Lustre

Peter J. Braam

braam@clusterfs.com

http://www.clusterfs.com

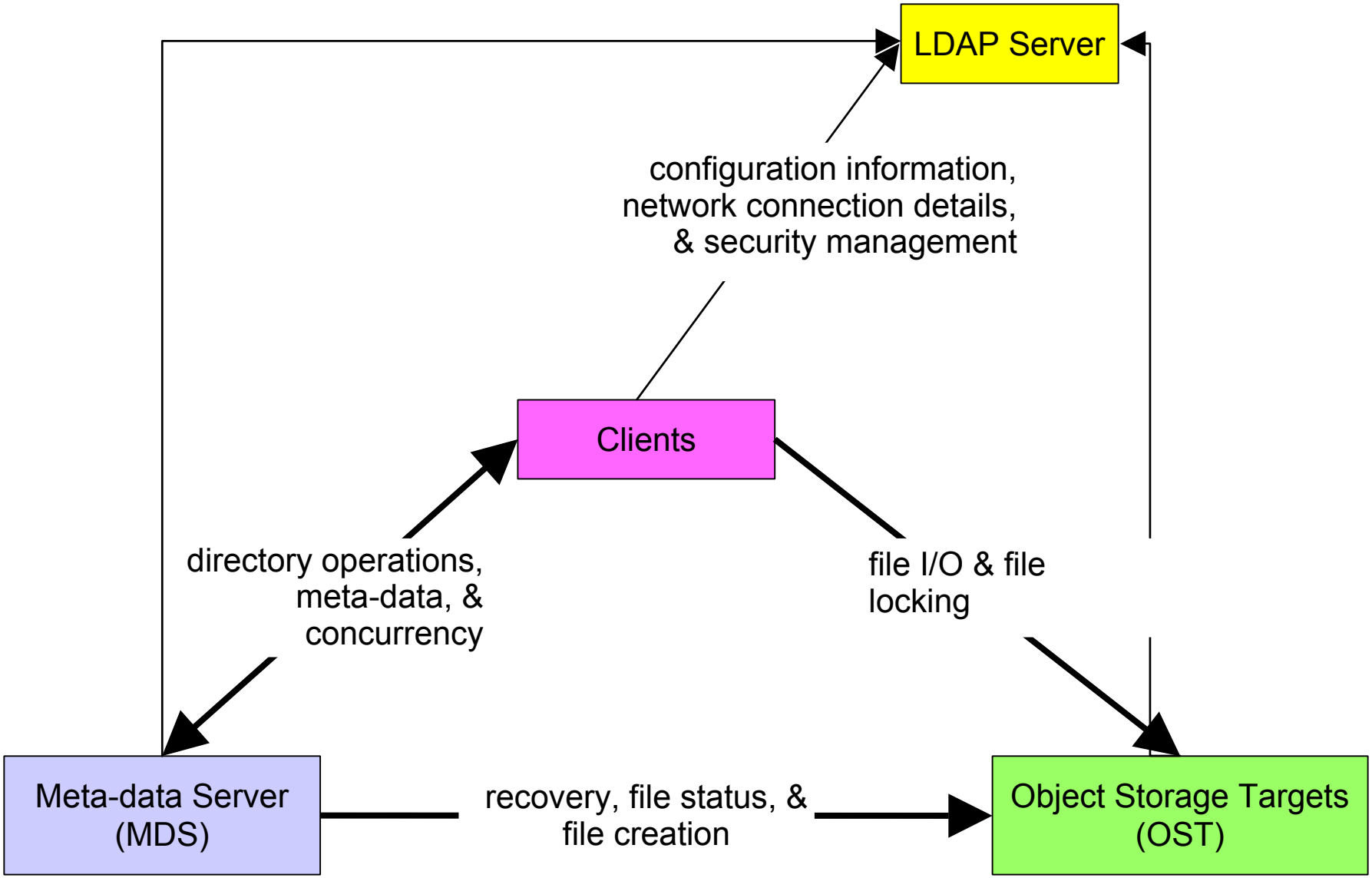
Cluster File Systems, Inc

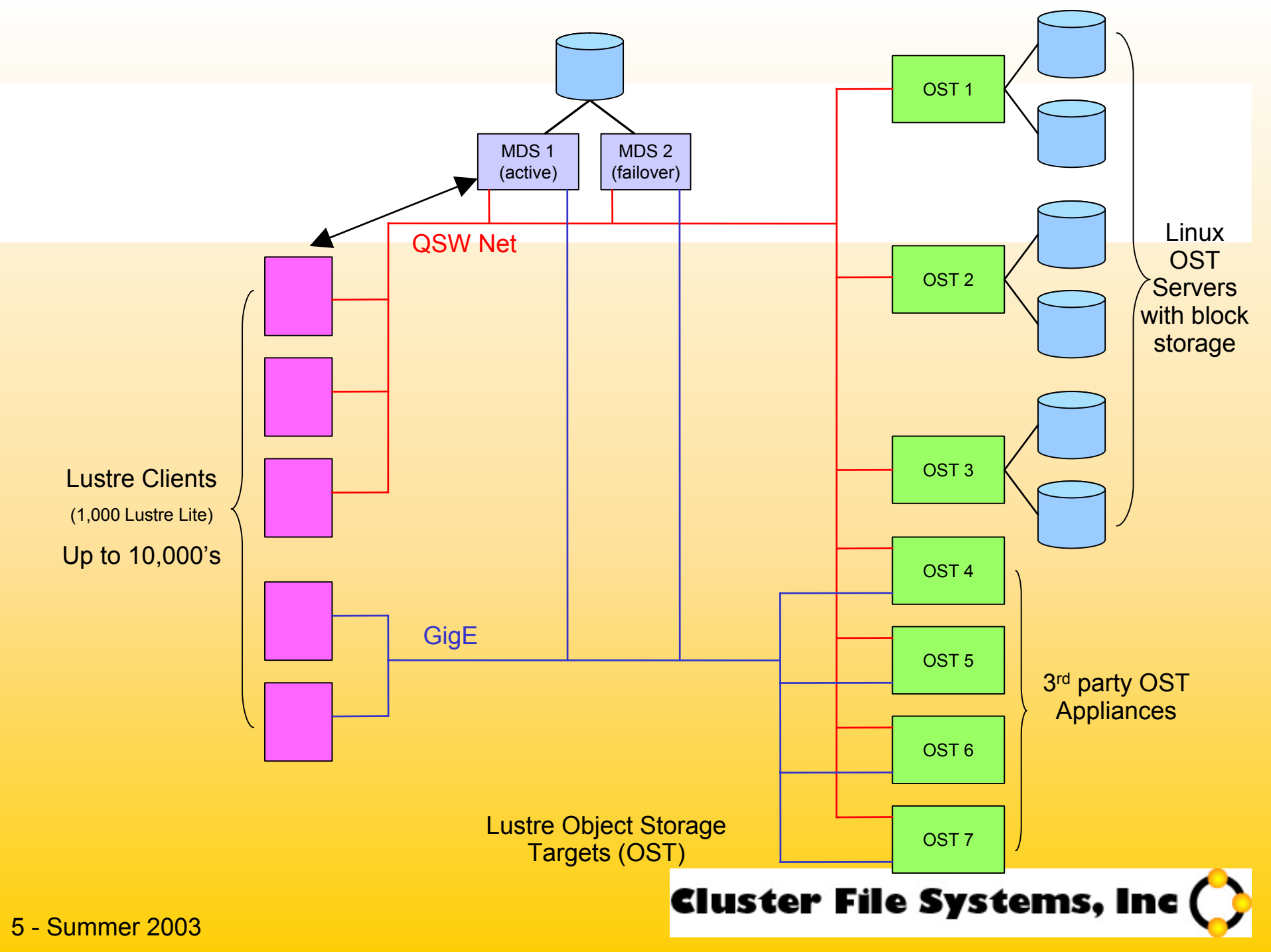
Topics

- Lustre introduction
- CFS, partners, friends
- Towards 1.0
- Lustre beyond a few months from now
- Conclusions

Lustre Introduction

- A fairly elaborate storage architecture
- Includes cluster file system for Linux
  - Stable on 2.4.x, making rapid progress for 2.6
- POSIX compliant
- Layering of object protocols
- Distributed lock management
- Separate metadata and file data servers
- No single points of failure


LDAP Server
configuration information,
network connection details,
& security management
Clients
directory operations,
meta-data, &
concurrency
file I/O & file
locking
Meta-data Server
(MDS)
recovery, file status, &
file creation
Object Storage Targets
(OST)



Cluster File Systems, Inc

CFS, partners, friends

CFS

- About 20 people
  - 2 executives, 1 program manager, 1 admin
  - 7 core team
  - ~10 other technical staff, mostly new
- Focused only on Lustre
- Mostly an internet based company
- Mostly a government contractor
- Practically everything is open source

Areas of focus

- AD – advanced development
  - Throw one away
  - Pathforward project with HP & Intel
  - Red Storm with Cray & Internal projects
