SNIA DSI '16 DATA STORAGE INNOVATION CONFERENCE
Innovation in Storage Products, Services, and Solutions
June 13-15, 2016 | Marriott San Mateo | San Mateo, CA

MarFS: A Scalable Near-POSIX File System over Cloud Objects Background, Use, and Technical Overview

Gary Grider
HPC Division Leader

June 2016

LA-UR-16-20920

Los Alamos NATIONAL LABORATORY EST.1943

NNSA

Eight Decades of Production Weapons Computing to Keep the Nation Safe

Maniac

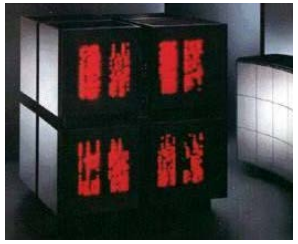
IBM Stretch

CDC

Cray 1

Cray X/Y

CM-2



CM-5

SGI Blue Mountain

DEC/HP Q

IBM Cell Roadrunner

Cray XE Cielo

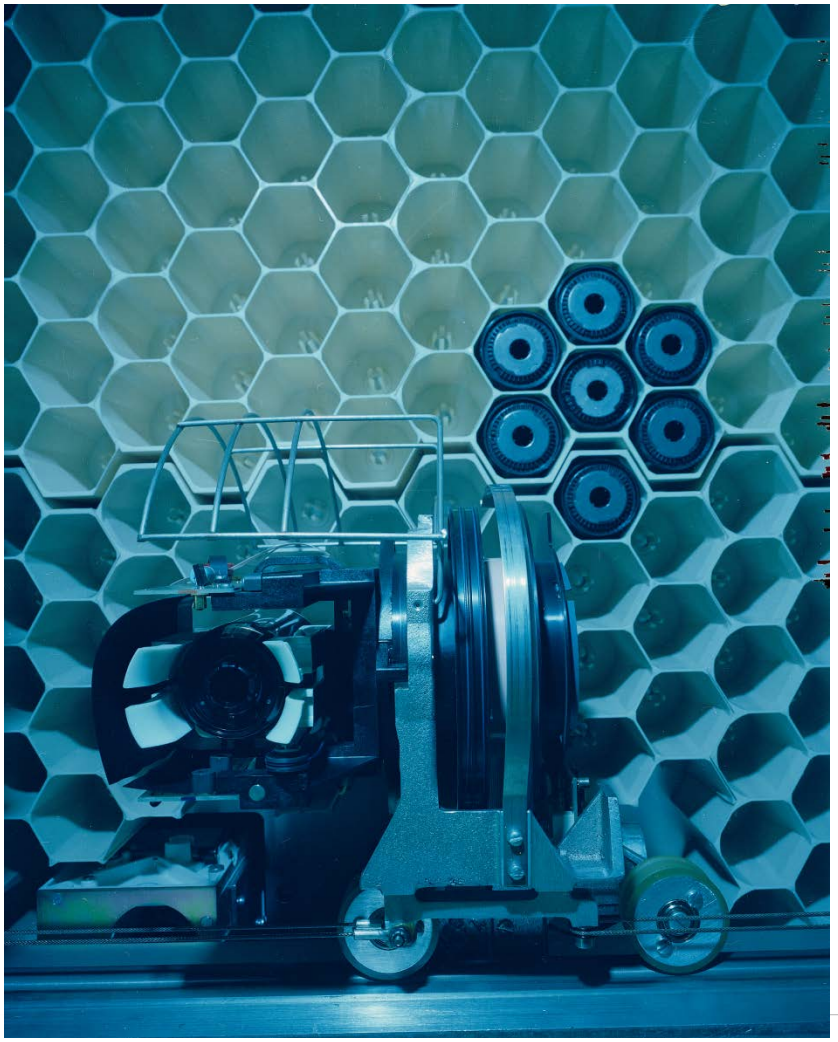
Cray Intel KNL Trinity

Ziggy DWave

Cross Roads

CROSS ROADS
An APEX Collaboration

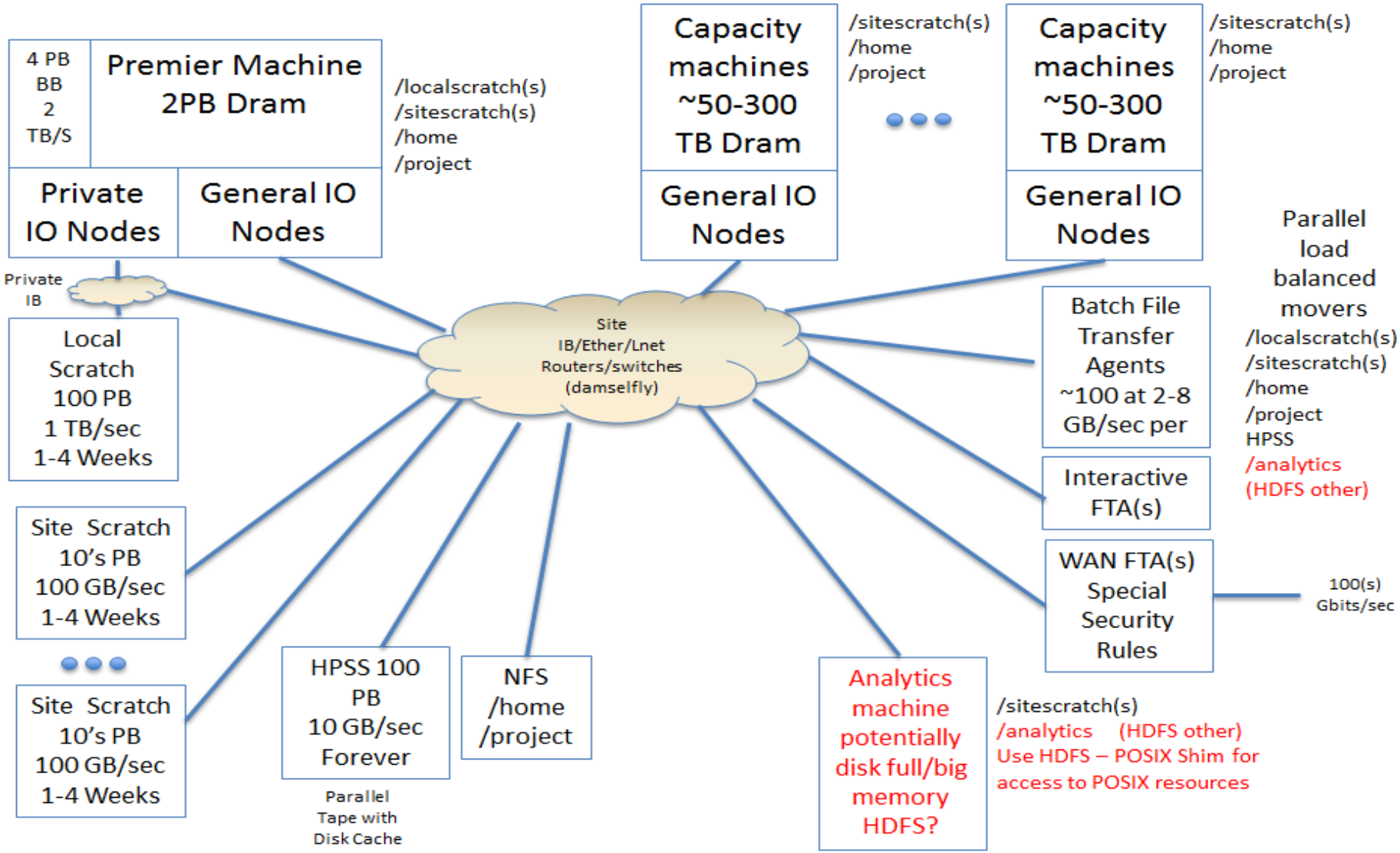
HPC Storage Background



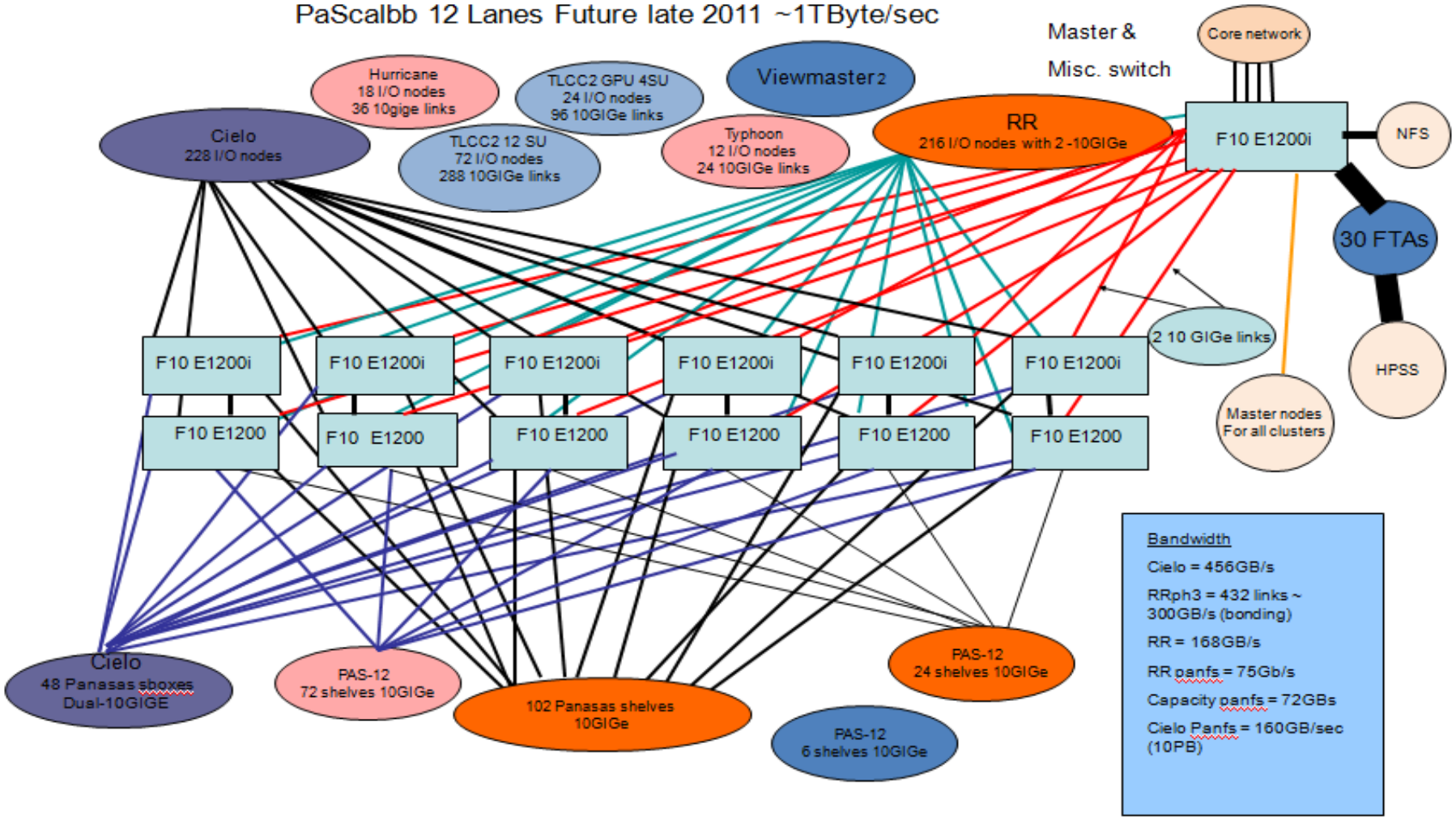
IBM Photostore

IBM
3850

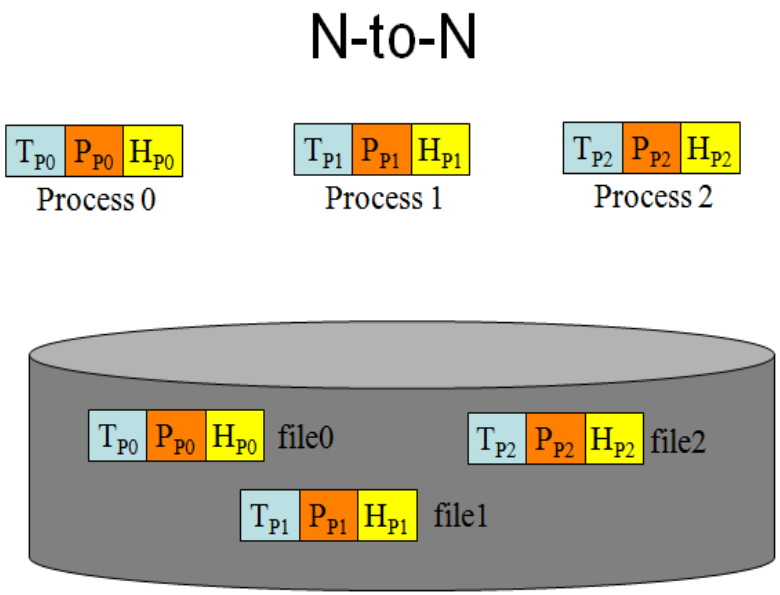
Simple View of our Computing Environment

