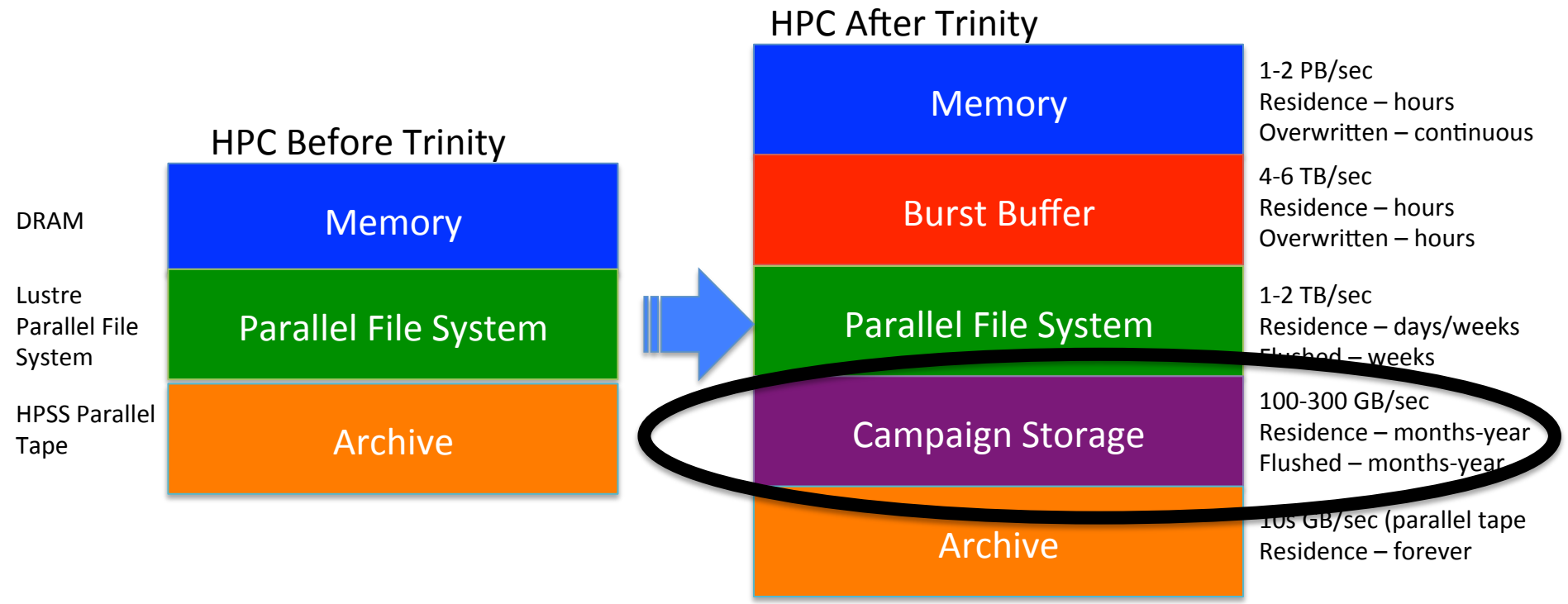
MarFS

A Near-POSIX File System using Scale Out Commercial/Cloud for Data and Many POSIX File Systems for Metadata (SPOOFS)

Gary Grider, LANL

LA-UR-15-24159
05/2015

Campaign Storage?
Scalable POSIX on Object



What is it?

- It provides a Near-POSIX global name space over many POSIX and non POSIX data repositories (Scalable object systems - CDMI, S3, etc.)
- It provides scalability of 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 should scale just like many scalable object systems
- Metadata should scale like N POSIX name spaces both across the tree and within a single directory (like Lustre D&E 1+2)
- It is friendly to object systems by
  - Spreading very large files across many objects
  - Packing many small files into one large data object

