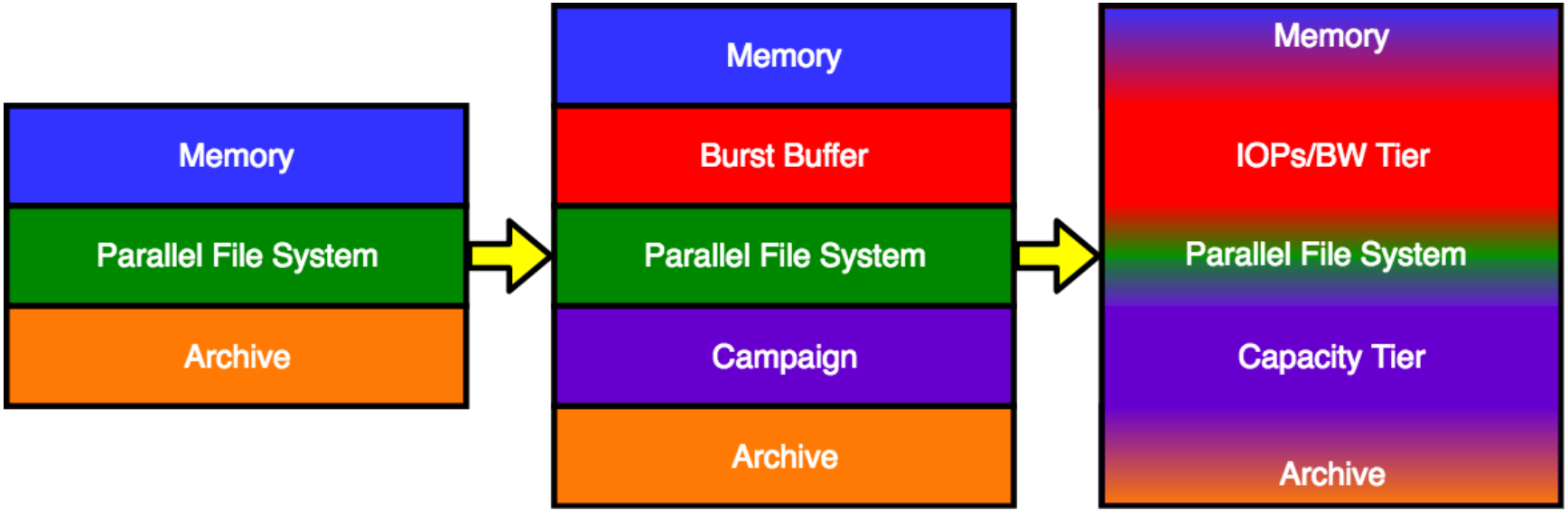
MarFS and Multi-Tier Erasure

Garrett Ransom (HPC-SYS)
Gary Grider (HPC)

05/15/18

Operated by Los Alamos National Security, LLC for the U.S. Department of Energy's NNSA

HPC Storage Trends at LANL



HPC Storage Trends at LANL – Capacity Tier

- **Data sets are growing faster than long-term storage can support**
  - Trinity: 2PB of memory / 4PB of flash
  - Crossroads: (maybe) 4PB mem / 10-100PB flash
  - HPSS Archive ~60PB Total, near capacity
- **Bandwidth of archive is a limiting factor**
  - Usable bandwidth of traditional archive (HPSS) ingest is roughly a few GB/sec
  - HPSS ingests data much faster than it recalls
  - Storing petabytes of job checkpoints is infeasible

Implementing a Capacity Tier

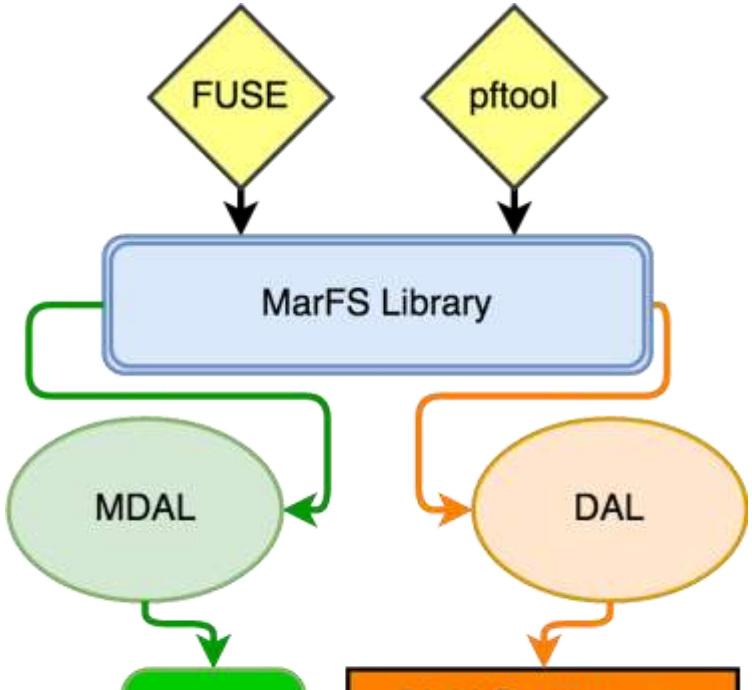
- **Tape is likely not the approach to take**
  - Tape is effective for truly cold data, not data sets that require more recall
  - Designing tape storage solutions is complex
- **Object Storage seems promising**
  - Flat namespace allows easy scalability
  - Erasure coding allows for cheaper disk media
- **Object Storage has limitations**
  - Machines love object-IDs, people generally don’t
  - Potentially billions $ in applications expecting ‘POSIX-like’ file trees

What is MarFS?