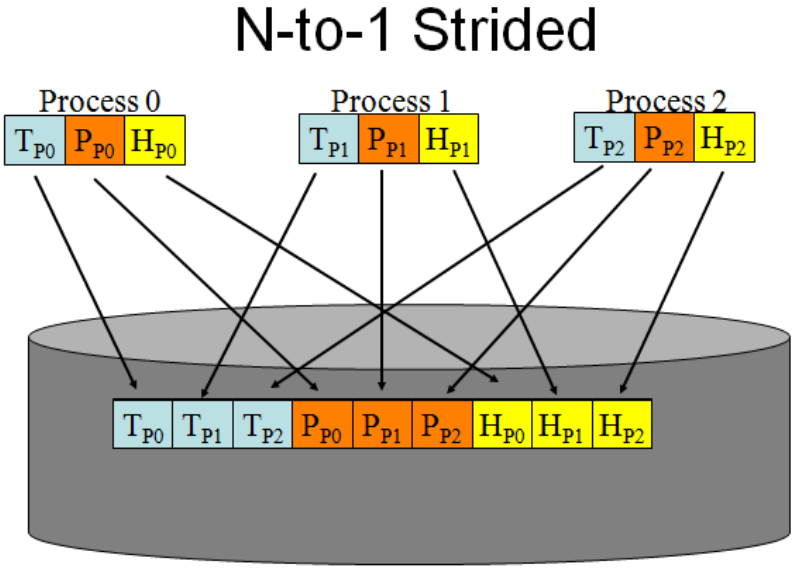


HPC Network Circa 2011
Today high end is a few TB/sec



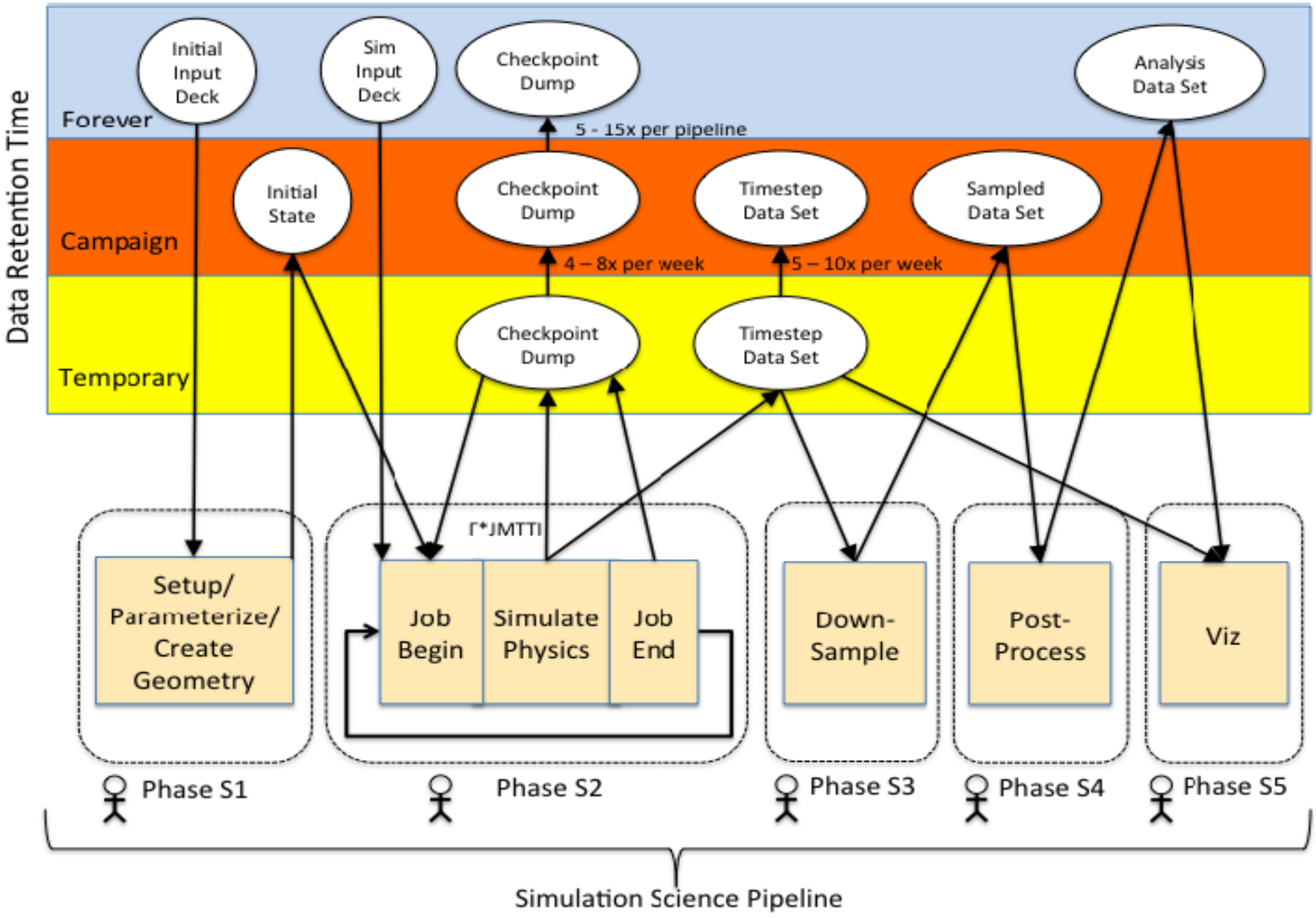
HPC IO Patterns





- 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 1 PB DRAM)

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.

