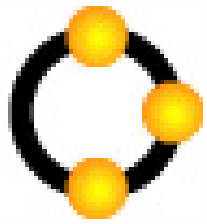
Lustre

Peter J. Braam
braam@clusterfs.com
http://www.clusterfilesystems.com

Cluster File Systems, Inc



Talk overview

- What is Lustre
- The big plan
- The small plan
- Discussion

What is Lustre?

A new storage architecture

- Use object oriented storage concepts
- Advanced cluster file system
- Scalable clustering ideas

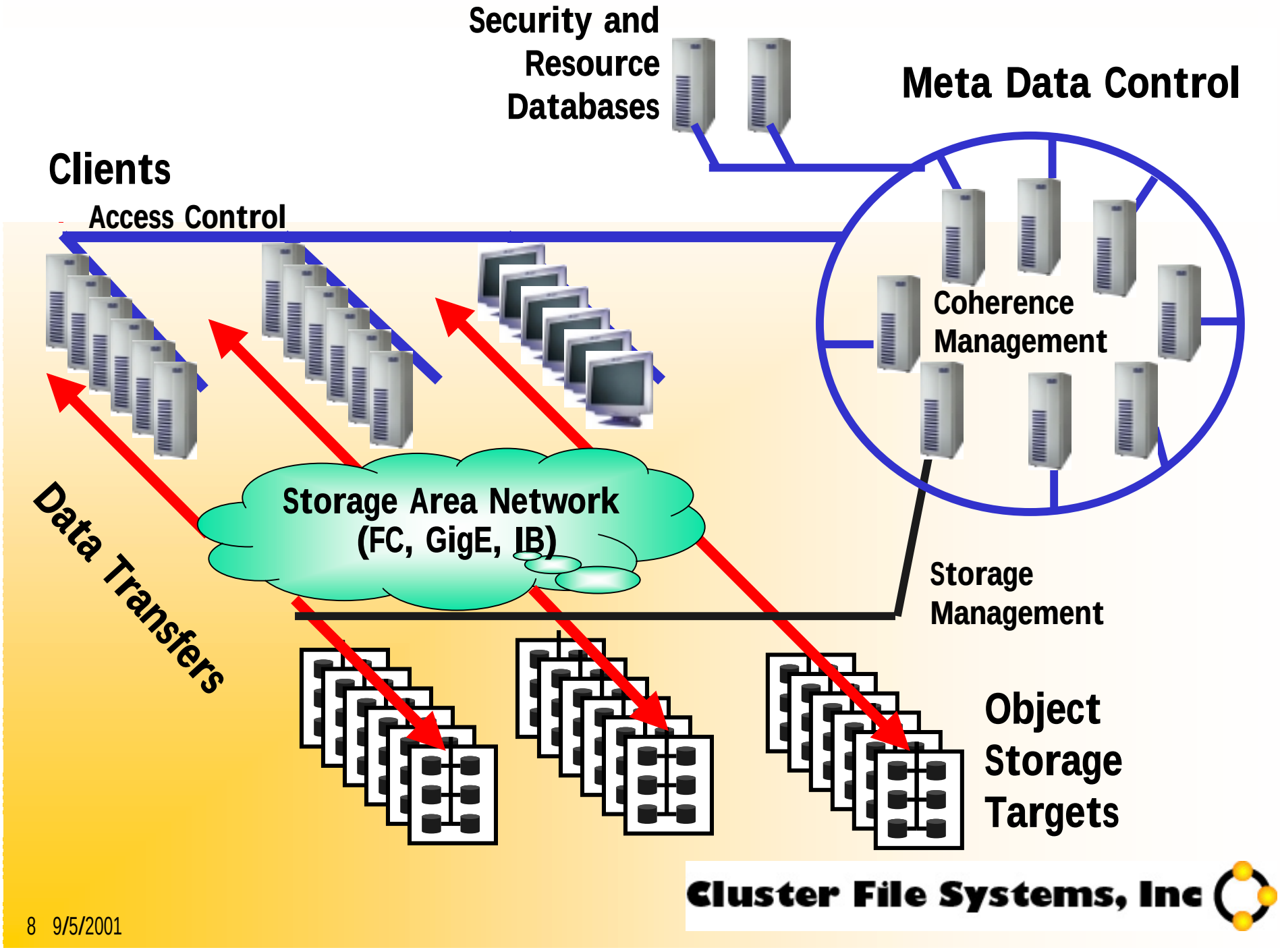
Project history

- Started between CMU – Seagate – Stelias Computing
  - Another road to NASD style storage
  - NASD now at Panasas – originated many ideas
- Los Alamos
  - More research
  - Nearly built little object storage controllers
  - Currently looking at genomics applications
- Sandia, Tri-Labs
  - Can Lustre meet the SGS-FS requirements?