- PQ – production quality
  - Keep us honest
  - LLNL, PNNL & Internal 1.0 project
- TP
  - Testing and performance

Thank you

- CMU, Linux people, Seagate & Lee Ward
  - Lots of initial design feedback
- Terry Heidelberg, Mark Seager, HP, PNNL
  - Make Lustre “Lite” real early at LNNL & PNNL
- Gary Grider & Bill Boas & several others
  - For dampening the turmoil and politics
- Dell, Cray, DDN, BlueArc
  - Make new development possible

CFS challenges – until now

- Manage
  - More than 2 developers
  - More than 0 customers
- The software engineering process
  - 8 months of intensive improvements
  - Track source, bugs, deliverables, tasks, hours
  - Entire company is on the web
- We feel reasonably organized

CFS challenges now – QA

- We do a lot, but not nearly enough
- Are building 10-15 people test team
  - Tracking stability of all changes is 24 hr job
- Are building better tests
  - File I/O: real jobs find bugs, test programs don't
  - Metadata: test suite has every bug we've found
  - Existing tests don't cut it

CFS challenges now - business

- Switch to a support model
  - Development contracts may slow down in 2004
- Looking at options for more development
  - Looking at grid and WAN storage management
  - Windows
  - Other Unixes
  - Key issue: what is a wise investment?