Workflow from APEX Procurement Docs
A High Throughput/UQ Pipeline

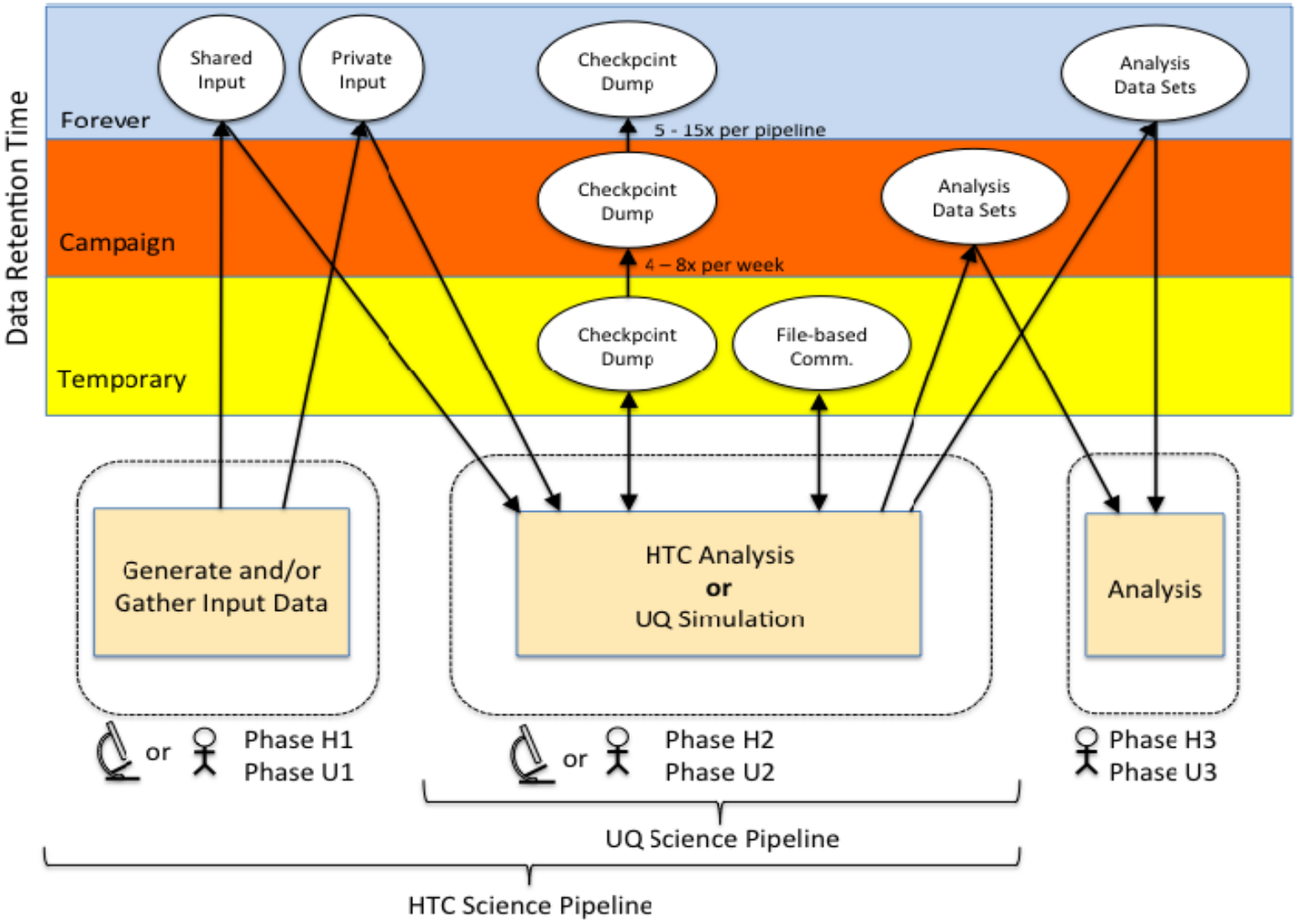


Figure 2: Example HTC and UQ workflow

APEX Workflows

LANL, NERSC, SNL

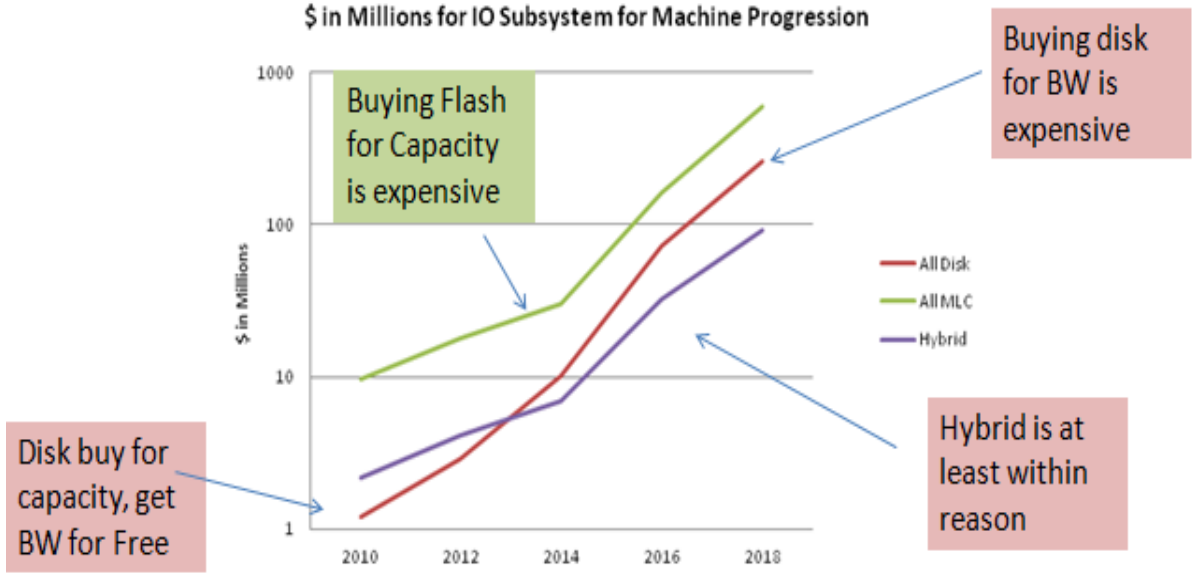
SAND2015-10342 O
LA-UR-15-29113

Workflow Data That Goes With the Workflow Diagrams

	Tri-Labs workload							NERSC workload						
	LANL				SNL		LLNL							
Workflow	EAP	LAP	Silverton	VPIC	Dakota A	Dakota S	Pending	ALS	CESM	GTS	HipMer	Materials	MILC	Sky Survey
Workflow type	Sim	Sim	Sim	Sim	Sim/UQ	UQ	Pending	HTC	Sim	Sim	HTC	Sim	Sim	HTC
Site Workload percentage	60	5	15	10	10 to 15	10 to 15	100	< 1	3	1	< 1	7	5	< 1
Representative workload percentage	20	2	5	3	4	4	33	3	6	6	7	19	11	3
Number of Cielo cores	65536	32768	131072	70000	131072	65536		100	2064	16384	960	2400	100000	24
Number of workflow pipelines per allocation	1 to 15	1 to 5	1 to 10	5 to 10	100 to 1000	50 to 200		10760	8	100	100	100	1000	21000
Number of simultaneous allocations	20 to 25	2 to 3	2 to 3	2 to 3	5 to 10	5 to 10	8							
Anticipated increase in problem size by 2020	10 to 12x	8 to 12x	6x	10x	2 to 4x	1x		1x	16 to 23x	5x	1x	10 to 25x	1x	1x
Anticipated increase in workflow pipelines per allocation by 2020	1x	1x	1x	1x	2 to 8x	2x		5x	3x	1x	50x	1x	?	2.38x
Data retained (percentage of memory)	910.00	3050.00	1205.00	545.25	415.00	25.07		285.63	835.37	15.57	100.54	135.42	103.38	11.57
