Lustre

Scalable Clustered Object Storage

Peter J. Braam

braam@clusterfs.com

http://www.clusterfilesystems.com

Cluster File Systems, Inc

The first 3 years…

- 1999 CMU – Seagate – Stelias Computing
- 2000 Los Alamos, Sandia, Livermore:
  - need new File System
- 2001: Lustre design to meet the SGS-FS requirements?
- 2002: things moving faster
  - Lustre on MCR (1000 node Linux Cluster – bigger ones coming)
  - Lustre Hardware (BlueArc, others coming)
  - Very substantial ASCI pathforward contract (with HP & Intel)

Key requirements

- I/O throughput – 100's GB/sec
- Meta data scalability – 10,000's nodes, ops/sec, trillions of files
- Cluster recovery – simple & fast
- Storage management – snapshots, HSM
- Networking – heterogeneous networks
- Security – strong and global

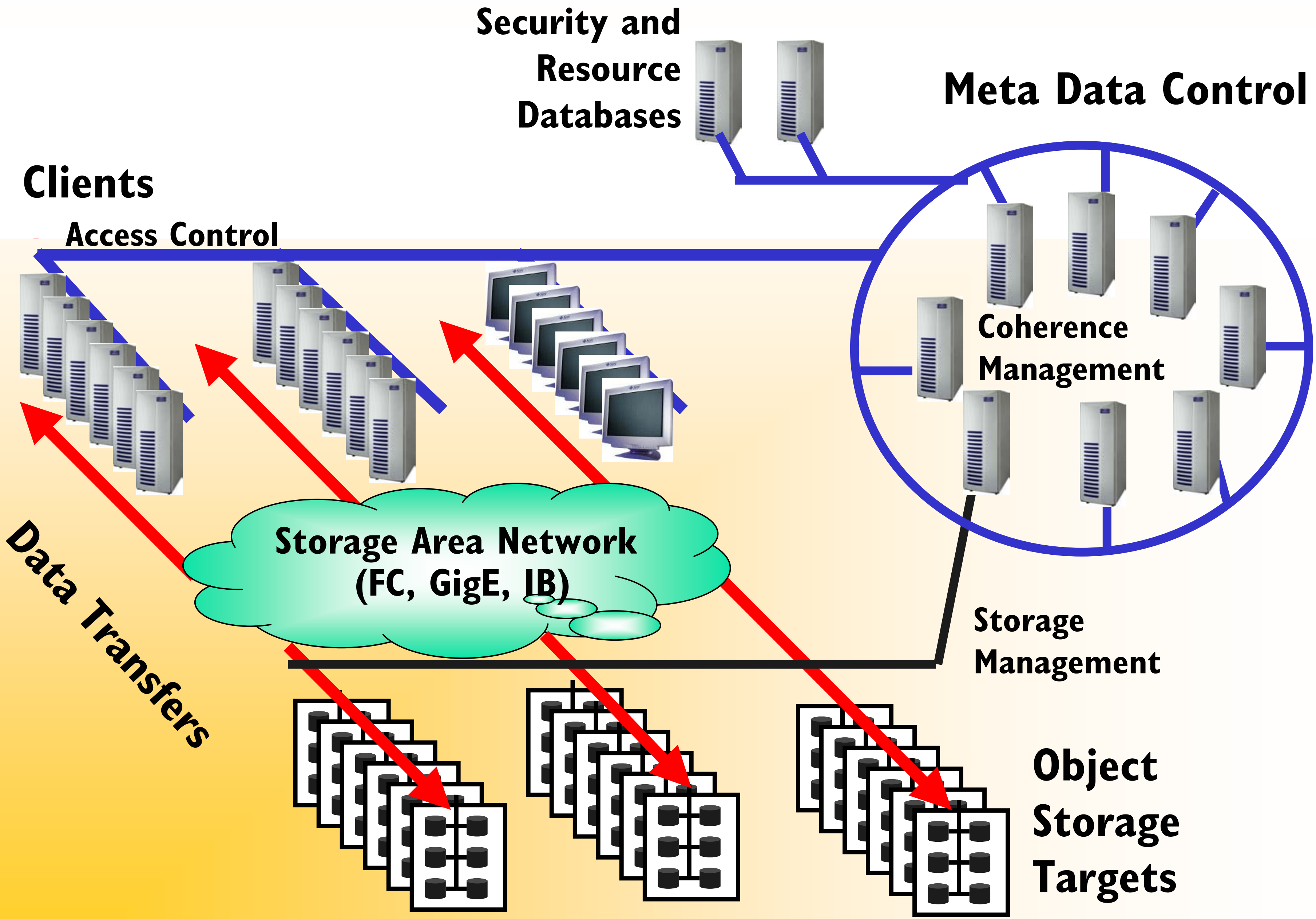
Cluster File Systems, Inc

Approach

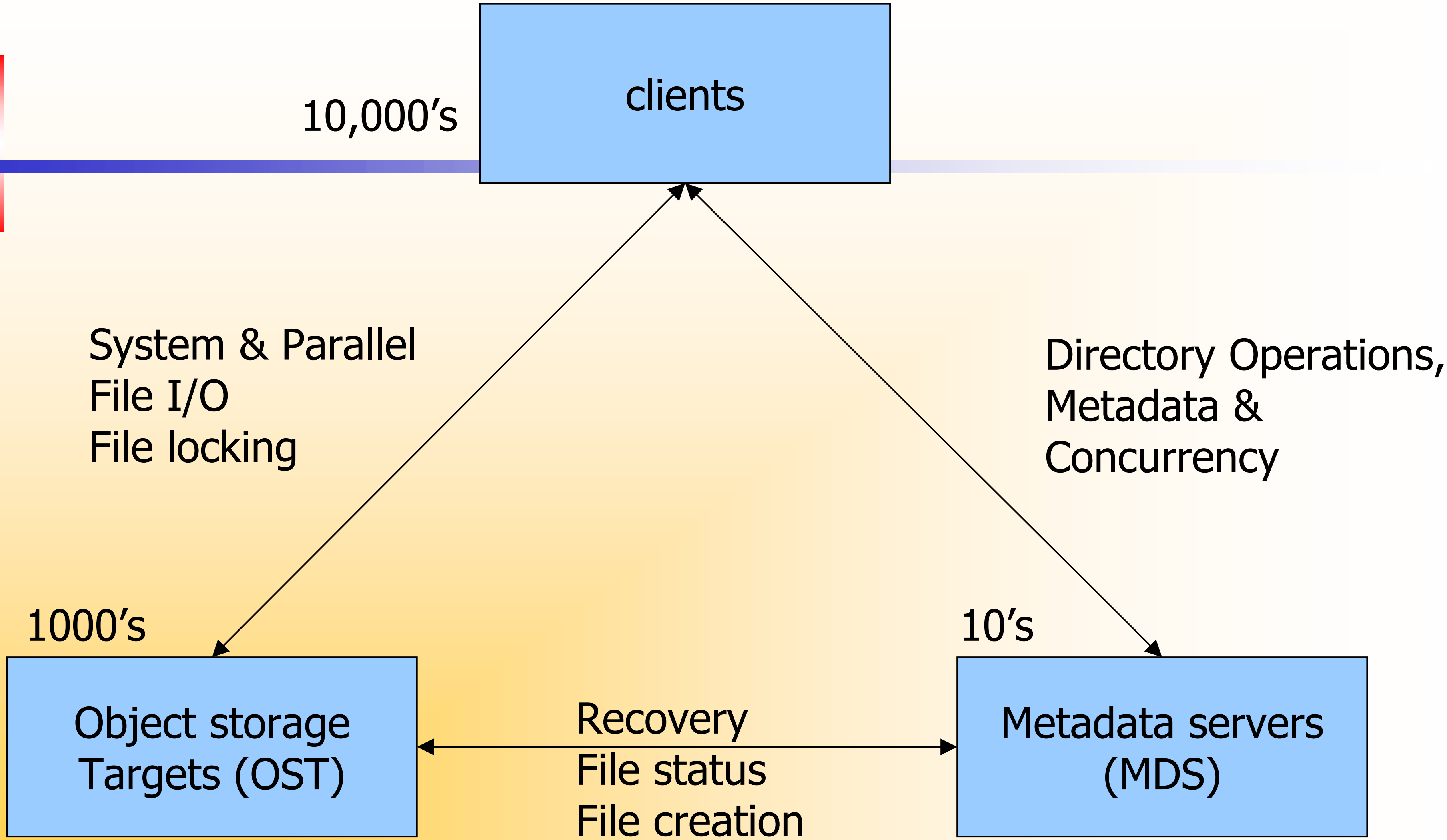
- Initially Linux focused
- Was given blank sheet
- Learn from successes
  - GPFS on ASCI White
  - TUX web server, DAFS protocol
  - Sandia Portals Networking
  - Use existing disk file systems: ext3, XFS, JFS
- New protocols
  - InterMezzo, Coda

Cluster File Systems, Inc

Lustre



