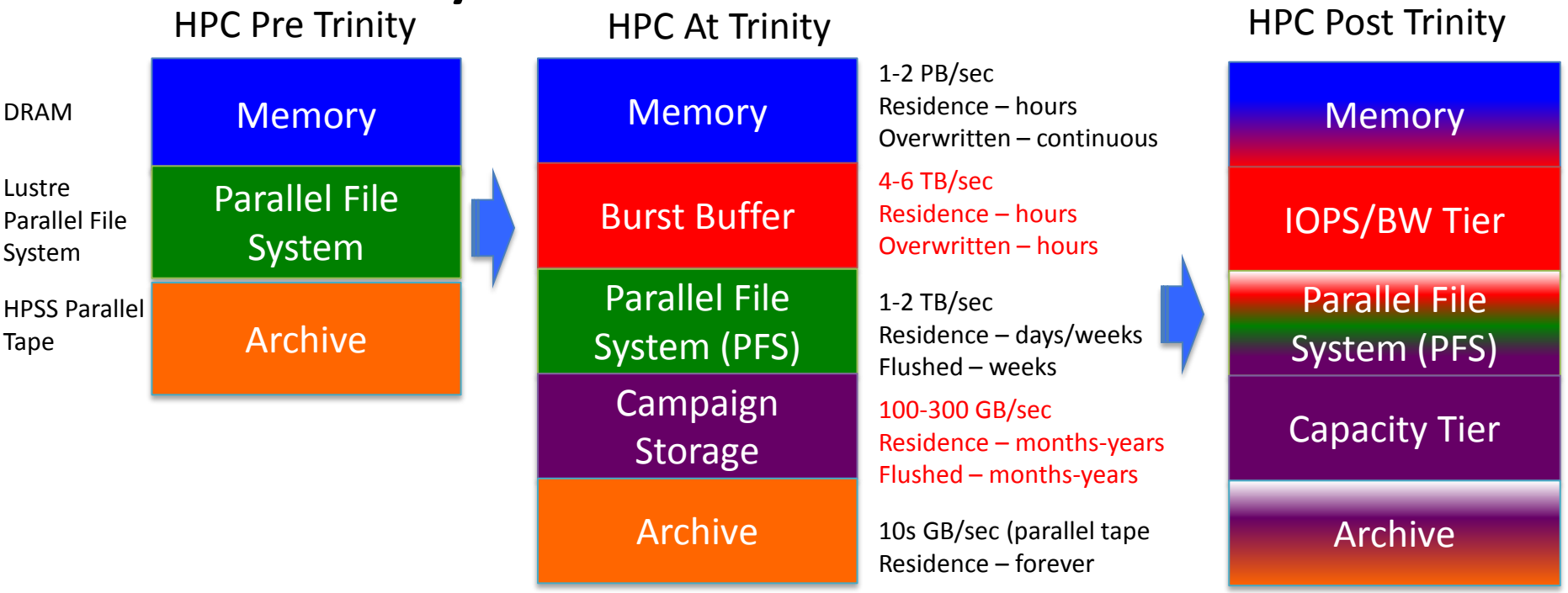
MarFS: A Scalable Near-POSIX File System over Cloud Objects

Kyle E. Lamb
HPC Storage Team Lead
Nov 18th 2015

LA-UR-15-27431

Why Do We Need a MarFS



- 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 ?