The real world

- 3 of the top 8 supercomputers run Linux.
- Lustre runs on all 3.
  - LLNL MCR: 1,100-node ia32 cluster (#3)
  - LLNL ALC: 950-node ia32 cluster (#6)
  - PNNL EMSL: 950-node ia64 cluster (#8)
- Installing in 2003-2004:
  - NCSA: 1,000 nodes
  - SNL/ASCI Red Storm: 8,000 nodes
  - LANL Pink: 1,000 nodes
- Chosen for ASCI PathForward SGS file system

Linux 2.6

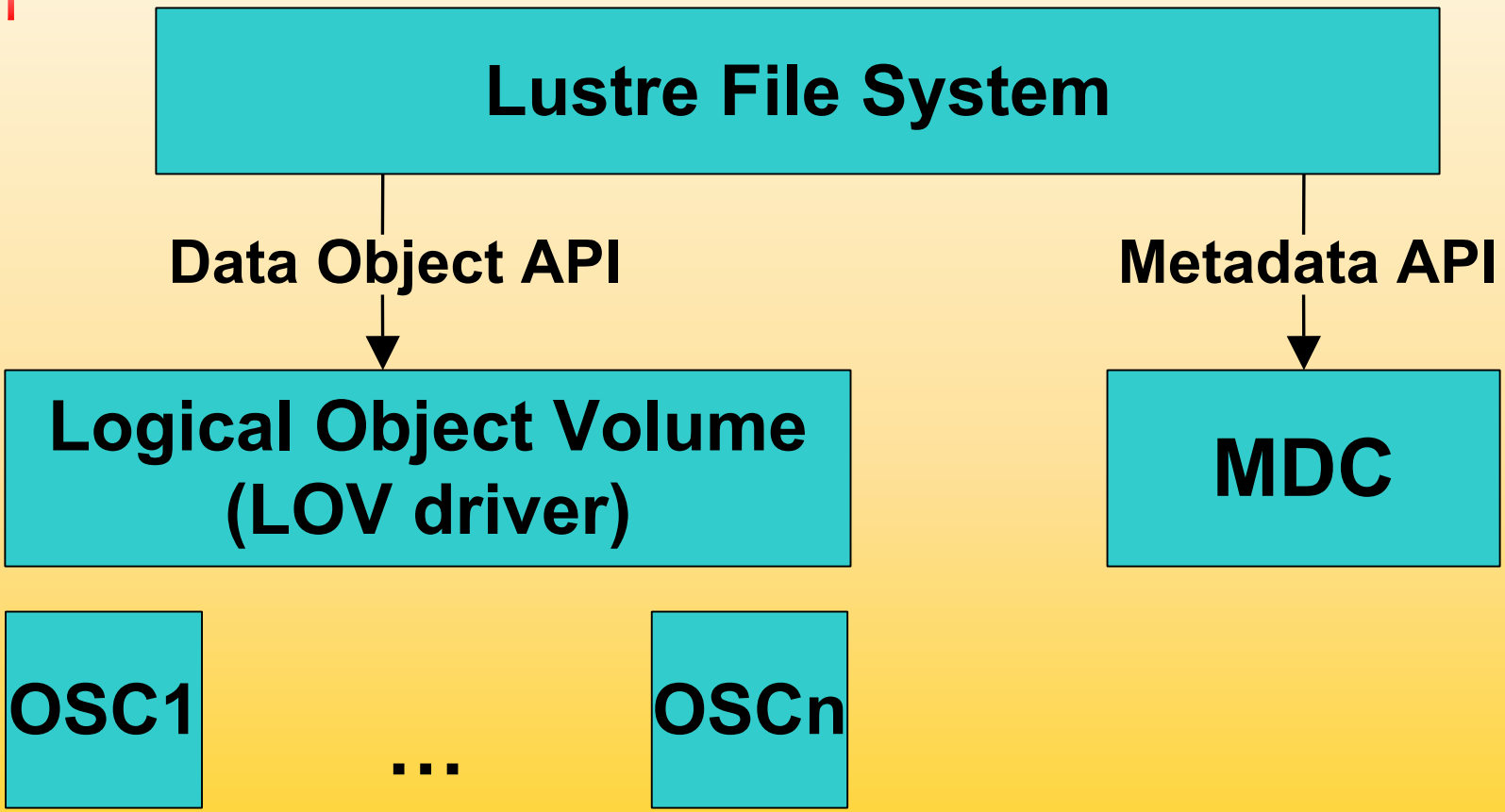
- Discussed remaining changes with Linus
  - Lustre basically ready for inclusion in 2.6
- All ext3 changes already in kernel
- 50% of VFS changes is now in

But …

- Have a fair amount of work before
  - Stability is rock solid
    - Eliminate “creeping doubt”
  - Performance is consistent
  - Build and installation process is smooth
- We need to manage this such that
  - New versions and updates can be trusted fast
- This is core focus of PQ team