During pipeline	**50.00**	**85.00**	**320.00**	**222.75**	**20.00**	**0.15**		**147.68**	**0.29**	**0.68**	**34.34**	**20.83**	**102.53**	**2.16**
Analysis	20.00	10.00	5.00	200.00	5.00			126.57			34.34	20.83		2.16
Checkpoint	30.00	75.00	210.00	18.75	10.00	0.15			0.29	0.68				
Input			70.00	5.00	5.00	0.00		21.10						
Out-of-core													102.53	
During Allocation	**240.00**	**210.00**	**480.00**	**142.50**	**345.00**	**0.00**		**21.10**					**0.62**	
Analysis	60.00	60.00	60.00	100.00	40.00			21.10					0.62	
Checkpoint	180.00	150.00	420.00	37.50	300.00									
Input	0.00	0.00		5.00	5.00	0.00								
Forever	**620.00**	**2755.00**	**405.00**	**180.00**	**40.00**	**24.93**		**116.84**	**835.08**	**14.89**	**66.20**	**114.58**	**0.23**	**9.41**
Analysis	500.00	2500.00	5.00	130.00	35.00	24.44		106.29	808.14	14.89	0.36	114.58	0.12	0.62
Checkpoint	100.00	250.00	400.00	50.00		0.49			26.94					
Input	20.00	5.00			5.00			10.55	0.00		65.83	0.00	0.12	8.79

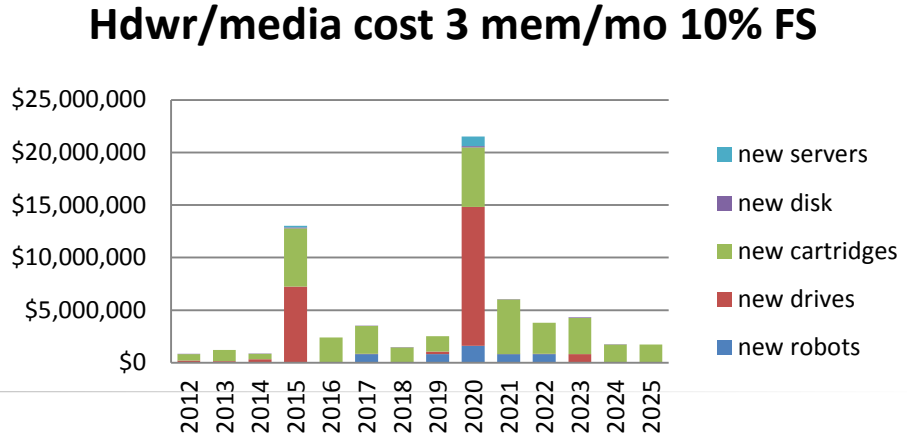
Enough with the HPC background
Why do we need one of these MarFS things?

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

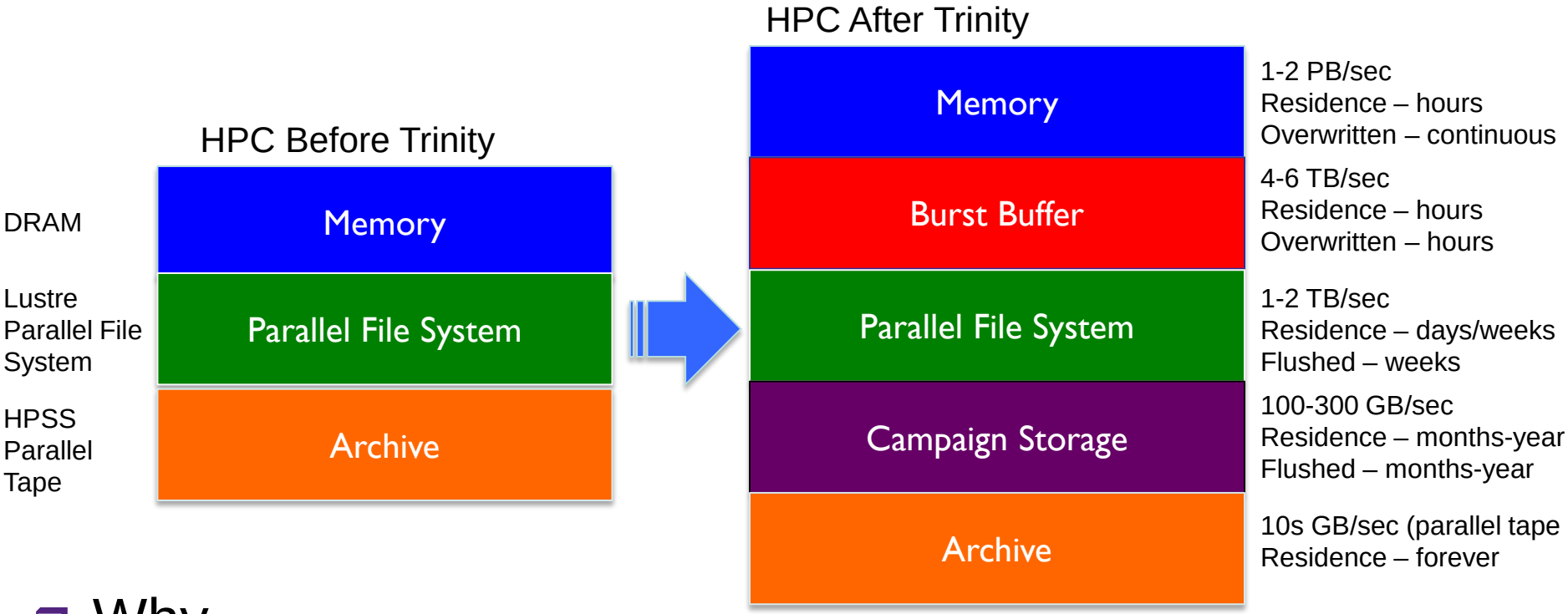


- Economic modeling for large burst of data from memory shows bandwidth / capacity better matched for solid state storage near the compute nodes

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



What are all these storage layers?
Why do we need all these storage layers?

