Lustre Architecture
Miami, April 2007

Peter J. Braam, PhD
Founder, CEO & President
Cluster File Systems, Inc.

V 1.0, pjb, 2007-04

Cluster File Systems, Inc™

Contents

- **Recent & in progress development**
  - Metadata
  - IO & file system
  - Scalability
  - Recovery
- **Robustness**
- **Possible future development**
  - User level servers
  - pNFS

Caveats

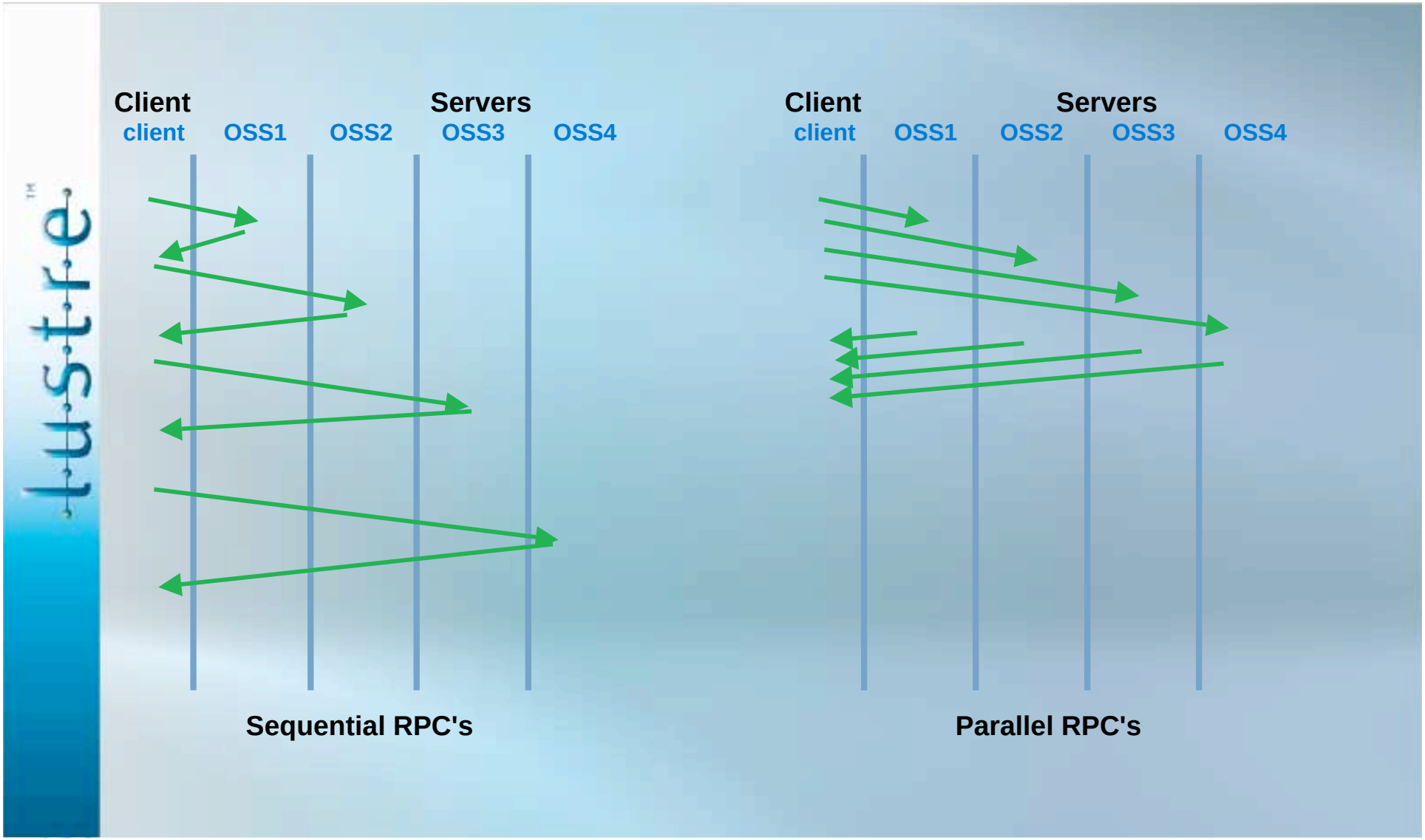
- **This is a one hour presentation**
  - I kept about 20 minutes for questions
  - There is much more going on
- **Deliberately I will say little about delivery dates**
  - Speak with Peter Bojanic

Metadata

Parallel Operations

- **Operations to the OSS nodes**
  - Obtaining file sizes of open files
  - Deletion of objects
  - Lock acquisitions – when possible
- **Dramatic performance consequences**
  - Parallel processing on the servers
- **Typical use cases**
  - “ls -l”,
  - “rm”