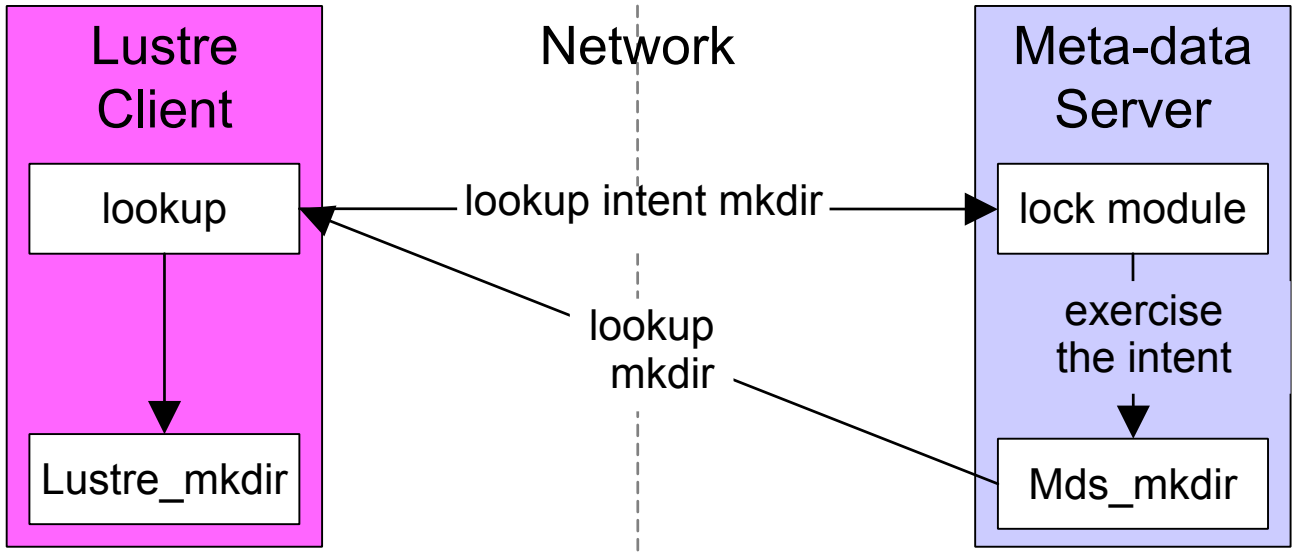
Lustre in the 1.0 timeframe

Lustre Client overview

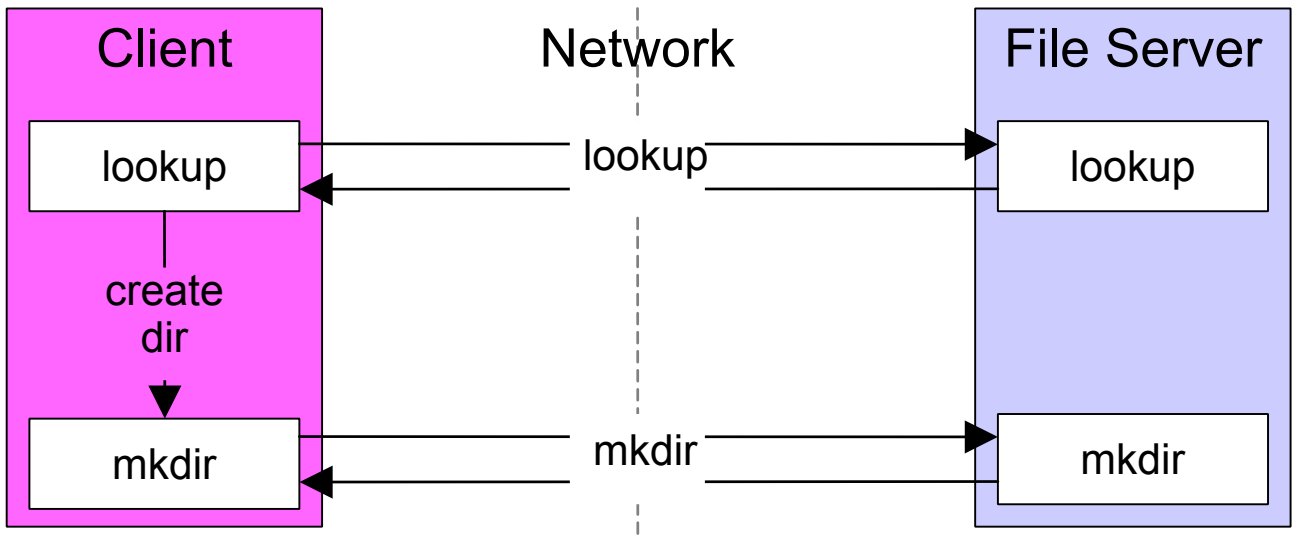


Issue 1: file i/o

- Sometimes see very good performance
  - Not consistently
  - Hence slowest drags us down
- Have pinpointed problems
  - Simplifications on backend:
    - direct I/O, extremely low CPU, very steady
  - Simplifications on client:
    - Treat OSC more like a block device
  - Some extra DMA in Elan networking



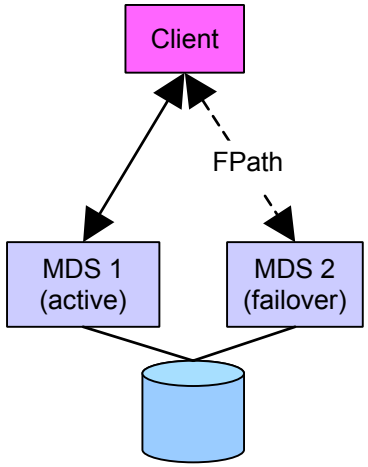
a) Lustre mkdir



b) Conventional mkdir

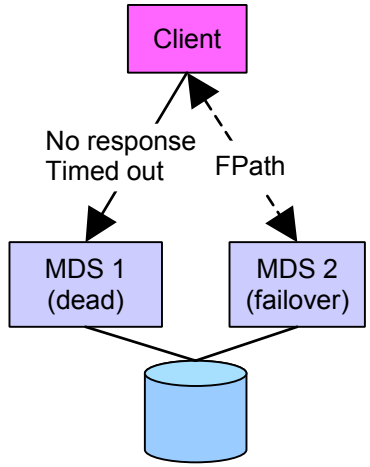
Issue 2: metadata fixes

- Metadata basically successful
- But - people find trouble for you
  - Like removing your cwd
  - Creating sockets or named pipes in our young baby
- Almost all MD fixes affect kernel patch
  - More elaborate to test and maintain
- Linus asked for small api changes
  - Backported immediately to 2.4