- **A near-POSIX interface layered over distinct metadata and data implementations**
  - Data stored as erasure coded objects
  - Metadata mirrored within a parallel file system (GPFS)
  - Object IDs stored as extended attributes of metadata files
- **Familiar semantics, fast metadata, stable objects**
  - Storing metadata to one or multiple POSIX file system gives us POSIX-style directory trees and permissions almost for free
  - Storing data as objects simplifies implementation and data protection
- **Highly leveraged on commercial or mature open products (for file metadata, for data storage, even for data movement) not much code!**
- **With tradeoffs, of course**
  - no update in place or file locking
  - restricted interactive use

The Structure of MarFS

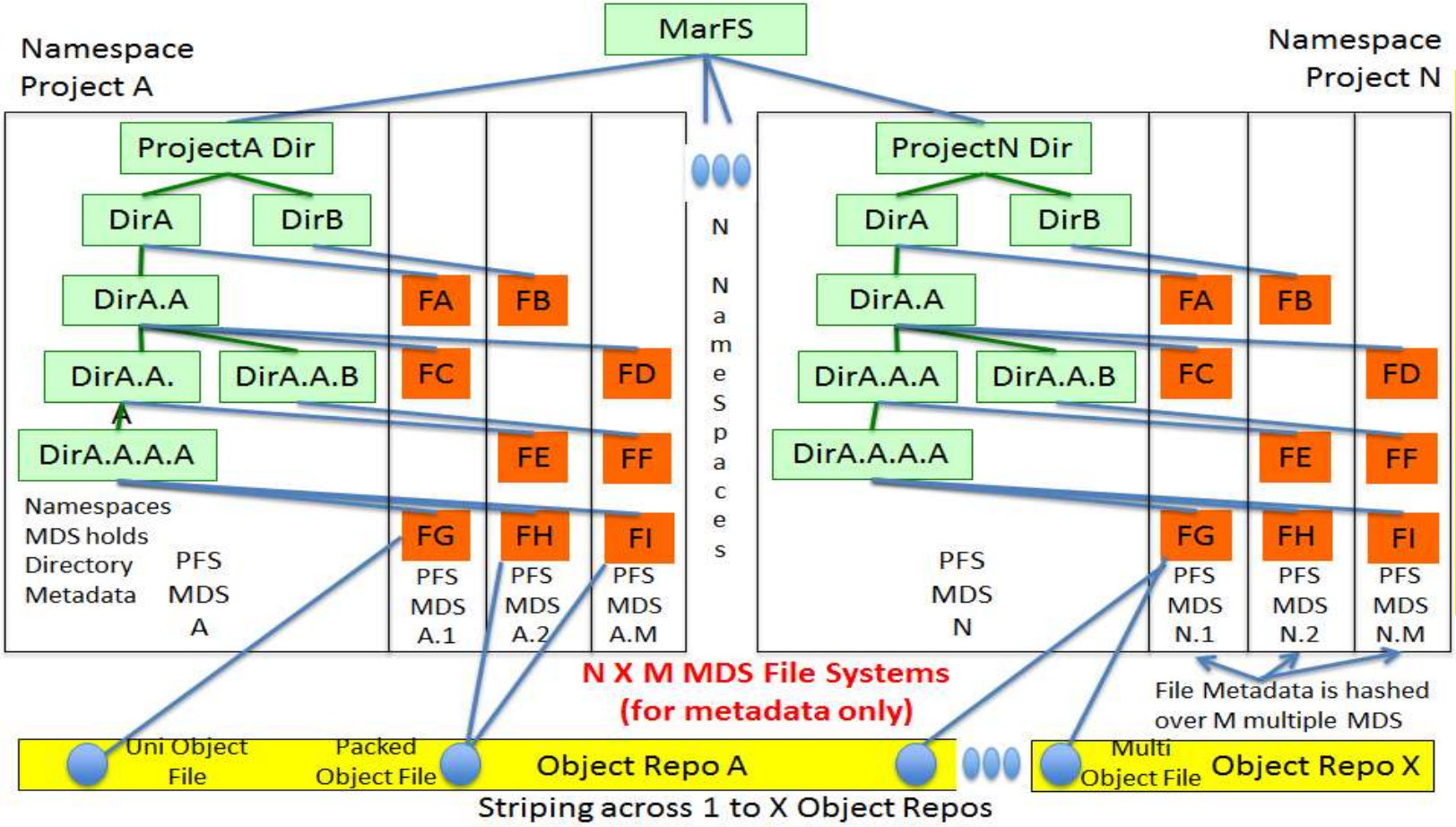
- **Pftool**
  - Parallel MPI file transfer utility
- **FUSE mount**
  - Provides user view of metadata
- **MarFS Library**
  - Heart of the software infrastructure
- **DAL/MDAL**
  - Abstraction layers atop data and metadata respectively
  - Allow easy swapping of underlying storage