Parallel operations



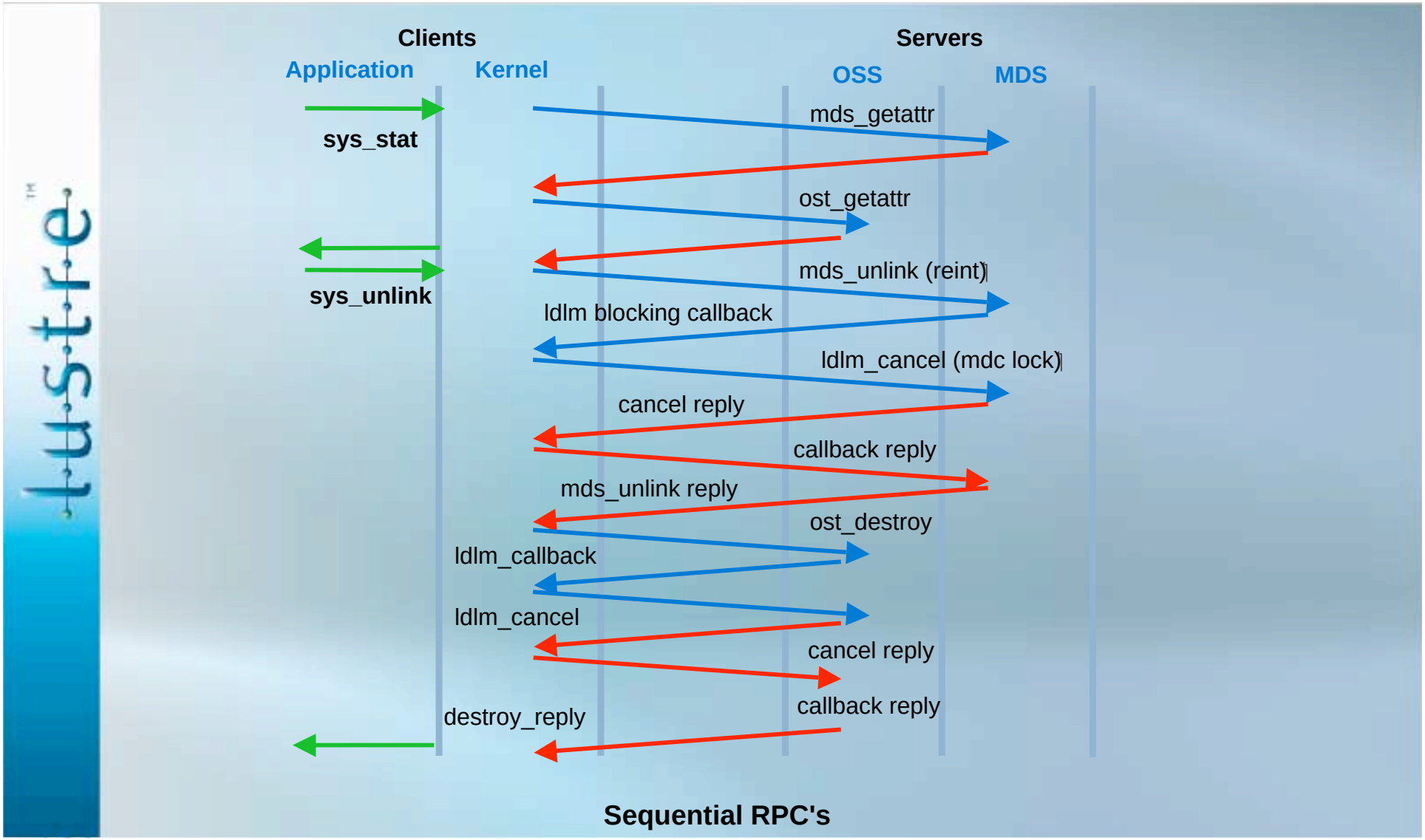
Early lock cancellation

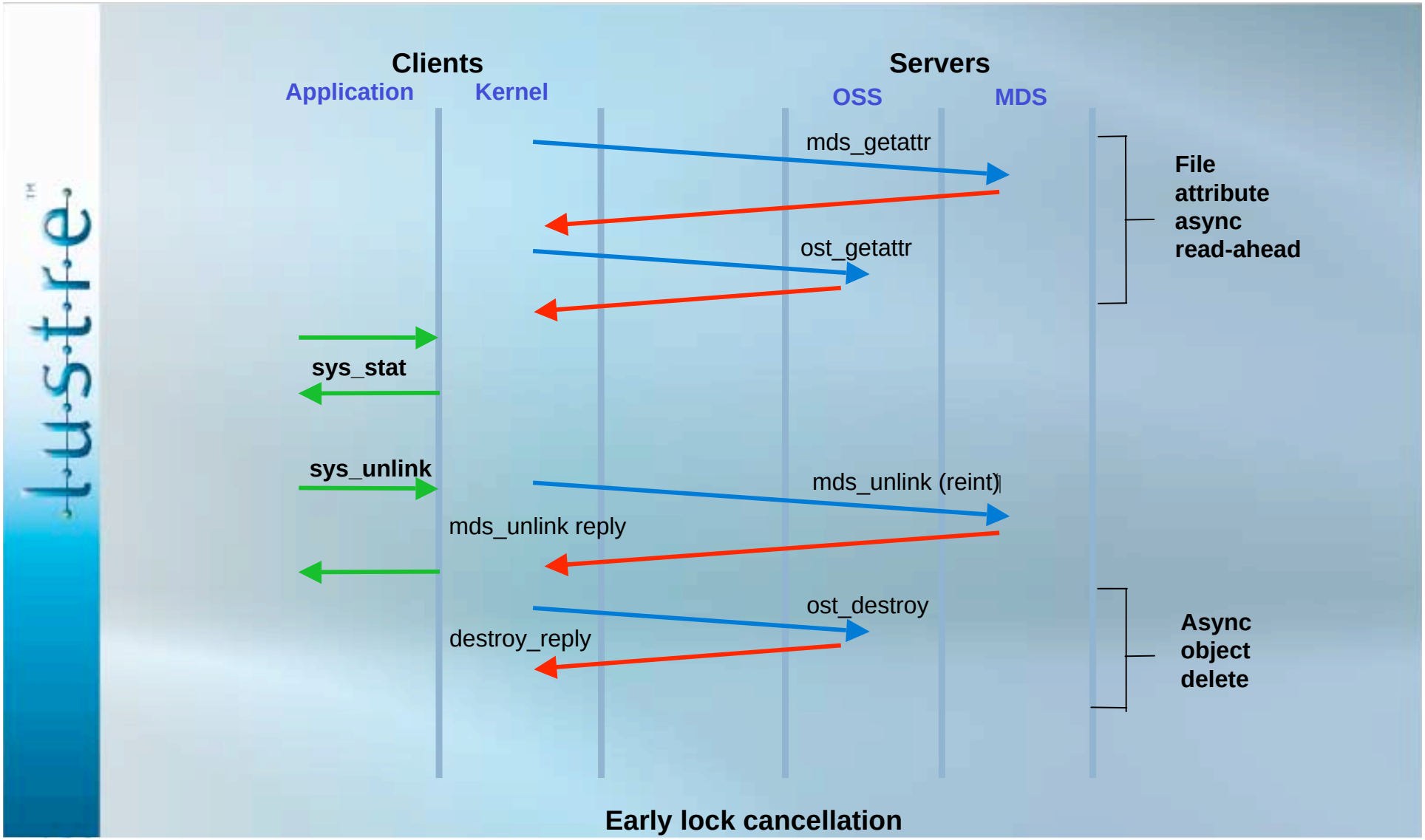
- **Clients obtain locks**
  - Sometimes it is known that they will see a cancellation callback
  - Early cancellations eliminate the callback
- **Performance improvement**
  - Serious RPC reduction
- **Use cases**
  - removing files, directories
  - rename
  - link
  - updating attributes
- **Example GNU “rm” file utility**
  - Does a stat on the file
  - Then makes an unlink system call

Directory & attribute read-ahead

- **RPC intensive patterns**
  - read directory pages
  - operate on all inodes in the page
- **Attribute read-ahead:**
  - fetch attributes from MDS and OSS in the background
- **Common use cases**
  - list all files in a directory
    - readdir
    - get mds inode attributes
    - get oss object attributes
  - remove all files in a directory
    - readdir
    - get mds inode, oss object attributes
    - unlink inode, object