Cluster File Systems, Inc



Lustre System

Cluster File Systems, Inc

Lustre Devices

- Lustre has numerous modules, all offering certain api's
  - Stacking is a key feature for Lustre
- Initially tried having "object devices"
- Devices now have drivers exporting up to 4 api's:
  - Administrative – a mandatory API
  - Object Storage
  - Metadata handling
  - Locking

Ingredient 1: Storage Networking

Lustre networking

- Currently runs over
  - TCP,
  - Quadrics
  - Myrinet
- Other networks in progress:
  - SAN's
  - I/B
  - NUMA interconnects (@ GB/sec)
  - Bus interconnects
  - Offload cards
  - SCTP

Lustre Network Stack

- Device|Library (Elan,TCP,...)
- Portal NAL's
- Portal Library
- NIO API
- Request Processing

Portal NAL's ← now: Elan & IP
soon: Sandia, GM

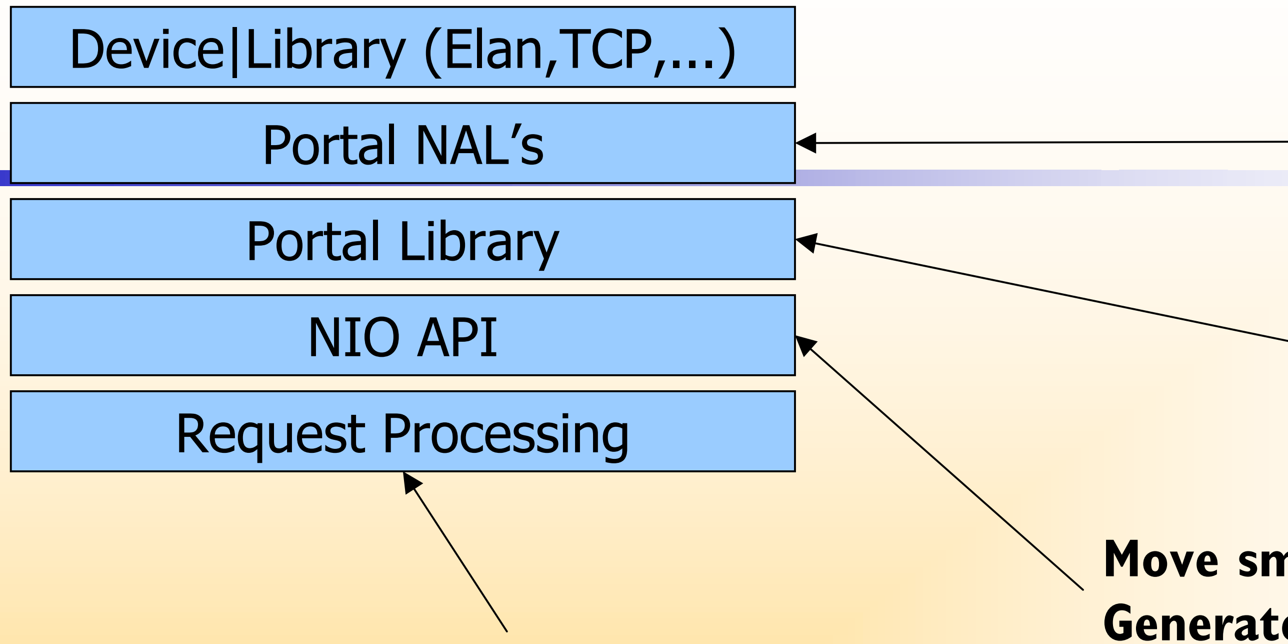
Portal Library ← Sandia's API
CFS improved impl.

NIO API ← Move small & large buffers
Generate events

Request Processing ←
0-copy marshalling libraries
service framework
client request dispatch
connection & address naming
generic recovery infrastructure

Cluster File Systems, Inc

Lustre Network Stack



now: Elan & IP
soon: Sandia, GM

Sandia's API
CFS improved impl.