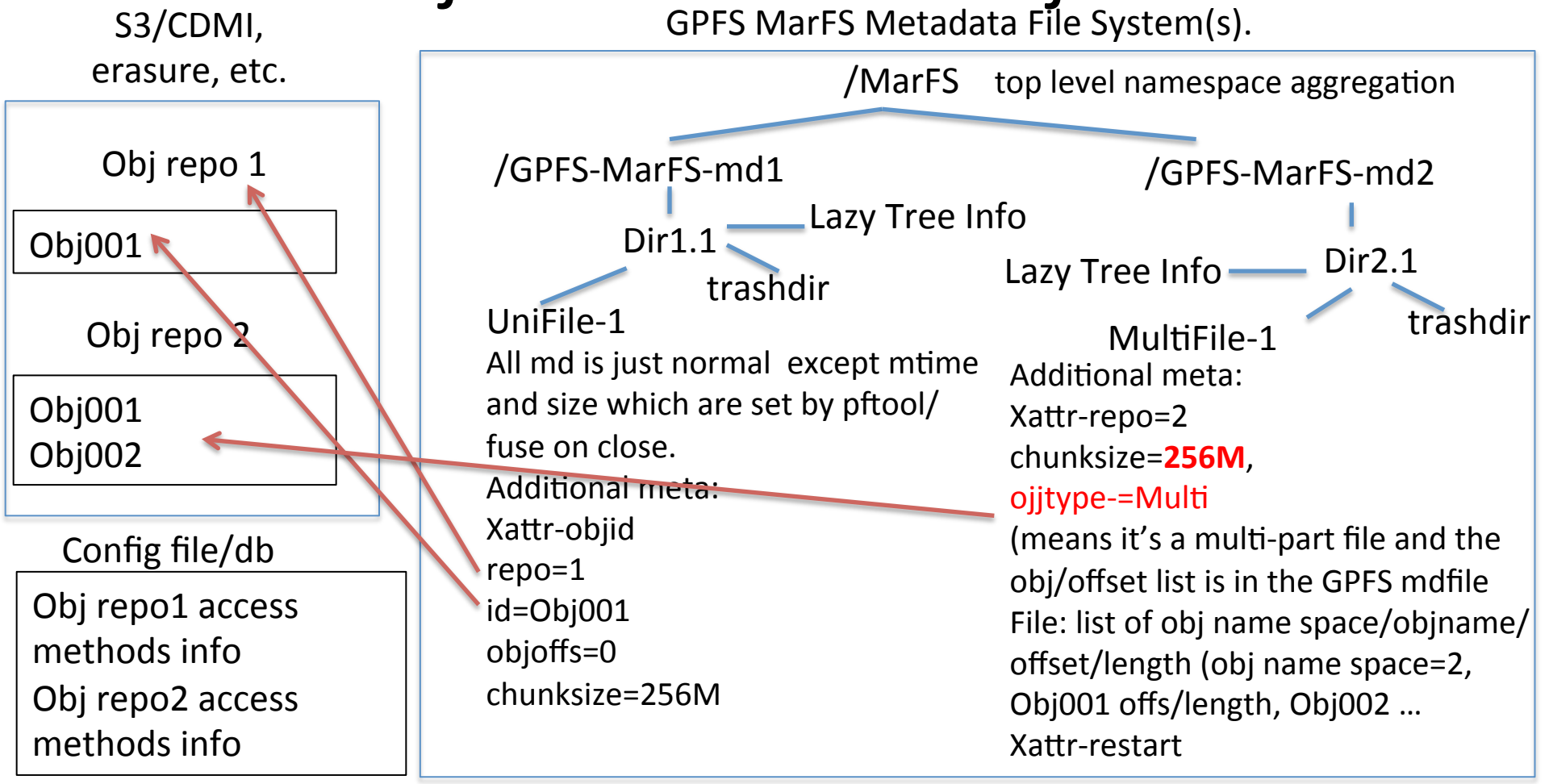
What it is not!

- Doesn't allow update file in place for object data repo's
- Currently scales across multiple metadata file systems (Lustre D&E 1) and within a single directory (Lustre D&E 2) by winter
- FUSE daemon does not check for or protect against multiple writers into the same file, the proper way to do that is to use the batch utility or library. The intended use of the FUSE is interactive metadata and read mostly use. Writing from FUSE is possible but FUSE will not create data objects that are packed or of a friendly size for Object systems. It will be possible to use batch utilities to reformat FUSE written files to be friendly to the object systems.