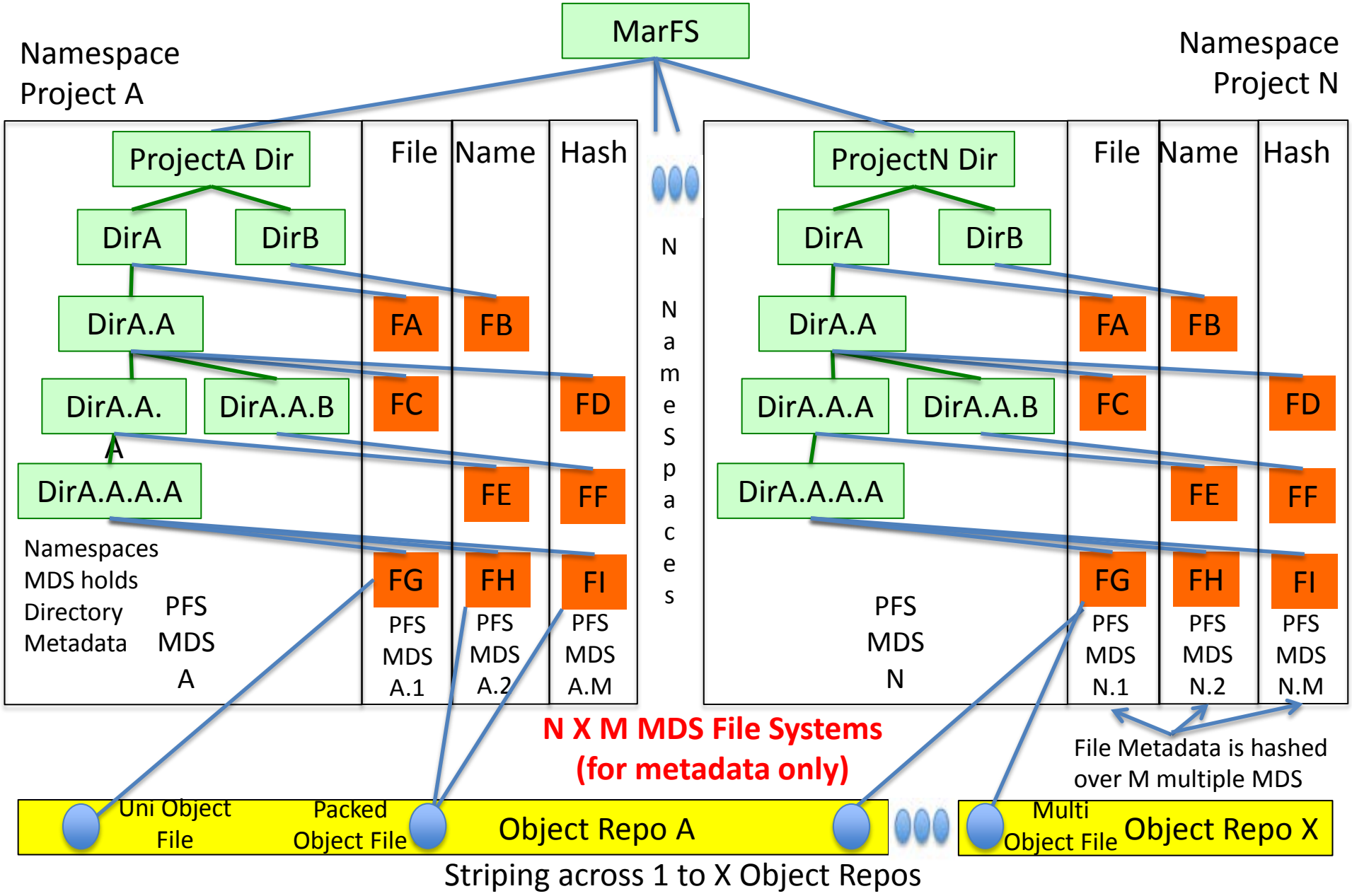
Factoids

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 with data residency measured in days or weeks

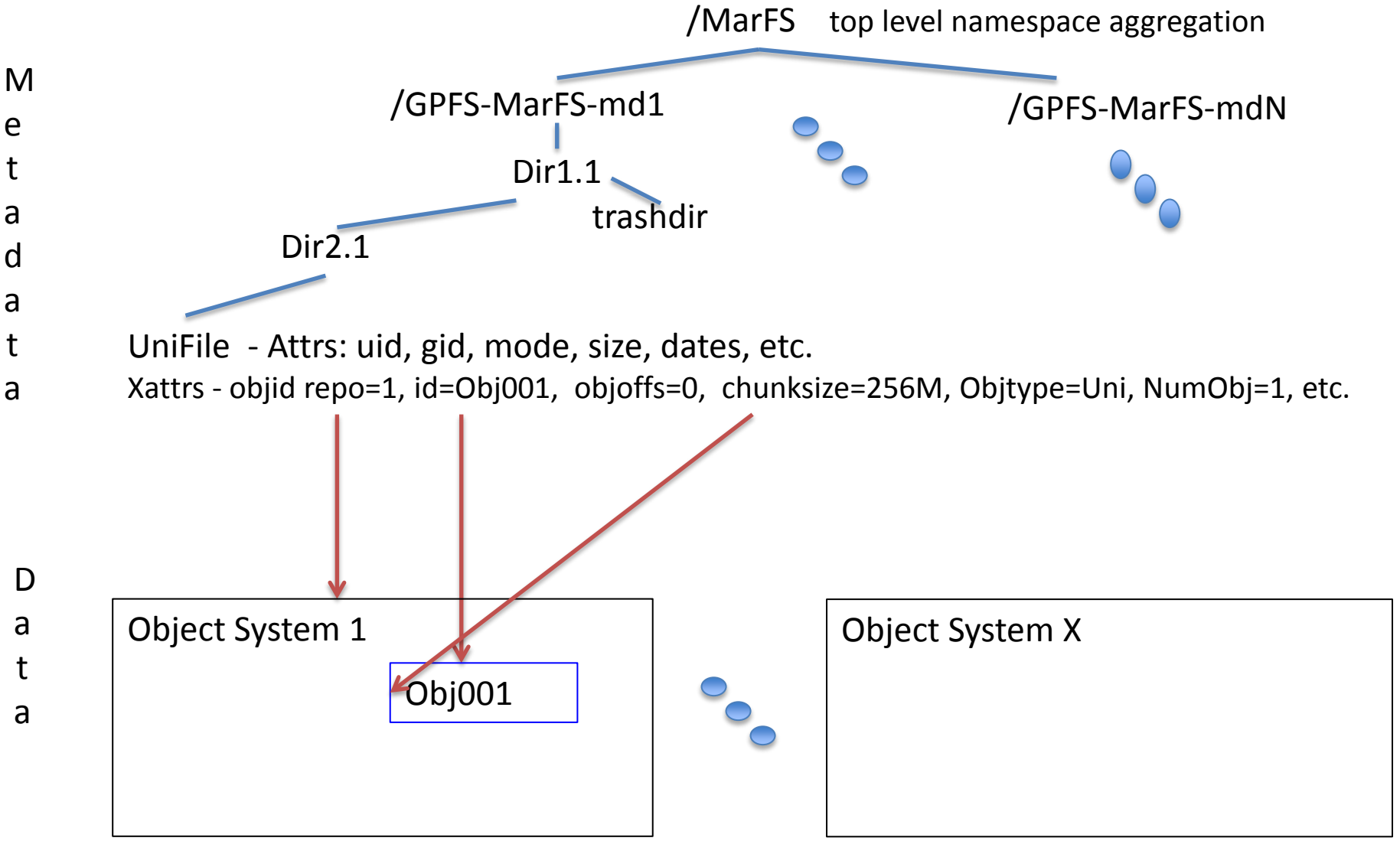
How about a Scalable Near-POSIX File System over Object Erasure?