Move small & large buffers
Generate events

0-copy marshalling libraries
service framework
client request dispatch
connection & address naming
generic recovery infrastructure

Cluster File Systems, Inc

Lustre Network Stack

- Device|Library (Elan,TCP,...)
- Portal NAL's
- Portal Library
- NIO API
- Request Processing

Portal NAL's:
now: Elan & IP
soon: Sandia, GM

Portal Library:
Sandia's API
CFS improved impl.

NIO API:
Move small & large buffers
Generate events

Request Processing:
0-copy marshalling libraries
service framework
client request dispatch
connection & address naming
generic recovery infrastructure

Portals

- Sandia Portals message passing
  - simple message passing API
  - support for remote DMA
  - support for plugging in device support
  - Network Abstraction Layers
- We have no definitive answers on best design of the NALs yet
- Not so suitable for SCSI layering

Initial performance figures

- Networking
  - OST can handle 40,000 requests/sec
  - Quadrics network: 340MB/sec
  - IP: 110MB/sec
- Client to disk
  - One client: 220MB/sec (5 threads)
  - All targets saturate, linear scaling, demo up to 1.5GB/sec

Lustre & SAN

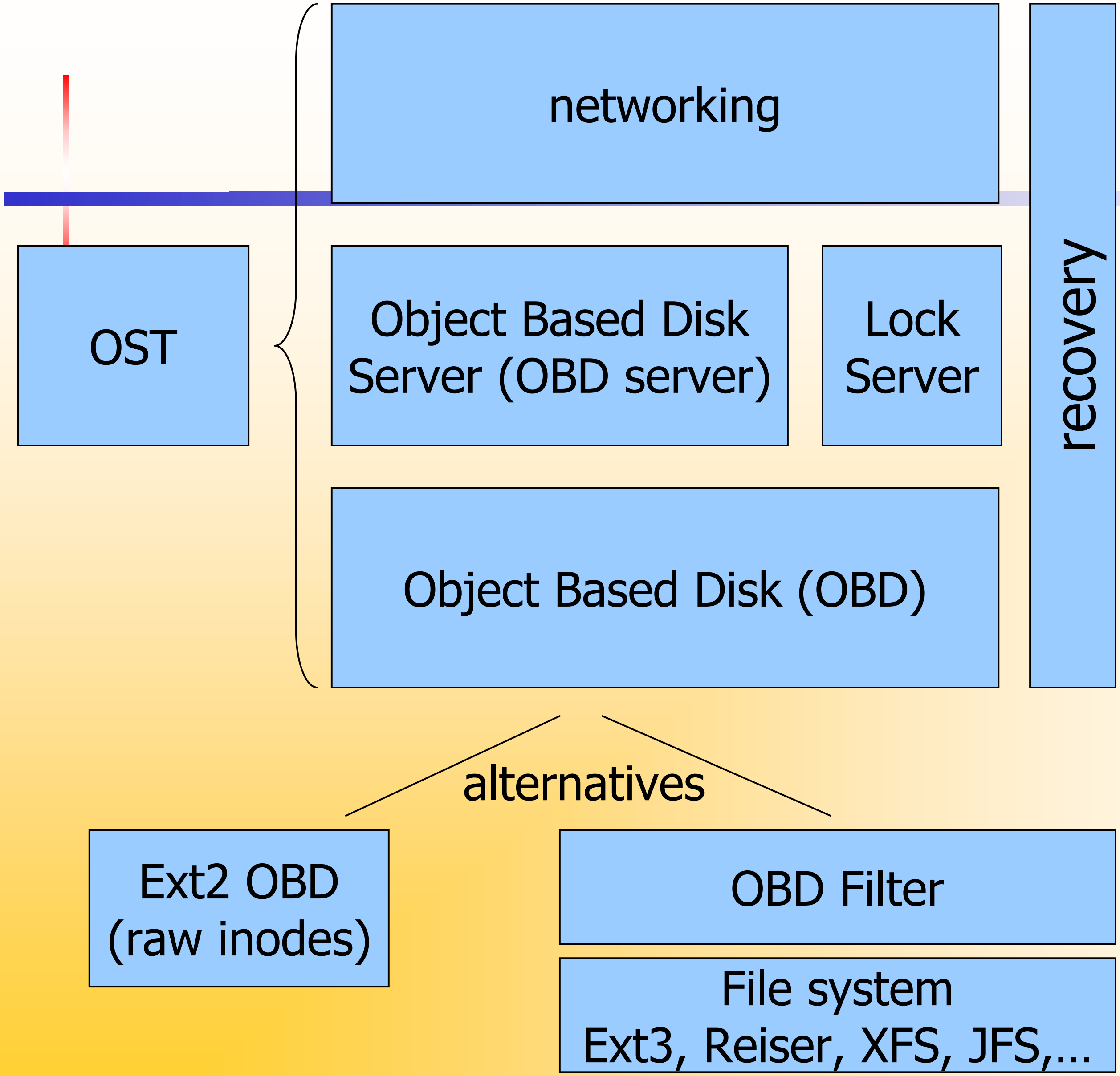
- From the galaxy to a 4 node Linux cluster
- Exploit SAN's – retain OST/MDS
  - TCP/IP: to allocate blocks, do metadata
  - SAN: for file data movement

Ingredient 2: 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

Object Storage Target



Cluster File Systems, Inc

How does object storage help?

File – I/O

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

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
  - Lock management must be spread over cluster
- This almost forces object storage controller approach