common “rm file” – wait for 8 RPC's





Patchless clients

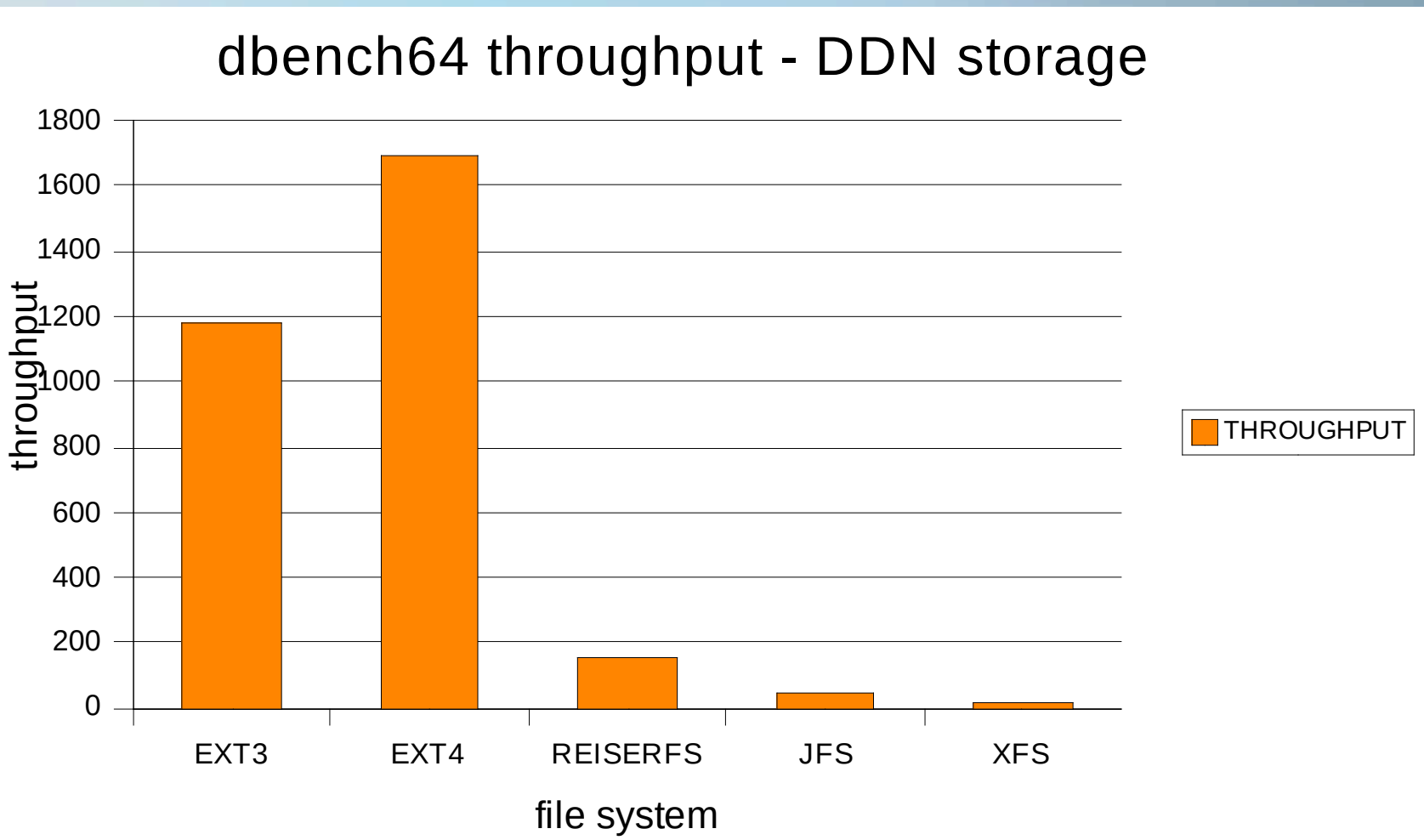
- **Lustre 1.6 supports patchless clients**
  - with kernels newer than 2.6.16
- **This adds some VFS / Lustre client interactions**
  - required without kernel patches
- **Example:**
  - patched “unlink” - just lustre_unlink operation
  - patchless “unlink” - lustre_lookup, then lustre_unlink
- **However, in practice “unlink” is done by GNU “rm”**
  - “rm” already does a lookup anyway
  - the extra interaction comes from the cache
  - no extra overhead to speak off!