Now

- Put in a proposal with Intel and XXX to build it
- 3 year project

Big Lustre picture



Cluster File Systems, Inc

Orders of magnitude

- Clients (aka compute servers)
  - 10,000's
- Storage controllers
  - 1000's to control PB's of storage (PB = 10**15 Bytes)
- Cluster control nodes
  - 10's
- Aggregate bandwidth
  - 100's GB/sec

Key issues: Scalability

- I/O throughput
  - How to avoid bottlenecks
- Meta data scalability
  - How can 10,000's of nodes work on files in same folder
- Cluster recovery
  - If something fails, how can transparent recovery happen
- Management
  - Adding, removing, replacing, systems; data migration & backup

Approach

- Andrew Project at CMU
  - 80's – file servers with 10,000 clients (CMU campus)
  - Key question: how to reduce foot print of client on server
  - By 1988 entire campus on AFS
- Lustre
  - Scalable clusters?
  - How to reduce cluster footprint of shared resources (scalability)
  - How to subdivide bottlenecked resources (parallelism)

Ingredients 1: object storage

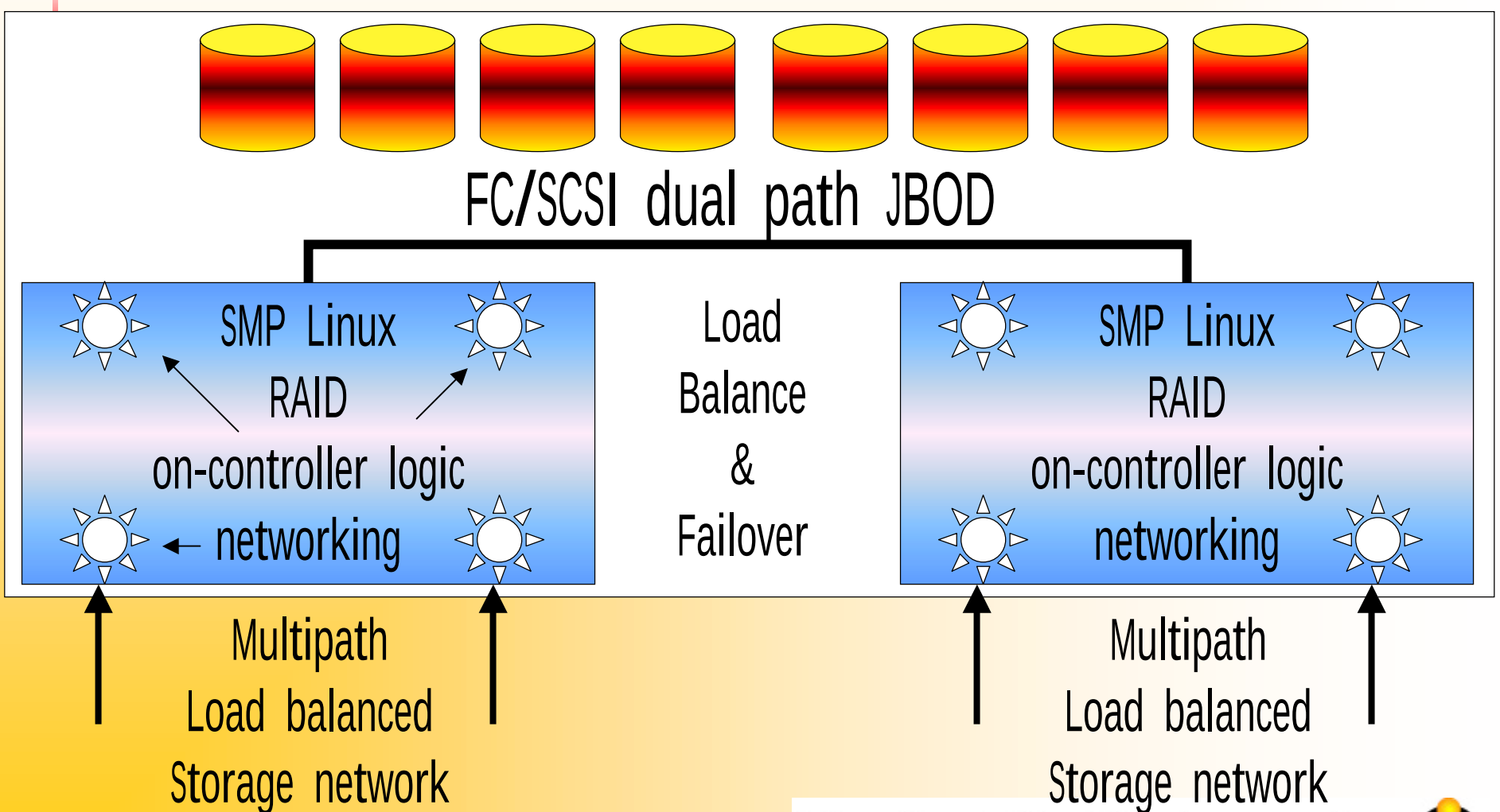
What is Object Based Storage?

