Los Alamos
NATIONAL LABORATORY
EST. 1943

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

National Nuclear Security Administration

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

LA-UR-16-22883

MarFS

A Near-POSIX Namespace Leveraging Scalable Object Storage

David Bonnie

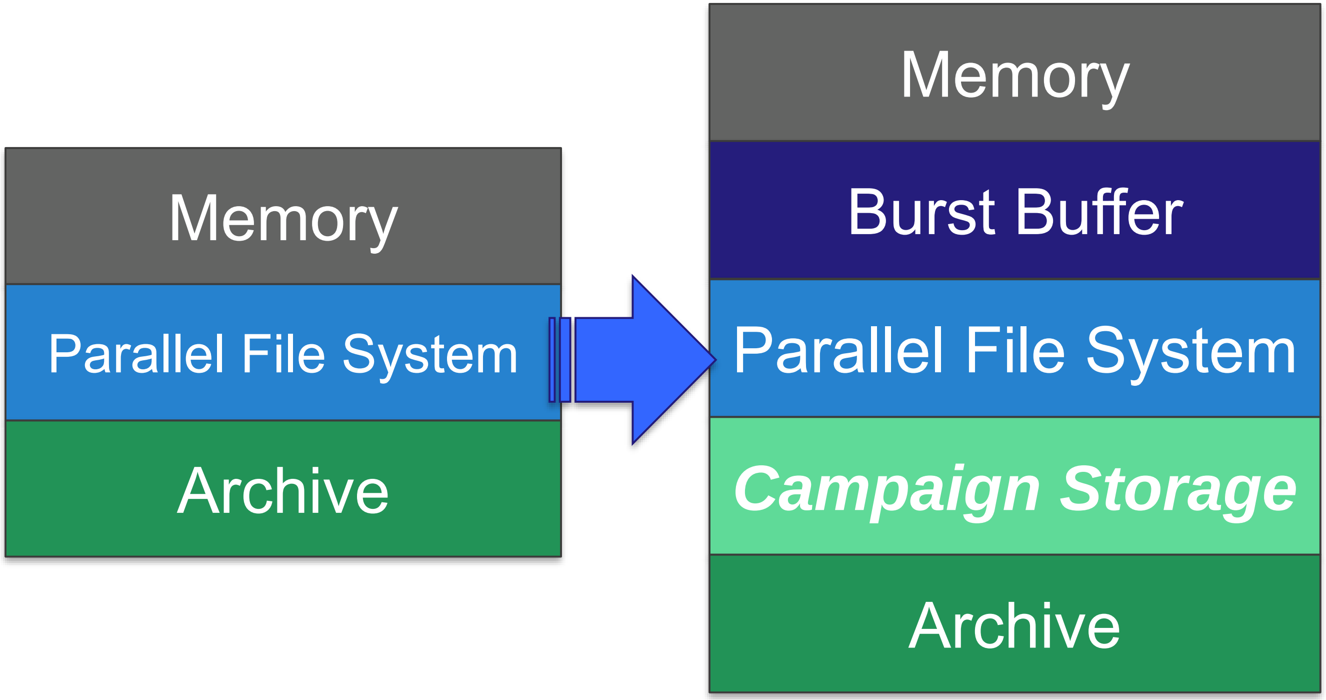
May 4th, 2016

Overview

- **What's the problem?**
- **What do we really need?**
- **Why existing solutions don't work**
- **Intro to MarFS**
  - What is it?
  - How does it work?
- **Status**
  - Current
  - Future

What's the problem?

- Campaign Storage (Trinity+ Version)



1-2 PB/sec
Residence – hours
Overwritten – continuous

4-6 TB/sec
Residence – hours
Overwritten – hours

1-2 TB/sec
Residence – days/weeks
Flushed – weeks

100-300 GB/sec
Residence – months-years
Flushed – months-years

10s GB/sec (parallel tape)
Residence – forever

What do we really need?

- Large capacity storage, long residency
- No real IOPs requirement for data access
- “Reasonable” bandwidth for streaming
- Metadata / tree / permissions management that’s easy for people and existing applications
- Do we need POSIX?

