Lustre Scalability

Eric Barton
Lead Engineer, Lustre Group
Sun Microsystems

Topics

HPC Trends

Architectural Improvements

General Performance Enhancements

HPC Trends

Processor performance / RAM growing faster than I/O

Relative number of I/O devices must grow to compensate

Storage component reliability not increasing with capacity

Failure is not an option – it’s guaranteed

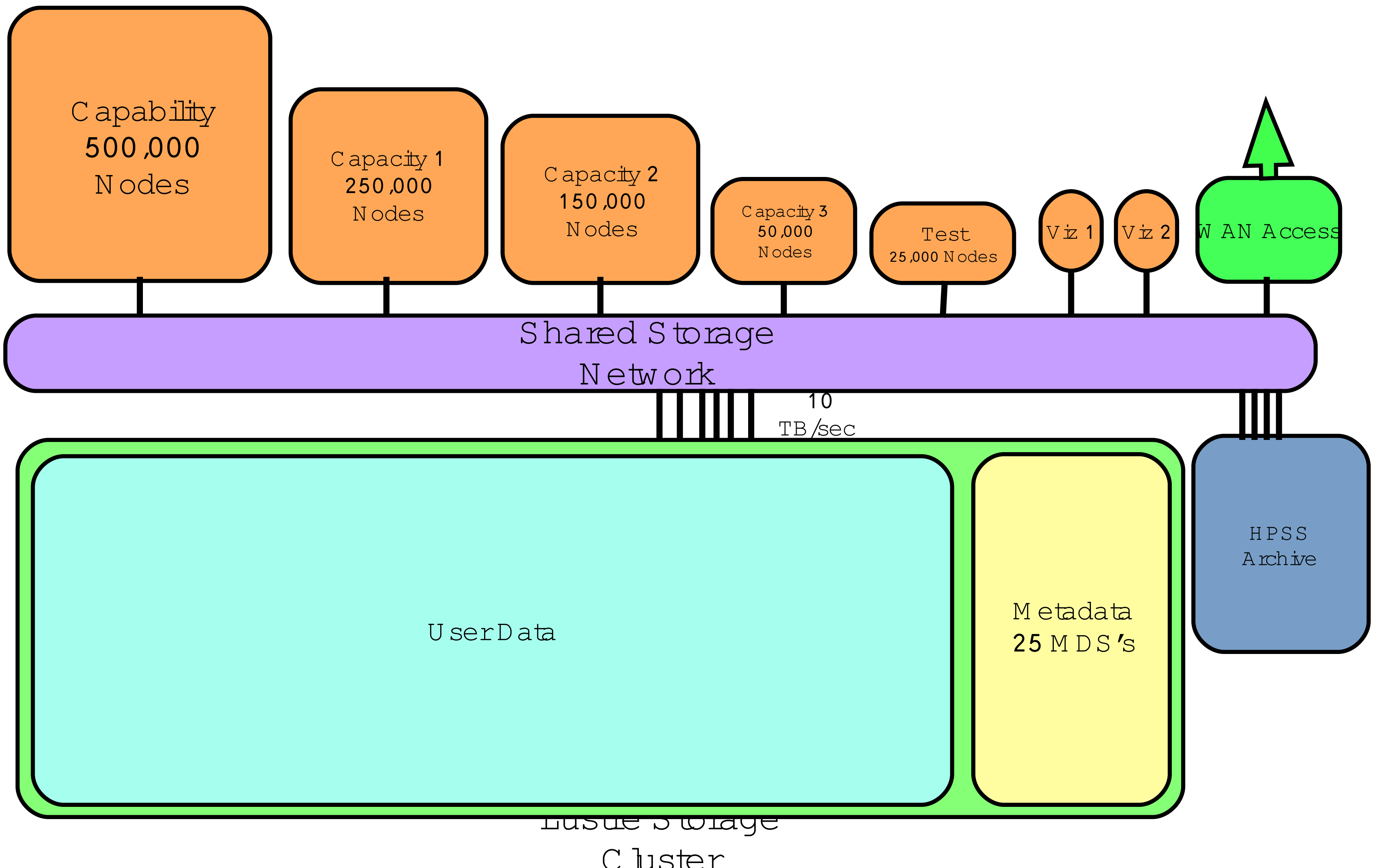
Trend to shared file systems

Multiple compute clusters