- Best of both worlds
  - Objects Systems
    - Provide massive scaling and efficient erasure
    - 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 and no update in place with Objects
  - How do we
    - scale POSIX name space to **trillions** of files/directories
    - leverage massive Object Scalability for "Capacity Tier) with 100X BW of HSM's but 1/10 BW of PFS
    - with many years of data protection longevity?
- Looked at
  - GPFS, Lustre, Panasas, OrangeFS, Cleversafe/Scality/EMC ViPR/Ceph/Swift, Glusterfs, General Atomics Nirvana/Storage Resource Broker/IRODS, Maginatics, Camlistore, Bridgestore, Avere, HDFS

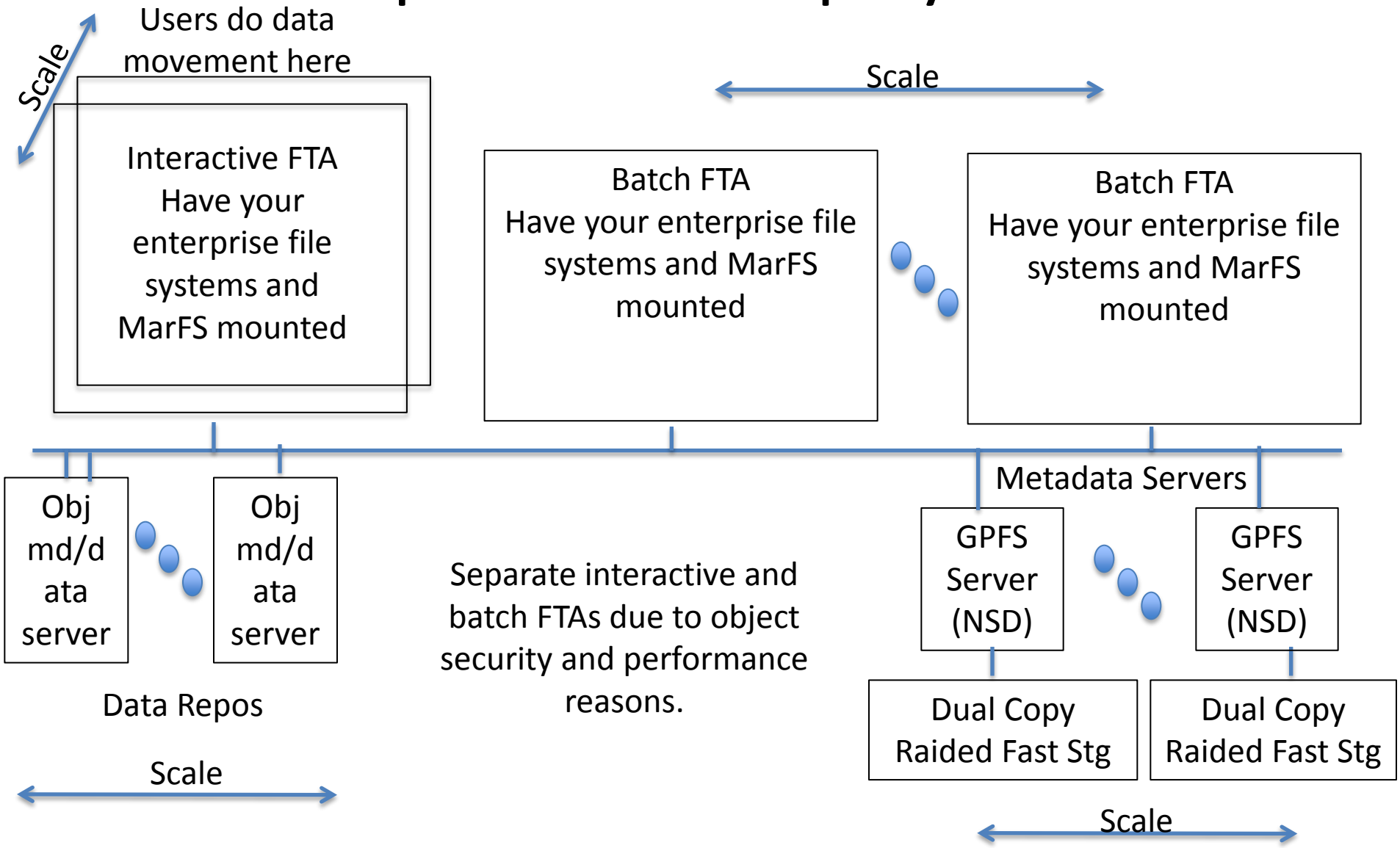
MarFS Scaling (Metadata and Data)