IO

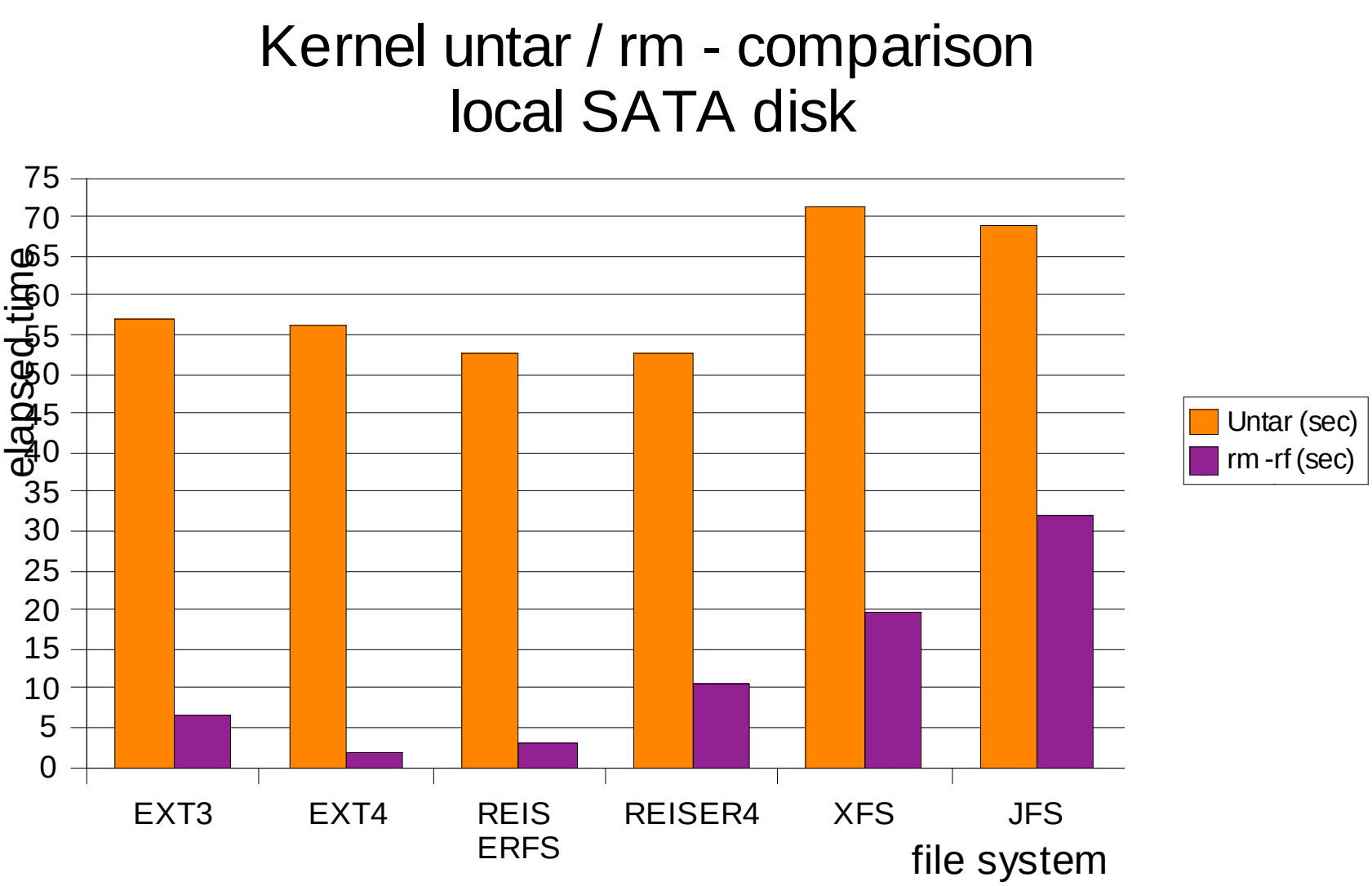
New disk allocator

- **Block allocation policies**
  - Write a little (e.g. <64K) before small offset (e.g. 64K)
    - Place the write in a “small file” area on the disk
  - Keep such small writes together
  - Large writes are aligned in 1-4MB chunks
  - Writes at significant offset are logically and physically aligned
- **Outcome – smoking performance**
  - It appears that this is the crux for small file performance
  - The secret of Reiser was to write things close together
- **Typical use cases**
  - liblustre
  - small file performance