Why existing solutions don't work

- So we need a high capacity, reasonable bandwidth storage tier…
  - Parallel tape is hard and expensive
  - Object solutions?
  - Big POSIX expensive, $$$ hardware
- Existing solutions don't make the right compromises (for us)
  - Petabyte scale files, and bigger
  - Billions of "tiny" files
  - Try to maintain too much of POSIX, this leads to complicated schemes, too many compromises

So what is MarFS?

- MarFS is a melding of the parts of POSIX we (people) like with scale-out object storage technology
  - Object style storage is scalable, economical, safe (via erasure), with simple access methods
  - POSIX namespaces provide people usable storage
- Challenges:
  - Objects disallow update-in-place (efficiently)
  - Access semantics totally different (permissions, structure)
  - Namespace scaling to billions/trillions of files
  - Single files in the many petabyte+ range

So what is MarFS?

- What are we restricting?
  - No update in place, period
  - Writes only through data movement tools, not a full VFS interface
    - 100% serial writes possible through FUSE, pipes?
- What are we gaining (through the above)?
  - Nice workloads for the object storage layer
  - Full POSIX metadata read/write access, full data read access

So what is MarFS?

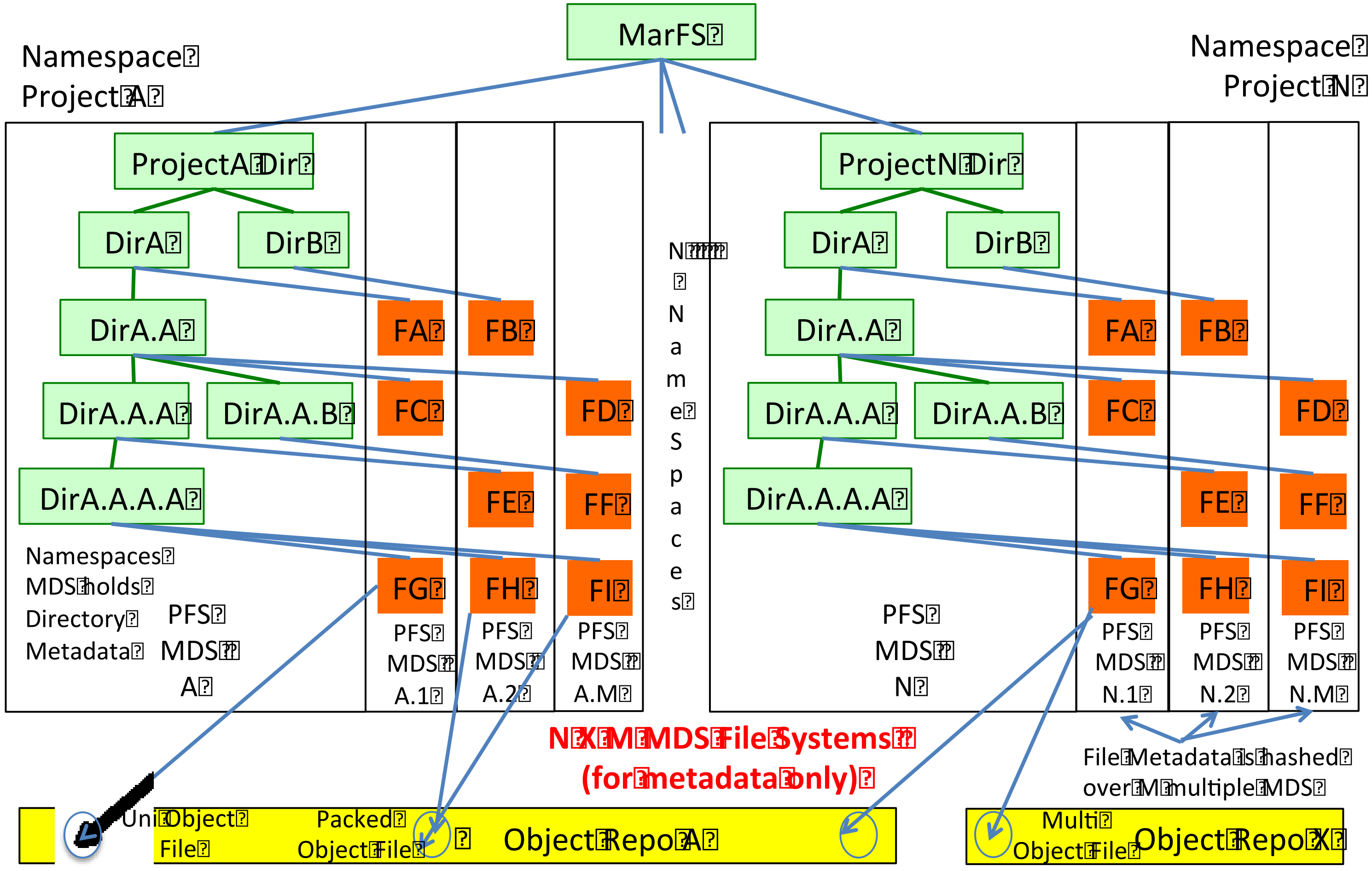
- Stack overview:
  - Smallish FUSE daemon for interactive metadata manipulation, viewing, etc
  - Parallel file movement tool (copy/sync/compare/list)
  - A handful of tools for data management (quotas, trash, packing)
  - Library that the above utilizes as a common access path
- Metadata stored in at *least* one global POSIX namespace
  - Utilizes standard permissions for security, xattr/sparse file support
- Data stored in at *least* one object/file storage system
  - Very small files packed, very large files split into "nice" size objects