MarFS Internals Overview



Simple MarFS Deployment



Open Source, BSD License
Partners Welcome

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

BOF: **Two Tiers Scalable Storage: Building POSIX-Like Namespaces with Object Stores**
Date: Nov 18th, 2015
Time: 5:30PM - 7:00PM
Ron: Hilton Salon A
Session leaders : Sorin Faibish, Gary A. Grider, John Bent

Thank You For Your Attention

Los Alamos NATIONAL LABORATORY EST. 1943

National Nuclear Security Administration

Backup

Won’t someone else do it?

- There is evidence others see the need but no magic bullets yet: (partial list)
  - Cleversafe/Scality/EMC ViPR/Ceph/Swift etc. are moving towards multi-personality data lakes over erasure coded 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. It also is making the trade off space for update in place which we can live without. Glusterfs is a way to unify file and object systems, MarFS is another, each coming at it from a different stance in trade space
  - General Atomics Nirvana, Storage Resource Broker/IRODS is 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.
  - Maginatics from EMC 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? The metadata rates design target way too low.
  - HDFS metadata doesn’t scale well.

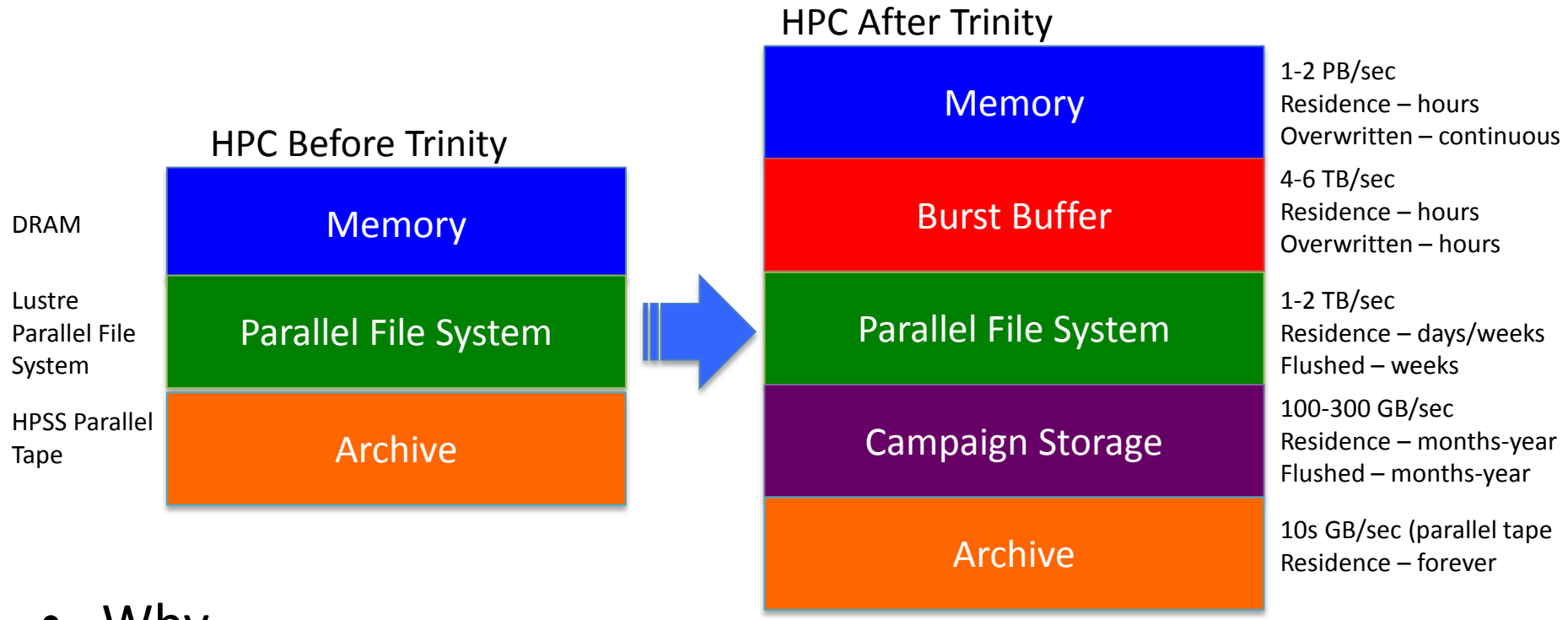
MarFS Requirements

- 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
- If use MarFS to combine multiple POSIX file systems into one mount point, any set of POSIX file systems can be used.
- 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
  - 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 are all these storage layers?
Why do we need all these storage layers?