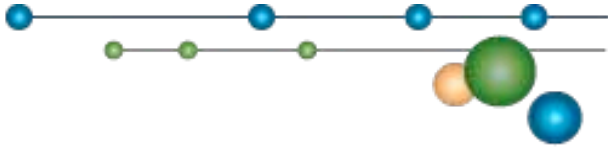
New allocator dbench64 throughput



kernel untar / remove with new allocator



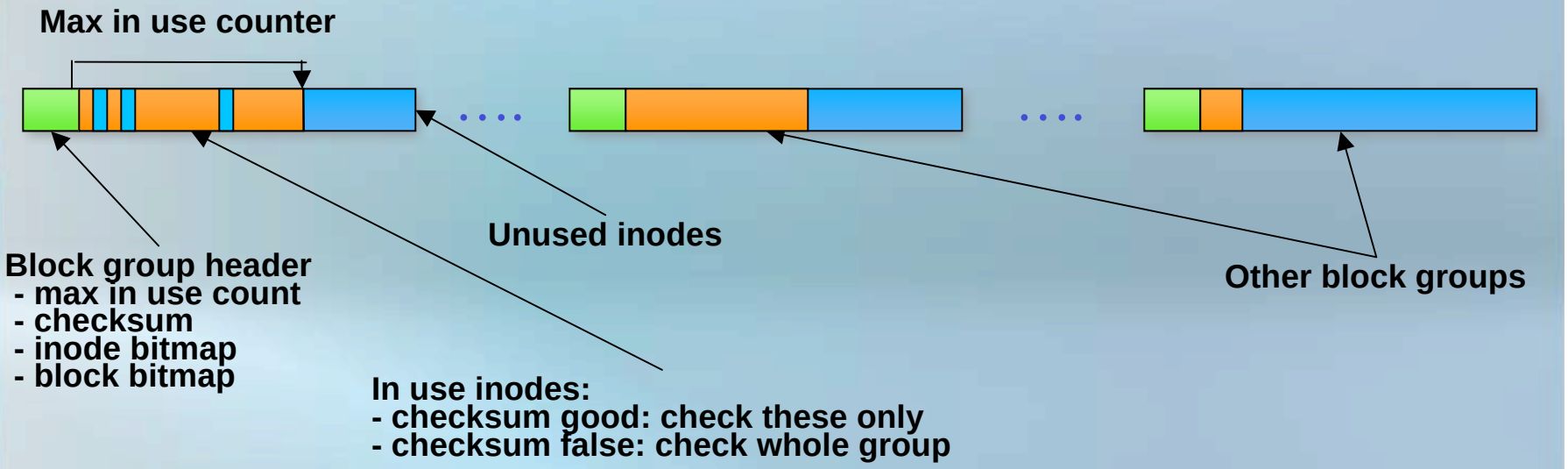
OSS writeback cache



- **Some jobs send very small IO's to the disk arrays**
  - aggregation is important
    - Lustre so far does no caching on the OSS
    - Liblustre clients have no cache (Linux clients do)
- **Lustre OSS servers will get a cache**

Fast FSCK & Format

- **FSCK has changed**
  - Previously fsck scanned all inodes
  - Now only inodes that are possibly in use
- **The most interesting part of this is a checksum**
  - The checksum indicates if the metadata that follows is consistent
  - If it is the counter can be used to check up to the maximum inode
- **Speedups of 4x to 10x**
  - Good, but fsck needs to disappear completely, it doesn't scale



IO and locking

- **Stripe locking**
  - Change from
    - Lock all stripe extents, do all IO in parallel, unlock all
  - To
    - For all stripes in parallel: lock, do IO, unlock
  - Holding locks from multiple servers
    - Can lead to cascading recovery events on many servers
    - Is necessary for truncate and O_APPEND writes
- **Disallow client locks under contention**
  - When an extent in a file sees concurrent access
    - Ask the client to write through to the server
  - This eliminates callback traffic and cache flushes

Stability & robustness

Architectural improvements for robustness

- **OSS server cleanup**
  - OSS had a hand-crafted IO path
    - Like directio, but bypassing locking, other kernel limitations
  - We have restructured this with normal VFS calls
    - Also in preparation for the user level server
- **Revisit the layering**
  - Many bugs appear to be caused by violating layers
  - E.g.
    - using an unpublished kernel api and misunderstanding it
    - Receiving callbacks close to the network, but needing to remove pages in the file system
  - On the server this is doable - implementation in progress
  - On the client we are still puzzling

