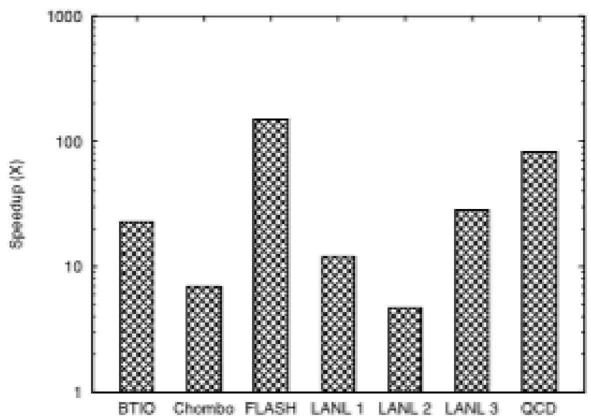
PLFS: A Checkpoint Filesystem for Parallel Applications

SC 2009

John Bent[*†], Garth Gibson[‡], Gary Grider[*], Ben McClelland[*],
Paul Nowoczynski[§], James Nunez[*], Milo Polte[†], Meghan Wingate[*]

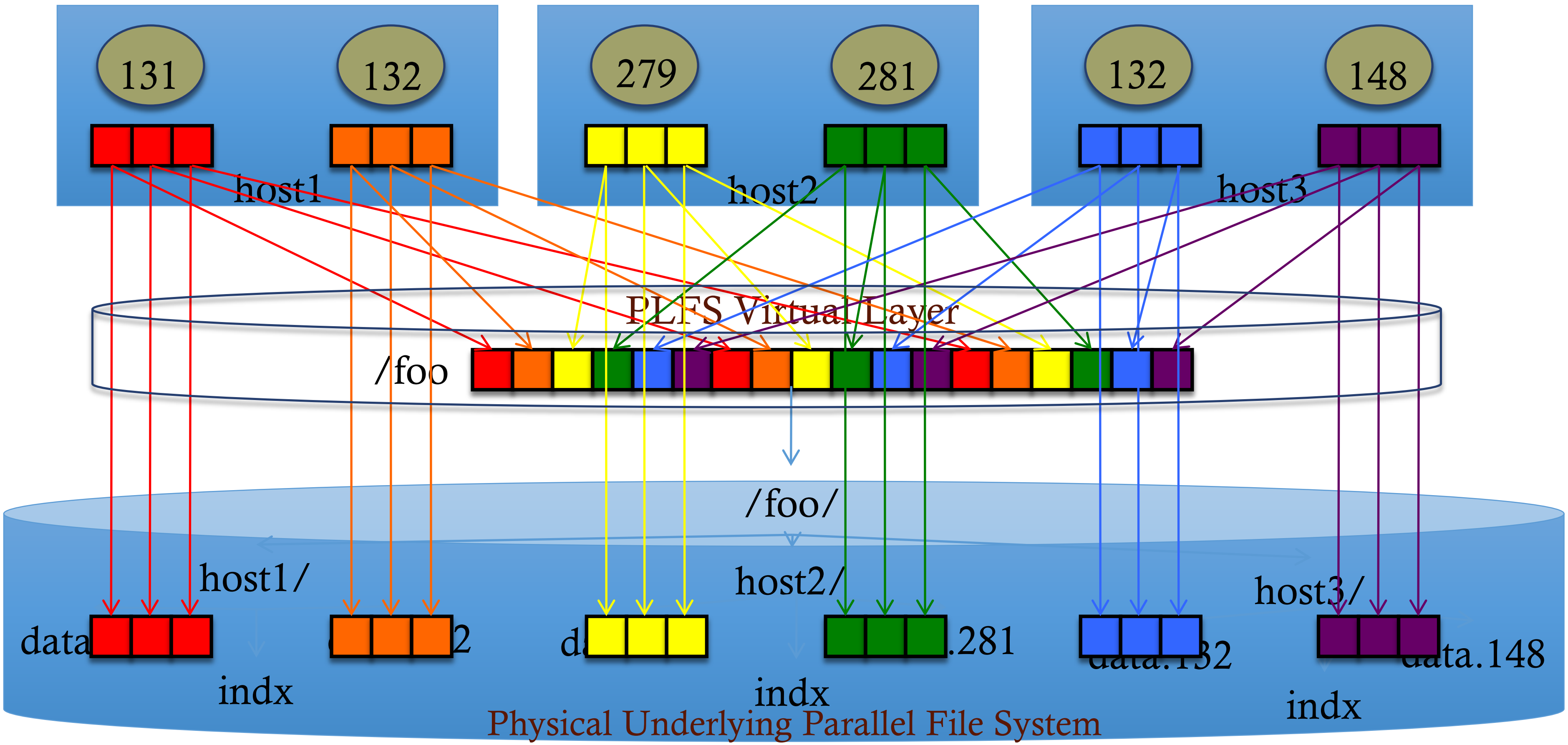
ABSTRACT

Parallel applications running across thousands of processors must protect themselves from inevitable system failures. Many applications insulate themselves from failures by checkpointing. For many applications, checkpointing into a shared single file is most convenient. With such an approach, the size of writes are often small and not aligned with file system boundaries. Unfortunately for these applications, this preferred data layout results in pathologically poor performance from the underlying file system which is optimized for large, aligned writes to non-shared files. To address this fundamental mismatch, we have developed a virtual parallel log structured file system, PLFS. PLFS remaps an application's preferred data layout into one which is optimized for the underlying file system. Through testing on PanFS, Lustre, and GPFS, we have seen that this layer of indirection and reorganization can reduce checkpoint time by an order of magnitude for several important benchmarks and real applications without any application modification.

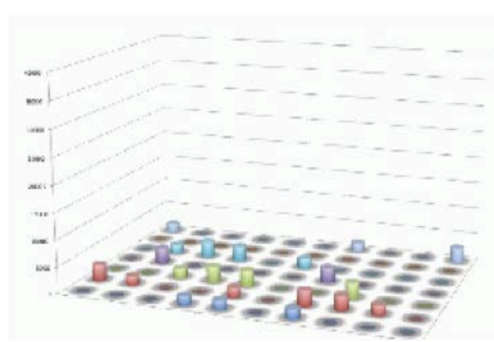


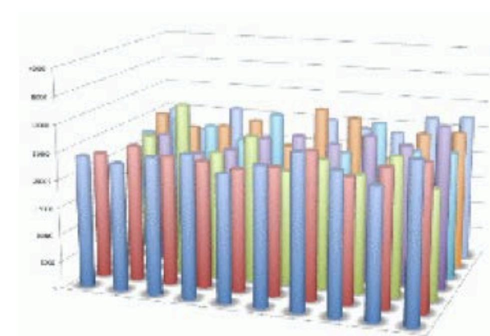
1: **Summary of our results.** *This graph summarizes our results which will be explained in detail in Section 4. The key observation here is that our technique has improved checkpoint bandwidths for all seven studied benchmarks and applications by up to several orders of magnitude.*