Scaling basics

- So how does the namespace scale?
  - Up to N-way scaling for individual directories/trees
  - Up to M-way scaling *within* directories for file metadata
  - Directory MD is abstracted to allow for alternate storage (kvs)
  - We're using GPFS, lists are easy, so it's manageable!

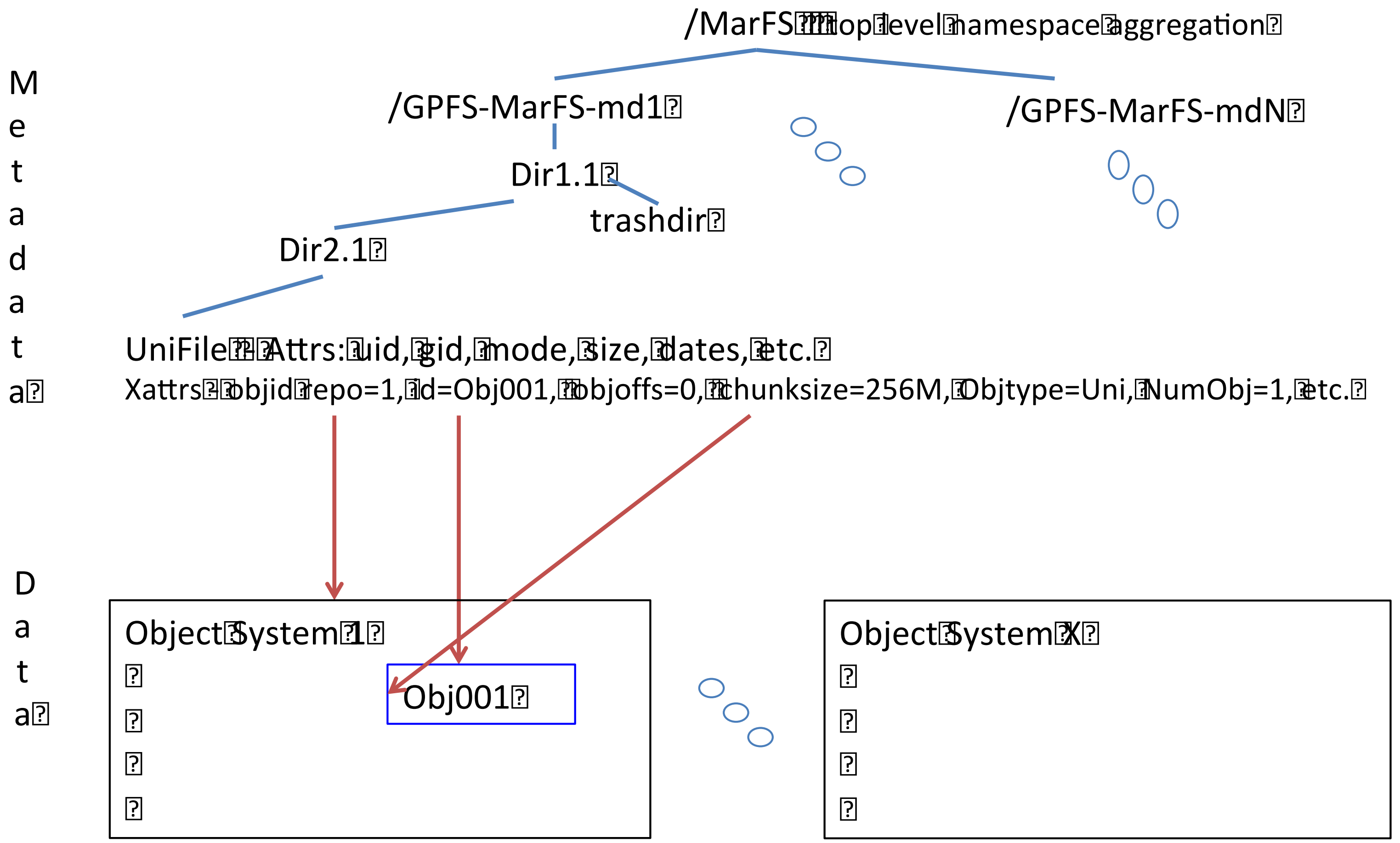
- How does the data movement scale?
  - No real limit on number of data storage repositories
  - New data can be striped within a repo and across repos
    - Repos can be scaled up and scaled out
    - New repos can be added at any time