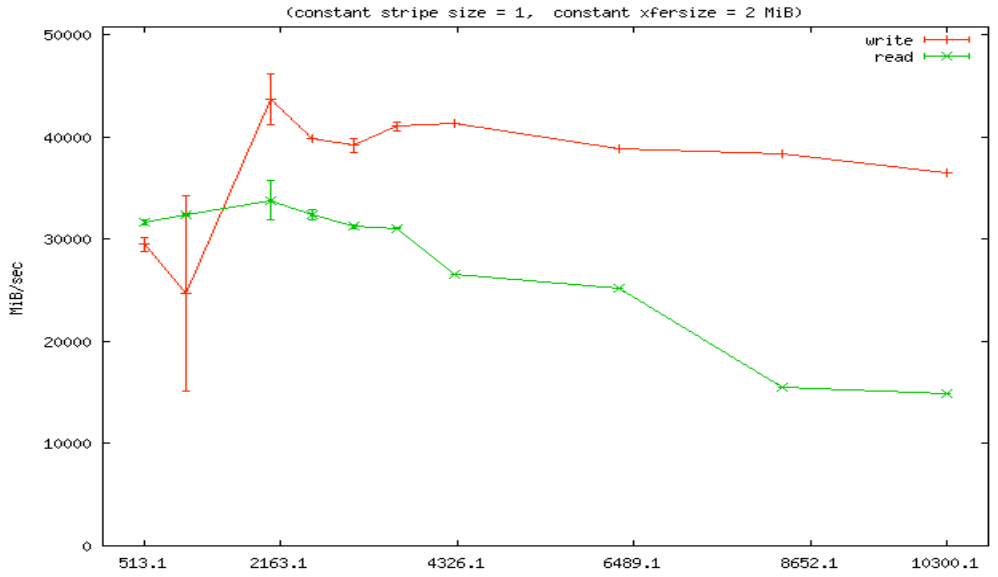
Future issues

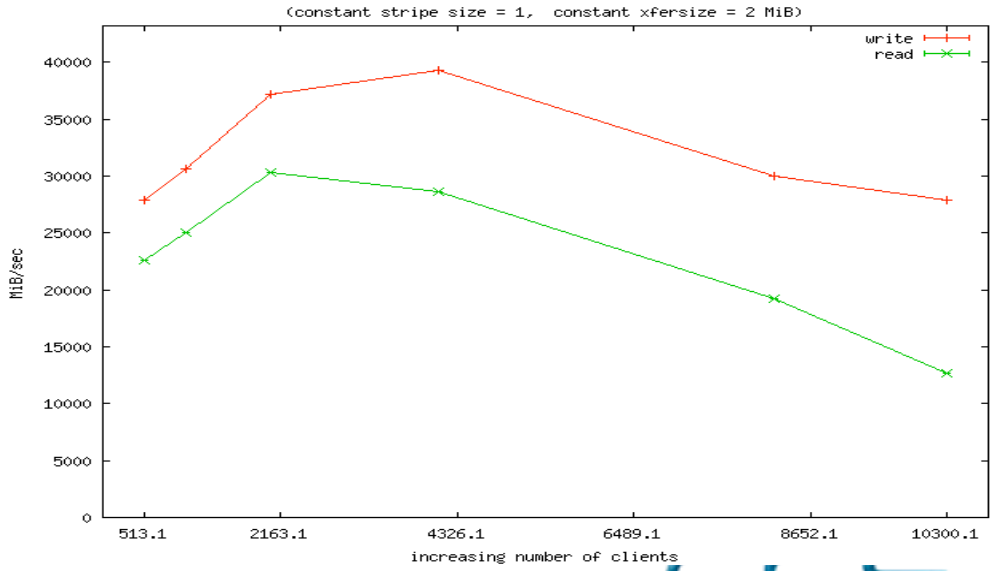
FS for 1PF system

- **Required 1TB/sec, FS will be many PBs**
- **CEA has servers: 2GB/sec**
  - Most promising solution: 500 OSS servers of this type
- **Lustre**
  - Already has installations with ~500 servers
  - Already has installations with ~2GB/sec servers
  - Already handling 25,000 clients on one FS in production today
- **10TB/sec requires some scalability improvements**

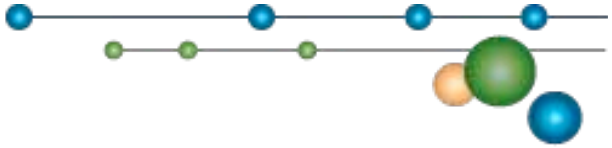
Red Storm – and why an LRE is useful

- ~40GB/sec
- File per process (top)
- Shared file (bottom)
  - 160 wide stripe
- Scales to 10,000 clients
- Array misconfigured for reads
  - Too much read ahead
- OST's misconfigured for shared file
  - Not enough disks