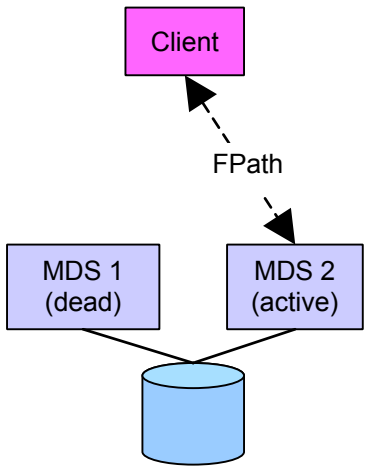


a) Normal functioning

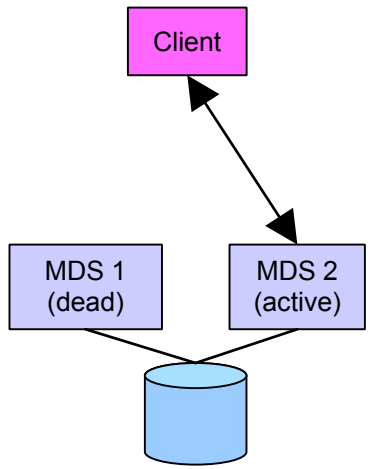
Fpath:
request path
in case of
MDS/OST
failover



b) MDS 1 fails to respond



c) Client finds new MDS



d) FPath connects newly active MDS 2

Issue 3: recovery

- Have had a breakthrough in testing
  - Expect rapid bug fixes
  - Goal: rebooting a server is 100% transparent
  - Failover & reboot have mostly same recovery
- Goal:
  - Client recovery never causes cluster problems
  - All recovery works 9 out of 10 cases
  - Should reduce downtime by 10x

Issue 4: testing

- Already discussed
  - No 1.0 until test regime can be trusted
- I/O tests
  - I2d_bench based
  - I2d_bench is a benchmark tool
    - Distributes I/O system calls across a cluster
- Recovery tests
- More frequent running
  - buffalo.lustre.org

Issue 5: configuration

- Yes:
  - mount –t lustre mds:/fileset /mnt/lustre
  - Recovery mostly free of client scripts