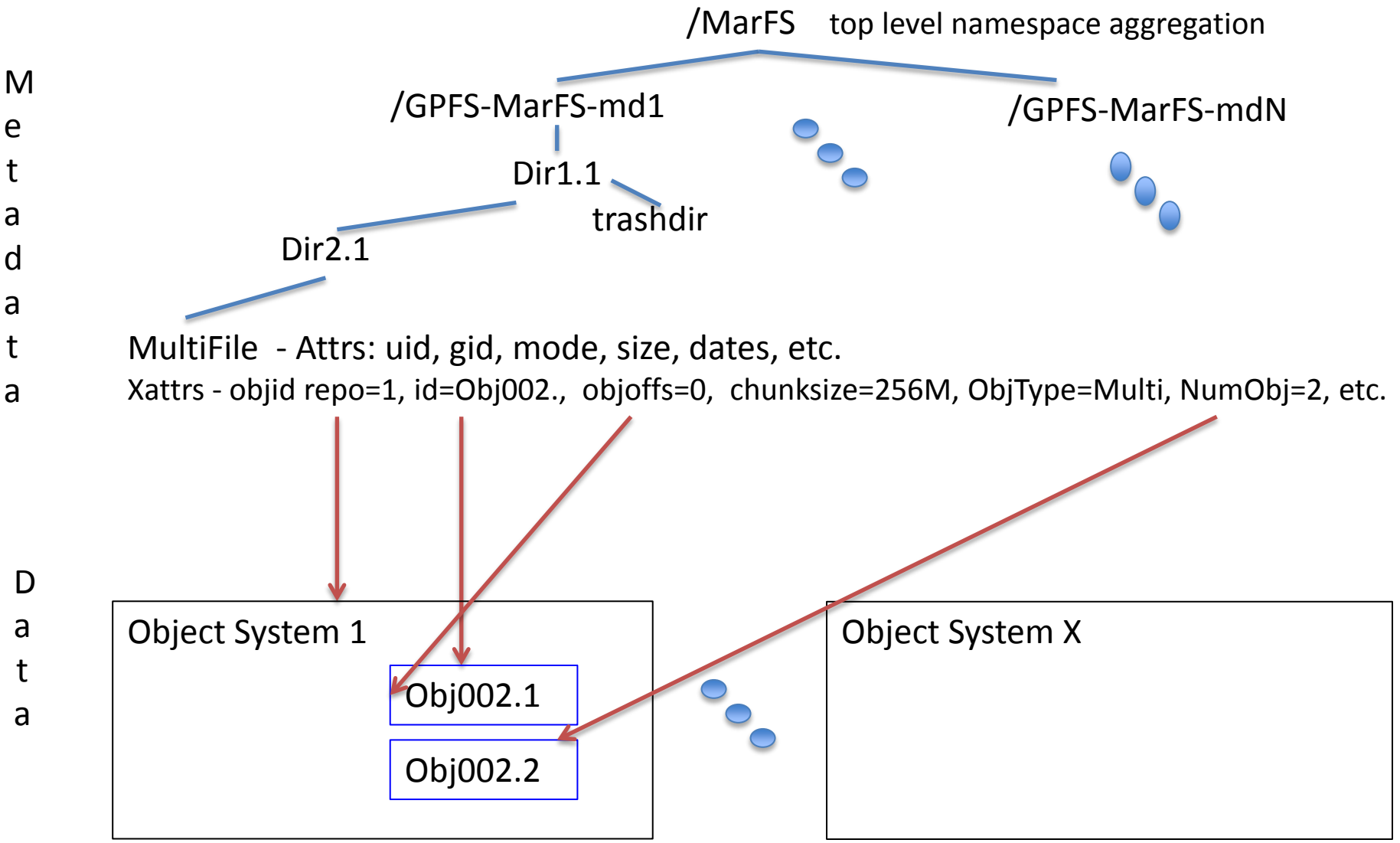
- Why
  - BB: Economics (disk bw/iops too expensive)

- 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)