Direct access from specialized systems

Storage scalability critical

HPC Center of the Future



Lustre Scalability

Definition

Performance / capacity grows nearly linearly with hardware

Component failure does not have a disproportionate impact on availability

Requirements

Scalable I/O & MD performance

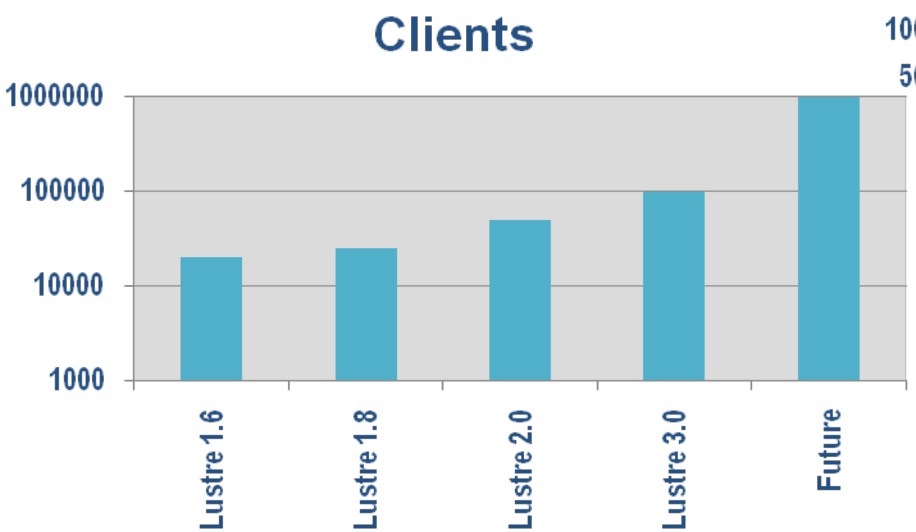
Expanded component size/count limits

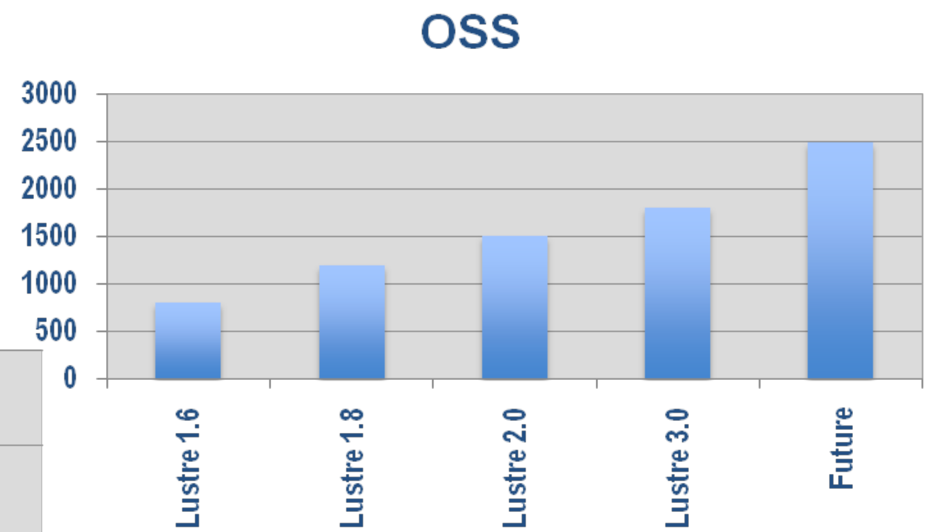
Increased robustness to component failure