Metadata

- **Metadata is just stored on multiple POSIX file systems that are visible to all ”mounters – batch and interactive data transfer agents”.**
- **Object info on objects that make up files are stored in extended attributes in the file metadata.**
- **Small files are packed into large 100MB-gigabyte size objects. Medium files are one file per object, and large files are chunked across multiple objects**
- **Lazy Quota is used**
- **All file deletes are trash canned**
- **Transfer in/out status is logged in metadata so re-startable transfers are possible**

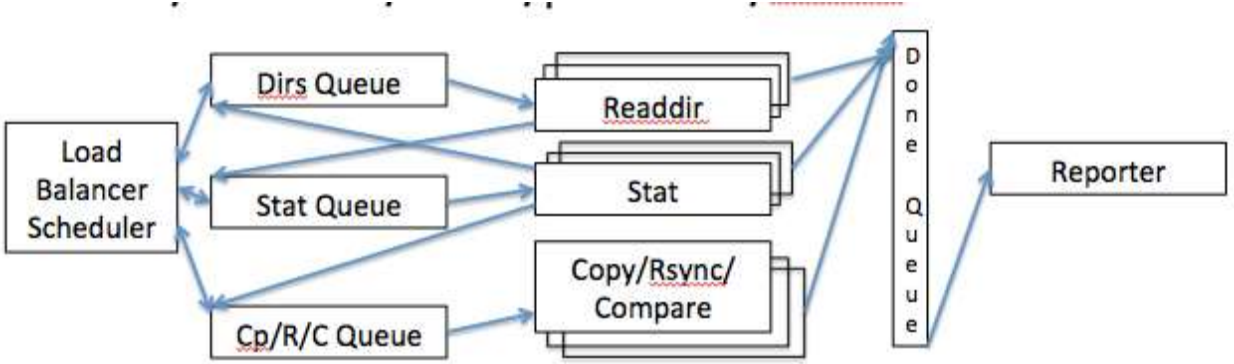
An example of metadata scaling: MarFS Scaling Test



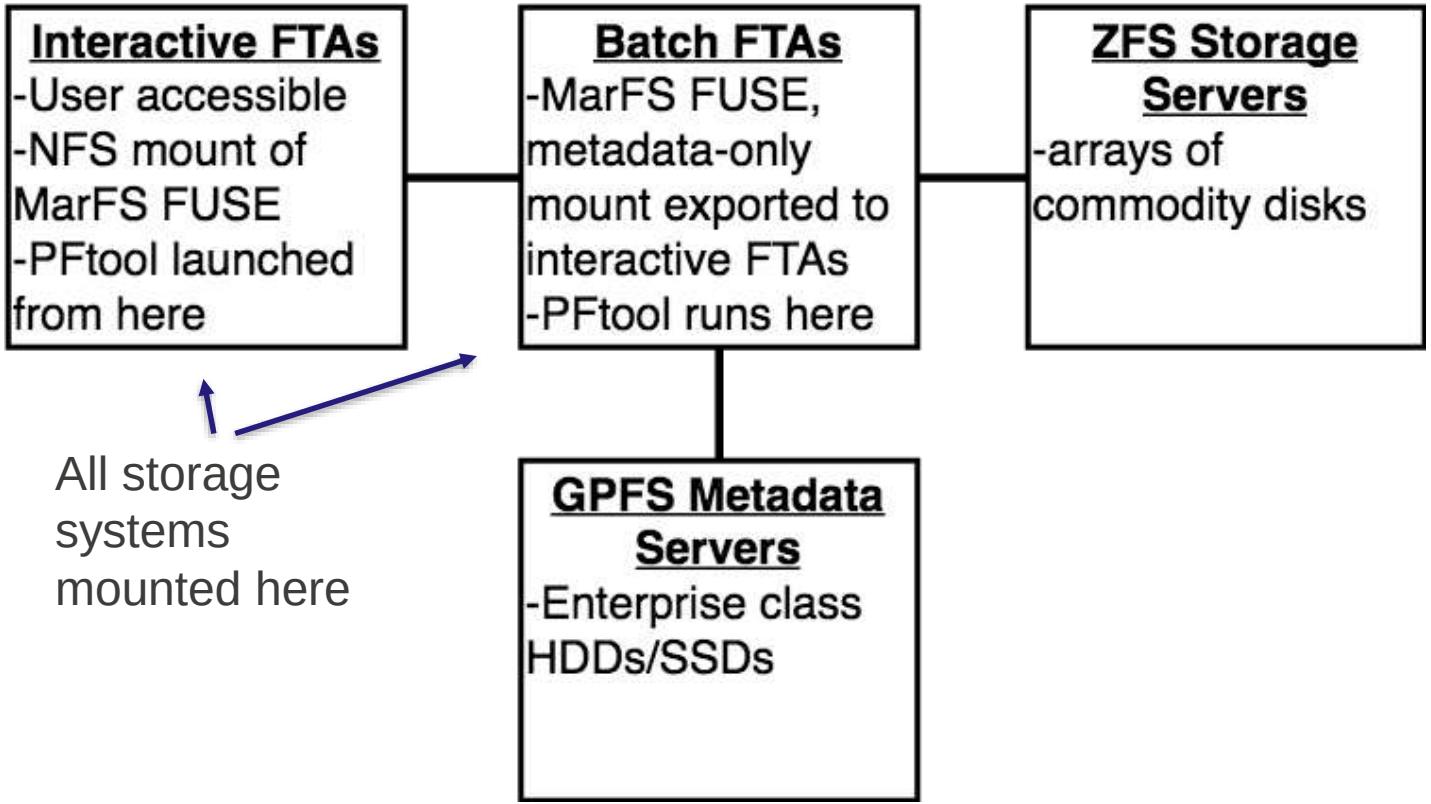
Scaling test on our retired Cielo machine:
835M File Inserts/sec
Stat single file < 1 millisecond
> 1 trillion files in the same directory