MarFS Scaling

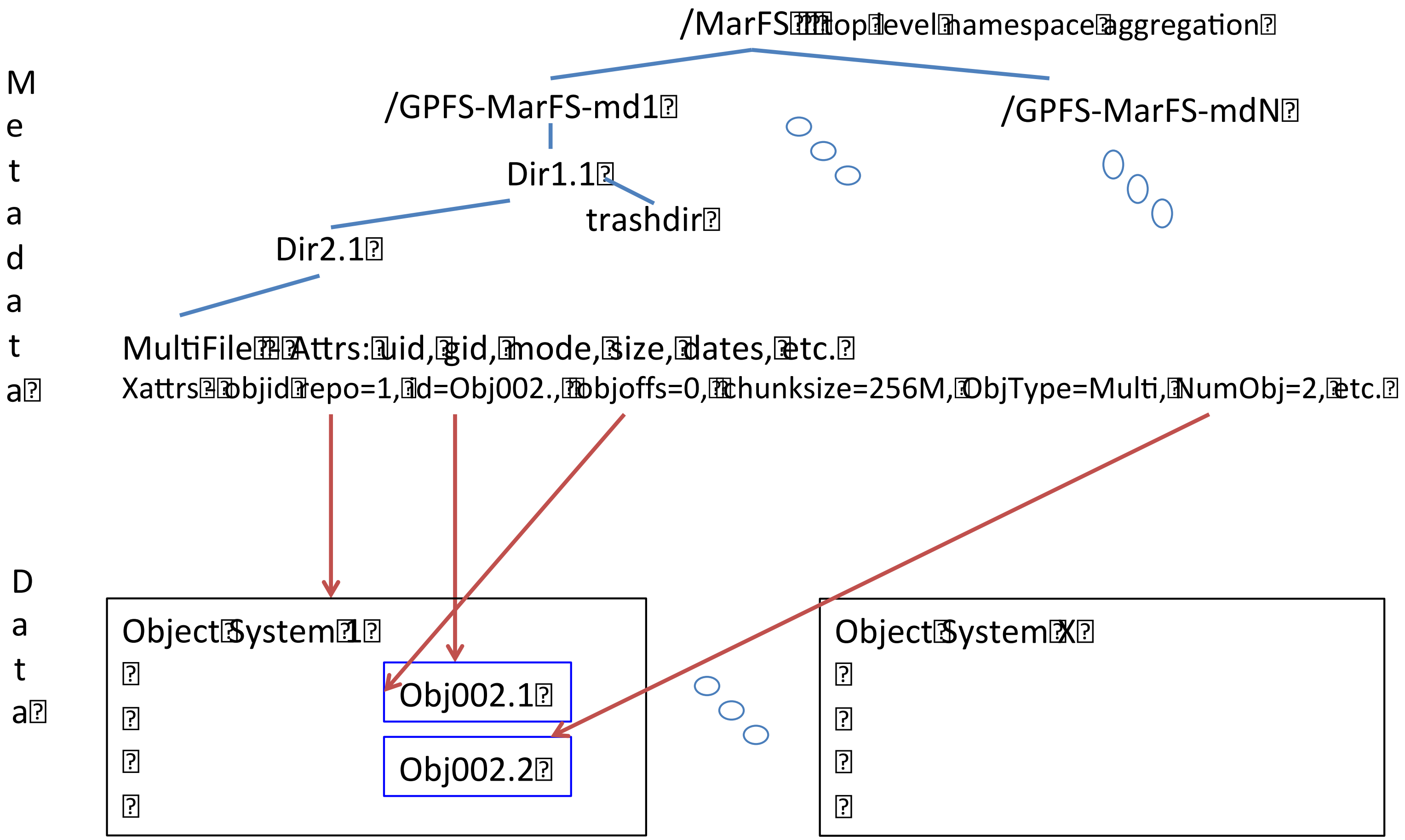


Striping across 1 to X Object Repos

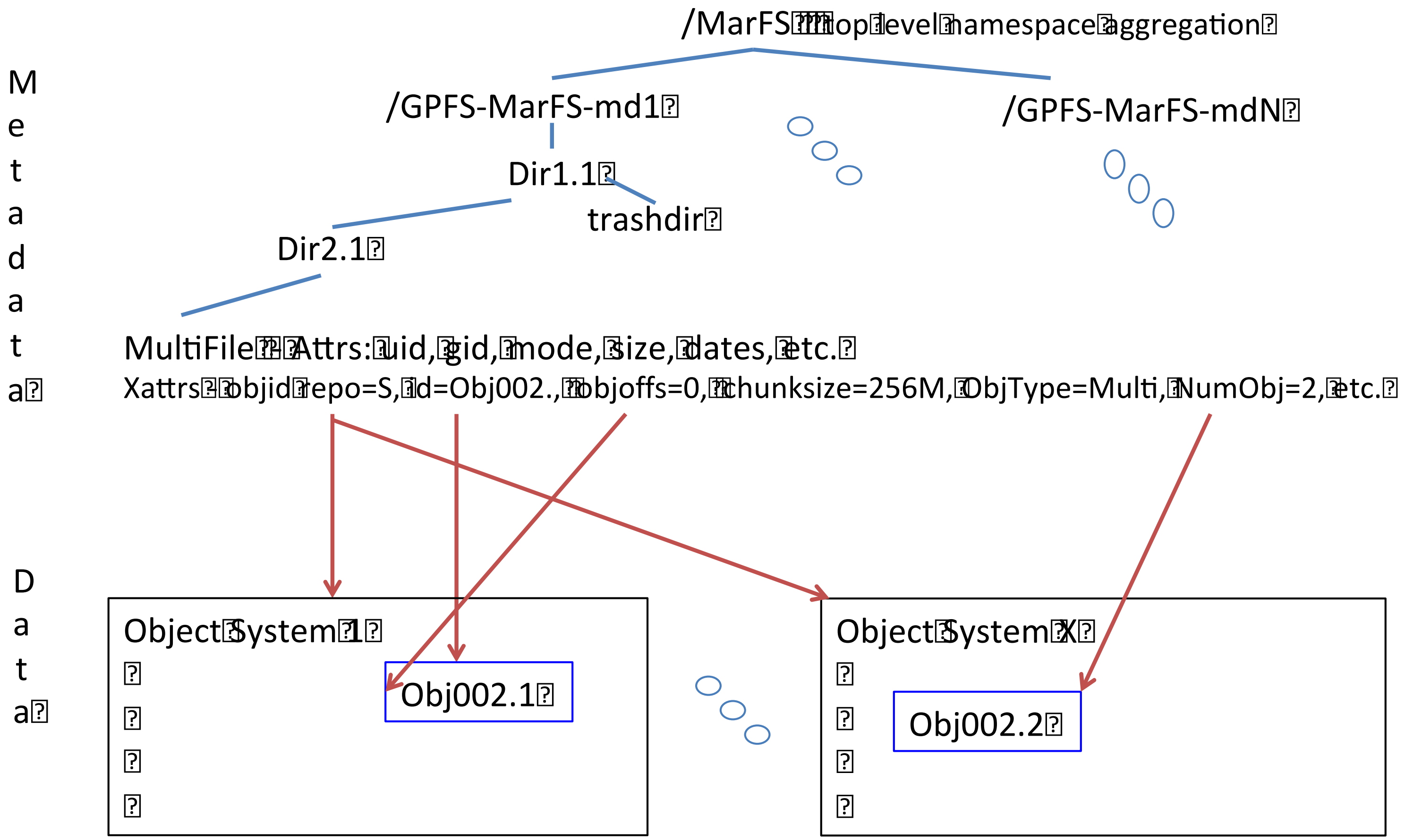
MarFS Internals Overview Uni-File


/MarFS top level namespace aggregation
/GPFS-MarFS-md1
/GPFS-MarFS-mdN
Dir1.1
trashdir
Dir2.1
Metadata
UniFile - Attrs: uid, gid, mode, size, dates, etc.
Xattrs - objid repo=1, id=Obj001, objoffs=0, chunksize=256M, Objtype=Uni, NumObj=1, etc.
Data
Object System 1
Obj001
Object System X