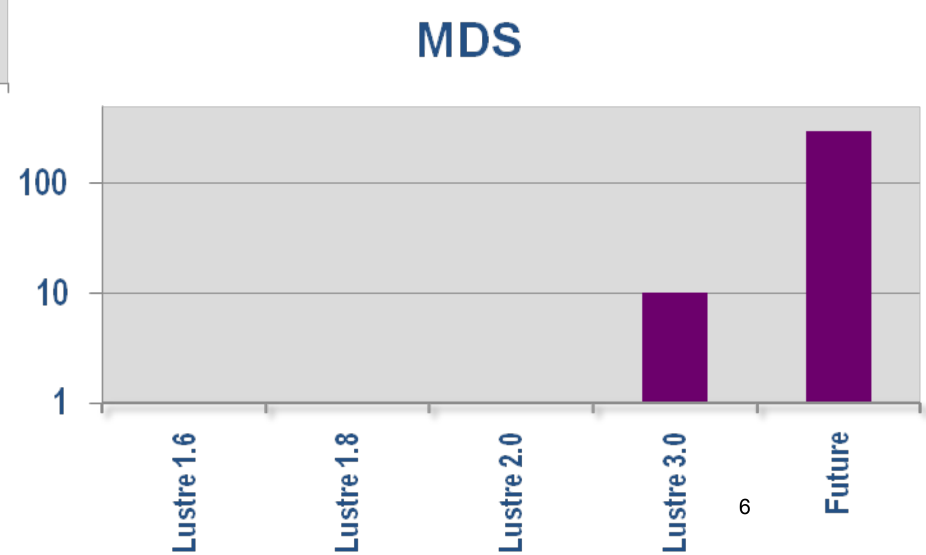
Overhead grows sub-linearly with system size

Timely failure detection & recovery

Lustre Scaling







Architectural Improvements

Clustered Metadata (CMD)