Data movement - PFtool

- Parallel data copy, compare, etc. done by Pftool
- Open source mature parallel data movement package
- Load balance readdir, stat, and copy/compare
- Packs small files and chunks large ones
- Re-startable for lots of small files and even for one very large one.



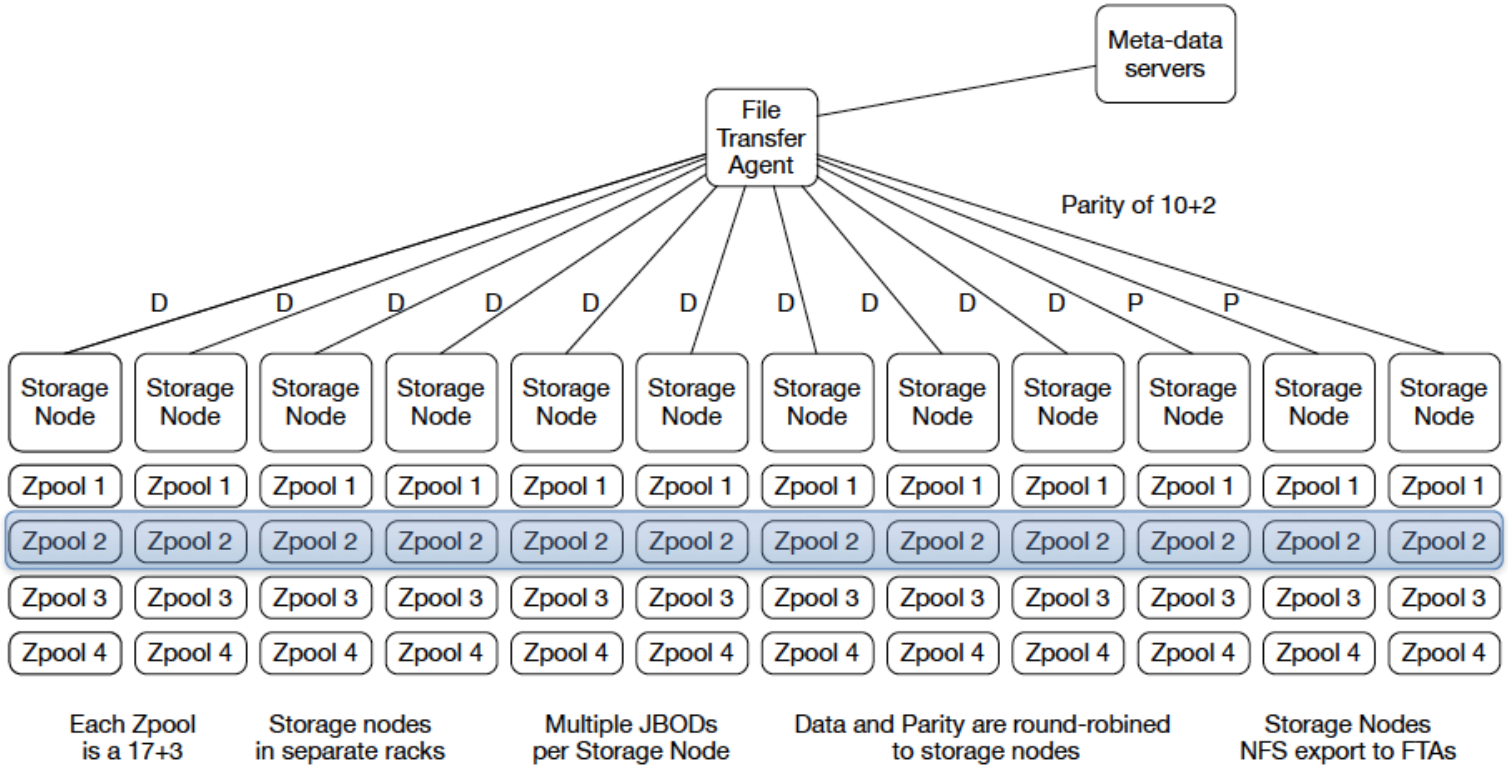
Our Deployment



Multi-Component

- **Current data storage solution for Campaign**
  - Integrated via the MarFS DAL
  - Stores data as pseudo-objects
- **Cross-server erasure coding atop ZFS pools**
  - Allows failure tolerance at both the disk and server level
  - More reliability allows the use of cheaper disk
  - Erasure coding performed through Intel’s Storage Acceleration Library (isa-l)
- **Performance through parallelism**
  - Threaded I/O to multiple servers