- Object Based Storage Device
  - More intelligent than block device
- Speak storage at “inode level”
  - create, unlink, read, write, getattr, setattr
  - iterators, security, almost arbitrary processing
- So…
  - Protocol allocates physical blocks, no names for files
- Requires
  - Management & security infrastructure

Components of OB Storage

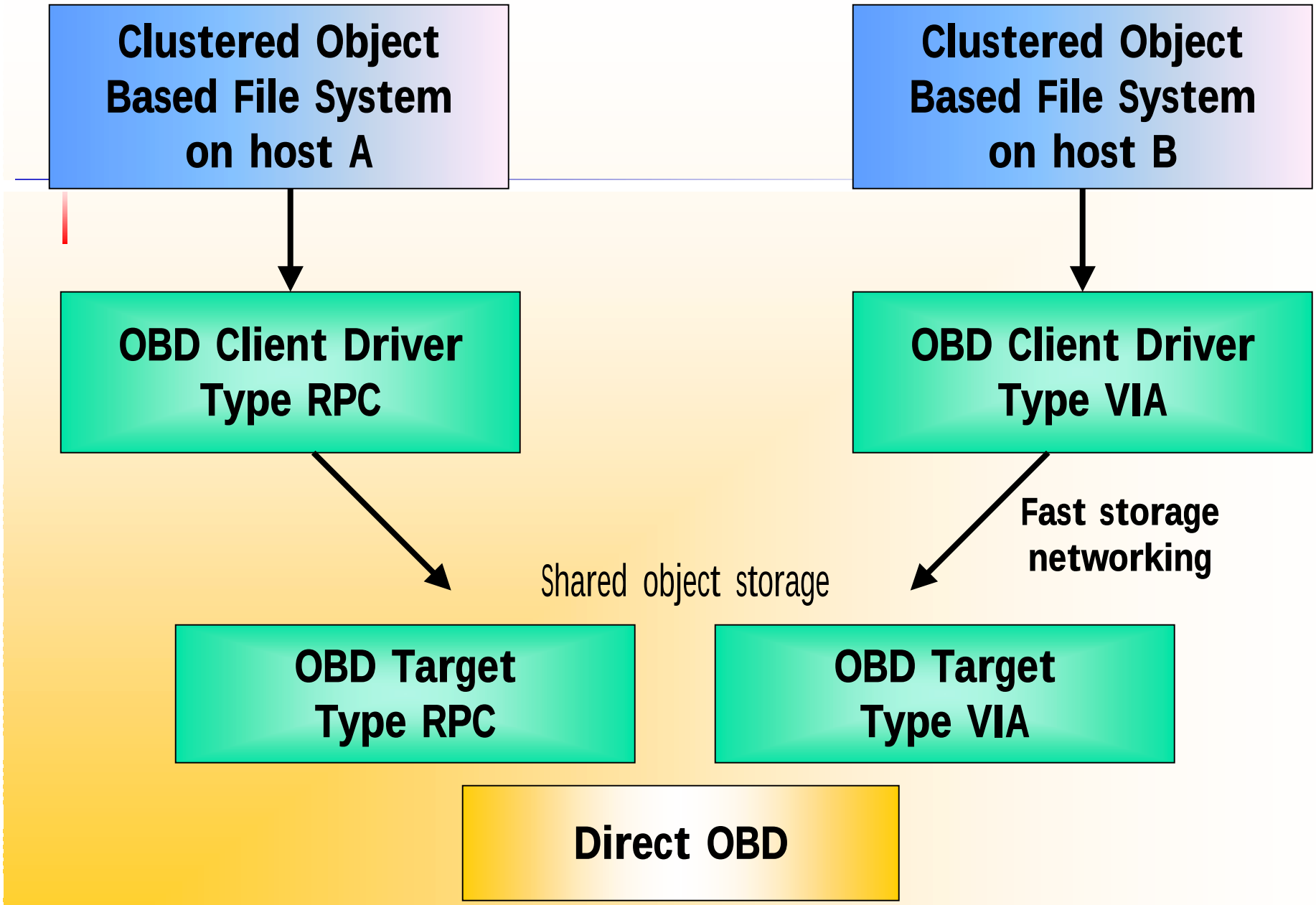
- Storage Object Device Drivers
  - class drivers – attach driver to interface
    - **Targets, clients** – remote access
    - **Direct drivers** – to manage physical storage
    - **Logical drivers** – for intelligence & storage management
- Object storage applications:
  - (cluster) file systems
  - Advanced storage: parallel I/O, snapshots
  - Specialized apps: caches, db's, filesrv