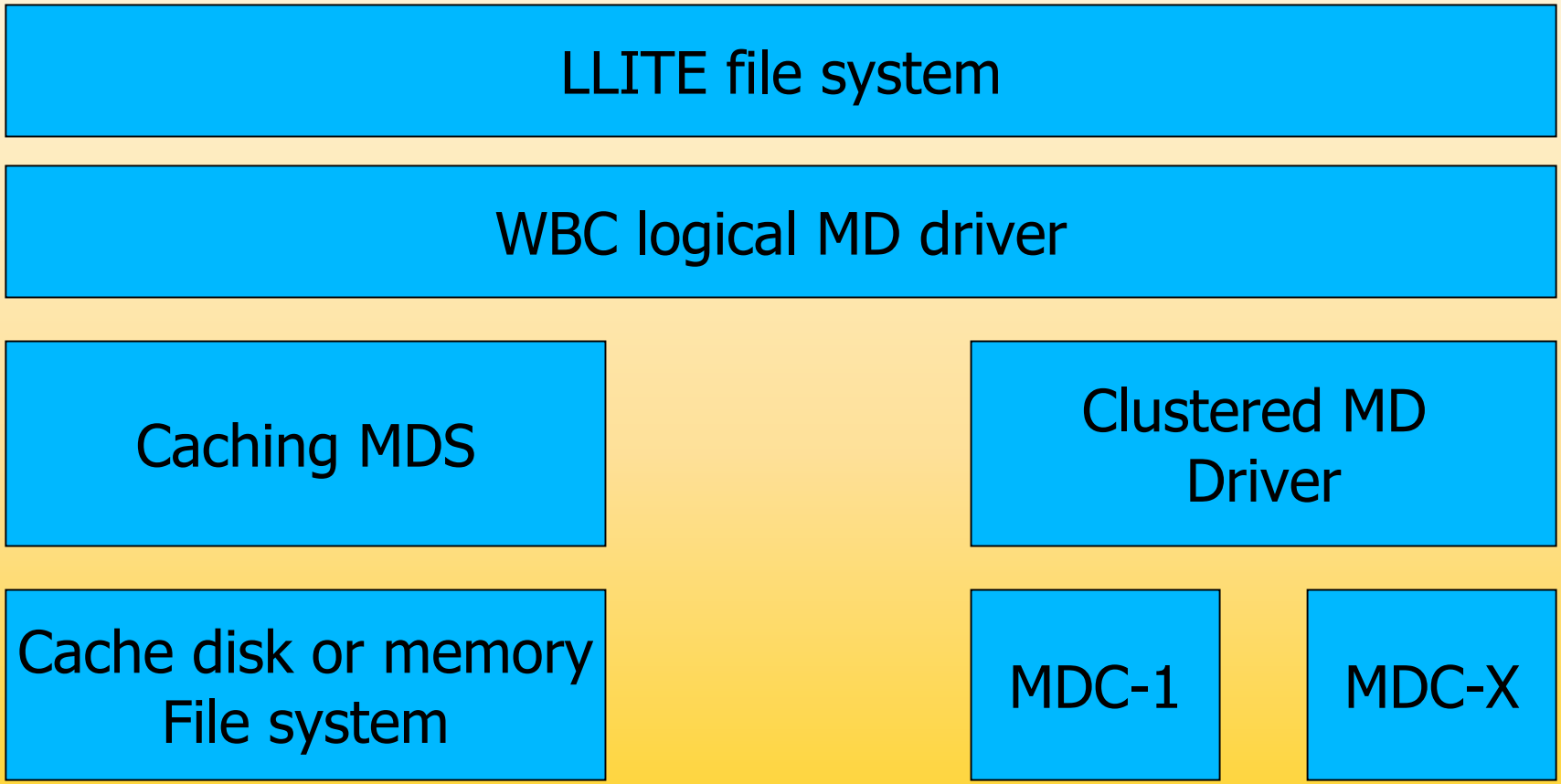
Lustre – future

Metadata

Metadata

- Performance improvements on the way
  - Better locking
  - Fewer rpc's
- MD writeback cache
  - Very fast updates, memory cache or
  - Persistent like AFS
  - Found extremely simple solution
    - Run a local MDS
- Working prototype late 2003

Caching & clustering



Clustering metadata (Pathforward)

- Again found very simple solution
  - Logical clustering metadata driver
  - Very similar to LOV

OST improvements

Caching OBD

- Extremely simple
  - Logical caching driver
  - Uses local object store
  - Uses normal OST client (osc)
- Implementation in collaboration with HP
  - First version will be read only cache

Redundant OST (PNNL)

- Replicating OBD
  - RAID 1 object raid
  - Goal redundancy
- Re-build for RAID 1
  - Cornerstone part of Lustre recovery
    - Build log items
    - Transmit to and use on other systems
    - Wait for commit of remote before cleaning up
  - Could probably become a WAN sync

Other platforms

LibLustre

- POSIX stdio library (userspace)
  - Currently only TCP support
  - Runs on Linux, Windows, everything
  - Will be very transparent (except mmap, exec)
- Liblustre lives in libbsdio
  - This is BSD VFS compiled in userspace
  - Liblustre extremely similar to BSD kernel client
  - Of course this choice is deliberate

LibWinFS – win32 access library

- Win32 Lustre library
- Components
  - Build interceptor (MS explains how)
    - Detours papers shows how
  - Wine expresses Win32 api in POSIX stdio
    - Use Wine FS components on Windows
    - Glue to cygwin liblustre on Windows

User level OST (& MDS?)

- Build user level Portals server framework
- Build prototype of OST
  - Show lock management
  - Look at recovery
  - Look at configuration
- Deep question: why kernel servers?
  - A must if client and server on one system
    - Caching MDS, caching OBD
  - For large servers perhaps no good reason