Why do it/Isn't it hard to do?

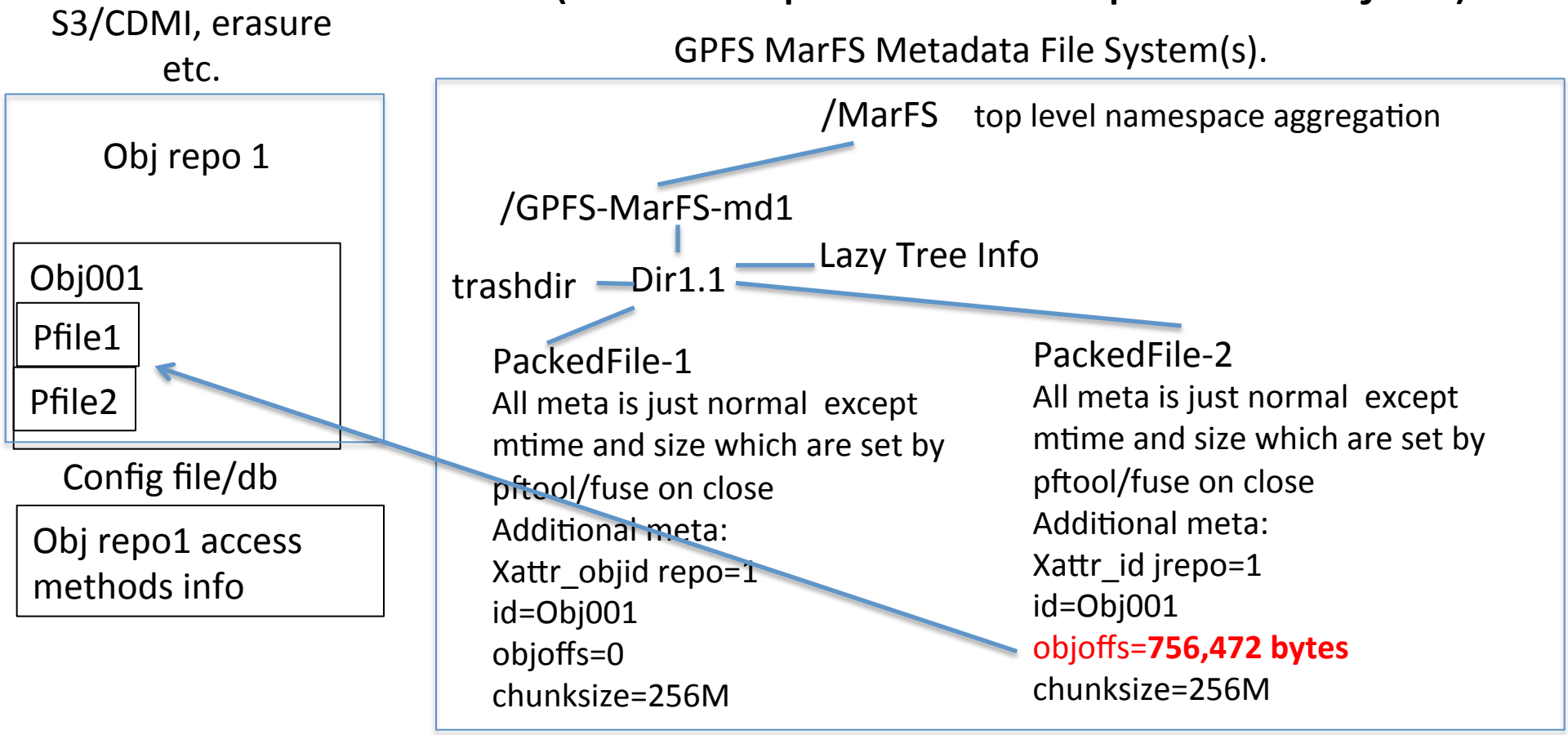
- The need
  - Objects Systems provide massive scaling with convenient and efficient erasure techniques but their use is friendly only to applications, not to people. People need folders and many of the powers that POSIX metadata gives them. Leveraging the best of POSIX namespace mgmt in a scalable way over the best of Object Systems has huge economic appeal. Update in place is one thing we can learn to live without though. We would like to have an open source solution in this space to set the bar high for more integrated solutions to beat someday.
- The challenges
  - There are many challenges to provide this capability including the mismatch of POSIX and Object metadata, security, update in place semantics, and efficient object sizes. Further, the HPC need includes billions of files in a single directory and single files that are even zetabytes in size, so Scale is a huge factor.