Decouples Logical from Physical

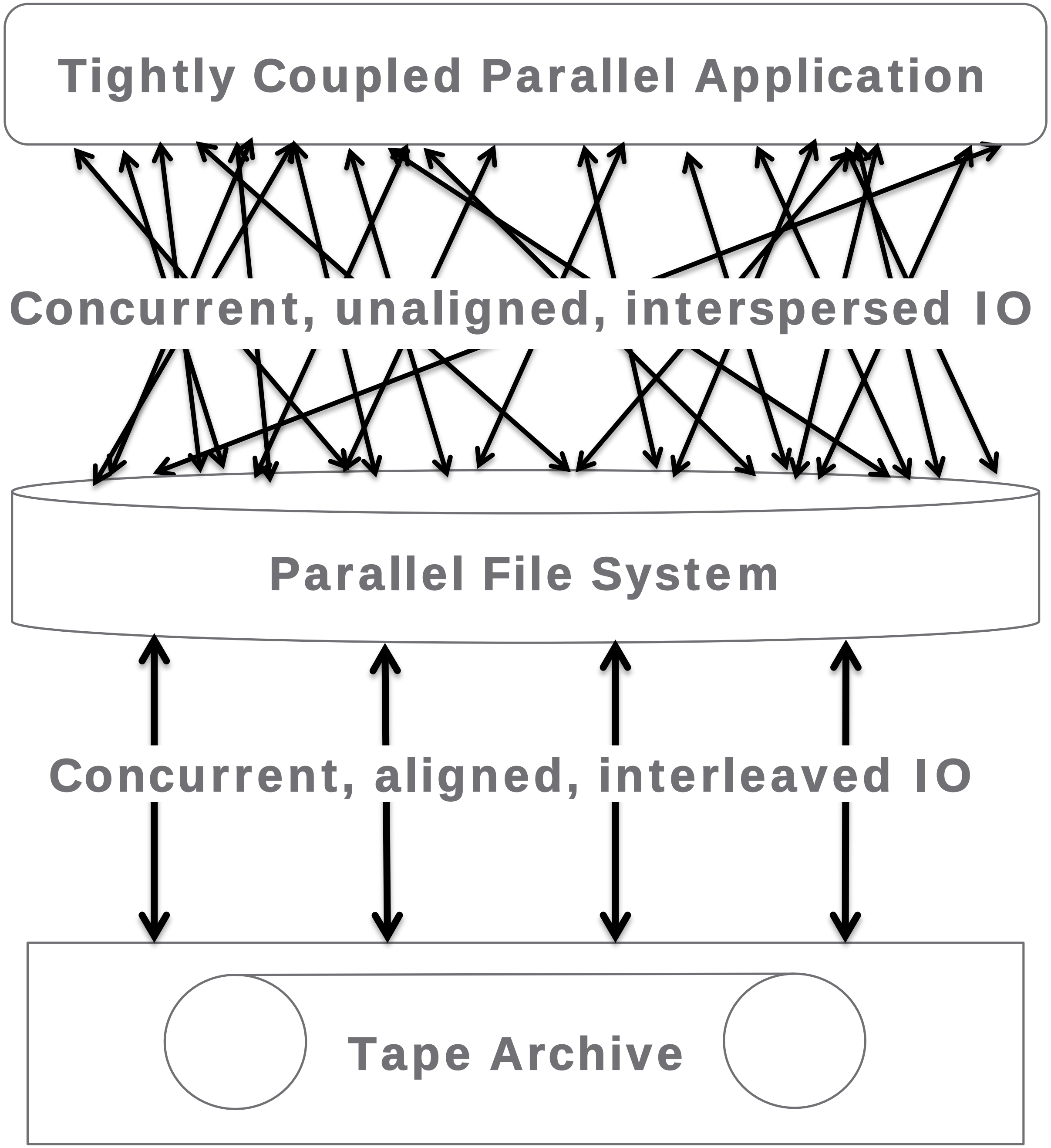


- N to 1 is fixed by PLFS
- But won’t scale to exascale (N-N unscalable (billions of files)

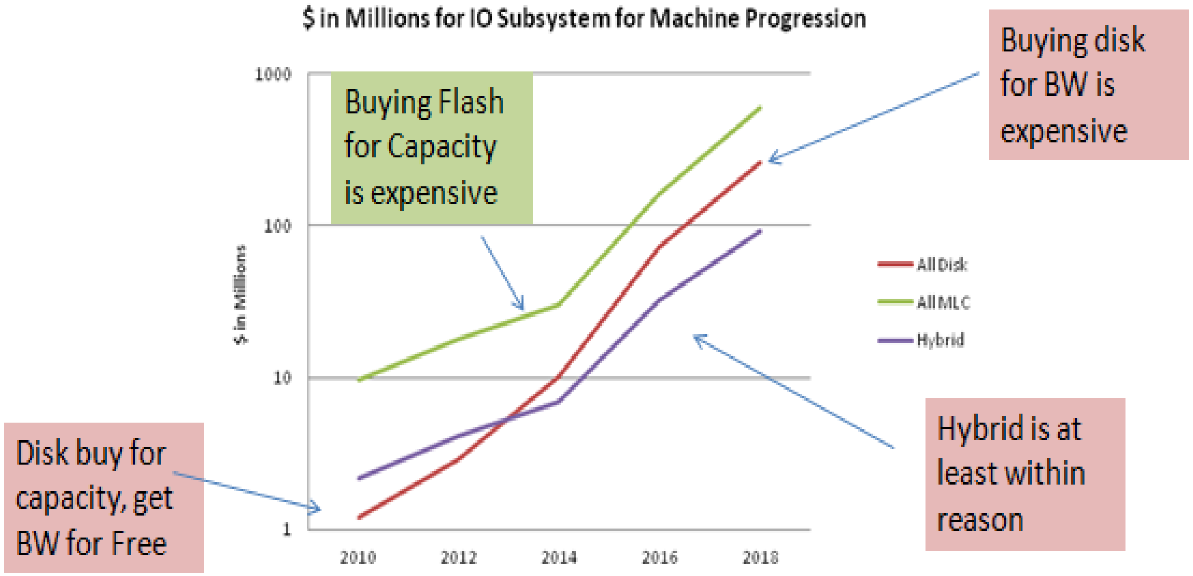




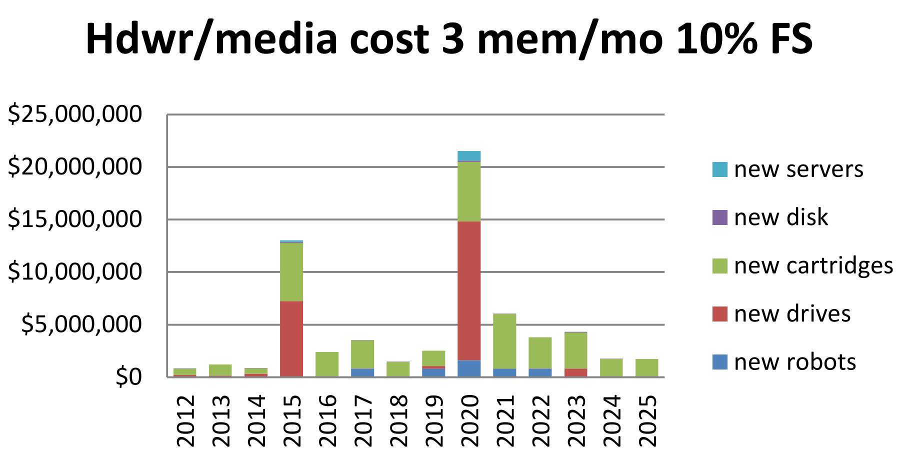
HPC Storage Stack, 2002-2015



Economics have shaped our world
The beginning of storage layer proliferation 2009