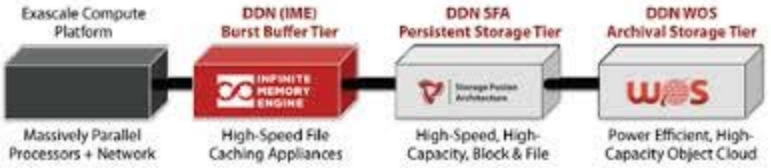


- Why
  - BB: Economics (disk bw/iops too expensive)
    - Campaign: Economics (PFS Raid too expensive, PFS solution too rich in function, PFS metadata not scalable enough, PFS designed for scratch use not years residency, Archive BW too expensive/difficult, Archive metadata too slow)

The Hoopla Parade circa 2014

DDN STORAGE

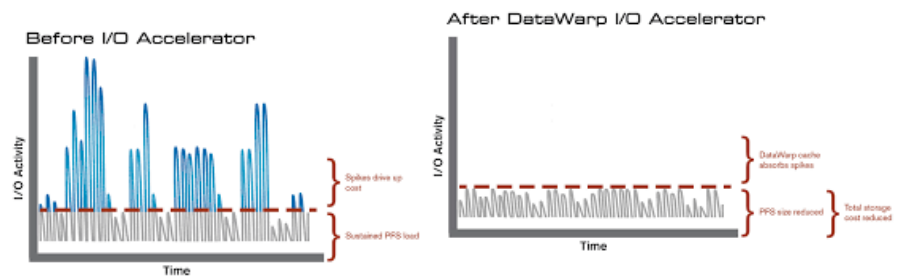
INFINITE MEMORY ENGINE™



CRAY
THE SUPERCOMPUTER COMPANY

Data Warp

Powered By Dilithium Crystals



Isn't that too many layers just for storage?

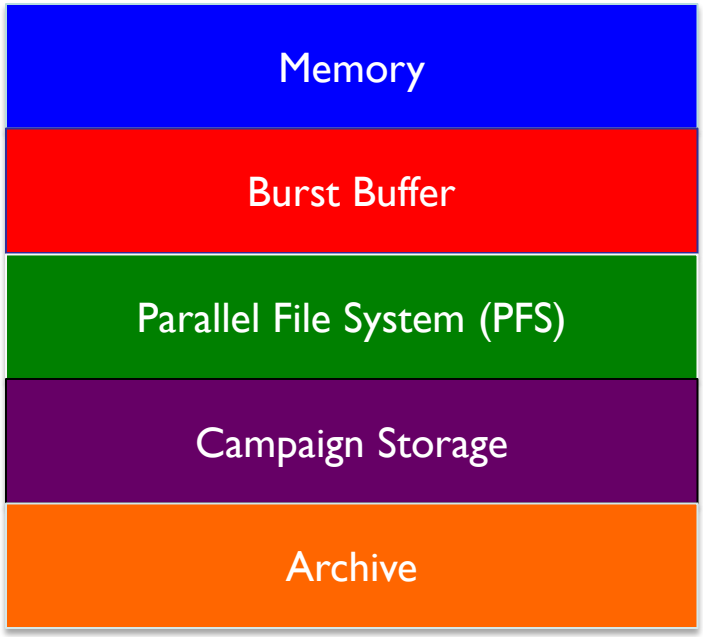
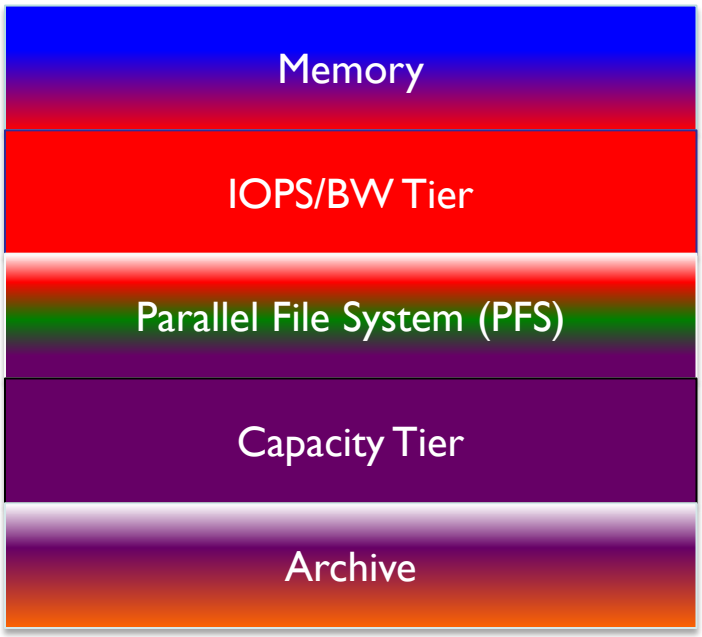


Diagram courtesy of John Bent EMC



Factoids (times are changing!)
LANL HPSS = 53 PB and 543 M files

Trinity 2 PB memory, 4 PB flash (11% of HPSS) and 80 PB PFS or 150% HPSS)

Crossroads may have 5-10 PB memory, 40 PB solid state or 100% of HPSS

- If the Burst Buffer does its job very well (and indications are capacity of in system NV will grow radically) and campaign storage works out well (leveraging cloud), do we need a parallel file system anymore, or an archive?
- Maybe just a bw/iops tier and a capacity tier.
- Too soon to say, seems feasible longer term

RIP PFS ?

We would have never contemplated more in system storage than our archive a few years ago

I doubt this movement to solid state for BW/IOPS (hot/warm) and SMR/HAMR/etc. capacity oriented disk for Capacity (cool/cold) is unique to HPC

Ok – we need a capacity tier
Campaign Storage

-Billions of files / directory
-Trillions of files total
-Files from 1 byte to 100 PB
-Multiple writers into one file

What now?

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?

- 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

Won’t someone else do it, PLEASE?

- There is evidence others see the need but no magic bullets yet: (partial list)
  - Cleversafe/Scality/EMC ViPR/Ceph/Swift etc. attempting multi-personality data lakes over erasure objects, all are young and assume update in place for posix
  - GlusterFS is probably the closes thing to MarFS. Gluster is aimed more for the enterprise and midrange HPC and less for extreme HPC. Glusterfs is a way to unify file and object systems, MarFS is another, aiming at different uses
  - General Atomics Nirvana, Storage Resource Broker/IRODS optimized for WAN and HSM metadata rates. There are some capabilities for putting POSIX files over objects, but these methods are largely via NFS or other methods that try to mimic full file system semantics including update in place. These methods are not designed for massive parallelism in a single file, etc.
  - EMC Maginatics but it is in its infancy and isnt a full solution to our problem yet.
  - Camlistore appears to be targeted and personal storage.
  - Bridgestore is a POSIX name space over objects but they put their metadata in a flat space so rename of a directory is painful.
  - Avere over objects is focused at NFS so N to 1 is a non starter.
  - HPSS or SamQFS or a classic HSM? Metadata rates designs are way low.
  - HDFS metadata doesn’t scale well.

MarFS Requirements

- Linux system(s) with C/C++ and FUSE support
- MPI for parallel comms in Pftool (a parallel data transfer tool)
  - MPI library can use many comm methods like TCP/IP, Infiniband OFED, etc.
- Support lazy data and metadata quotas per user per name space
- Wide parallelism for data and metadata
- Try hard not to walk trees for management (use inode scans etc.)
- Use trash mechanism for user recovery

- Multi-node parallelism MD FS's must be globally visible somehow
- Using object store data repo, object store needs globally visible.
- The MarFS MD FS's must be capable of POSIX xattr and sparse
  - We use GPFS but you don't have to use GPFS, we use due to ILM inode scan features