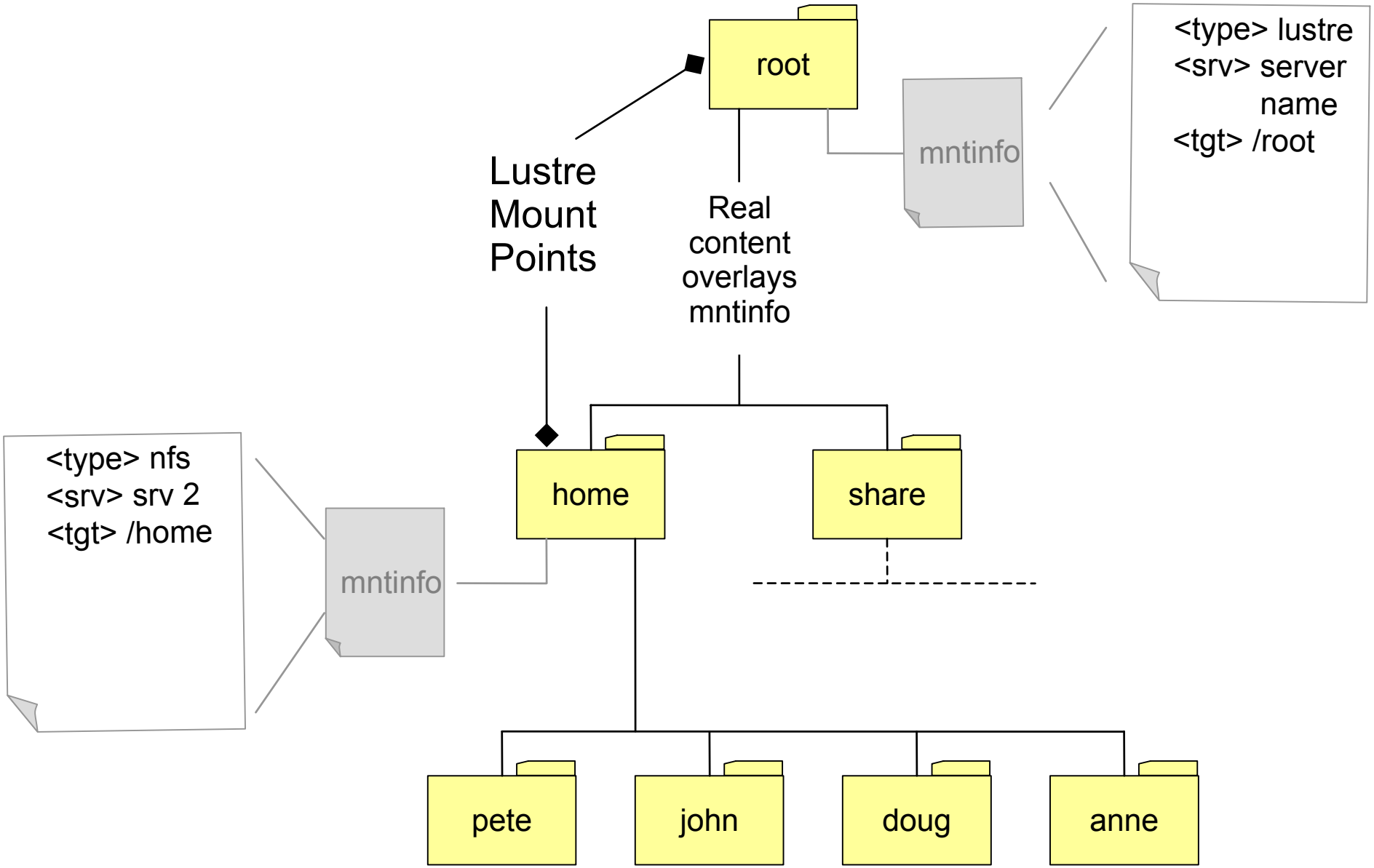
Storage Management

Storage management

- Dynamic addition & removal of OSTs
- Hot data migration to new OST's
- QOS
  - Some guarantees of quality
  - Better space management

HSM & Snapshots & Backup

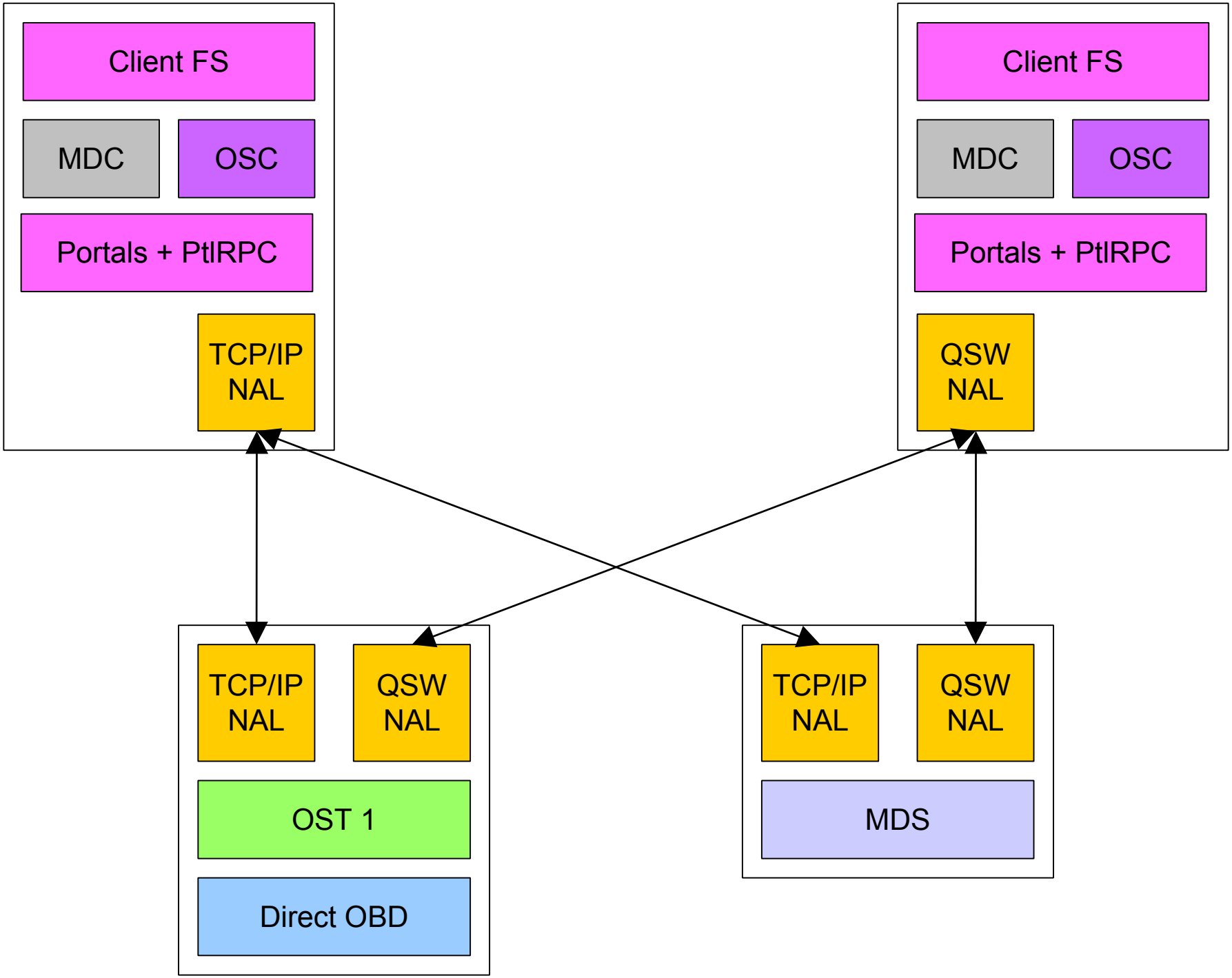
- Probably coming in 2004
- HSM
  - Probably XDSM api
- Snapshot
  - Very similar to Waffle snapshots
  - Probably integrated into ext3
- NDMP server
  - To make commercial backup work


root
Lustre
Mount
Points
Real
content
overlays
mntinfo
mntinfo
<type> lustre
<srv> server
name
<tgt> /root
<type> nfs
<srv> srv 2
<tgt> /home
mntinfo
home
share
pete
john
doug
anne