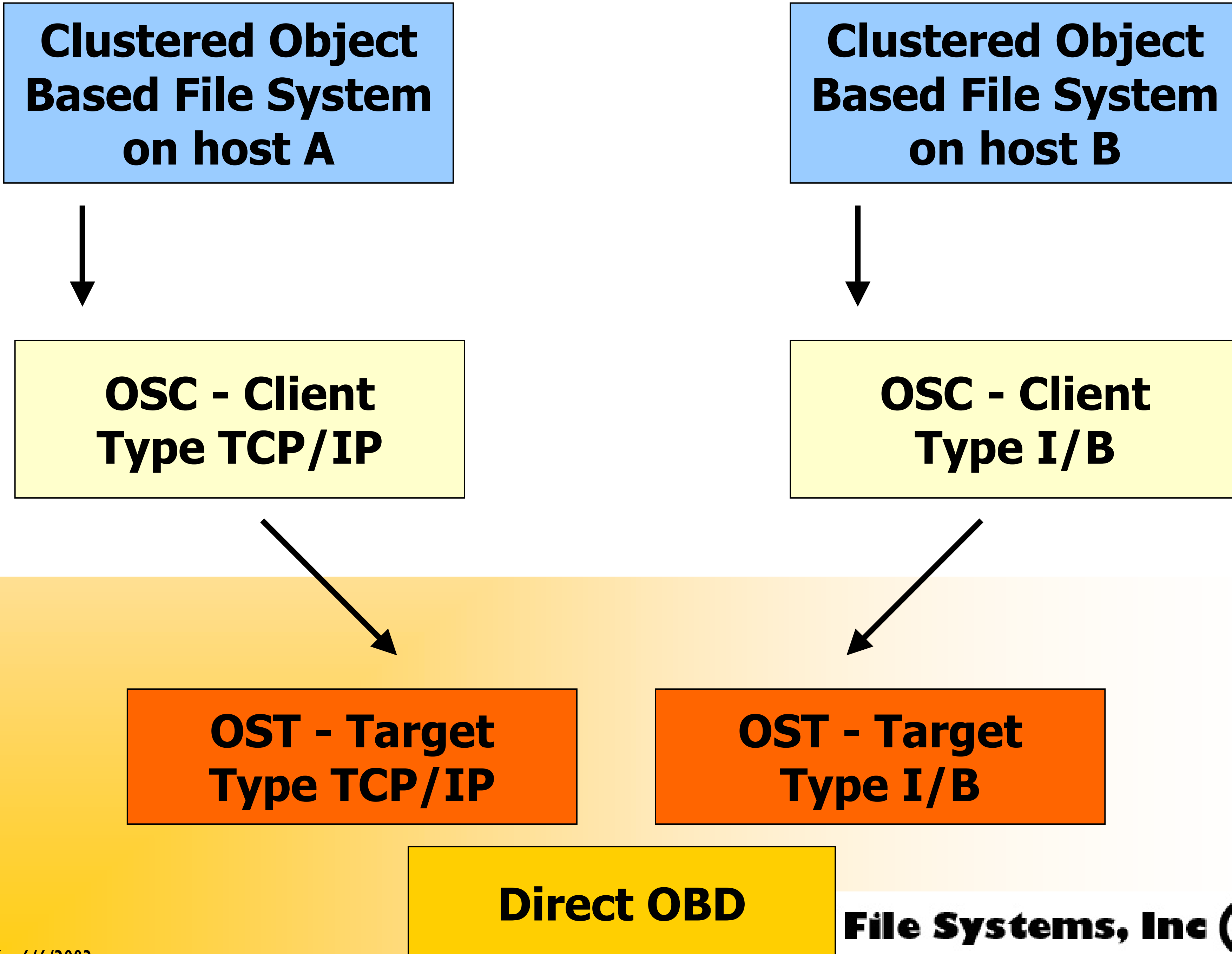
Ingredient 3: Storage Management

Components of OB Storage

- Storage Object Device Drivers
  - **Class driver** – attach driver to interface
  - **Targets, clients** – remote access
  - **Direct drivers** – to manage physical storage
  - **Logical drivers** – for intelligence & storage management
  - **Object storage “applications”** – eg. the file system

Examples of logical modules

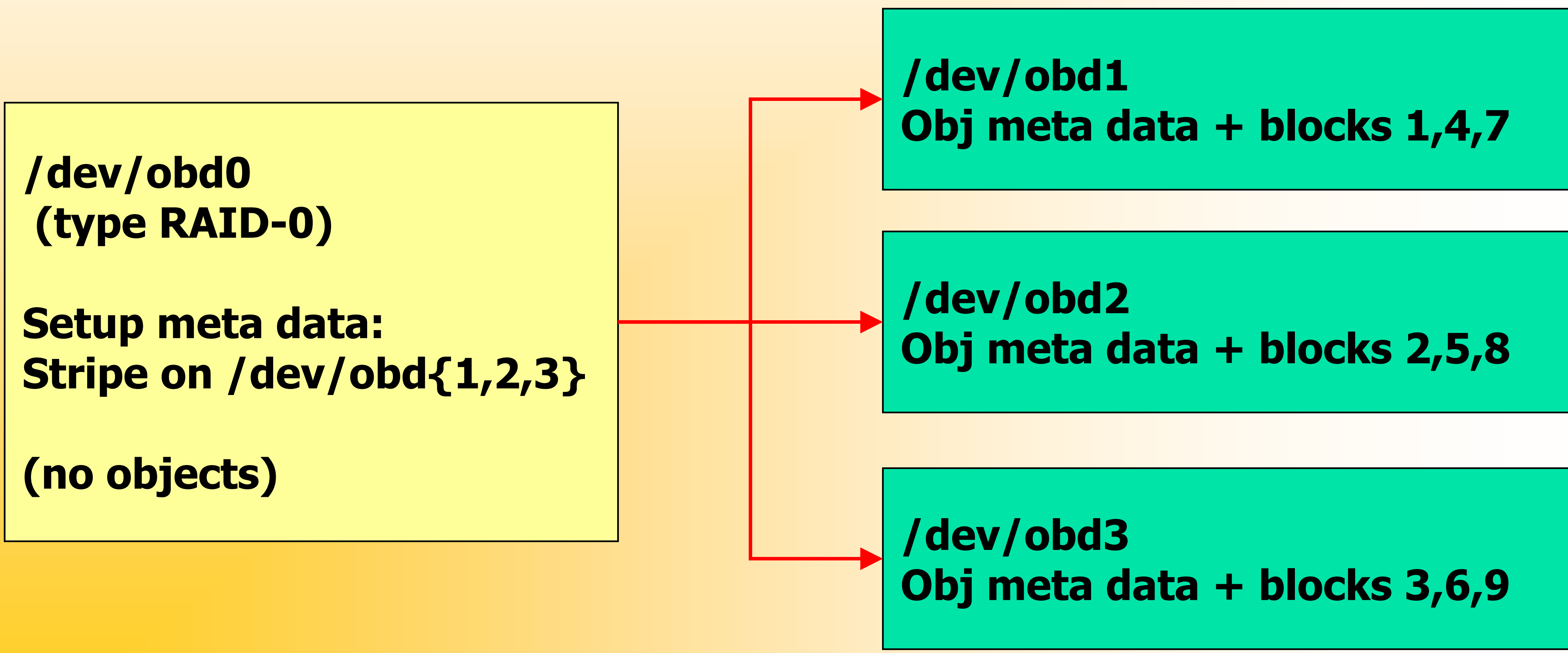
- Storage management:
  - System software, trusted
  - Often inside the standard data path,
  - Often involves iterators
  - Eg: security, snapshots, versioning data migration, raid
- Lustre offers active disks
  - almost arbitrary intelligence can be loaded into OST driver stack
  - Largely unexplored – LANL wanted to process gene matching