Inside the storage controller…



System Interface

- Modules
  - Load the kernel modules to get **drivers** of a certain **type**
  - Name **devices** to be of a certain type
  - Build **stacks** of devices with assigned types
- For example:
  - insmod obd_xfs ; obdcontrol dev=obd1,type=xfs
  - insmod obd_snap ; obdcontrol current=obd2,old=obd3,driver=obd1
  - insmod obdfs ; mount –t obdfs –o dev=obd3 /mnt/old


Clustered Object Based File System on host A
Clustered Object Based File System on host B
OBD Client Driver Type RPC
OBD Client Driver Type VIA
Fast storage networking
Shared object storage
OBD Target Type RPC
OBD Target Type VIA
Direct OBD

Examples of logical modules

- Tri-Lab/NSA: SGS – File system (see next slide)
  - Storage management, security
  - Parallel I/O for scientific computation
- Other requests:
  - Data mining while target is idle
  - LANL: gene sequencing in object modules
  - Rich media industry: prioritize video streams

I/O bandwidth requirements

- Required: 100's GB/sec
- Consequences:
  - Saturate 100's – 1000's of storage controllers
  - Block allocation must be spread over cluster
- This almost forces object storage controller approach

File – I/O

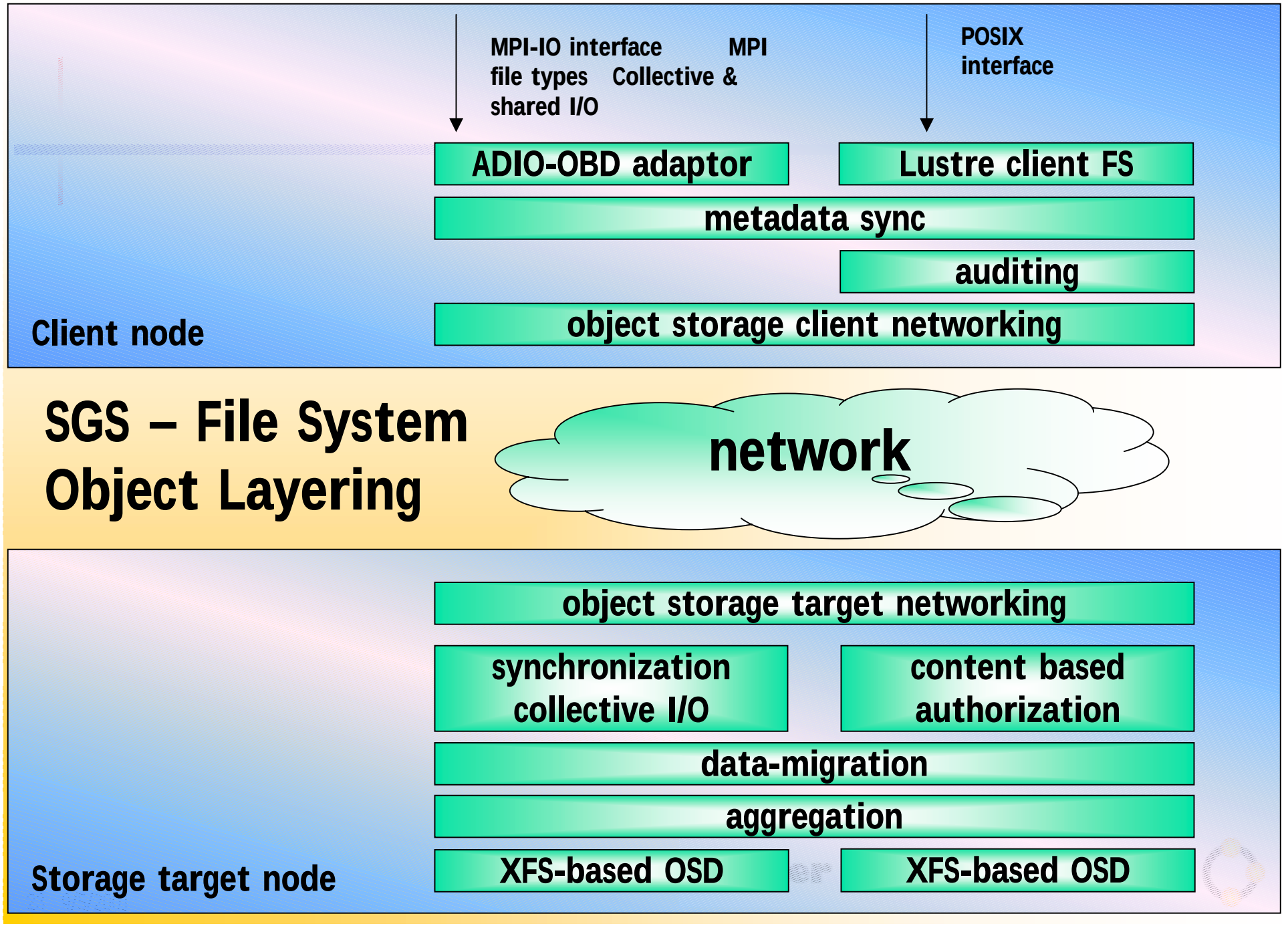
- Open file on metadata system
- Get obtain information
  - What objects on what storage controllers store what part of the file
  - Striping pattern
- Establish connection to storage controllers you need
  - Do logical object writes to OST
  - From time to time OST updates MDS with new file sizes