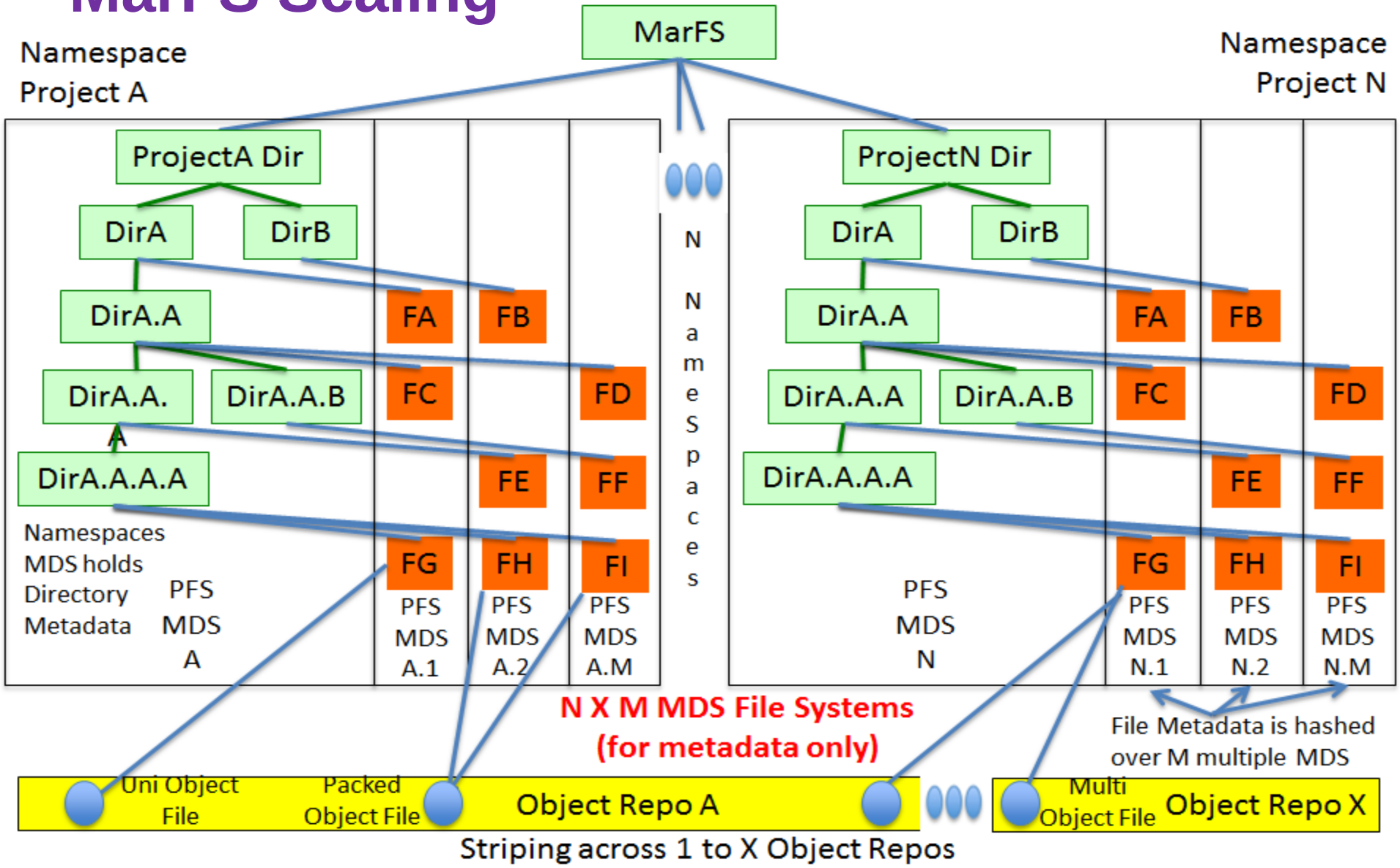
What is MarFS?

- Near-POSIX global scalable name space over many POSIX and non POSIX data repositories (Scalable object systems - CDMI, S3, etc.)
- It scales name space by sewing together multiple POSIX file systems both as parts of the tree and as parts of a single directory allowing scaling across the tree and within a single directory
- It is small amount of code (C/C++/Scripts)
  - A small Linux Fuse
  - A pretty small parallel batch copy/sync/compare/ utility
  - A set of other small parallel batch utilities for management
  - 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

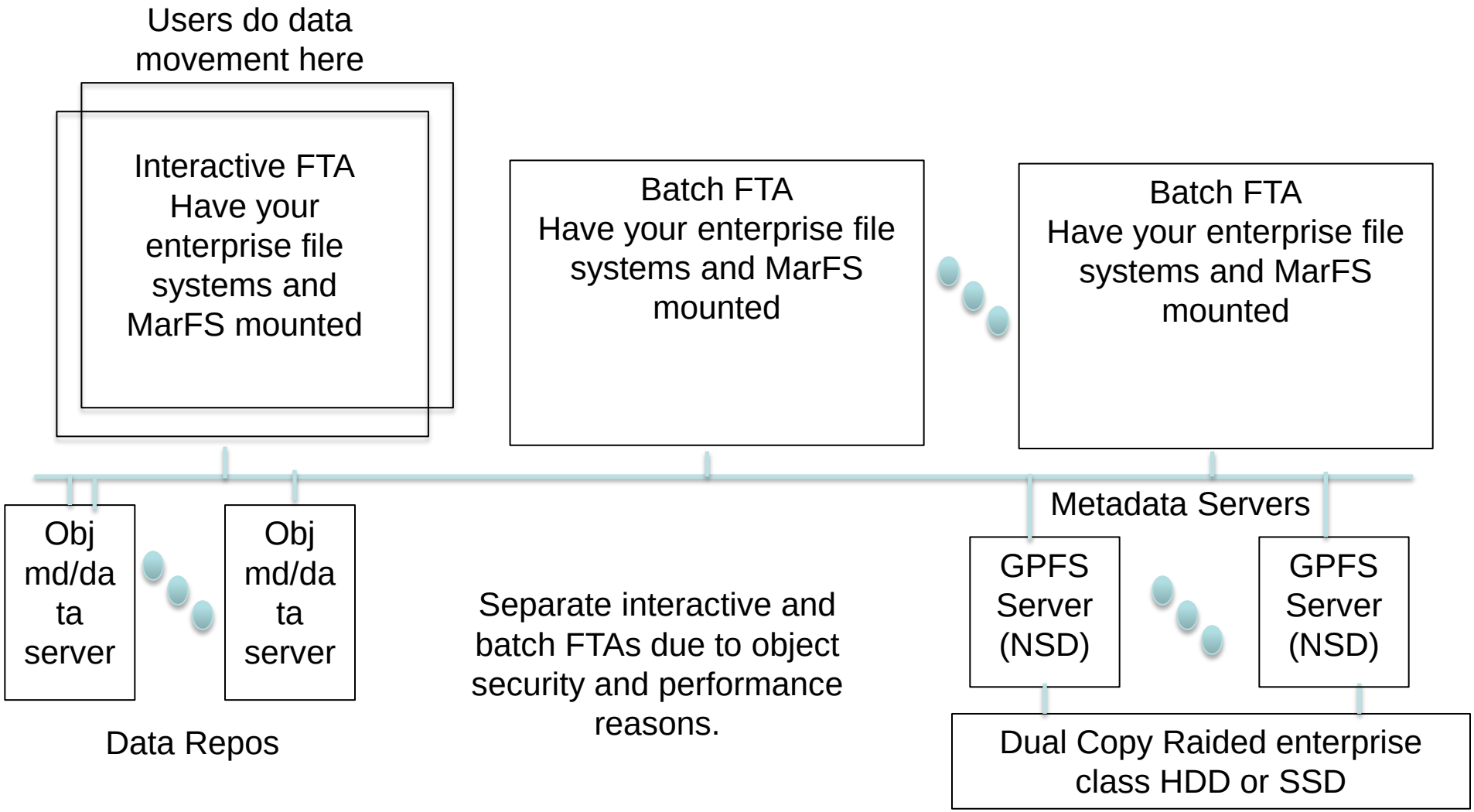
What it is not!

- Doesn't allow update file in place for object data repo's (no seeking around and writing – it isnt a parallel file system)
- The interactive use - FUSE
  - Does not check for or protect against multiple writers into the same file, batch copy utility or library for efficient parallel writing)
  - Fuse is targeted at interactive use
  - Writing to object backed files works but FUSE will not create data objects that are packed as optimized as the parallel copy utility.
    - Batch utilities to reshape data written by fuse