10s – 100s of metadata servers

Distributed inodes

Files local to parent directory entry / subdirs may be non-local

Distributed directories

Hashing ⇔ Striping

Distributed Operation Resilience/Recovery

Uncommon HPC workload

Cross-directory rename

Short term

Sequenced cross-MDS ops

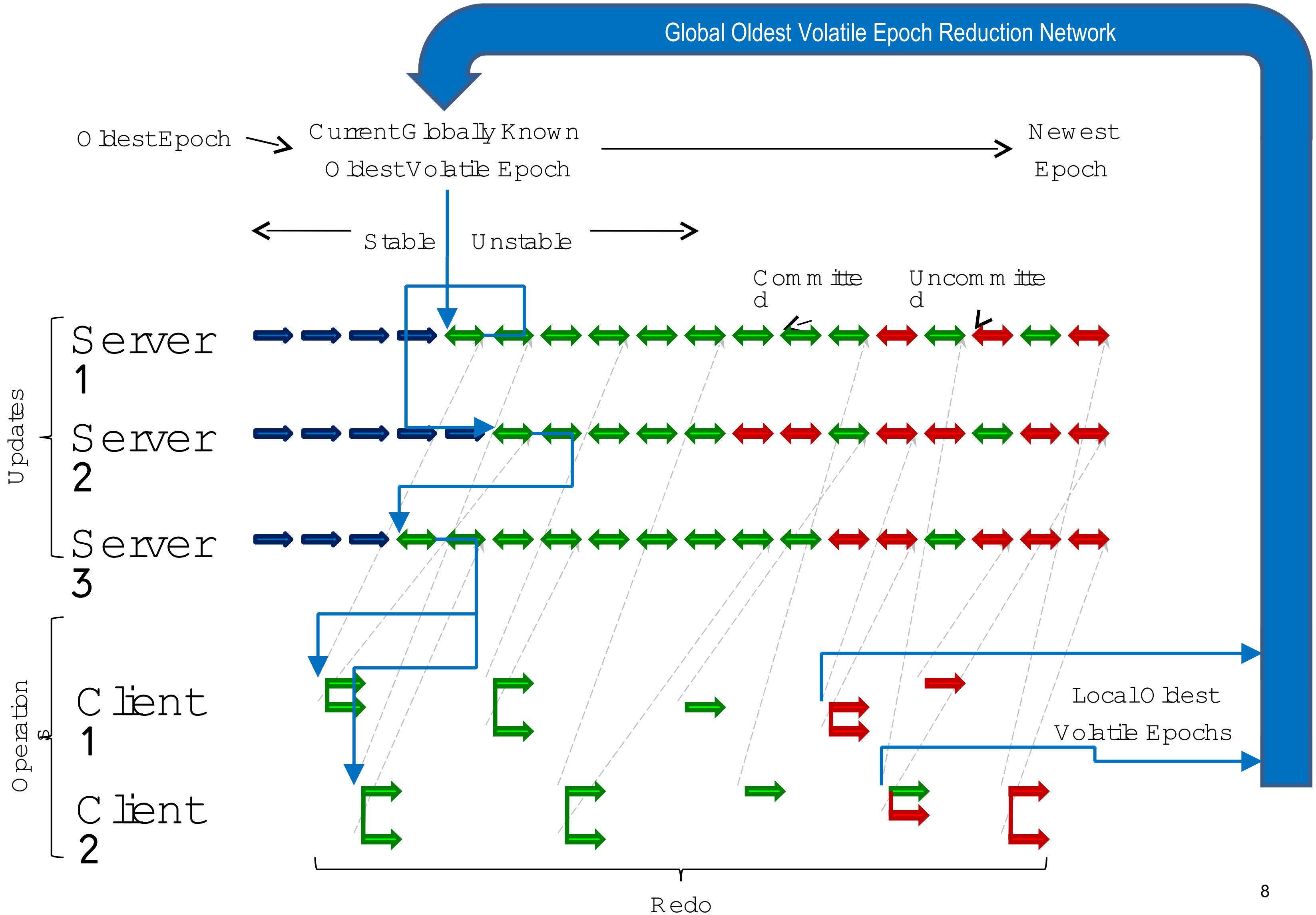
Longer term

Atomicity **C**onsistency **I**solation **D**urability

Non-blocking - deeper pipelines

Hard - cascading aborts, synch ops

Epochs


Global Oldest Volatile Epoch Reduction Network
Oldest Epoch
Current Globally Known Oldest Volatile Epoch
Newest Epoch
Stable
Unstable
Committed
Uncommitted
Updates
Server 1
Server 2
Server 3
Operations
Client 1
Client 2
Local Oldest Volatile Epochs
Redo