Multi-Component

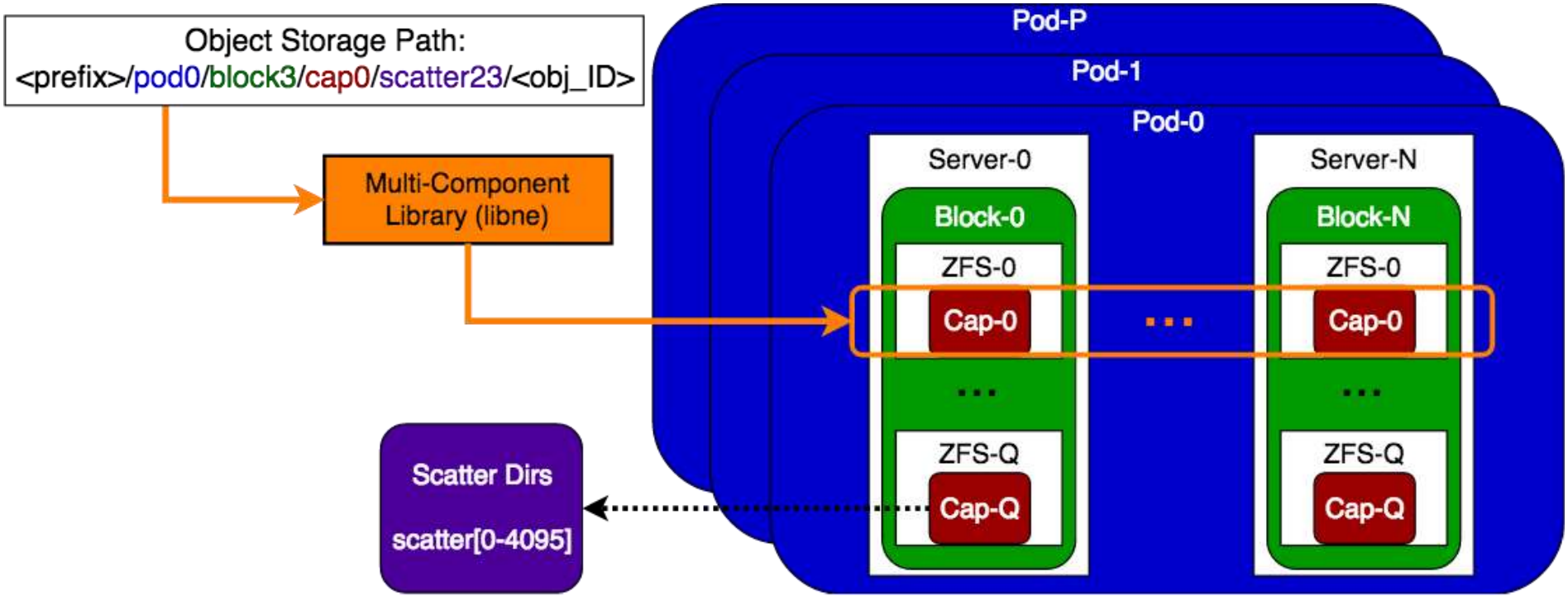


Thanks to Kyle Lamb, Dave Bonnie, and Jeff Inman

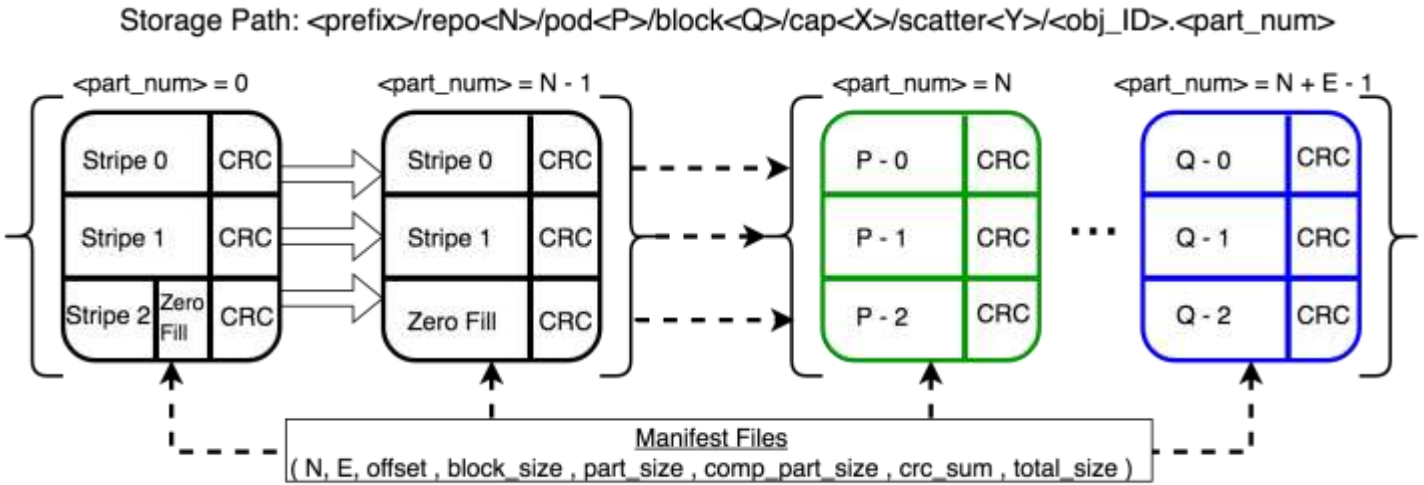
Multi-Component

- **Parallelism**
  - Objects hashed across all available servers
- **Transparency**
  - Object stripes simply stored as files in ZFS
- **Resiliency**
  - Data recoverable even after losing entire server racks

Multi-Component: Parallelism



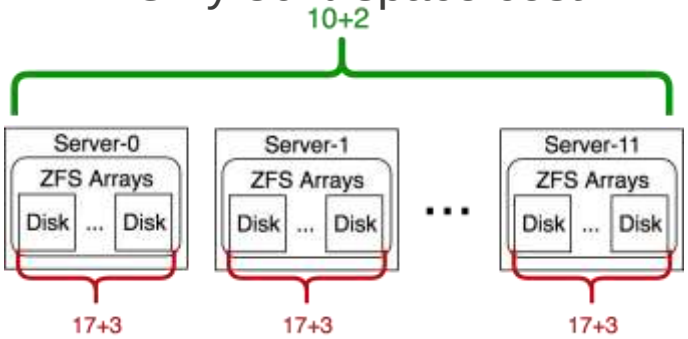
Multi-Component: Transparency



- **Storing to ZFS systems means that objects are plainly visible to administrators**
  - Each object ‘part’ is paired with a manifest file, providing data and erasure structure
  - Admins can literally ‘ls’ object parts and ‘cat’ manifest info
- **Utilities exist for interacting directly with objects**
  - Read/write data independent of the entire MarFS stack
  - Object integrity checks
  - Erasure rebuild of damaged objects