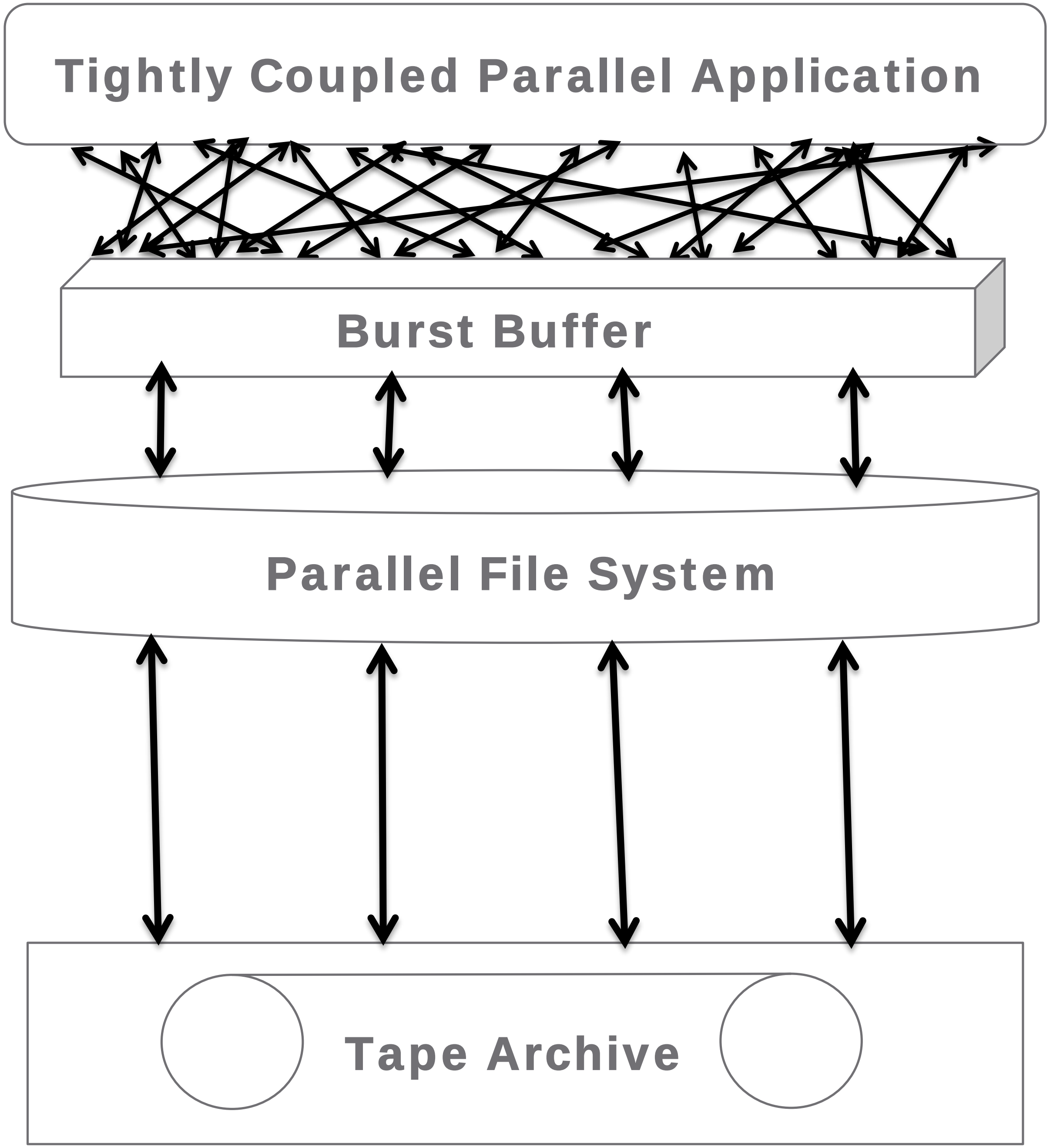


- Economic modeling for large burst of data from memory shows bandwidth / capacity better matched for solid state storage near the compute nodes



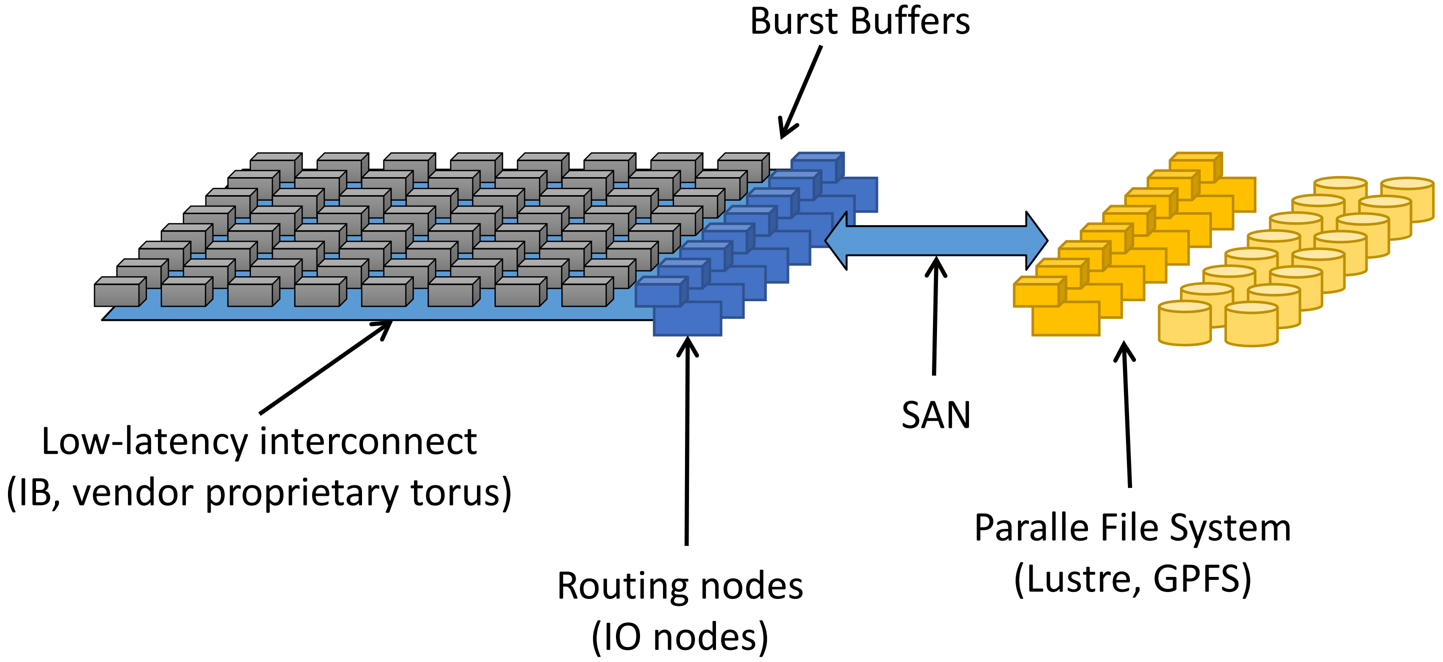
- Economic modeling for archive shows bandwidth / capacity better matched for disk