And…

- For everybody: locking
  - OST manages locks for file writes:
    - Lockf/flock locks
    - Internal cluster file system enforced locks
- For the labs – parallel I/O
  - Client and OST can carry logical parallel I/O modules
  - Translate file views to larger aggregates
  - Collective I/O
  - Disable locking

And…

- Storage management
  - Investigate versioning
  - HSM interfaces
- Security
  - Token interface between client, MDS, OST
  - Content based security etc.


MPI-IO interface
MPI file types
Collective & shared I/O
POSIX interface
ADIO-OBD adaptor
Lustre client FS
metadata sync
auditing
object storage client networking
Client node
SGS – File System Object Layering
network
object storage target networking
synchronization collective I/O
content based authorization
data-migration
aggregation
XFS-based OSD
XFS-based OSD
Storage target node

Ingredients 2: metadata handling

Basic picture

- Clients dealing with metadata cluster
- Low concurrency
  - Want write back caching
- High concurrency
  - Want load balancing in cluster
    - Subdivide directories etc with hashes
  - Want server handling of requests to limit lock revocations

Metadata cluster

- Responsible for storing:
  - Inodes
  - Directory content
  - File data: replaced with indirect objects on OST's
  - "Third party I/O model", on steroids
- Must execute:
  - Read only requests
  - Update requests
  - Lock requests

And…

- Load balancing
  - Hashing directories
    - Named based hashes allow client to target particular cluster node
    - Allow load balancing highly concurrent operation
    - E.g.: MDS1 serves filenames A-J, MDS2: K-Z
  - Resource allocation issues
  - Recovery issues
  - Little bit like ordinary cluster file system with NFS servers on top

And…

- Interaction with storage targets
  - Preallocation of objects
  - Recovery in case MDS or OST's fail