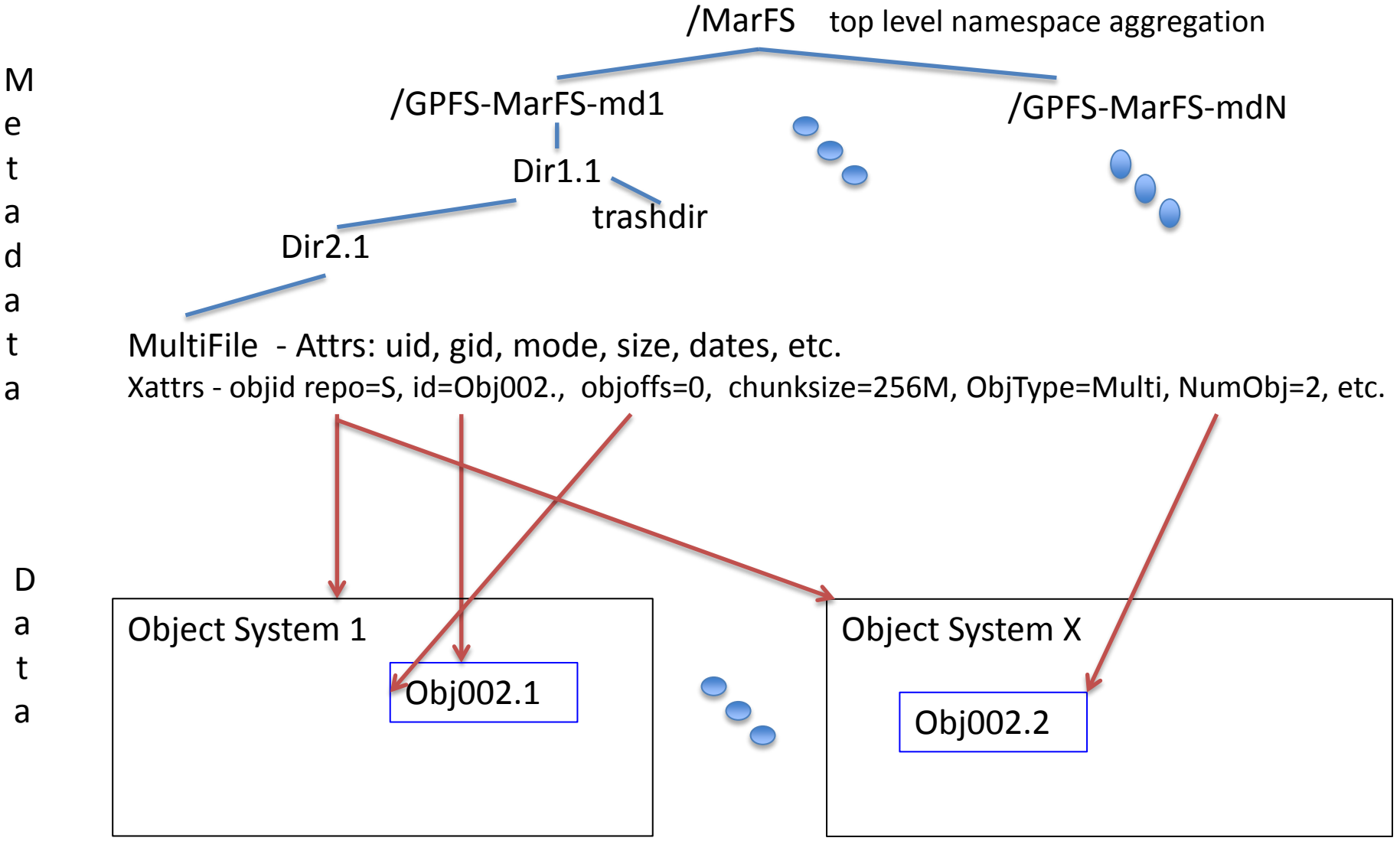
What it is not!

- Doesn’t allow update file in place for object data repo’s (no seeking around and writing – it isnt a parallel file system)
- FUSE
  - Does not check for or protect against multiple writers into the same file (when writing into object repos), use batch copy utility or library to do this efficiently)
  - 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