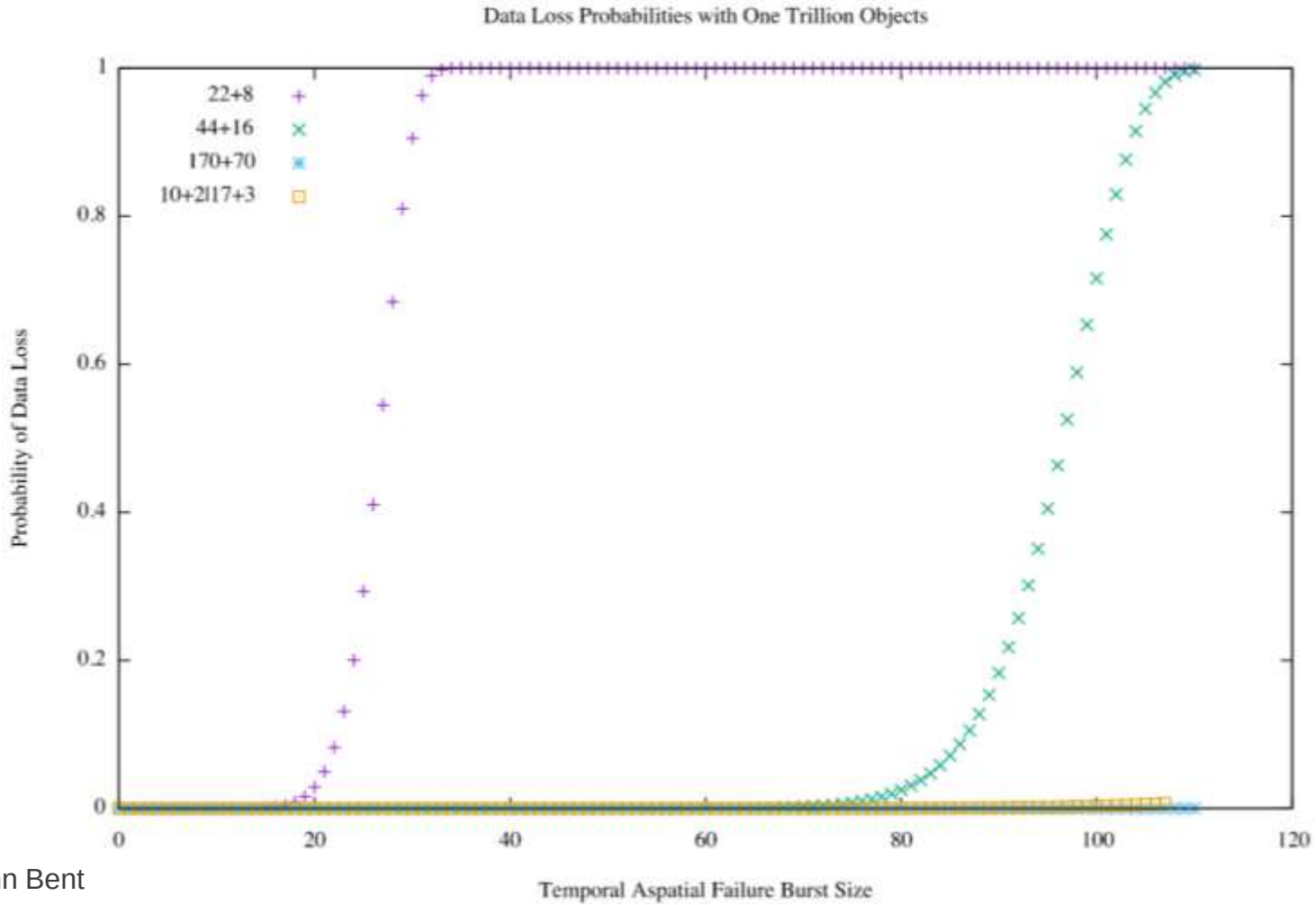
Multi-Component: Resiliency

- **Multi-tier erasure**
  - Multi-Component: **10+2** across servers
  - ZFS: **17+3** across disks
  - Tolerates min 11 and max 70 disk failures per stripe (set of 240 disks)
  - Only 30% space cost



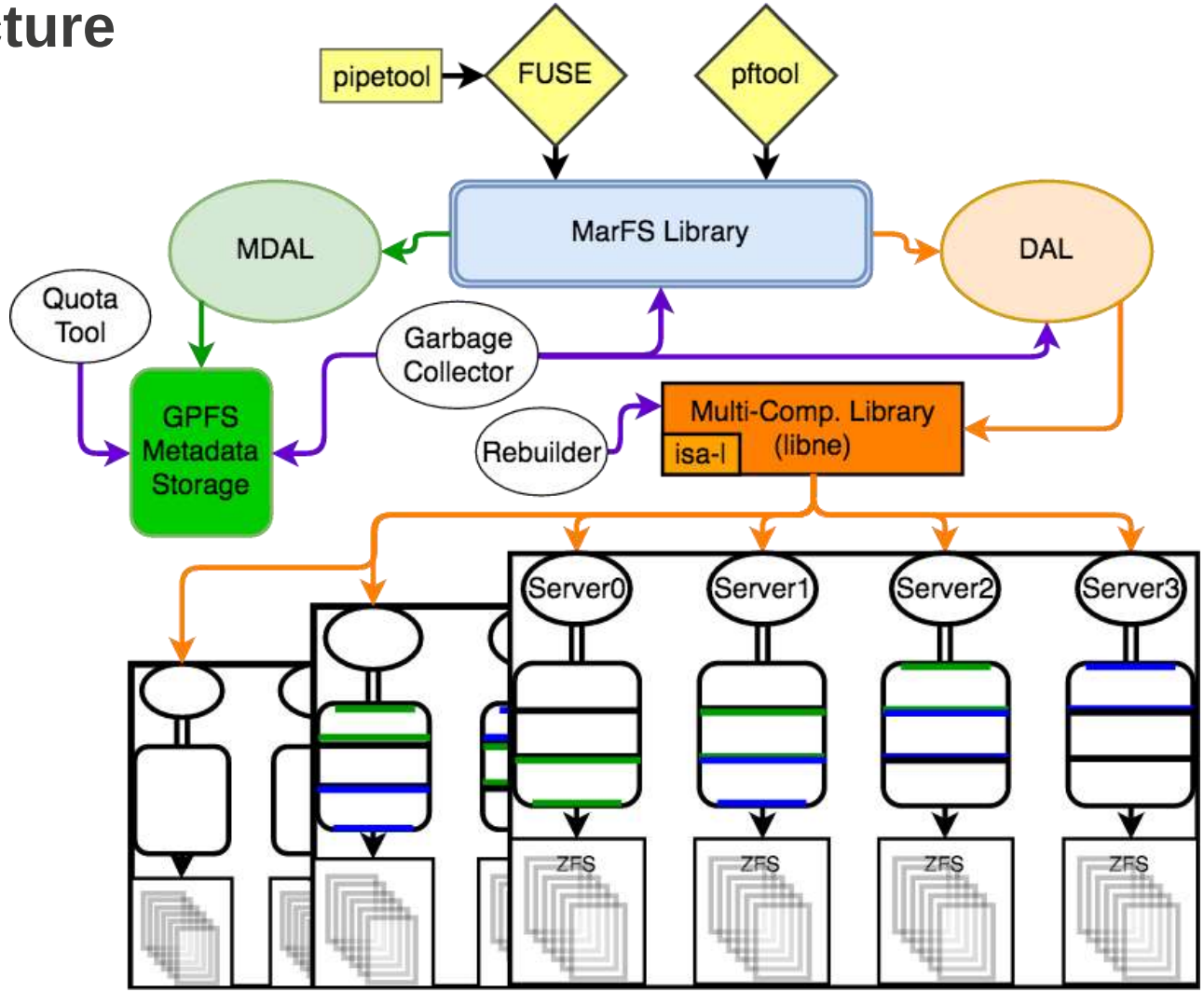
- **The Multi-Component Library (libne) automatically responds to read failures and makes use of erasure code if necessary**
- **'Degraded' writes are permitted up to a configurable safety threshold**
  - 10+2 can be written as 10+1 or 9+2 and later rebuilt to full width
- **Any degraded objects detected during read/write are logged**
  - A utility crawls these logs and calls into the Multi-Component Library to rebuild objects
  - The rebuilder utility can also be run against entire capacity-units to assess overall health or recover from catastrophic failures