Architectural Improvements

Fault Detection Today

RPC timeout

Timeouts must scale O(n) to distinguish death / congestion

Pinger

No aggregation across clients or servers

O(n) ping overhead

Routed Networks

Router failure can be confused with end-to-end peer failure

Fully automatic failover scales with slowest time constant

Many 10s of minutes on large clusters ☹

Failover could be much faster if “useless” waiting eliminated ☺

Architectural Improvements

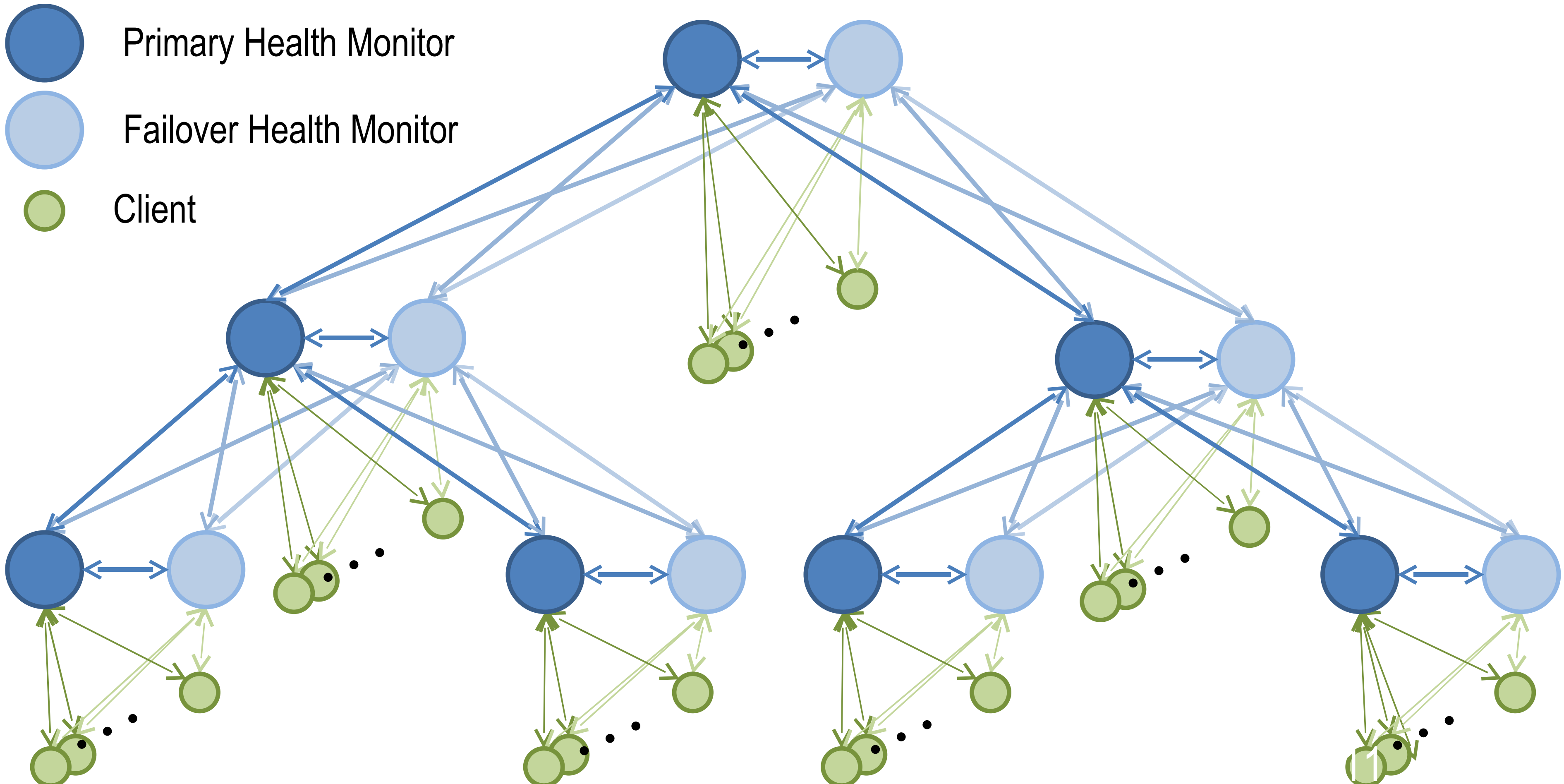
Scalable Health Network

- Burden of monitoring clients distributed – not replicated
  - ORNL – 35,000 clients, 192 OSSs, 7 OSTs/OSS
- Fault-tolerant status reduction/broadcast network
  - Servers <u>and</u> LNET routers
- LNET high-priority small message support
  - Health network stays responsive
- Prompt, reliable detection
  - Time constants in seconds
  - Failed servers, clients and routers
  - Recovering servers and routers

Interface with existing RAS infrastructure

- Receive and deliver status notification

Health Monitoring Network



Architectural Improvements

Metadata Writeback Cache

Avoids unnecessary server communications

Operations logged/cached locally

Performance of a local file system when uncontended

Aggregated distributed operations

Server updates batched and tranferred using bulk protocols (RDMA)

Reduced network and service overhead

Sub-Tree Locking

Lock aggregation – a single lock protects a whole subtree

Reduce lock traffic and server load

Architectural Improvements

Current - Flat Communications model

Stateful client/server connection required for coherence and performance

Every client connects to every server

O(n) lock conflict resolution

Future - Hierarchical Communications Model

Aggregate connections, locking, I/O, metadata ops

Caching clients

Aggregate local processes (cores)