HPC Storage Stack, 2015-2016

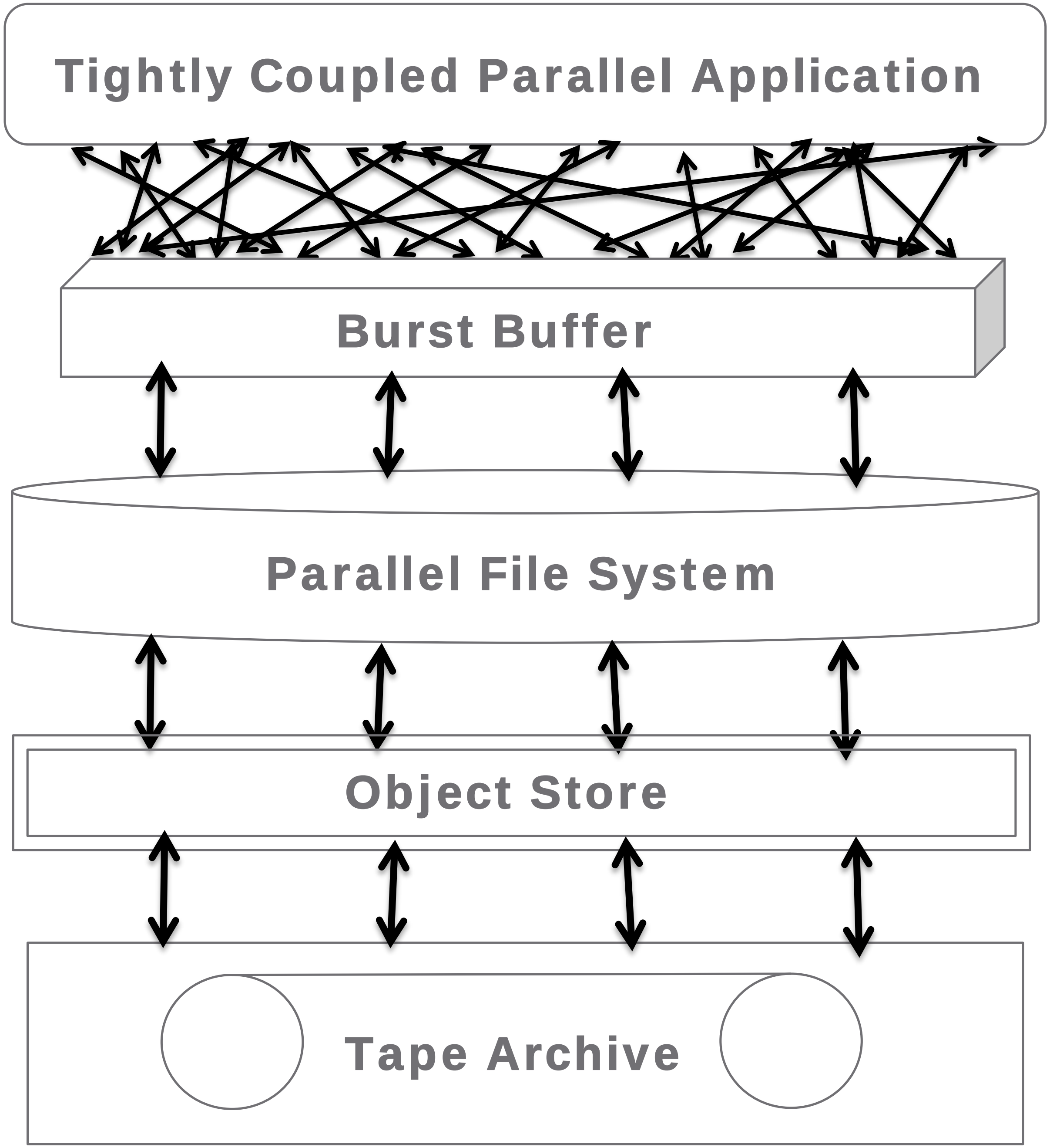


HPC environment

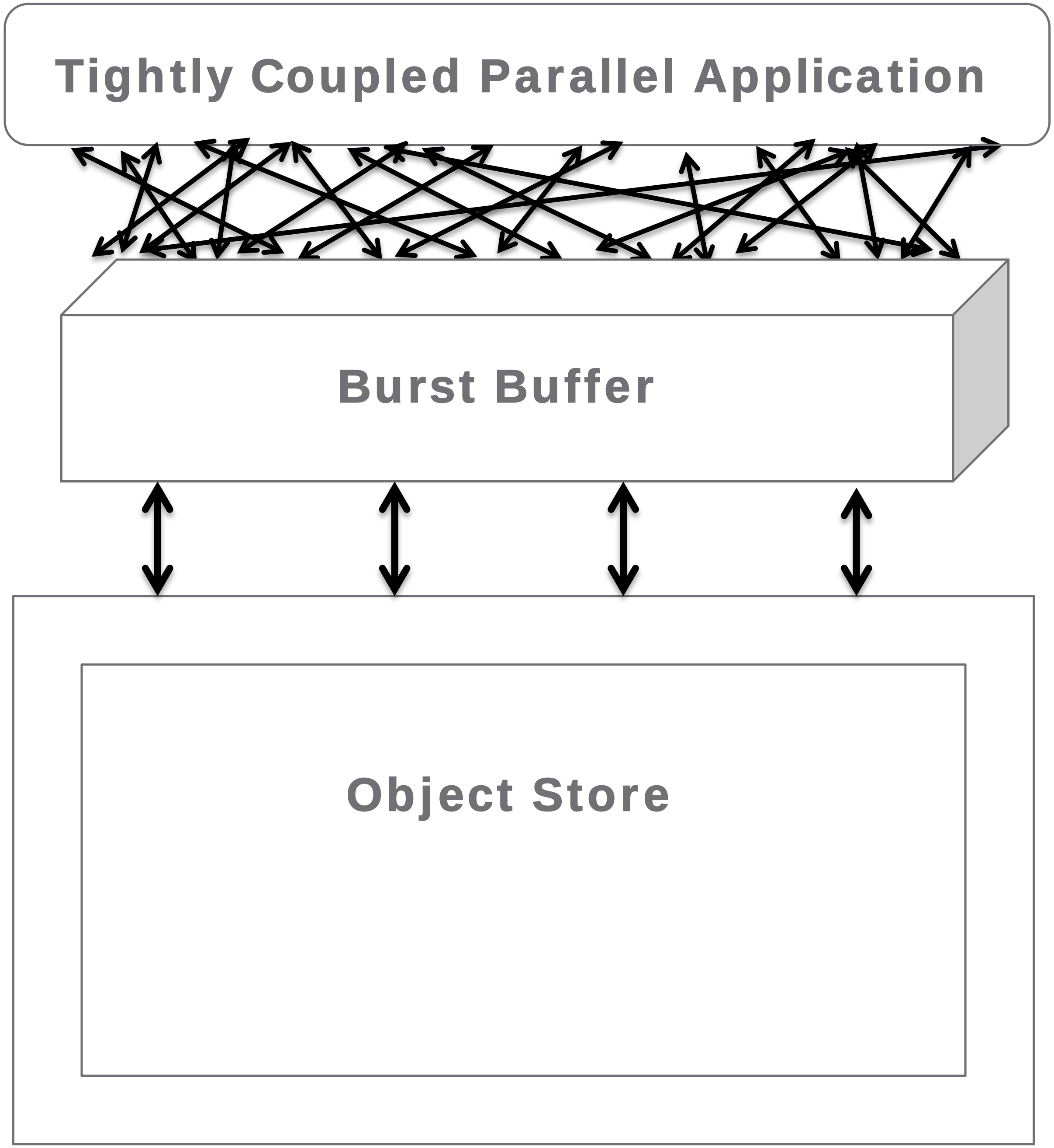
- Big difference from cloud: parallel, tightly coupled, extremely simple nodes to lower jitter and job failure due to tightly coupled behavior (one code syncs between all neighbors every 1 millisecond



HPC Storage Stack, 2016-2020



HPC Storage Stack, 2020-



What about the Capacity Tier: Won't cloud technology provide the capacity solution?

- Erasure to utilize low cost hardware
- Object to enable massive scale
- Simple minded interface, get put delete

- Problem solved --→ NOT

- Works great for apps that are newly written to use this interface
- Doesn't work well for people, people need folders and rename and …
- Doesn't work for the $trillions of apps out there that expect some modest name space capability (parts of POSIX)

How about a Scalable Near-POSIX Name Space over Cloud style Object Erasure: MarFS

- Best of both worlds
  - Objects Systems
    - Provide massive scaling and efficient erasure techniques
    - Friendly to applications, not to people. People need a name space.
    - Huge Economic appeal (erasure enables use of inexpensive storage)
  - POSIX name space is powerful but has issues scaling
- The challenges
  - Mismatch of POSIX an Object metadata, security, read/write semantics, efficient object/file sizes.
  - No update in place with Objects
  - How do we scale POSIX name space to trillions of files/directories