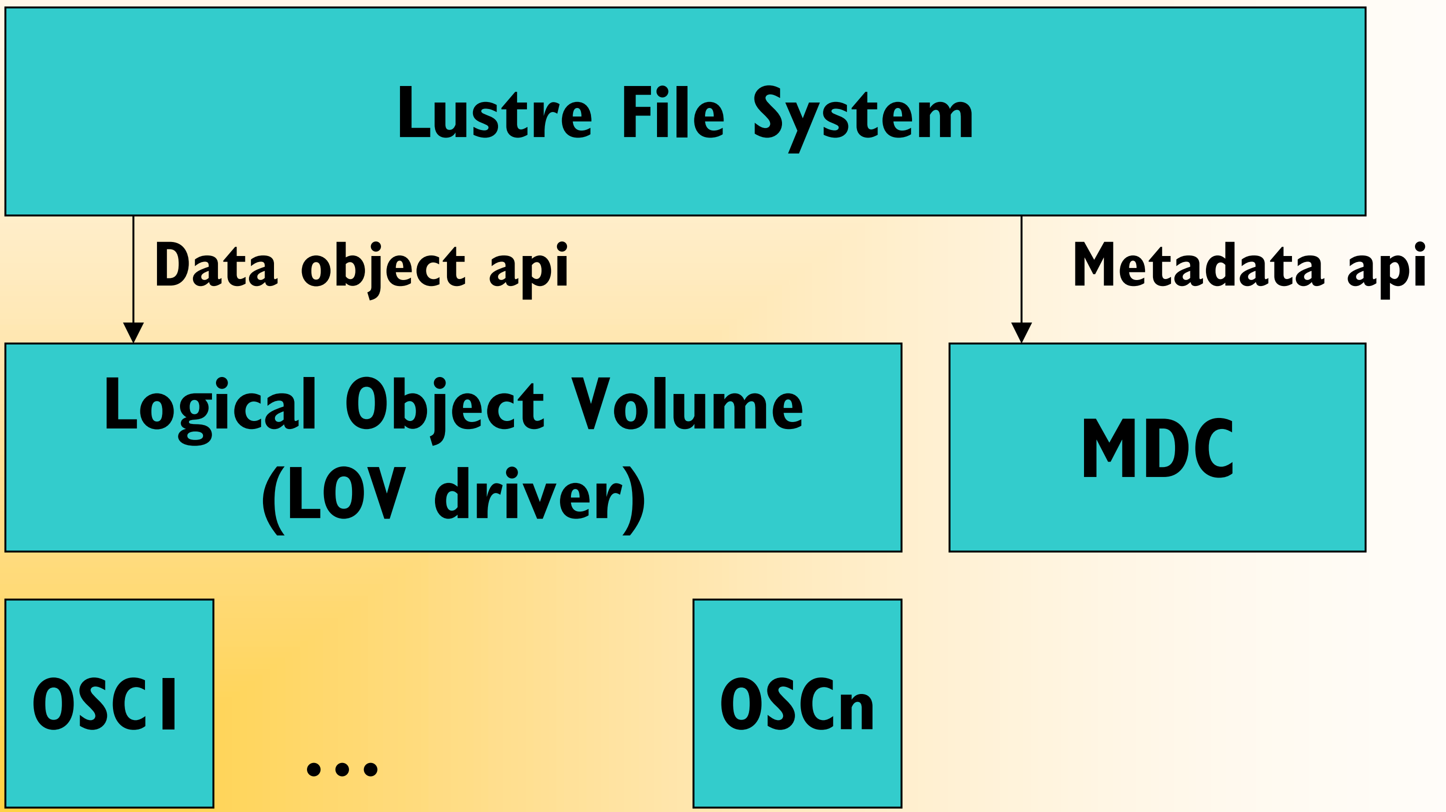
File Systems, Inc

LOV: striping & raid

Logical Object Volume Management:



Lustre Clients

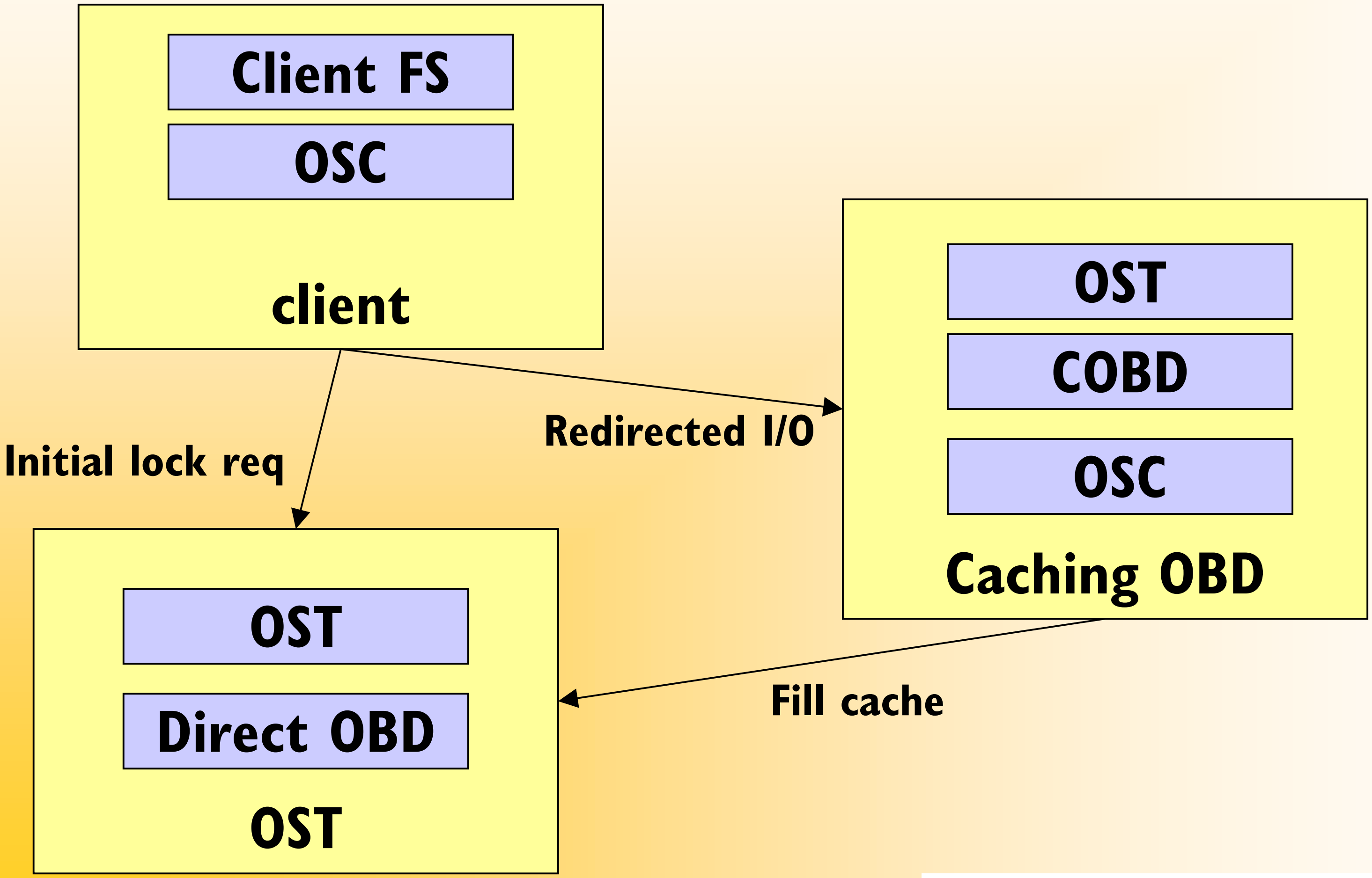


Lustre Collaborative Read Cache

- Add read scalability to system
- Read is preceeded by read-lock request
  - Lock server knows what is going to be read
  - Lock server knows who has that cached already
  - Lock server includes a referral

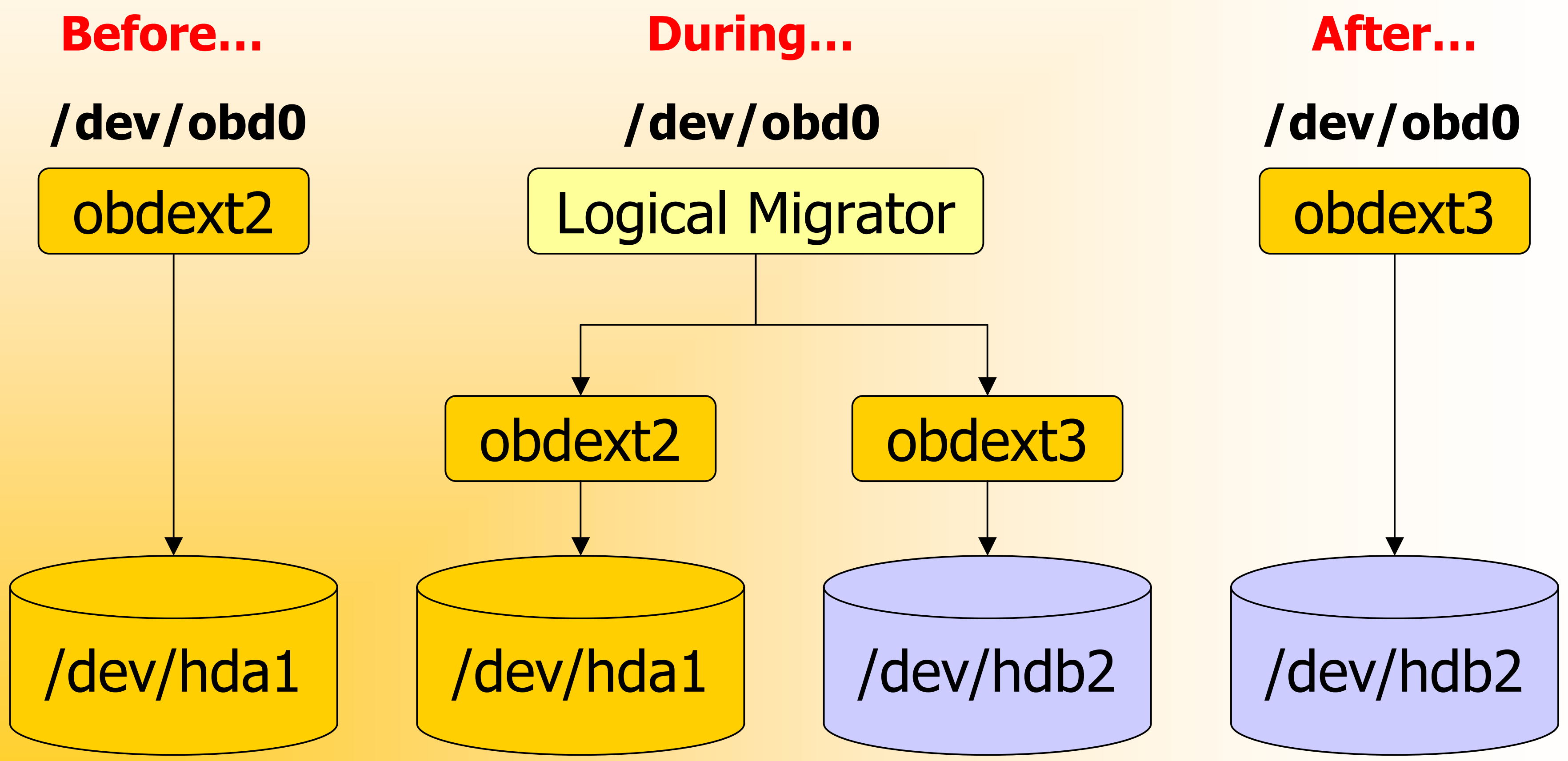
- Separate read cache servers, possibly in a tree
- Whole cluster acts as read-cache for each other

COBD – caching OBD



Example of management: hot data migration:

Key principle: dynamically switch object device types



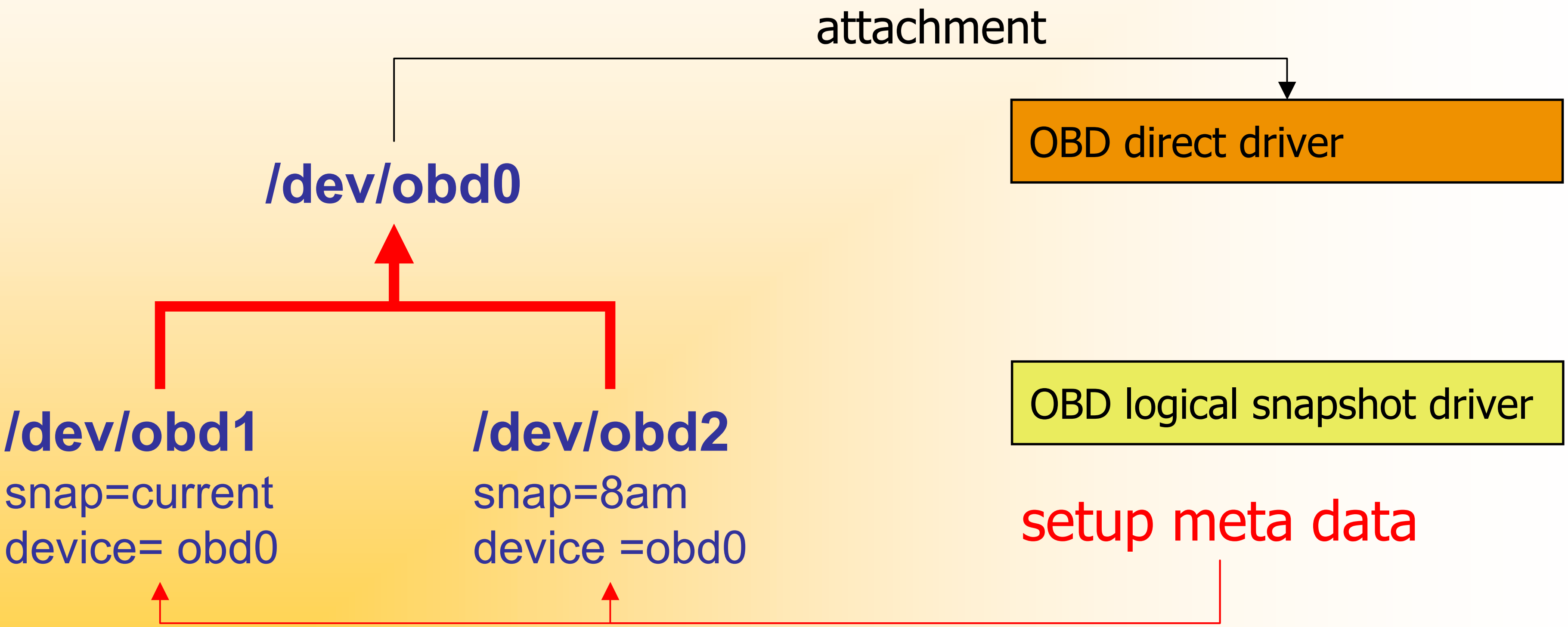
Cluster File Systems, Inc

Objects may be files, or not…

- Common case:
  - Object, like inode, represents a file
- Object can also:
  - represent a stripe (RAID)
  - bind an (MPI) File_View
  - redirect to other objects

Cluster File Systems, Inc

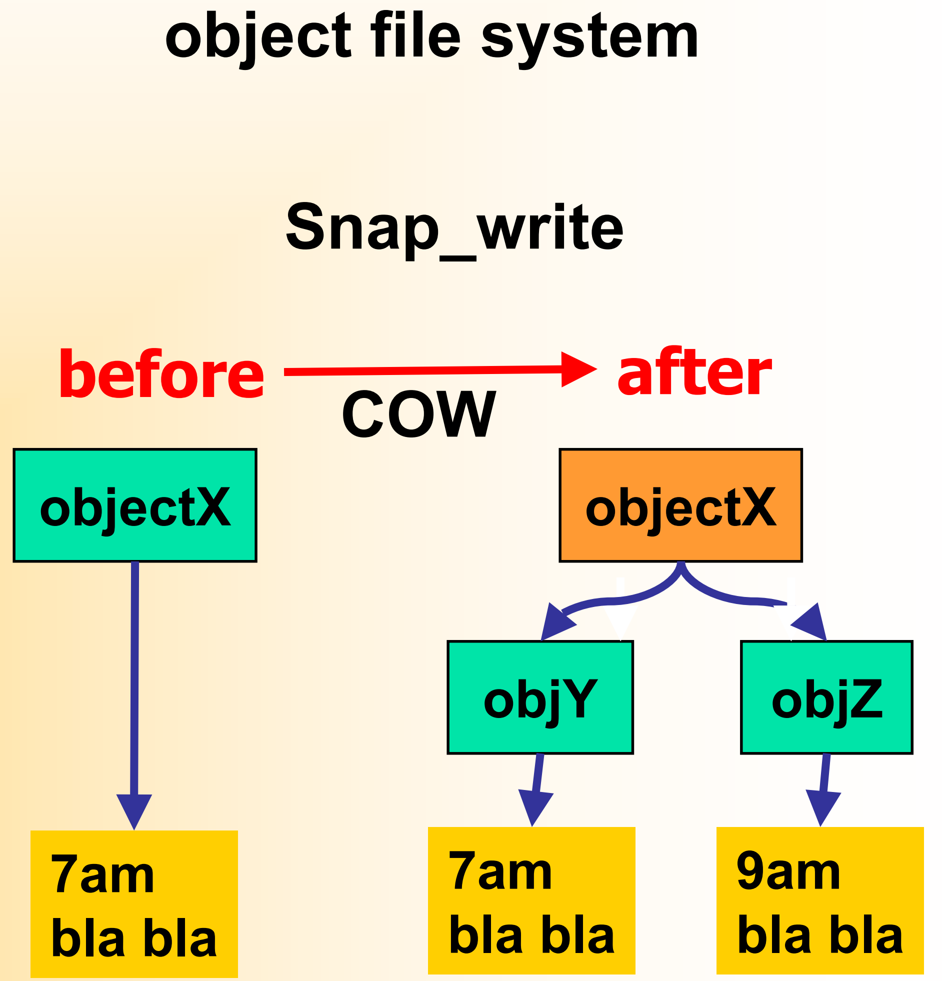
Snapshot setup



- **Result:**
  - /dev/obd2 is read only clone
  - /dev/obd1 is copy on write (COW) for 8am

Snapshots in action

- **Modify /mnt/obd/files**
- **Result:**
  - **new copy** in /mnt/obd/files
  - **old copy** in /mnt/obd/8am



Cluster File Systems, Inc

Ingredient 4: metadata handling

Intent based locks & Write Back caching

- Protocol adaptation between clients and MDS
- Low concurrency - write back caching
  - On client in memory updates with delayed replay on MDS
- High concurrency
  - Want single network request per transaction, no lock revocations
  - Intent based locks – lock includes all info to complete transaction