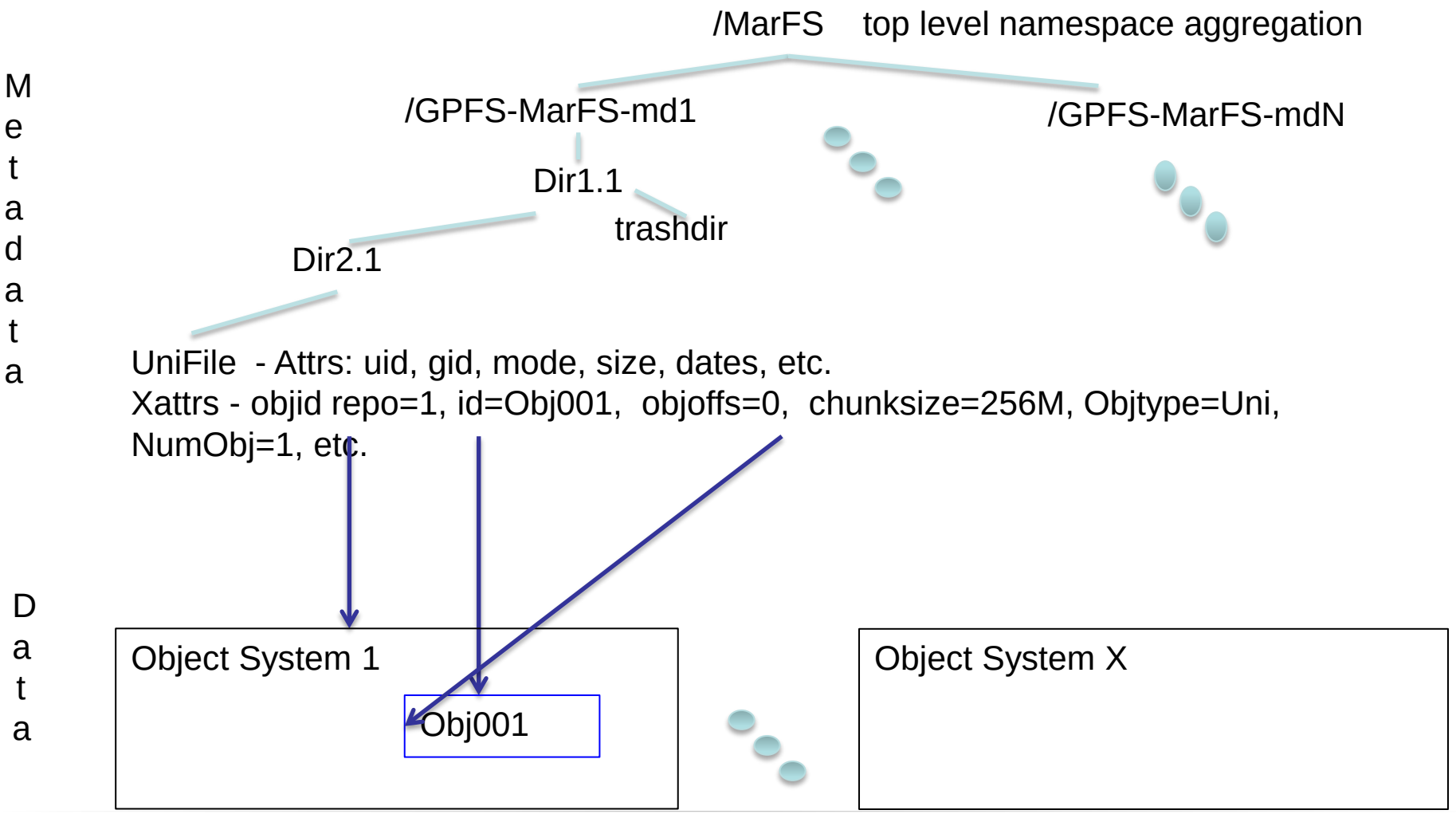
MarFS Scaling



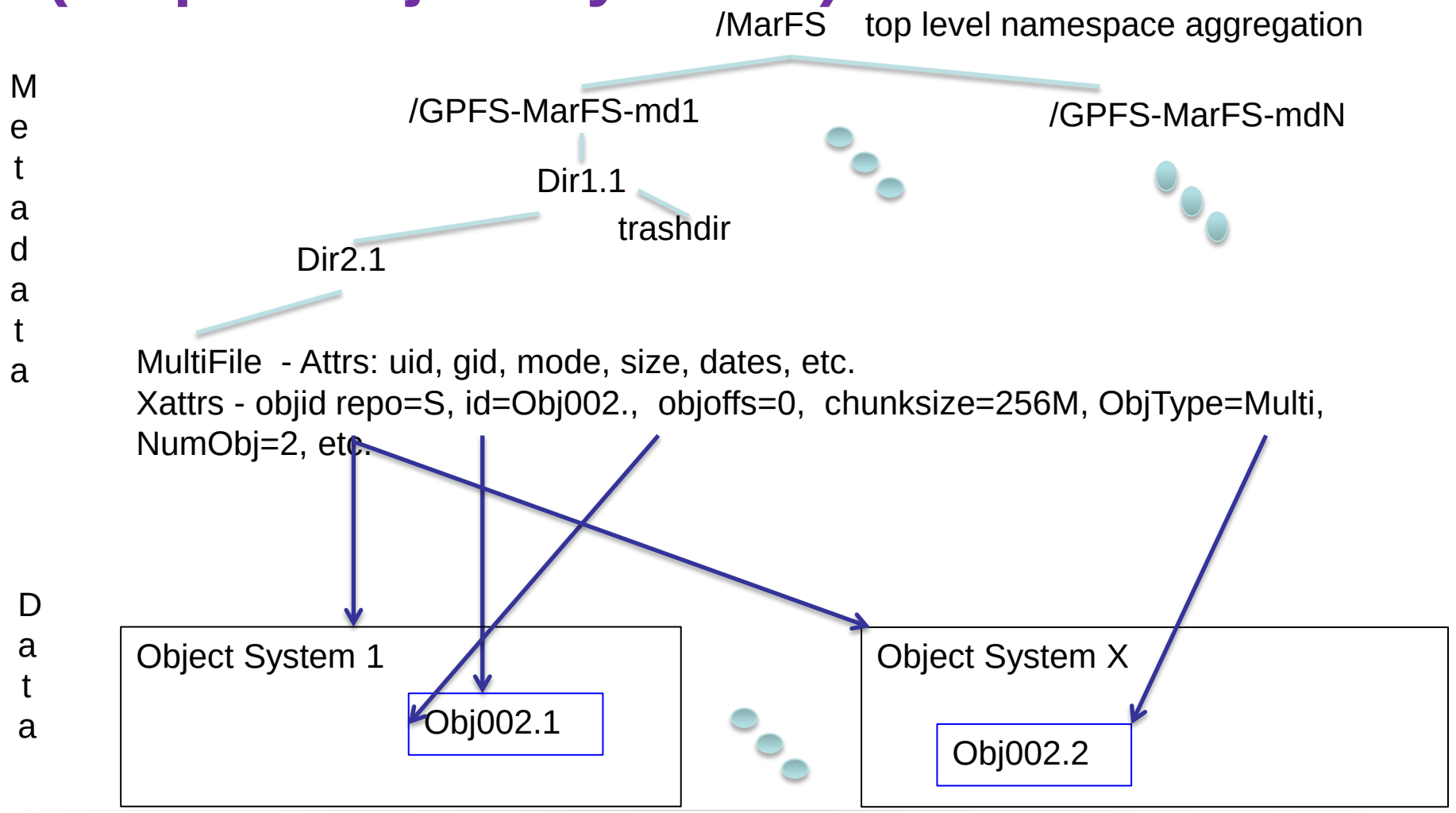
Simple MarFS Deployment



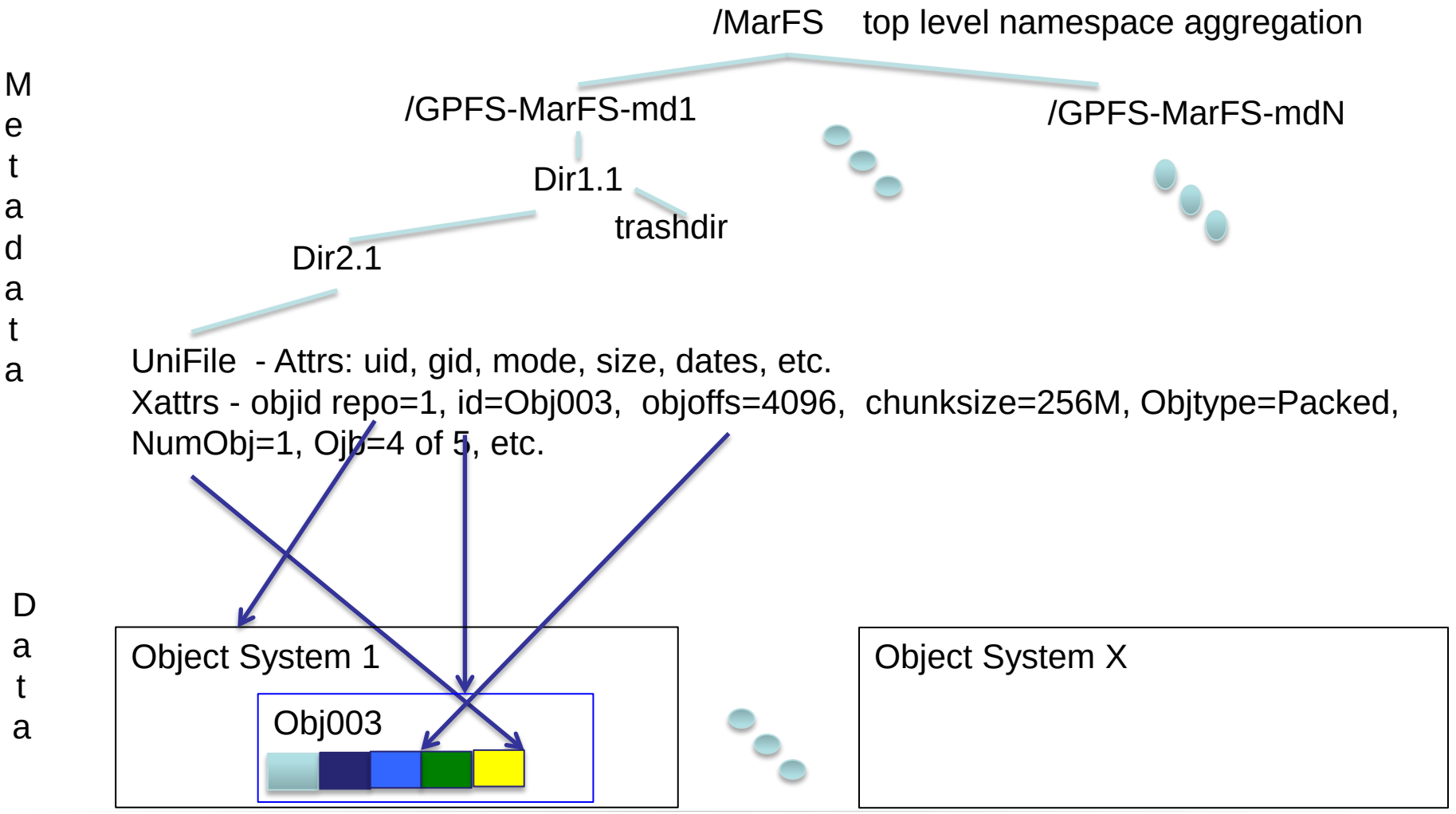
MarFS Internals Overview Uni-File



MarFS Internals Overview Multi-File (striped Object Systems)

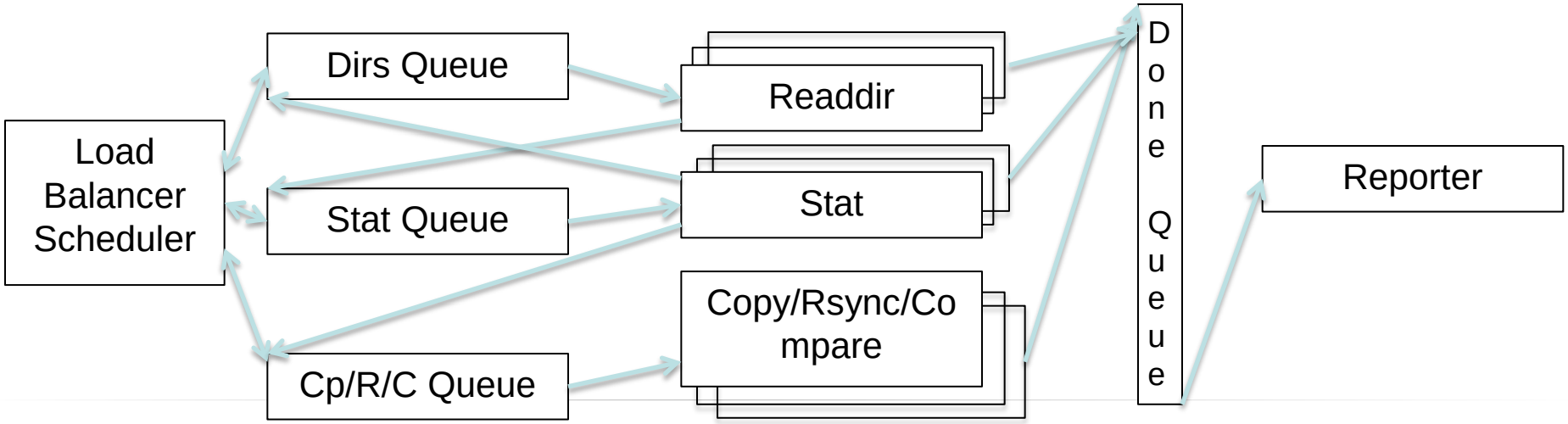