MarFS

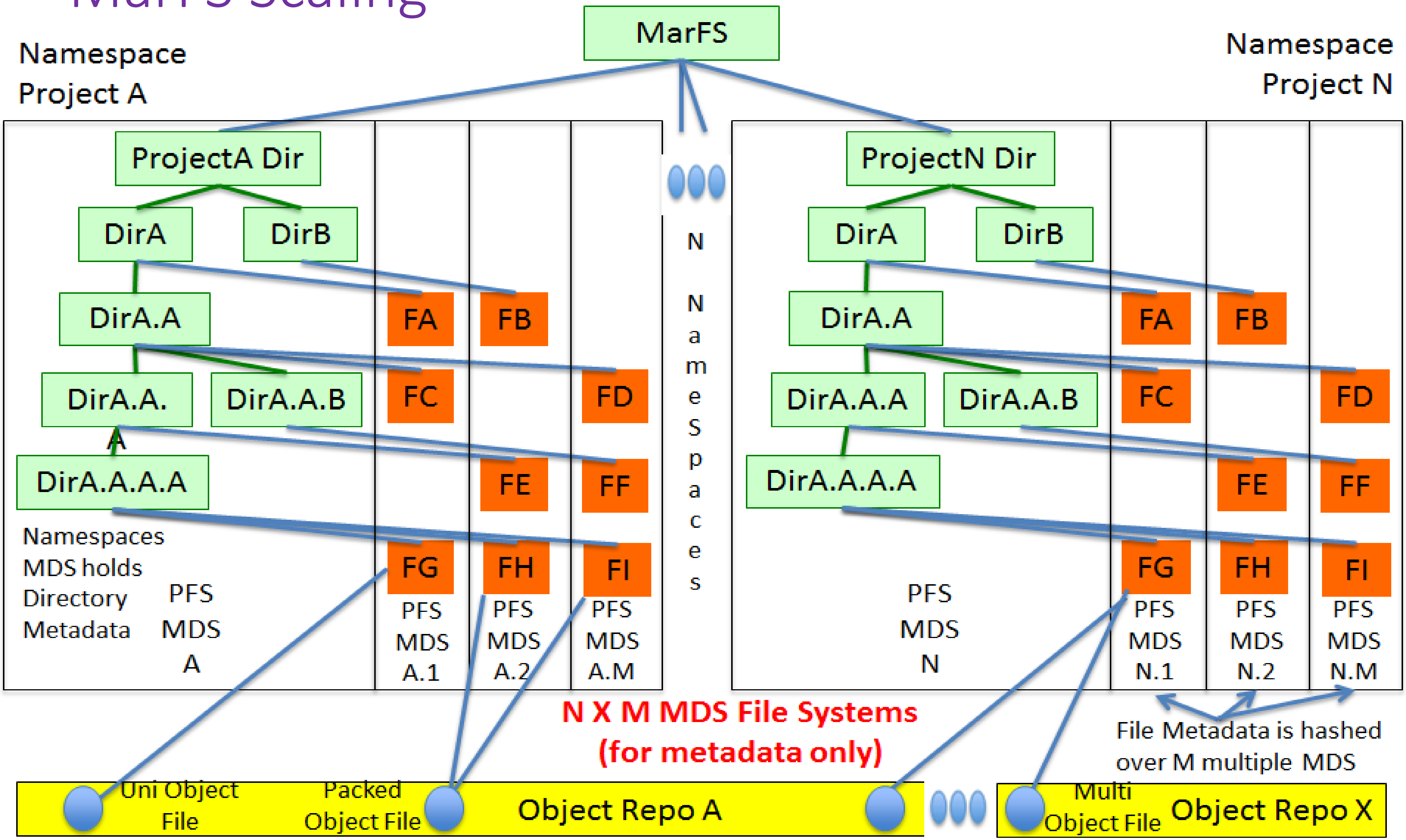
What it is

- 100-1000 GB/sec, Exabytes, Billion files in a directory, Trillions of files total
- Near-POSIX global scalable name space over many POSIX and non POSIX data repositories (Scalable object systems - CDMI, S3, etc.)
  - (Scality, EMC ECS, all the way to simple erasure over ZFS's)
- It is small amount of code (C/C++/Scripts)
  - A small Linux Fuse
  - A pretty small parallel batch copy/sync/compare/ utility
  - A moderate sized library both FUSE and the batch utilities call
- Data movement scales just like many scalable object systems
- Metadata scales like NxM POSIX name spaces both across the tree and within a single directory
- It is friendly to object systems by
  - Spreading very large files across many objects
  - Packing many small files into one large data object

What it isnt

- No Update in place! Its not a pure file system, Overwrites are fine but no seeking and writing.

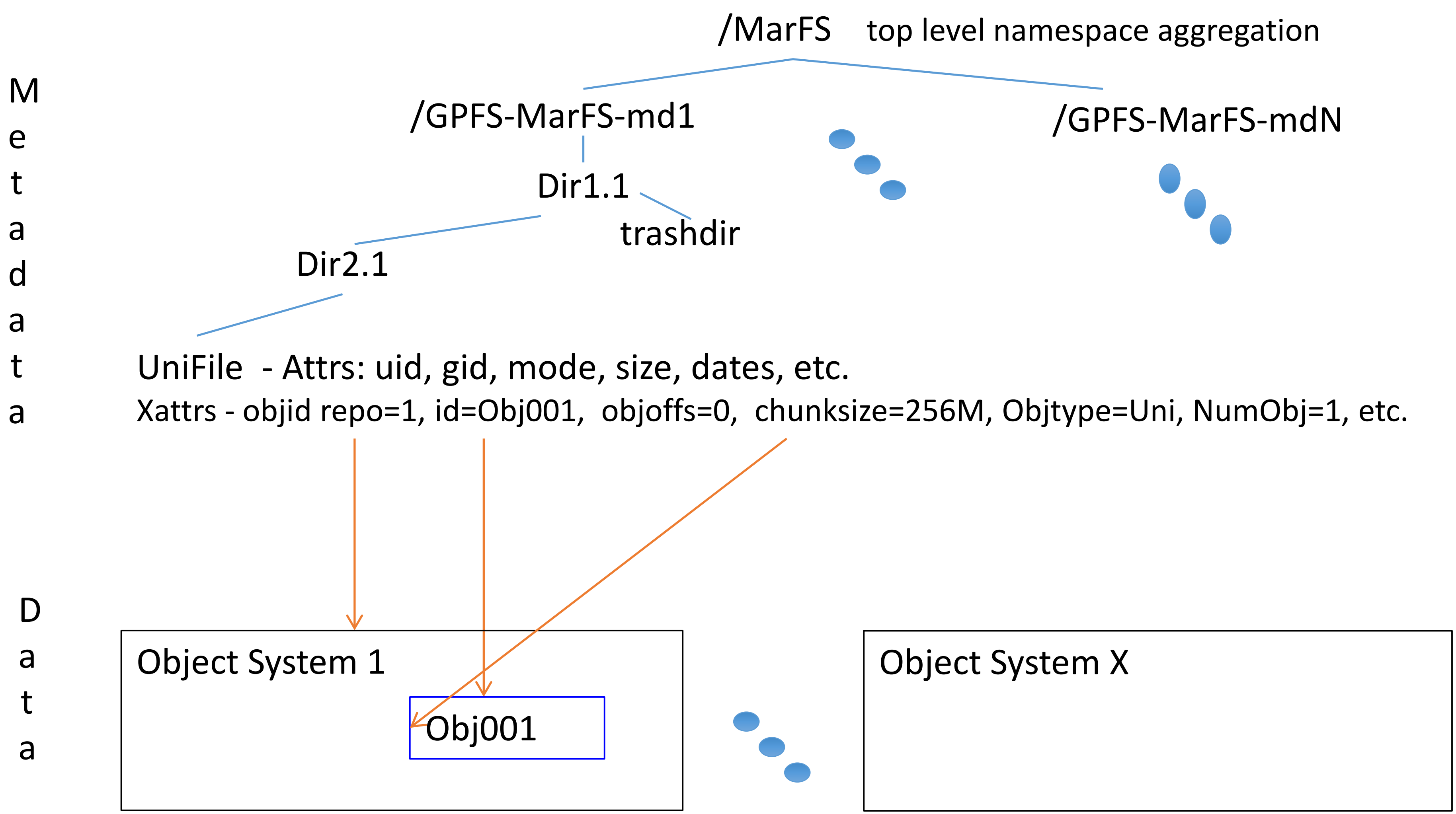
MarFS Scaling



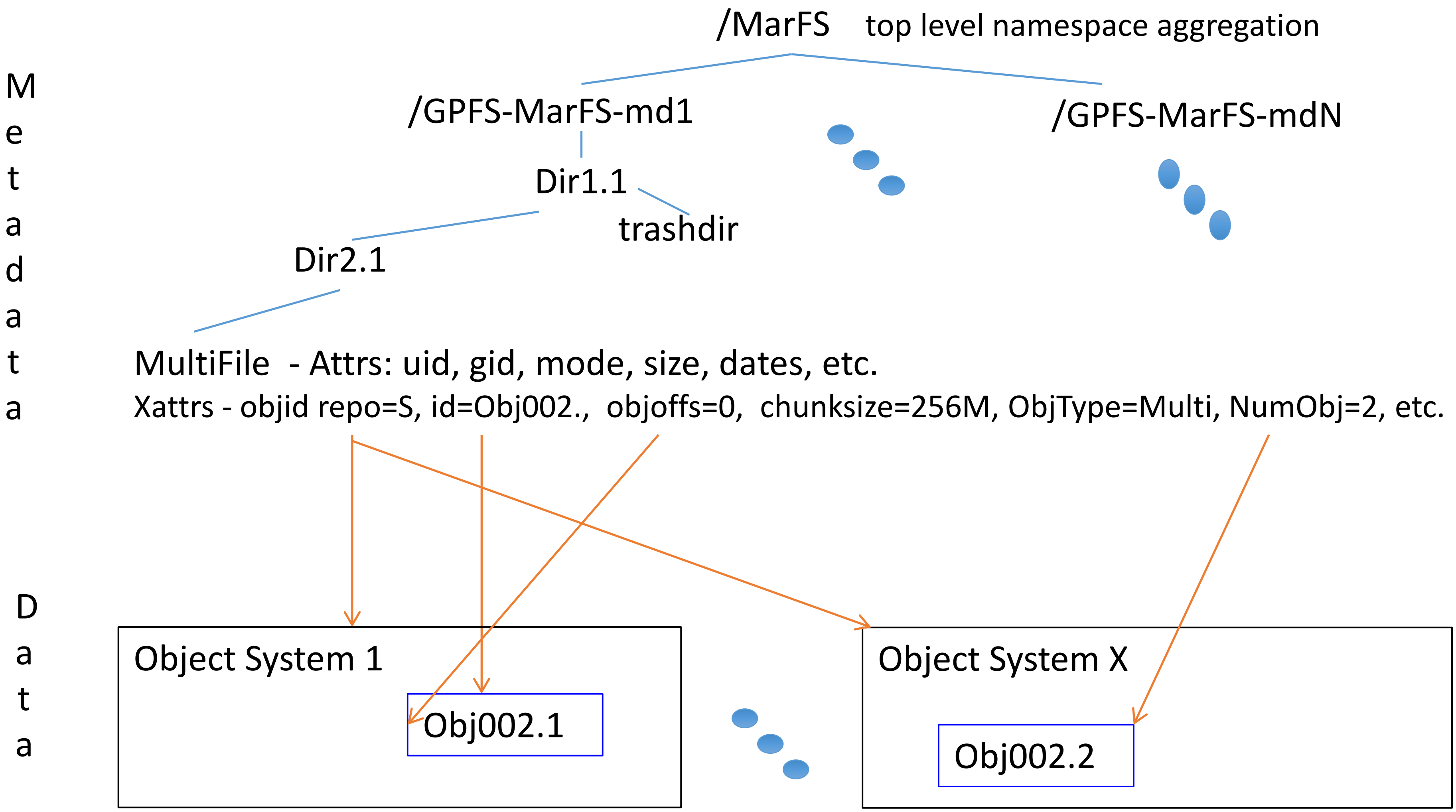
Scaling test on our retired Cielo machine:
835M File Inserts/sec Stat single file < 1 millisecond
> 1 trillion files in the same director