Linux VFS changes: intent lookups

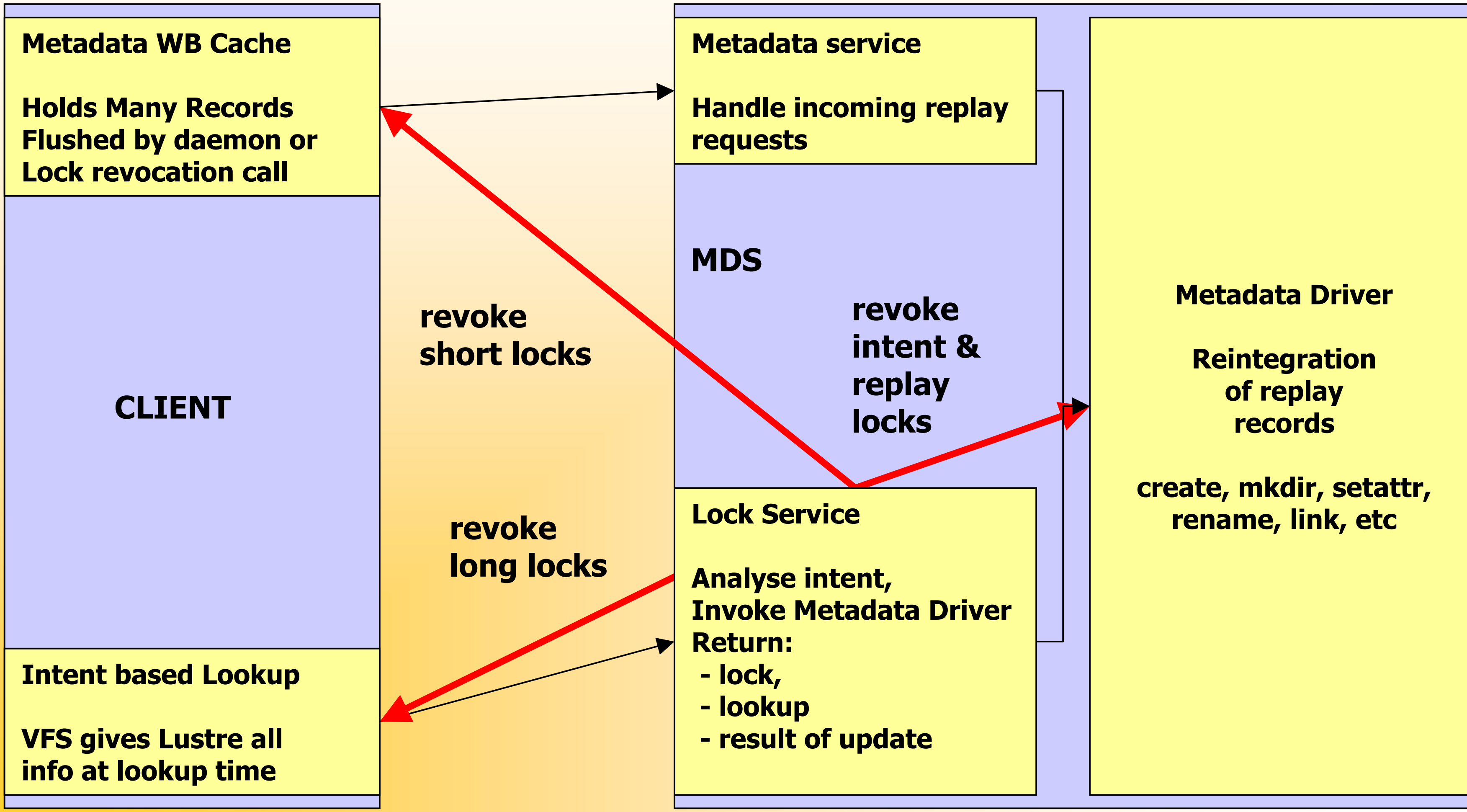
VFS	FS
sys_mkdir namei intent mkdir	Inode lookup operation /or/ Dentry revalidate operation FS arranges for 'mkdir' locks
Test if OK no: d_intent_release	Release lock
vfs_mkdir	Inode mkdir operation (use intent)
d_intent_release	Release lock

Cluster File Systems, Inc

Two types of metadata locks:

- Long locks –
  - Lock tail of pathname, help with concurrency
  - e.g. locking the root directory is BAD
    - so lock /home/peter & /home/phil separately
- Short Locks
  - Lock a directory subtree -help for delegation
  - e.g. a single lock on /home/phil is GOOD

Metadata updates



Cluster File Systems, Inc

Subdivision of metadata across cluster

- Directories:
  - hash by name
  - assign hash values to MDS cluster nodes
- Inodes:
  - Assign 16GB ext3 block groups to MDS cluster nodes
- Result:
  - many ops can proceed in parallel
  - Journaled metadata file system at the core

Recovery

- Client – MDS updates
  - Deals with lost replies, requests & disk updates
  - Replay mechanism: two phase response
- Locks
  - Forcefully revoke locks from dead clients
  - Re-establish existing locks with recovering services
- Recovery Interaction with storage targets
  - Preallocation of objects
  - Orphaned inodes and data objects, replay logs

Metadata odds and ends

Logical Metadata Drivers

- We have not forgotten about:
  - Local persistent metadata cache, like AFS/Coda/InterMezzo
  - Replicated metadata server driver
  - Remotely mirrored MDS

Light weight CFS

- Lightweight CFS
  - Export both interfaces from a file system
  - Results in shared ext3 file system
  - Combine with SAN approach

Conclusions - prospects

Cluster File Systems, Inc

Project

- Have 15 developers now – expect a few more
- Are working on deployment on
  - LLNL MCR cluster (1000 nodes) – with BlueArc OST's
  - PNNL IA64 cluster – HP system
  - End of 2002 – expect solid Lustre Lite 1.0

Lustre Feature Roadmap

Lustre Lite (Linux 2.4)	Lustre Lite Performance (2.5)	Lustre
2002	2003	2004
Single Failover MDS / OST	Metadata cluster	Metadata cluster
Basic Unix security	Basic Unix security	Advanced Security
File I/O very fast	Collaborative read cache	Storage management
Intent based scalable metadata	Write back metadata	Load balanced MD
POSIX compliant	Parallel I/O	Global namespace

Cluster File Systems, Inc

Lustre

- Great vehicle for advanced storage software
  - Things are _really_ done differently
- Leverage existing components
- Initial signs of performance and stability very promising