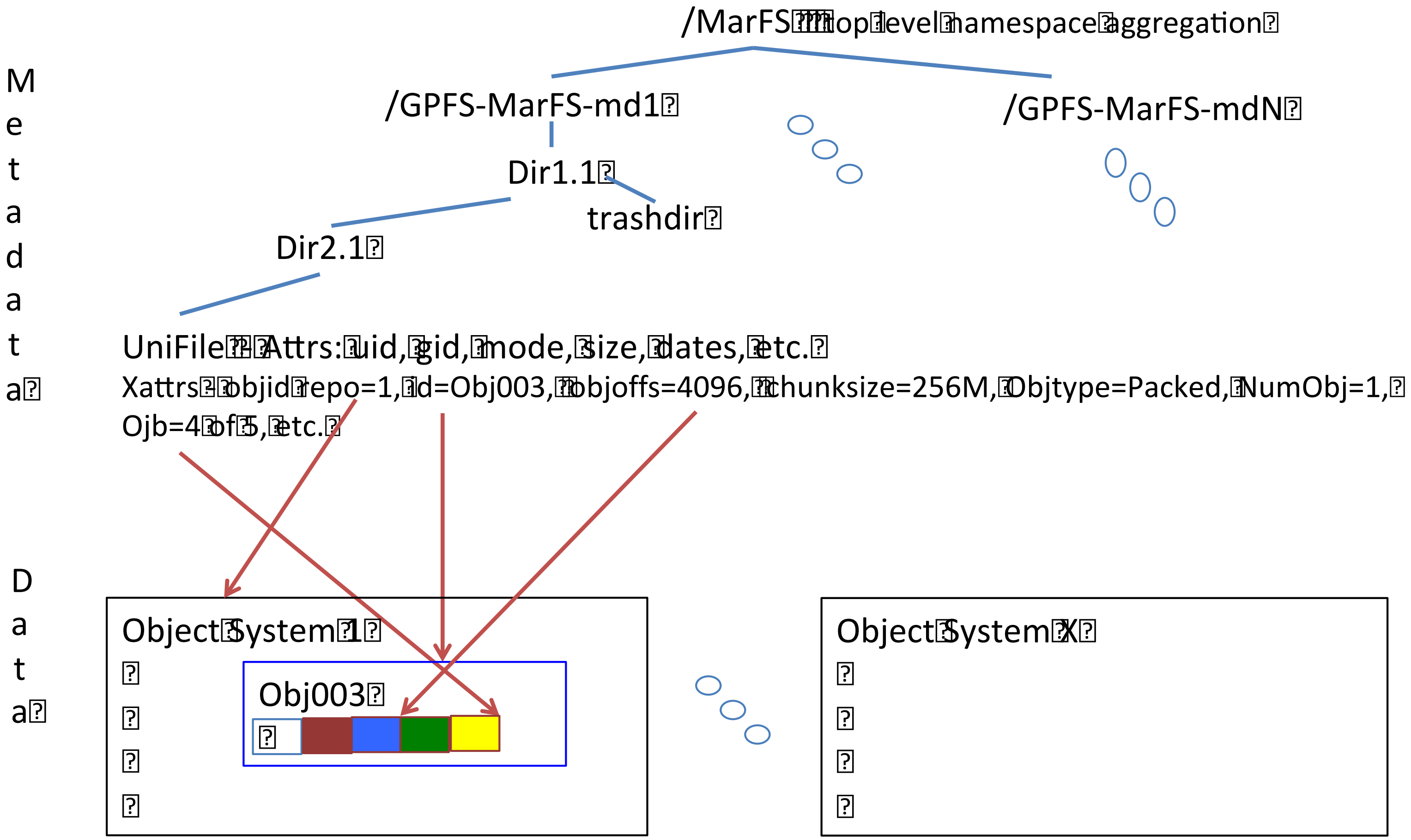
MarFS Internals Overview Multi-File


/MarFS top level namespace aggregation
M e t a d a t a
/GPFS-MarFS-md1
/GPFS-MarFS-mdN
Dir1.1
trashdir
Dir2.1
MultiFile - Attrs: uid, gid, mode, size, dates, etc.
Xattrs - objid repo=1, id=Obj002., objoffs=0, chunksize=256M, ObjType=Multi, NumObj=2, etc.
D a t a
Object System 1
Obj002.1
Obj002.2
Object System X

MarFS Internals Overview Multi-File (striped Object Systems)


/MarFS top level namespace aggregation
/GPFS-MarFS-md1
/GPFS-MarFS-mdN
Dir1.1
trashdir
Dir2.1
MultiFile - Attrs: uid, gid, mode, size, dates, etc.
Xattrs - objid repo=S, id=Obj002., objoffs=0, chunksize=256M, ObjType=Multi, NumObj=2, etc.
Metadata
Data
Object System 1
Obj002.1
Object System X
Obj002.2

MarFS Internals Overview Packed-File


/MarFS top level namespace aggregation
/GPFS-MarFS-md1
/GPFS-MarFS-mdN
Dir1.1
trashdir
Dir2.1
Metadata
UniFile - Attrs: uid, gid, mode, size, dates, etc.
Xattrs - objid repo=1, id=Obj003, objoffs=4096, chunksize=256M, Objtype=Packed, NumObj=1,
Ojb=4 of 5, etc.
Data
Object System 1
Obj003
Object System X

Current Status

- Where are we now?

- Open Science runs completed without real issue, ~4 PB scale system
  - ~2-3 GB/s bandwidth, utilized Scality RING storage
  - Discovered edge-case bugs with varied workloads
- Next system is currently being deployed, ~30 PB scale
  - ~28 GB/s bandwidth, also utilizing Scality RING storage

- Packing in parallel movement utility in progress

Future work

- Metadata in-directory scaling
  - Billion files in a directory…
- Compression / encryption within MarFS
- Data protection *in* MarFS – erasure on erasure, dual copy, etc
- Other access methods
  - HPSS, Globus, etc
- Migration tools (background movement)
- Dual copy would allow for DR opportunities on tape/offline media, tools to support this
- Alternate views of metadata (files within date, related project files, etc)

Learn more!