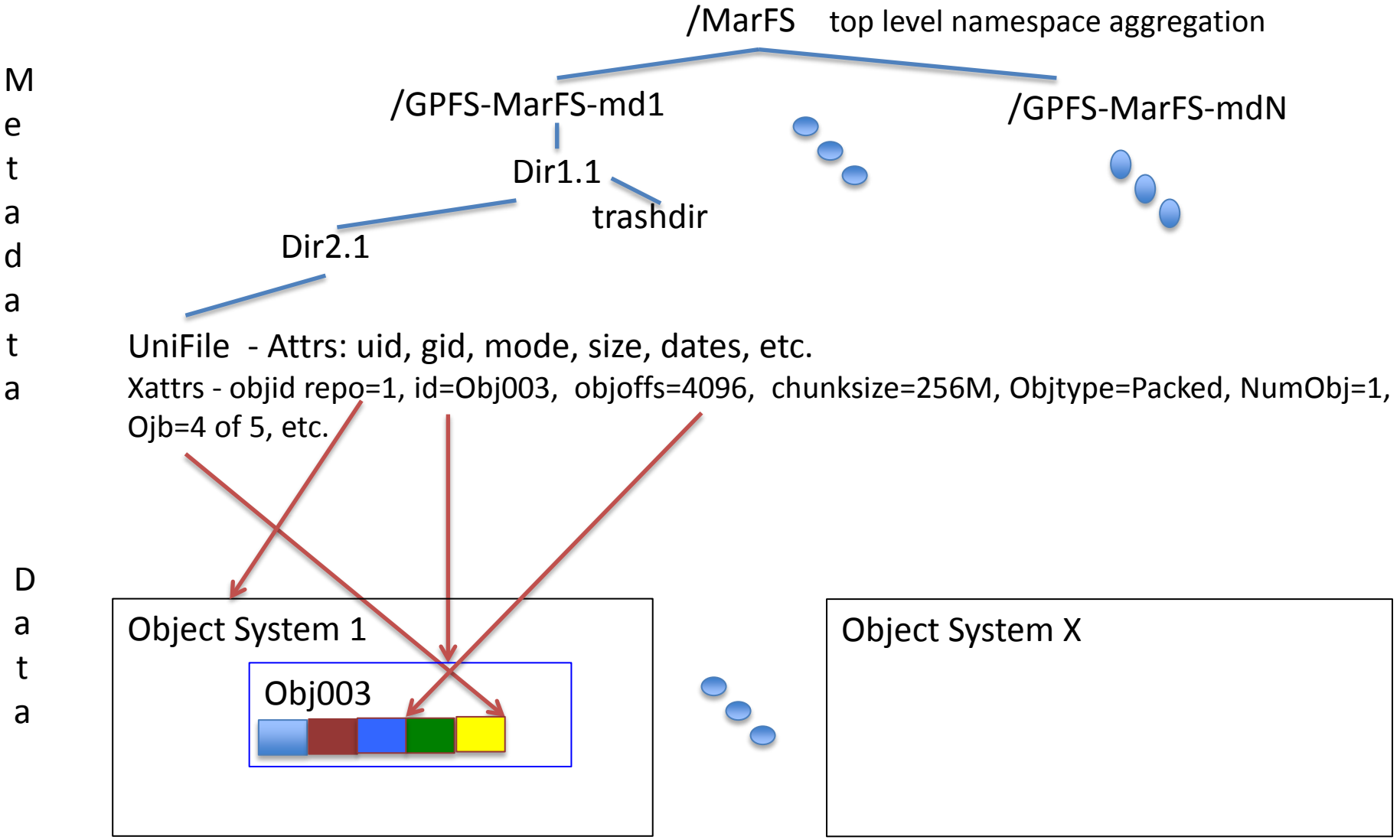
MarFS Internals Overview Multi-File



MarFS Internals Overview Multi-File
(striped Object Systems)



MarFS Internals Overview Packed-File



Configuration

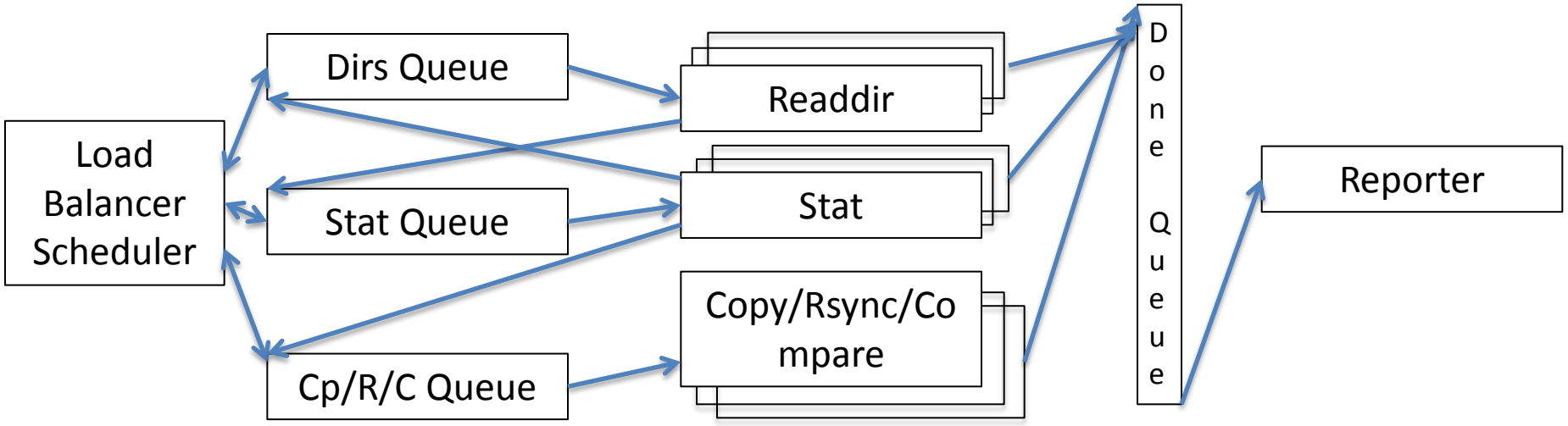
- Top level MarFS mountpoint
- Stanza for every name space/metadata file system you want to bring into MarFS
  - Describes how metadata is to be handled in this part of the MarFS system
  - Describes where data is to be put for files of various sizes/shapes (into which data repo)
- Stanza for every Repo - object system/area of object system, or other access method for data
  - Describes how data is to be stored into this data repo, chunksizes/methods/etc.