This

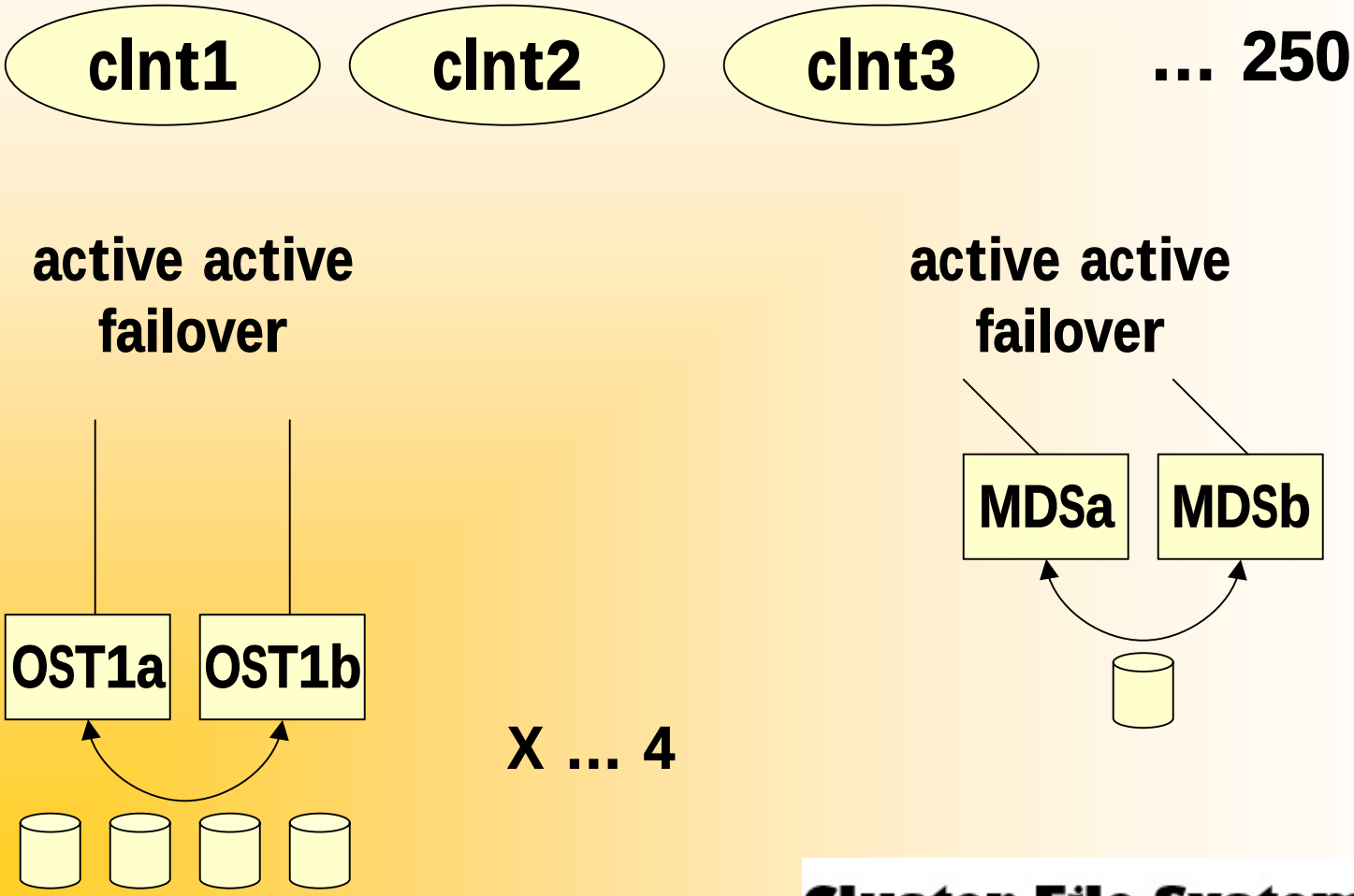
- Is a 3 year project
- Has hard parts:
  - Meta cluster, load balancing, recovery
  - Tuning
  - Management infrastructure

Small Plan

Lustre light

- NO
  - No meta cluster with single, failover metadata server
  - No advanced security do basic Unix security
  - No storage management
  - No parallel I/O
  - No write back caching for metadata
  - Minimize locking support (like NFS)
- What remains?
  - A pretty attractive open source cluster file system
  - Something that can evolve into Lustre