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

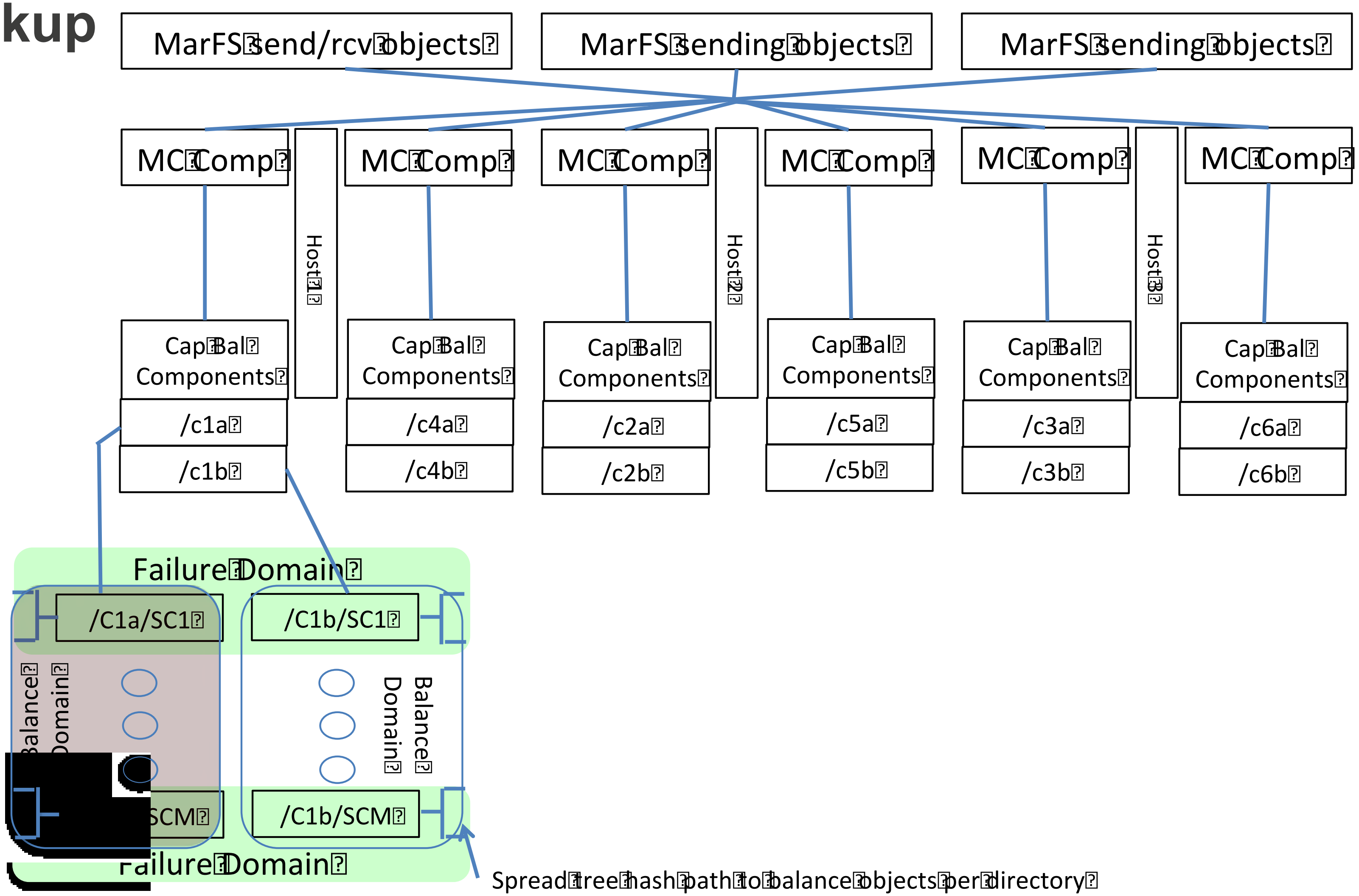
Open Source
BSD License
Partners Welcome

MAY THE FOURTH

Thanks for your attention!

BE WITH YOU

Backup



Spread tree hash path to balance objects per directory