Striping across 1 to X Object Repos

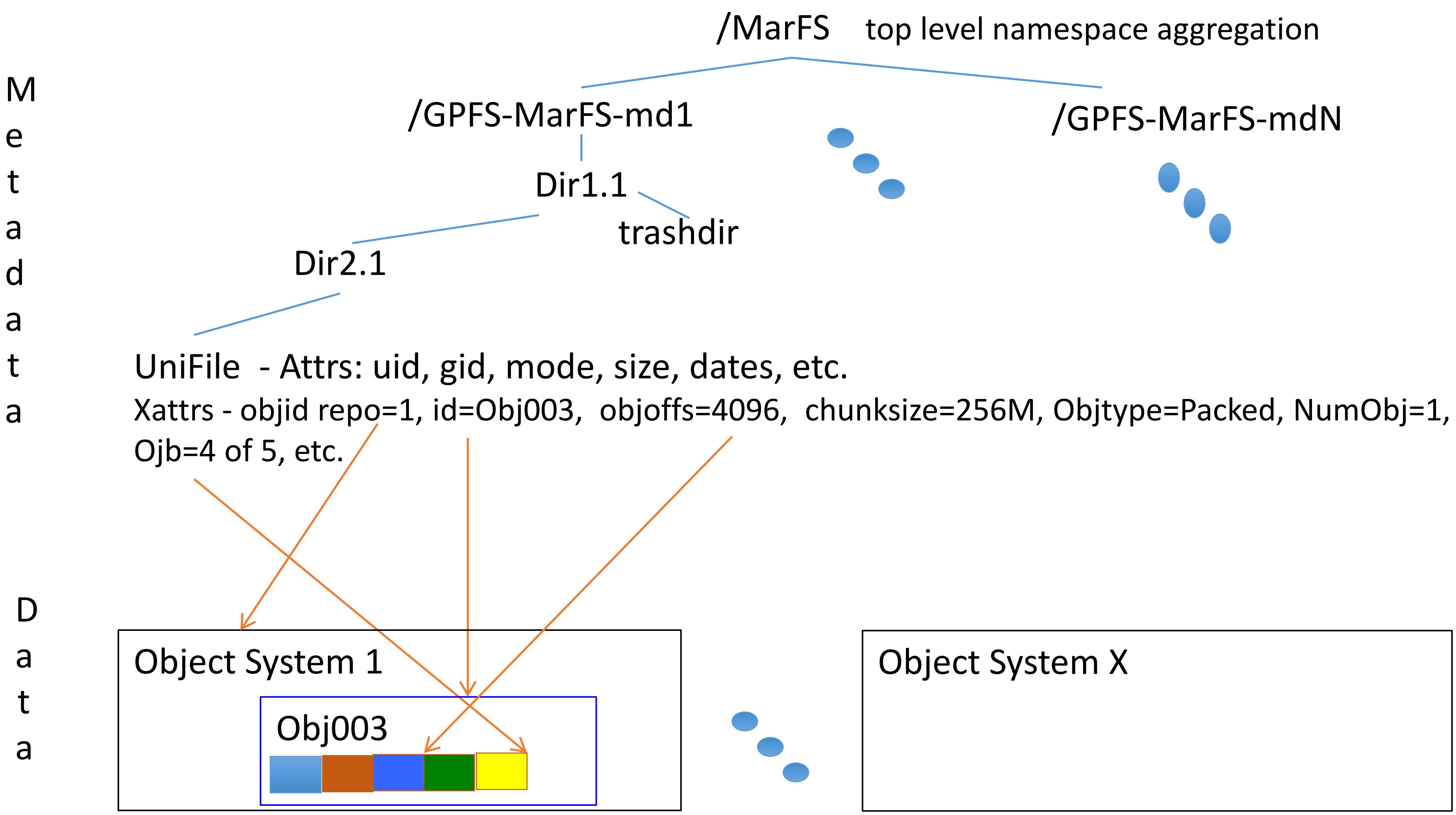
MarFS Internals Overview Uni-File



MarFS Internals Overview Multi-File (striped Object Systems)