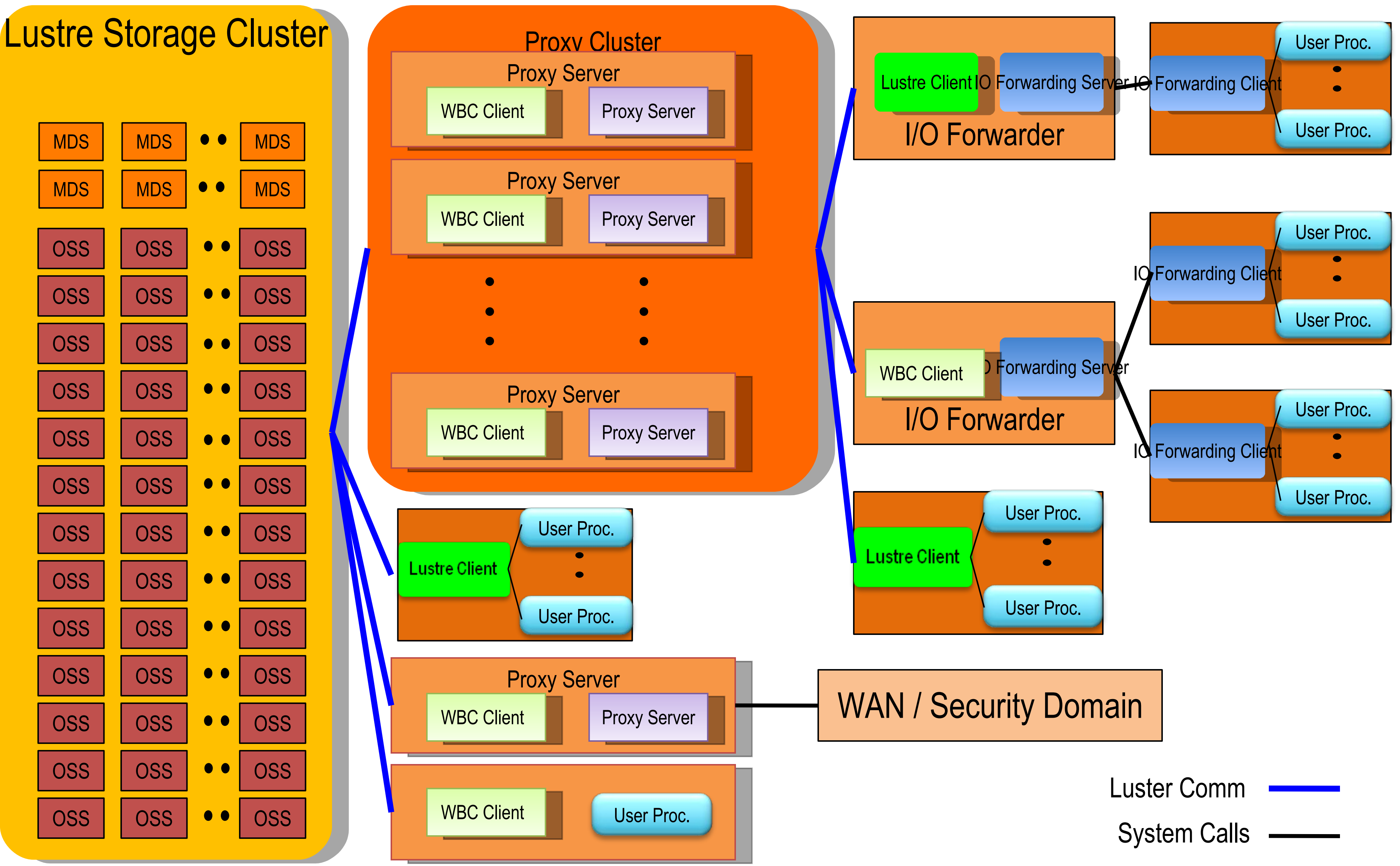
I/O Forwarders scale another 32x or more

Caching Proxies

Aggregate whole clusters

Implicit Broadcast - scalable conflict resolution

Hierarchical Communications



Performance Improvements

Network Request Scheduler (NRS)

Much larger working set than disk elevator

Higher level information - client, object, offset, job/rank

Prototype

Initial development on simulator

Scheduling strategies - quanta, offset, fairness etc.

Testing at ORNL pending

Future

Exchange global information - gang scheduling

QoS - Real time / Bandwidth reservation (min/max)