File-sets

- AFS Volumes will be available
- Carefully designed to offer
  - AFS advantages
  - SUN autofs4 advantages
- Very simple

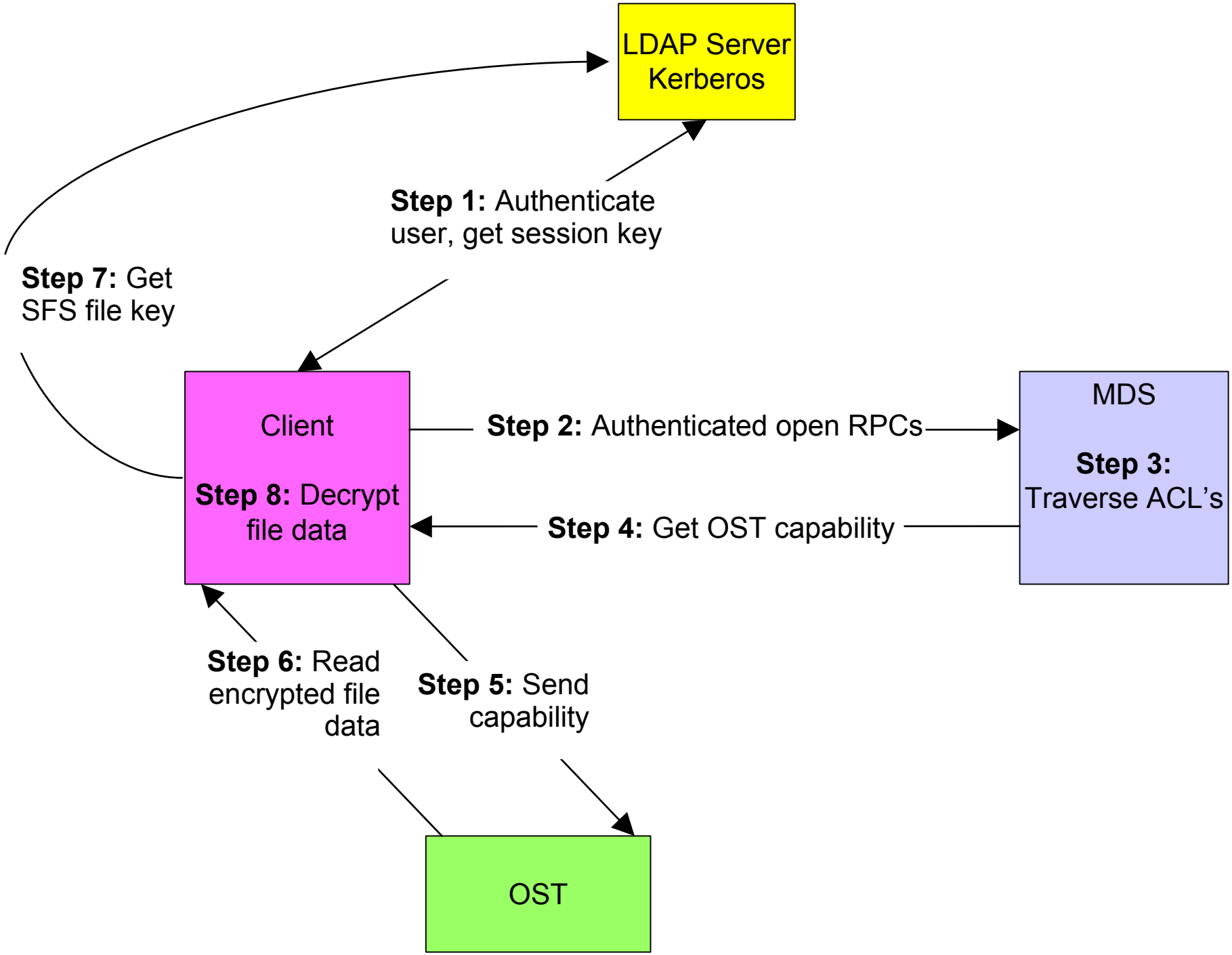
Networking


Client FS
MDC
OSC
Portals + PtlRPC
TCP/IP NAL
Client FS
MDC
OSC
Portals + PtlRPC
QSW NAL
TCP/IP NAL
QSW NAL
OST 1
Direct OBD
TCP/IP NAL
QSW NAL
MDS

Networking

- Myrinet
  - Late 2003
  - Several non-competing efforts here
- Infiniband
  - LANL
  - CFS would love user space I/B NAL

Security


LDAP Server
Kerberos
Step 1: Authenticate user, get session key
Step 7: Get SFS file key
Client
Step 8: Decrypt file data
Step 2: Authenticated open RPCs
MDS
Step 3: Traverse ACL's
Step 4: Get OST capability
Step 6: Read encrypted file data
Step 5: Send capability
OST

Security

- Authentication
  - GSS (e.g. Kerberos, PKI, or simpler)
- Authorization
  - POSIX ACL's
- Privacy
  - Client side file crypto with "project keys"
  - This is STK's SFS
- Composition of existing technology

Conclusions

Lots of progress & lots of work

- Two years ago: I said it would NEVER work
  - That wasn't true
- Becoming mature and solid is hard
  - But there is fast progress
  - Customers keep us honest
- Path Forward Effort
  - Demands radically new technology
- We are lucky & having fun!