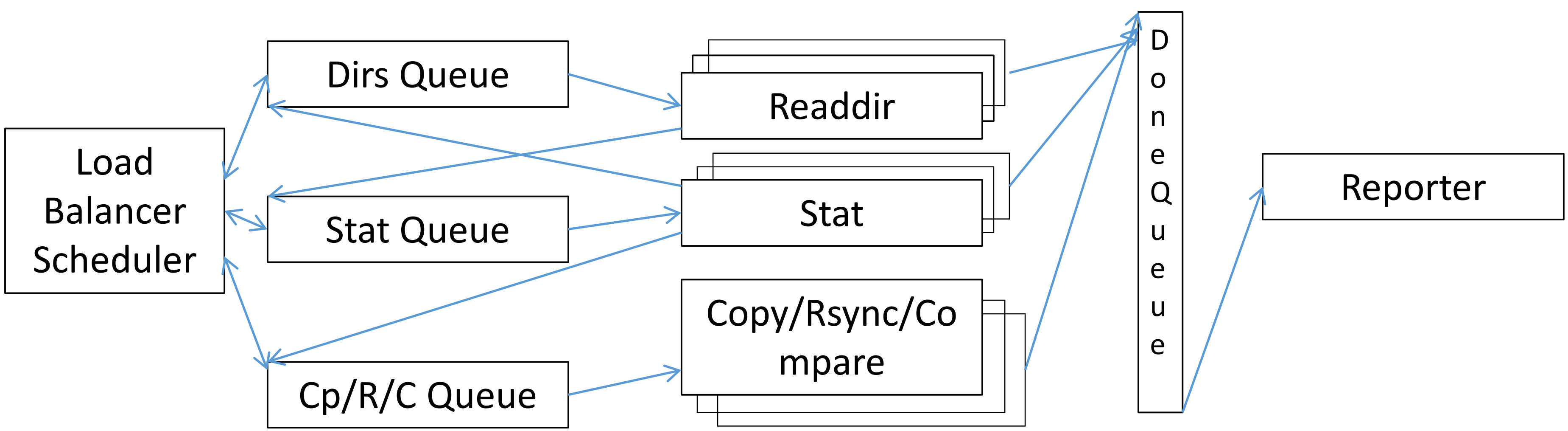


MarFS Internals Overview Packed-File

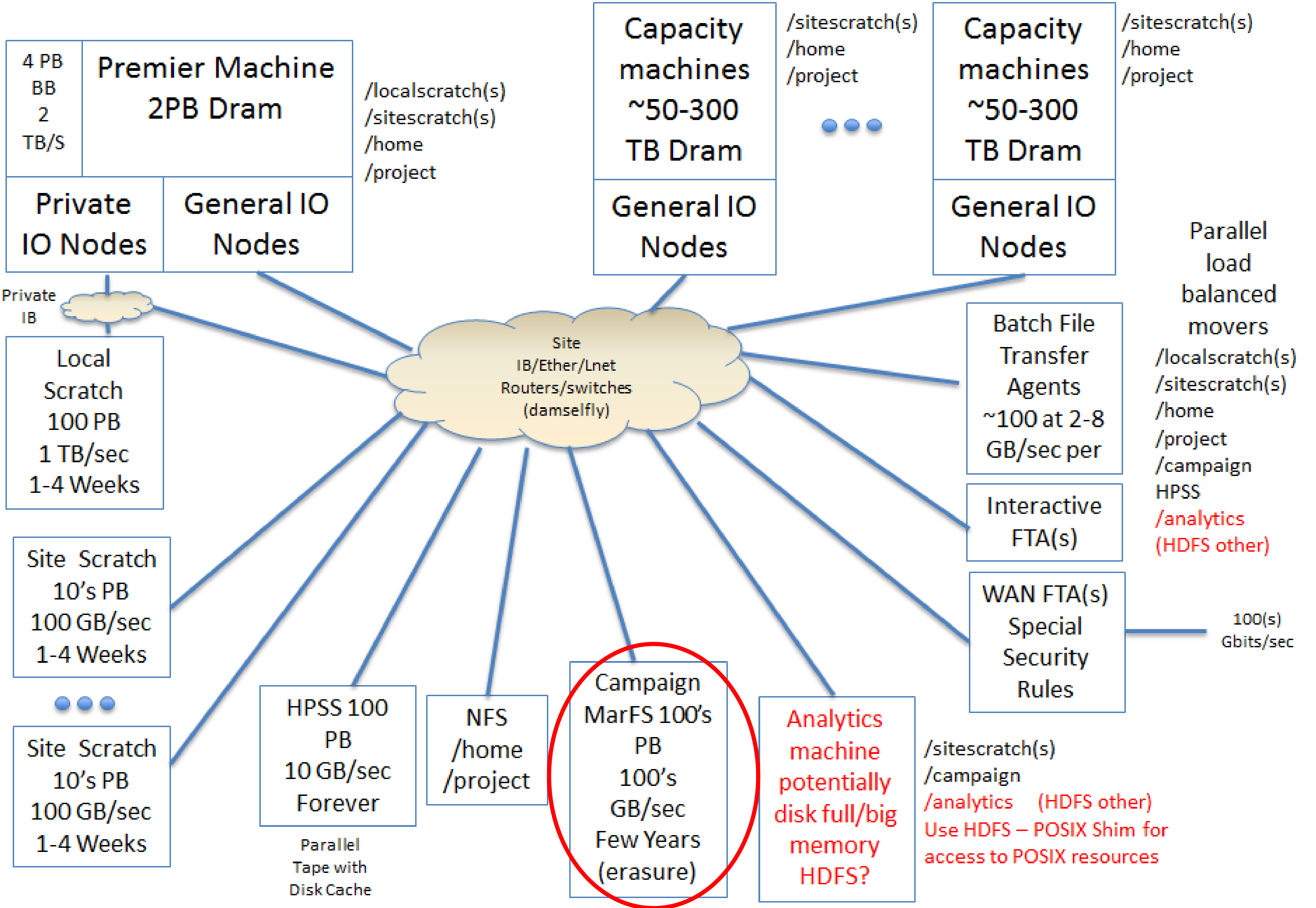


Pftool – parallel copy/rsync/compare/list tool

- Walks tree in parallel, copy/rsync/compare in parallel.
    - Parallel Readdir's, stat's, and copy/rsinc/compare
  - Dynamic load balancing
  - Restart-ability for large trees or even very large files
  - Repackage: breaks up big files, coalesces small files
  - To/From NFS/POSIX/parallel FS/MarFS



How does it fit into our environment in FY16



Not Just LANL Developing This Tier

SPECTRA Products · Features

Spectra Logic Announces Lustre Archive Campaign Storage Solution

Posted on Tuesday, November 15th, 2016 at 6:00 am.
Written by Spectra Logic

f 2 · g+ 1 · in 22

Spectra's Lustre archive solution, powered by Campaign Storage HSM, offers new architecture for scalable high-performance archive based on industry standard workflows and technologies

Salt Lake City, UT, SC-16, #1401 – November 15, 2016 – Spectra Logic, the deep storage experts, today announced a new archive solution for Lustre file systems widely deployed in high-performance computing (HPC), education and government supercomputers, and data centers. The solution, which includes embedded hierarchical storage management (HSM) software by Campaign Storage LLC, works with Spectra's BlackPearl® Deep Storage Gateway, Intel's Lustre file system, and an industry-standard policy manager. It offers seamless Lustre Archive functionality with a high-performance archive search and HSM management tool allowing nearly unlimited scalability, performance, monitoring and flexibility.