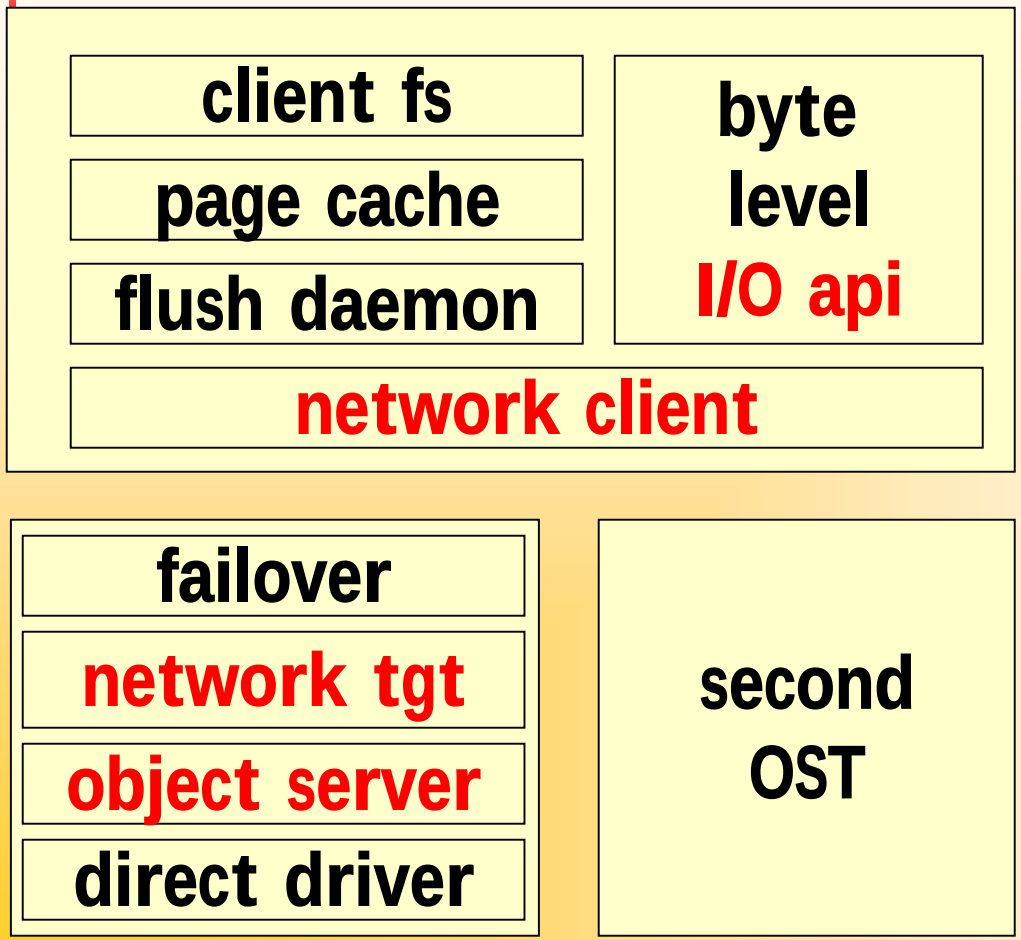
Picture…



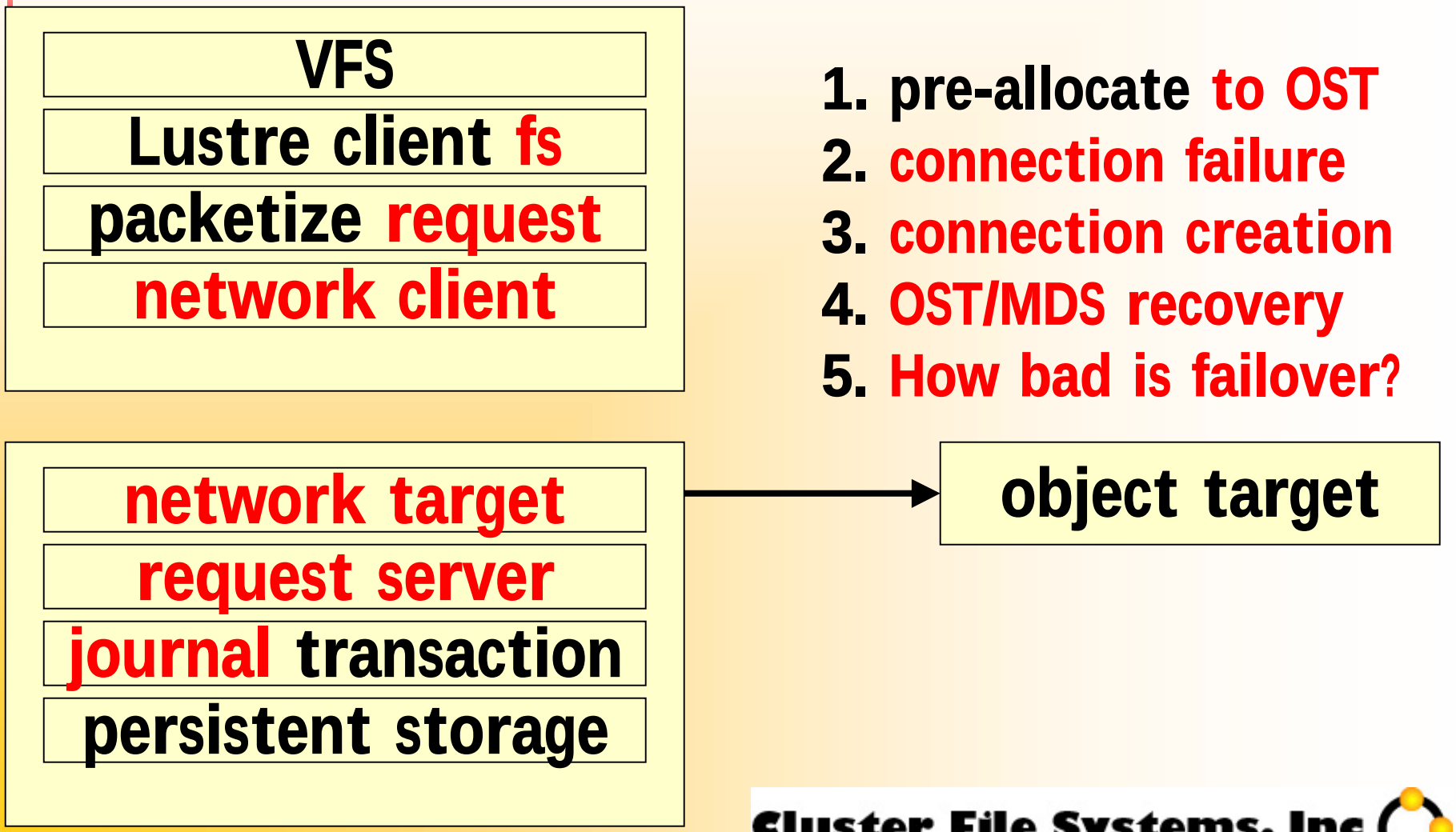
What is there, what do we need?