MarFS Internals Overview Packed-File

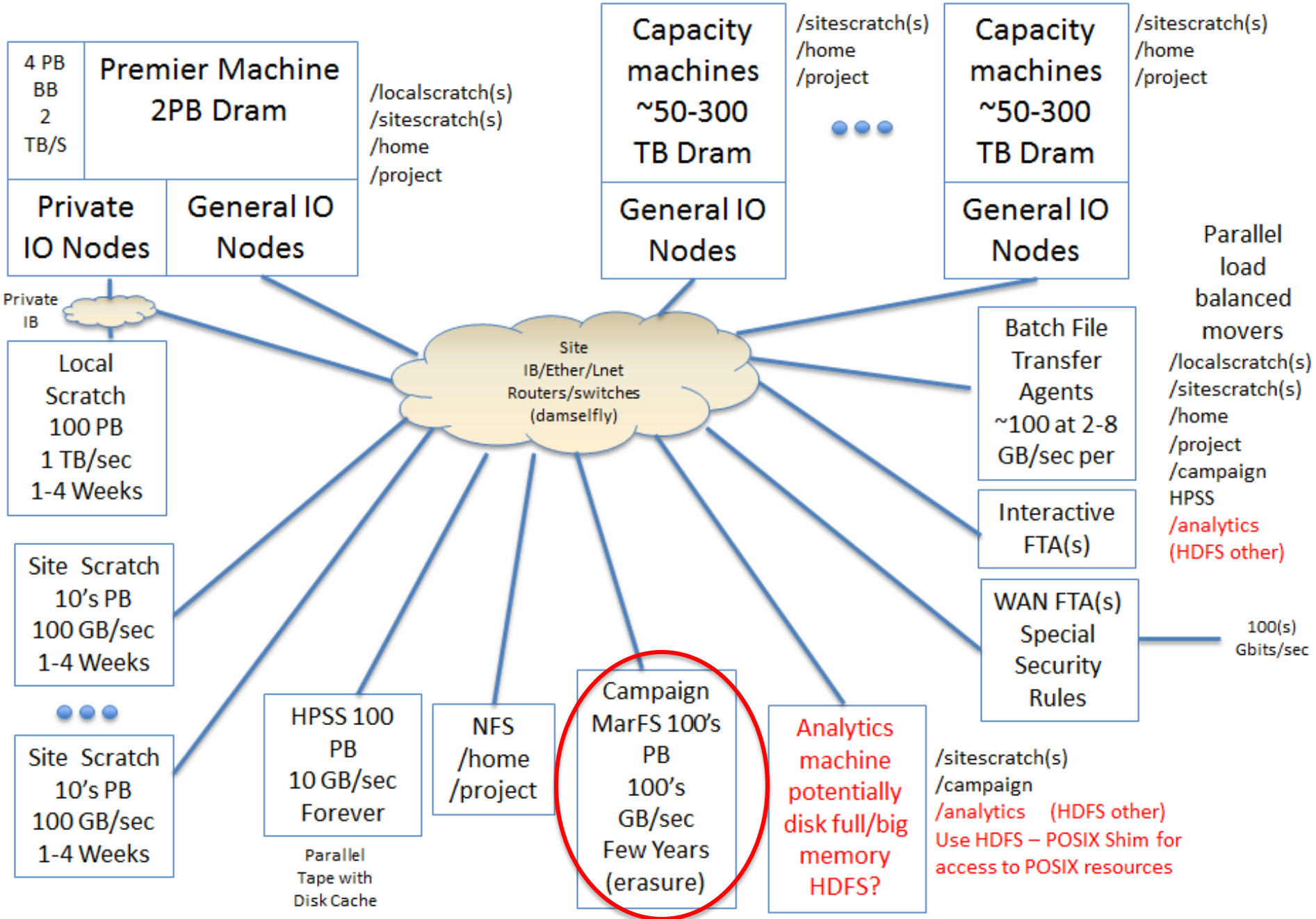


Pftool – parallel copy/rsync/compare/list tool

- 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



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

National Nuclear Security Administration