Recoverability

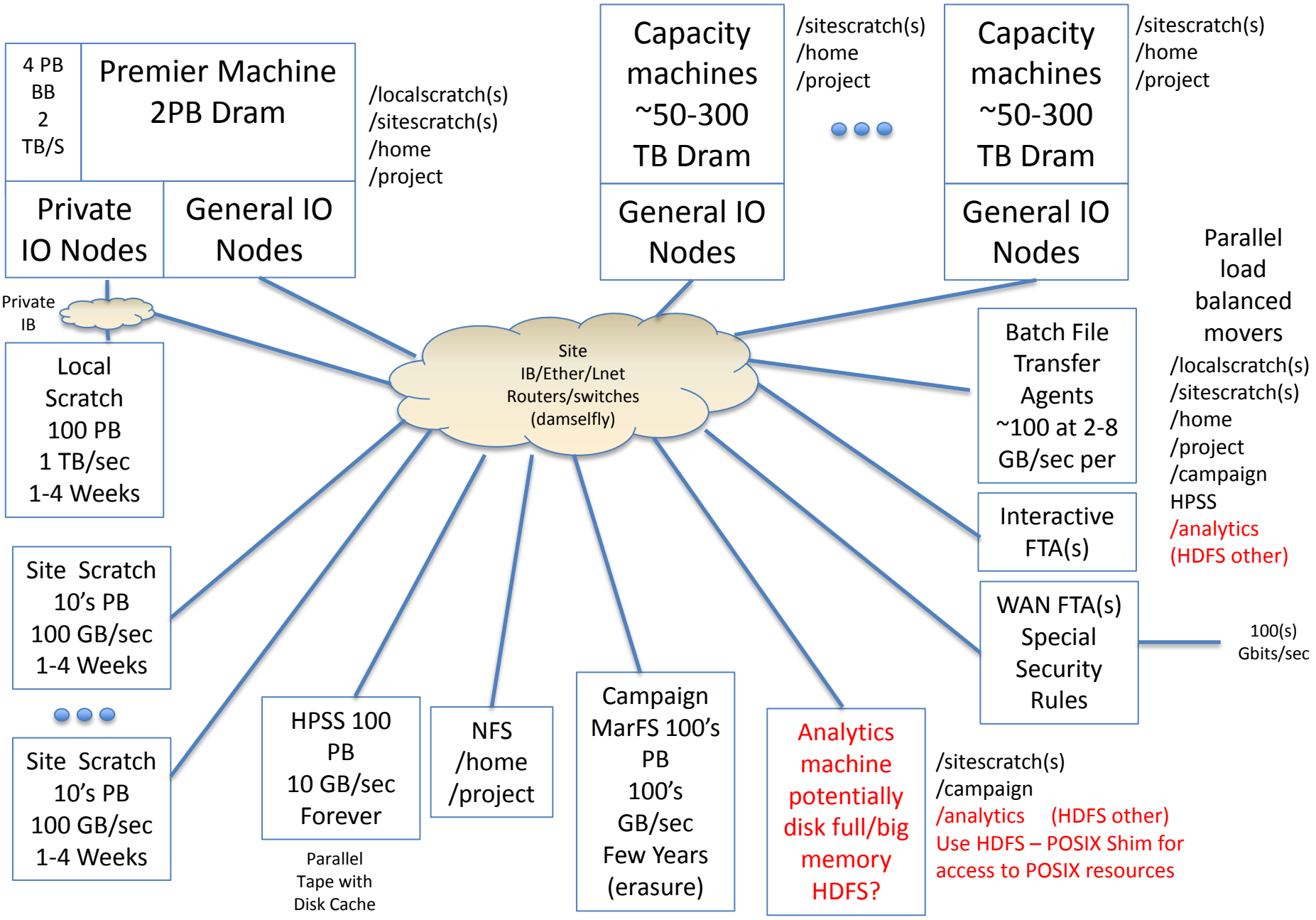
- Parallel backup of the metadata file system(s) backs up the metadata for MarFS
  - (METADATA ONLY)
- Objects are encoded with CREATE TIME info
  - (path,uidgid,mode,times, MarFS/other xattrs, etc.
- Parallel backup of object metadata
  - Object name has some recovery info in it, so just listing/saving the objects/buckets is useful
  - Any other info your object server will allow you to back up

Pftool

- A highly 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
  - Repackaging: breaks up big files, coalesces small files
  - To/From NFS/POSIX/parallel FS/MarFS



How does it fit into our environment (circa FY16) ?



Security Model

- All POSIX security is obeyed by MarFS
- Addition special security can be added by configuration to manage what parts of the name space allow metadata and data update/read
  - and you can control those special permissions for interactive and batch separately per name space.
    - rm – read metadata wm – write metadata rd – read data wd – write data ud – unlink data
    - Can lock down data read/write separately from metadata read/update
    - Value is not stored with the file, real time, fast way to control access.
- Object Security is provided by the following methods
  - Password for Object Server access is stashed safely, can be time based, crypto securely sent to Object Server on every request.
  - Encryption in the data path to objects can be turned on
  - Encryption at rest could be implemented and is on the futures list.
  - Protecting the trash is essential as well

Futures

- File data versioning – using data pointers in trash
- Dual copy/MarFS erasure to enable erasure on erasure
- Metadata update logging, investigate how this might be done and what the cost is
- Compression and Encrypton in MarFS (for repos that don’t compress)
- Offline optimizations/sorting/indexing of attrs and user xattrs etc.
- Append or sparse support, need to consider carefully, hard to do because of book keeping
- Other access methods HPSS, Globus, other.
- HDFS alternate access of same data, via java hdfs lib
- Offline deep reconcile/repack – if trash is lost
- Semi-direct – store data into parallel file system file(s) for Globus or other parallel N to 1 write staging