Uni Object and Multi Object File



Directories are just GPFS dirs, permissions are just permissions, (so all md ops just work (link, chown, chmod, etc.)
Links/etc. are just links, xattr restart is a way to know that the file was being written, the file size can be updated lazily
Unifiles have no GPFS mdfile data, just metadata, multipart files have GPFS file data, unlinking files moves them to trashdir for reclaim in batch
Lazy Tree Info is populated by batch processes lazily (using ilm) holds total bytes stored, quota, and any other access assists like how big dirs are, even an index of dirs/files etc.
Think large objects like many Megabytes so the object metadata for the object server is manageable.

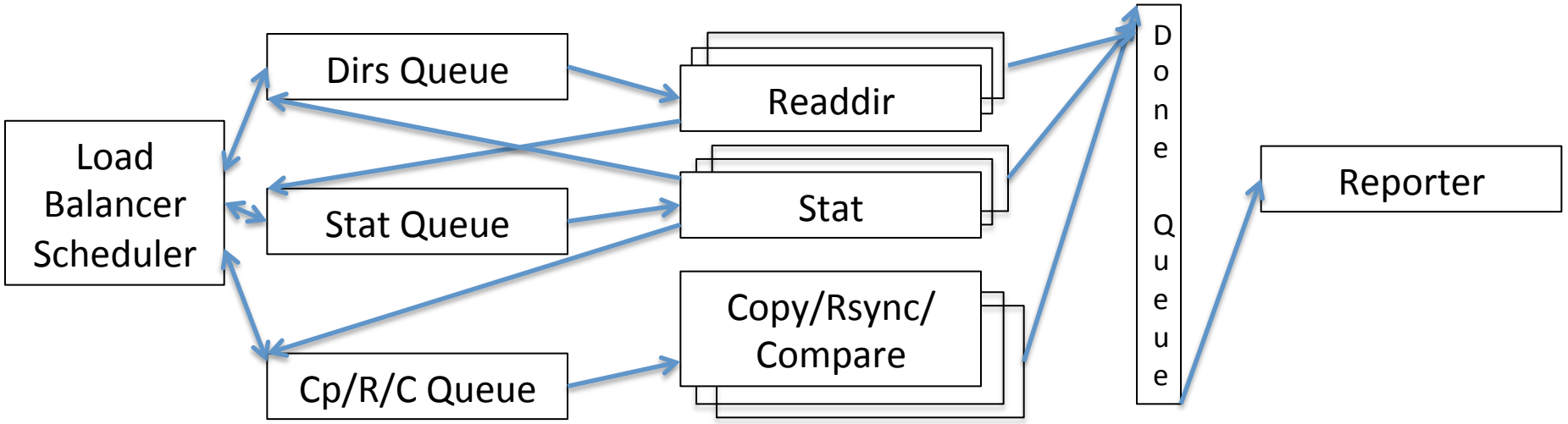
Packed File (same as previous but packed object)



When files are overwritten (they have to be completely over written, no update in place, enforced by pftool and fuse), trunc'd/unlink'd files are moved to trash and clean up can repack and get space back in batch, not done interactively. Packed objects can get trash in then as well.

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



MarFS Requirements

- Linux system(s) with C/C++ and FUSE support
- MPI for parallel communication in Pftool (a parallel data transfer tool, see https://github.com/pftool/pftool). Thus, most any MPI library with a C interface can be used.
  - Communications with the MPI library can utilize many communications methods like TCP/IP, Infiniband OFED, etc.
- All servers need to see all metadata file systems and all object server system directly or via forwarding