Spectra Logic Campaign Storage LLC

The Register® Biting the hand that feeds IT

DATA CENTER · SOFTWARE · SECURITY · TRANSFORMATION · DEVOPS · BUSINESS · PERSONAL TECH · SCIE

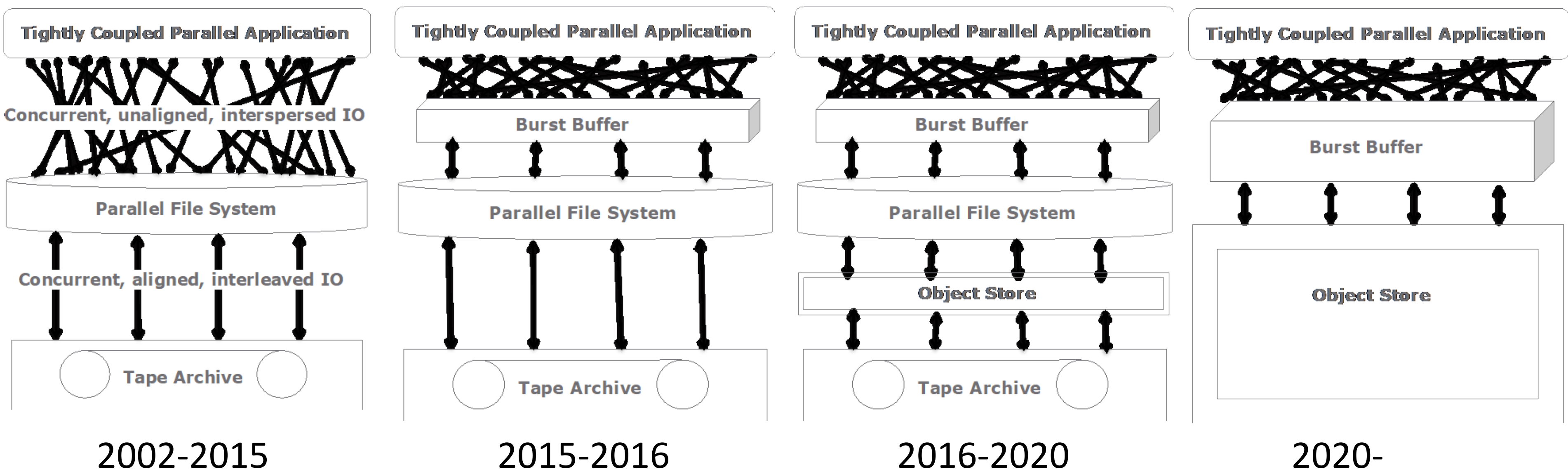
Data Center ▸ Storage

Seagate serves up three layer ClusterStor sandwich

Online disk archive for HPC and Big Data

Seagate ClusterStor A200

From 2 to 4 and back again



;login:
THE USENIX MAGAZINE

Serving Data to the Lunatic Fringe: The Evolution of HPC Storage

;login: issue: Summer 2016, Vol. 41, No. 2

Authors:
John Bent, Brad Settlemyer, and Gary Grider

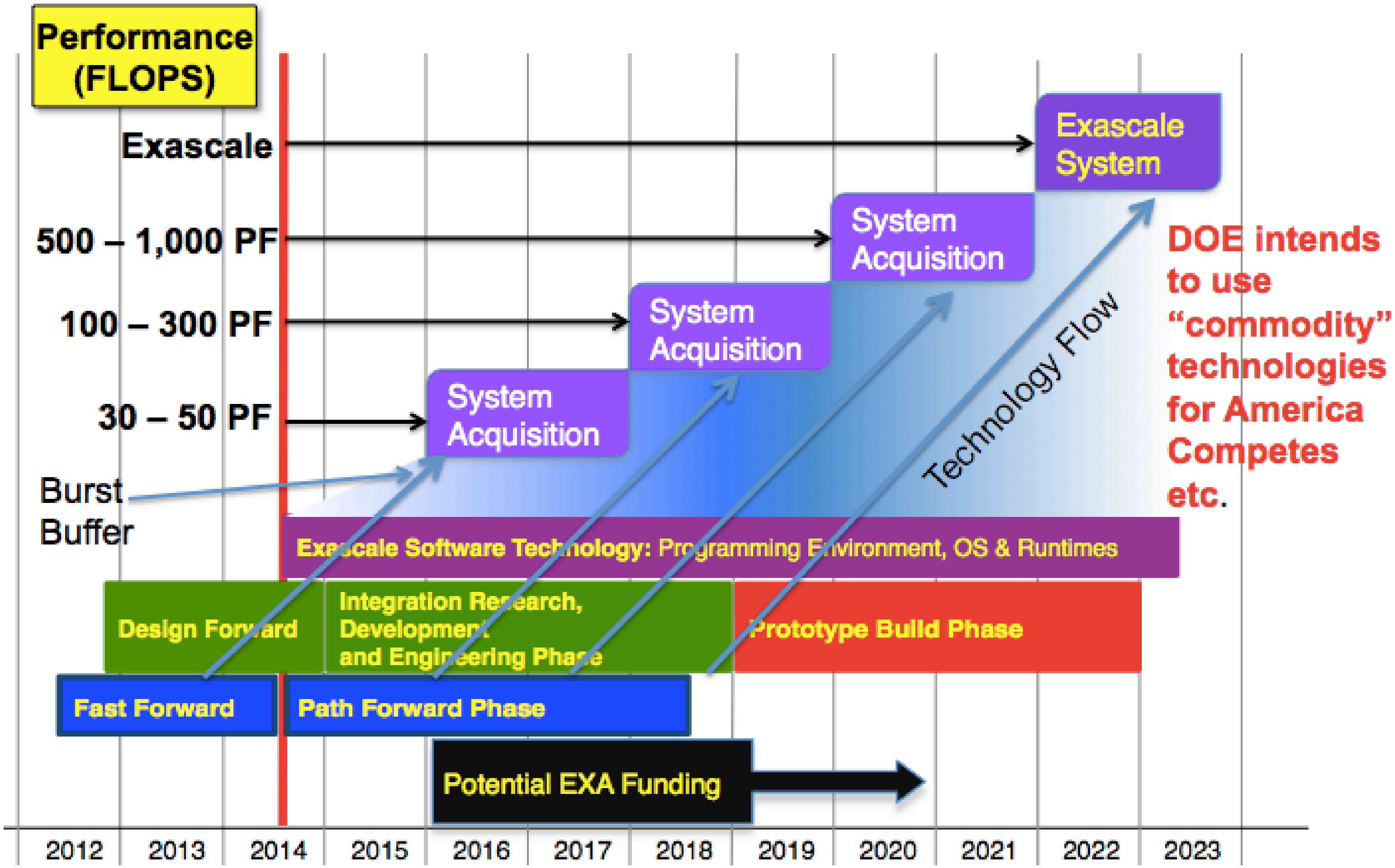
Article Section: STORAGE

Before the advent of Big Data, the largest storage systems in the world were found almost exclusively within high performance computing centers such as those found at US Department of Energy national laboratories. However, these systems are now dwarfed by large datacenters such as those run by Google and Amazon. Although HPC storage systems are no longer the largest in terms of total capacity, they do exhibit the largest degree of concurrent write access to shared data. In this article, we will explain why HPC applications must necessarily exhibit this degree of concurrency and the unique HPC storage architectures required to support them.

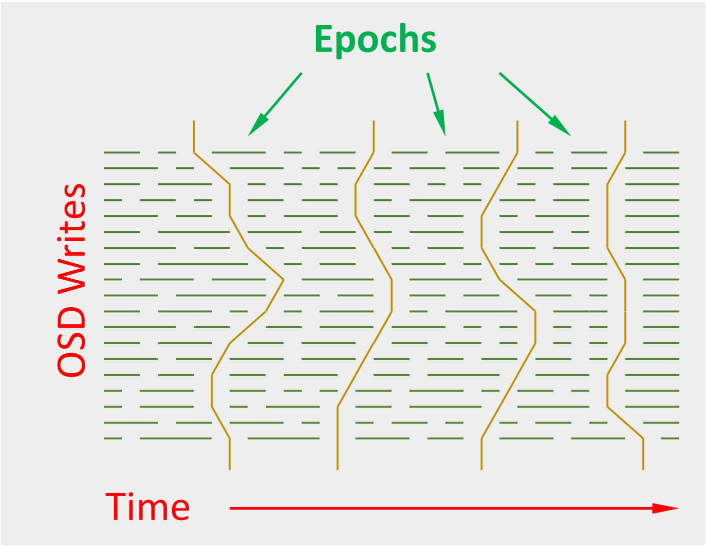
DOE Exascale Computing and Future Considerations

- R&D and integration required to deploy Applications on Exascale computers in 2023+
- Partnership involving: Government, Computer industry, DOE laboratorie, Academia
- Target System Characteristics
  - **1-10 Billion** degrees of concurrency
  - **20-30 MW** Power requirement for one machine (machine only)
  - **<300** cabinets
  - **Development and execution time productivity improvements**
  - **100 PB working sets**
  - **Checkpoint times < 1 minute (constant failure)**
  - **Storage systems need to be very reliable as the machine wont be**
  - **Leverage->Exploit new technology, dense flash/SCM/low latency high bandwidth byte addressable networks**

Exascale Computing Timeline
System Perspective

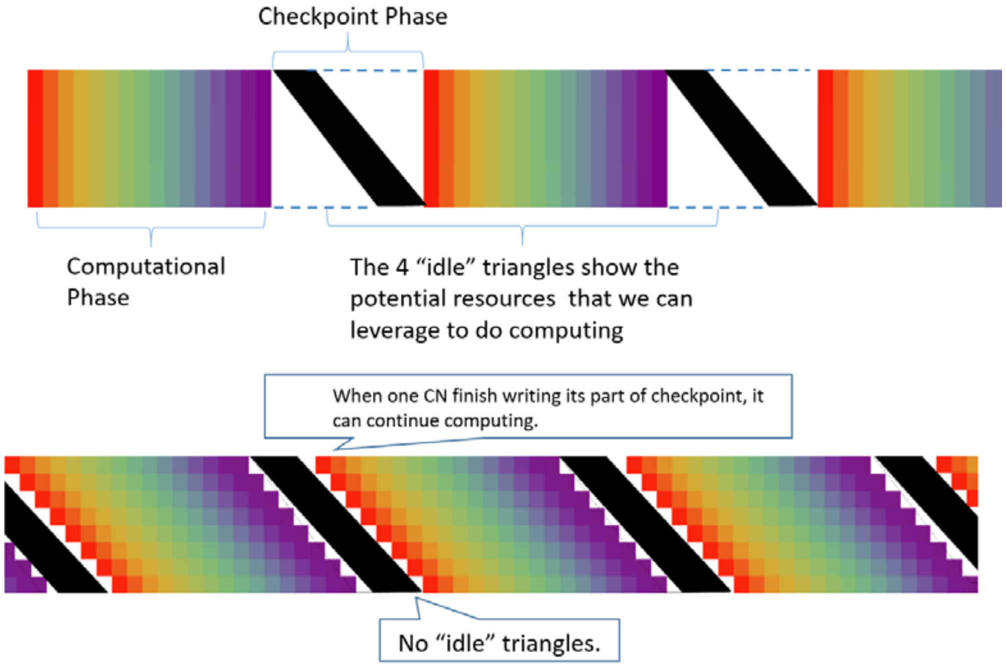


Transactional/Versioning Coupled with Async Programming Models



Spread the load over as much time as the app will allow

Save every microsecond



Exploiting New Techology (dense flash, SCM, one sided networks)

- Todays storage stacks do not allow full exploitation of even flash latencies never the less SCM – trapped IOP's in software
- Must move away from thin client to heavy server to block storage (Lustre, Ceph, MongoDB).
- These stacks are incredibly thick. Client KV/Object -> MDS/Access Server->KV/Object Server->Kernel File System->block or network block.
- Head towards embedding more of this in the client and provide light weight one side enabled servers.
- Applications use Middleware (HDF5 for example) to form Objects that go directly to light weight object servers, no real server just light weight kvs/object on the hardware.
  - HDF5 (HDF Group) Vol interface is an example
  - MDHIM (LANL) MultiDimensional Hierarchical Middleware is a user space distributed KVS middleware
  - DeltaFS (CMU/LANL) is a user space file system Middleware

Open Source
BSD License
Partners Welcome

https://github.com/mar-file-system/marfs
https://github.com/pftool/pftool)

Thank You For Your Attention

Los Alamos NATIONAL LABORATORY EST. 1943

NNSA National Nuclear Security Administration

Attribution & Feedback

The SNIA Education Committee thanks the following Individuals for their contributions to this Tutorial.

Authorship History

Gary Grider July 2017

Additional Contributors

Please send any questions or comments regarding this SNIA Tutorial to ***tracktutorials@snia.org***