- Black components are there
- Red is missing

What do we need? – file I/O

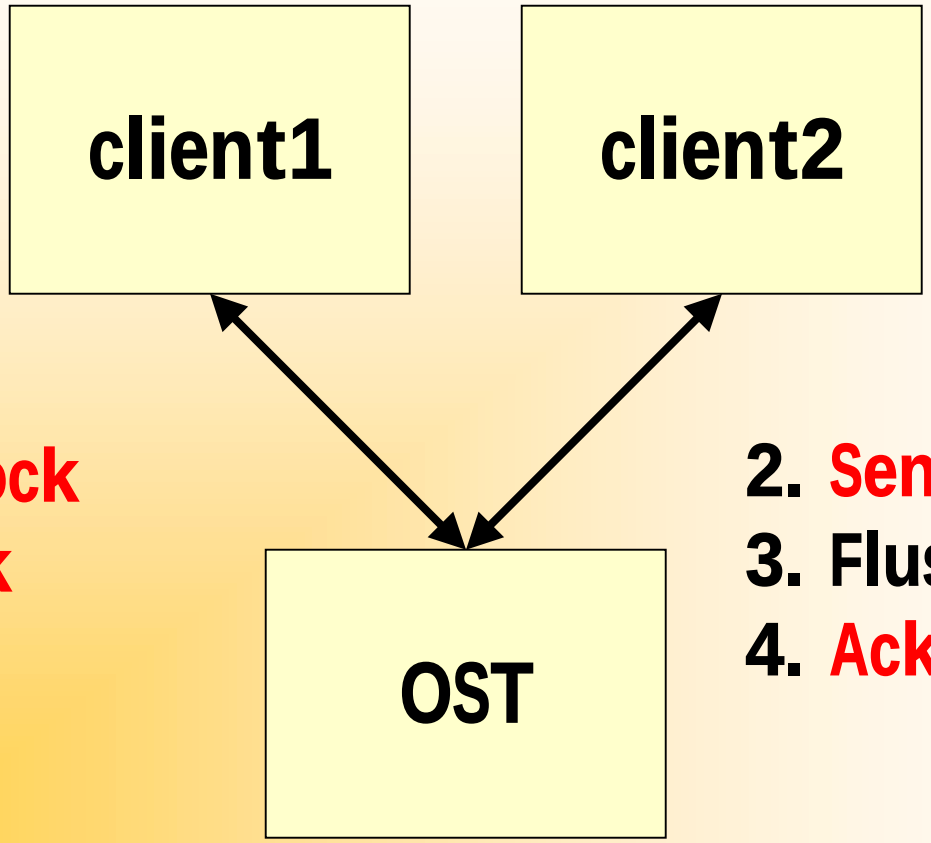


What do we need? – meta data



1. pre-allocate to OST
2. connection failure
3. connection creation
4. OST/MDS recovery
5. How bad is failover?

What do we need? – Locking



1. Request lock
4. Grant Lock
6. Start I/O

2. Send revoke
3. Flush cache
4. Ack revoke

Leads to cluster Unix semantics for file writes

Enthusiastic?

- Definitely!
  - We'll have a working clusterfs soon.
  - We have a roadmap for serious improvements.
- But…
  - Have we overlooked something?
  - How serious is recovery from network failures?
- Thanks Mark, for kicking this discussion off!