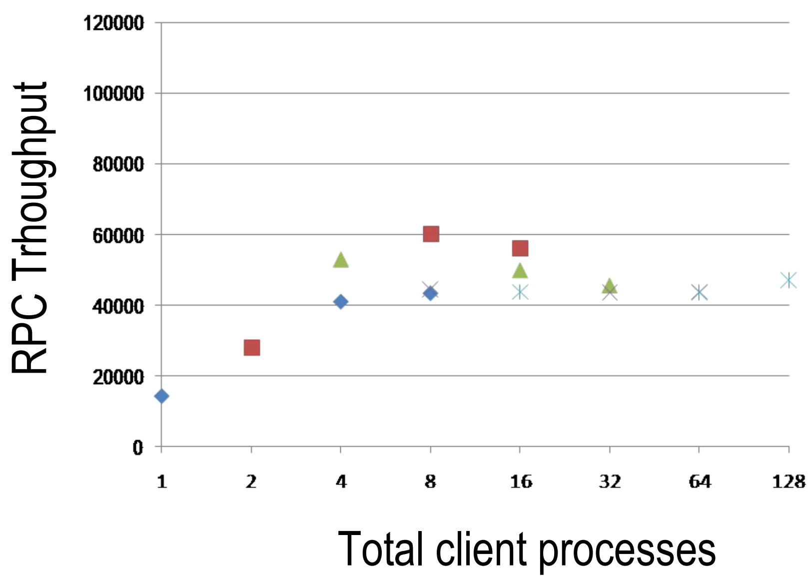
Performance Improvements

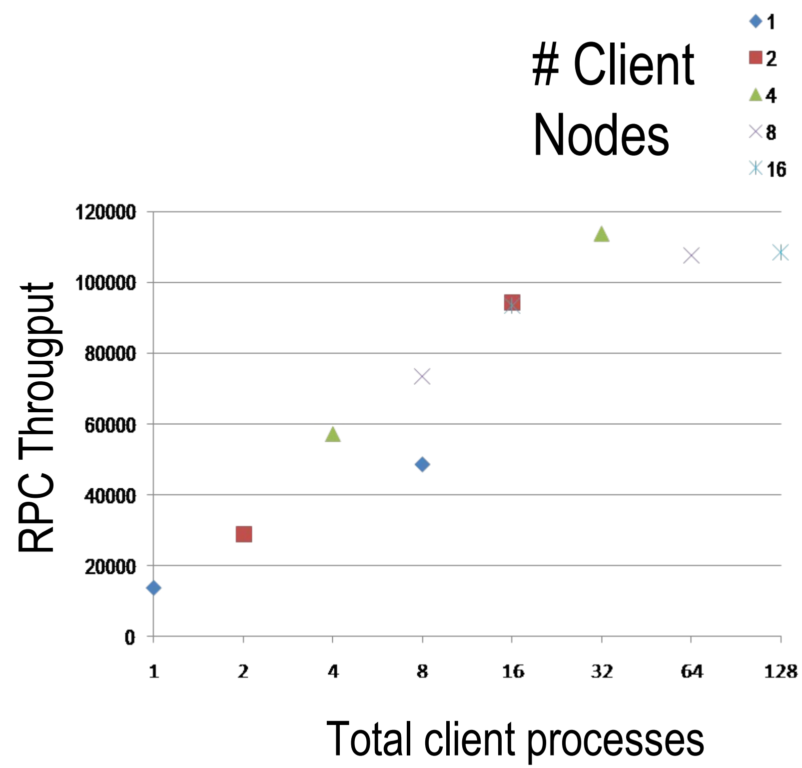
SMP Scaling

Improve MDS performance / small message handling

CPU affinity

Finer granularity locking





Performance Improvements

Metadata Protocol

Size on MDT (SOM)

- Avoid multiple RPCs for attributes derived from OSTs
- OSTs remain definitive while file open
- Compute on close and cache on MDT

Readdir+

- Aggregation
  - Directory I/O
  - Getattrs
  - Locking

Performance Improvements

ZFS

Remove ldiskfs size limits

End-to-end data integrity

Hybrid storage

Channel bonding

Combine multiple Network Interfaces

Failover

Capacity

Performance Improvements

Rebuild performance

Frequent disk failures

Rebuild quickly to prevent data loss on next failure

Disk group remains in operation during rebuild

Avoid using OST during rebuild

Speed rebuild

Amdahl's law

ZFS rebuild improvements

Lustre Scalability

Attribute	Today	Future
Number of Clients	10,0000s Flat comms model	1,000,000s Hierarchical comms model
Server Capacity	Ext3 - 8TB	ZFS - Petabytes
Metadata Performance	Single MDS	CMD
Recovery Time	RPC Timeout - O(n)	Health Network – O(log n)

THANK YOU

Eric Barton
lustre-devel@lists.lustre.org