Multi-Component: Resiliency



Thanks to John Bent

The Big Picture

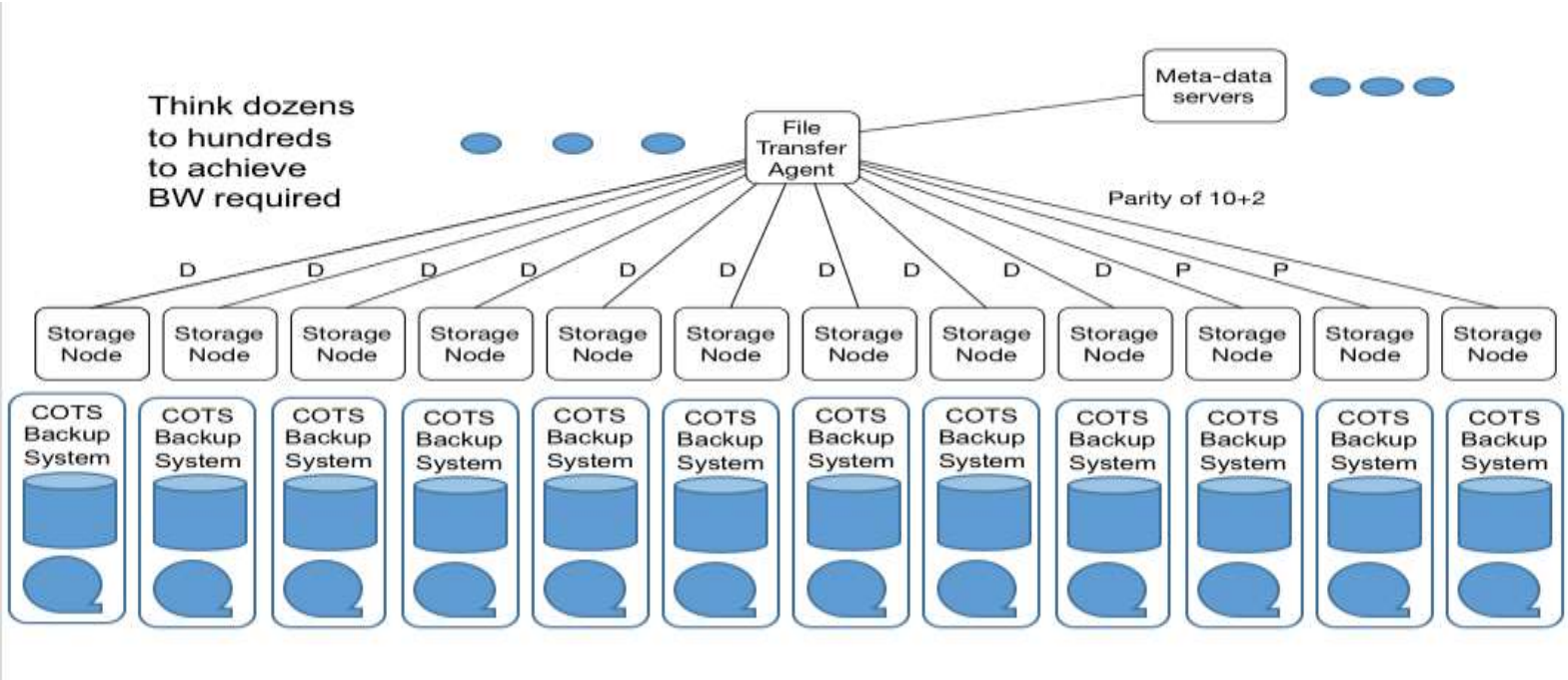


Current Status

- **MarFS Multi-Component has been in production use for two years now**
  - 5 interactive and 25 batch FTAs
  - 60PB of total storage
  - Roughly 25 GB/sec aggregate bandwidth for both read and write
  - NFS seems to be the bottleneck
- **Additional deployments in progress for this year**
- **Infiniband RDMA based Multi-Component currently in development**
  - Initial testing yielded 35 GB/sec with a fraction of the mpi ranks

We have a scalable metadata system and a parallel erasure system that works over distributed files, why not put the files on tape, lots of products put files on tape.

We get a Parallel RAIT enabled Archive for very little effort.



Github Organization – https://github.com/mar-file-system

- **MarFS – https://github.com/mar-file-system/marfs**
- **LibNE – https://github.com/mar-file-system/erasureUtils**

Pftool – https://github.com/pftool/pftool

Los Alamos
NATIONAL LABORATORY
EST.1943

Thank You

Delivering science and technology
to protect our nation
and promote world stability