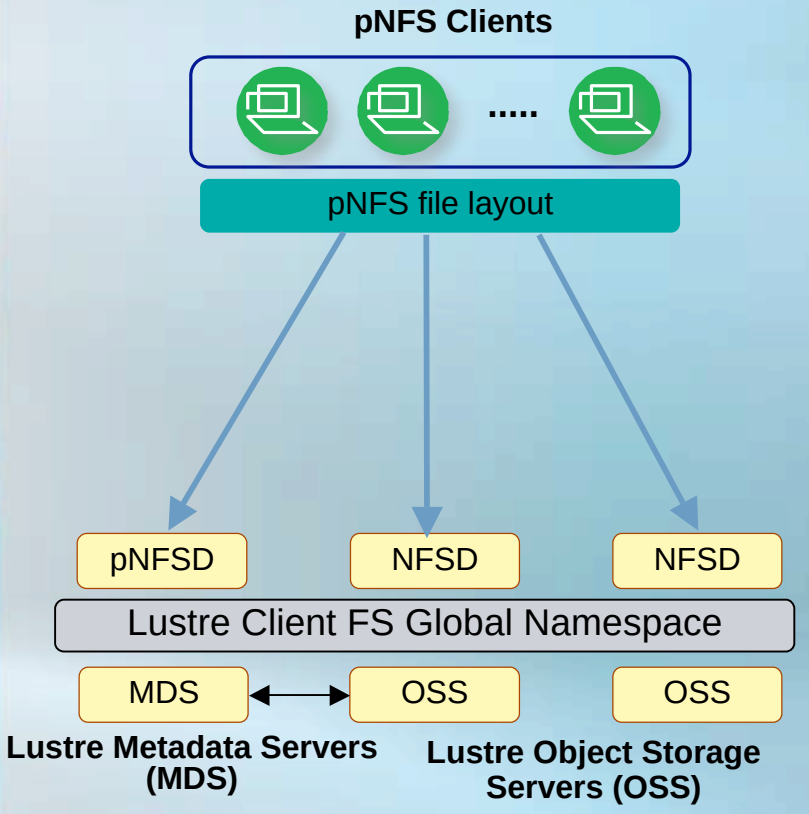
User level servers



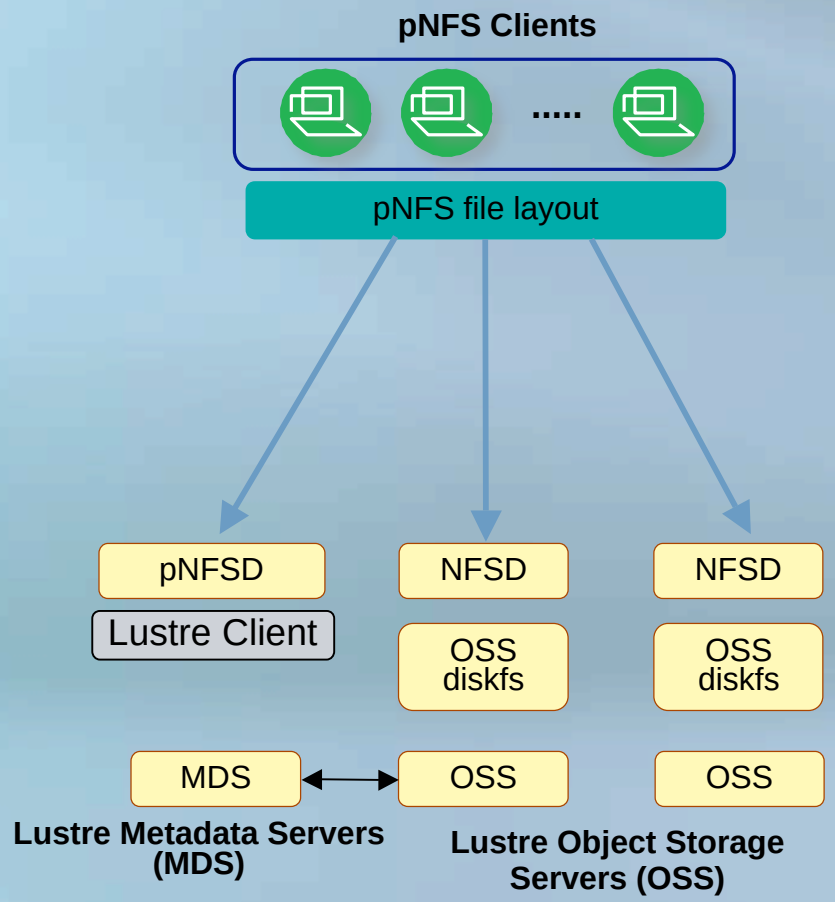
- **The Solaris OSS port layers the OSS server on ZFS**
  - The server will be a user space server
  - It will not use any custom interfaces to the file system
- **On Linux we are exploring the same**
  - Layer on ext4
  - Preparations
    - Give ext4 / Linux the capability of concurrent writes to one file
    - Improve the direct IO / VM cache relationship
- **Evaluate the performance**
  - For this we have written a simple server simulation program
    - pios – Parallel IO Simulator
- **High likelihood of success**
  - If so the OSS will become a user space server

pNFS - direct from the OSS file systems?

- **User level servers allow pNFS from disk FS**
  - Less overhead? More reliability?



pNFS layered on Lustre



pNFS layered on disk file system

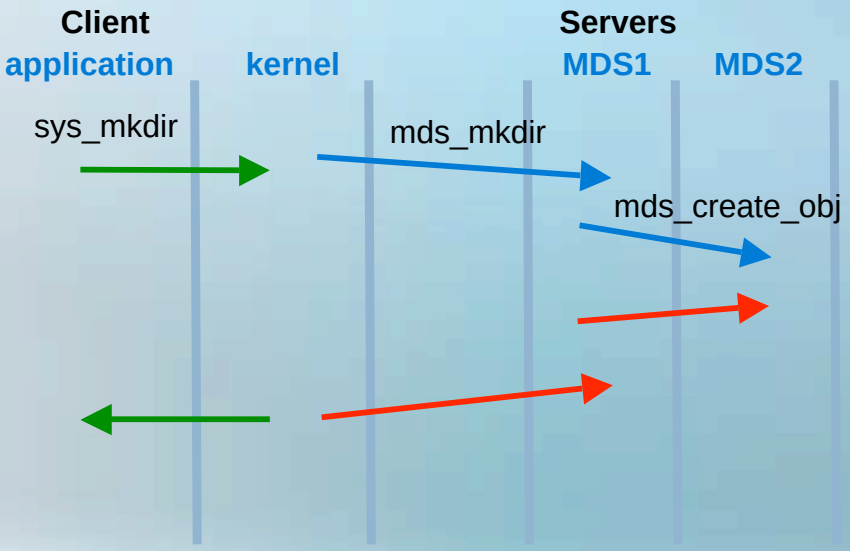
Metadata - 2007 & 2008

- Clustered metadata will go into Lustre 2.0
- Client does not get many changes
- Server saw a re-write

- Show a diagram of typical interactions
- Scalability results

Making a new directory

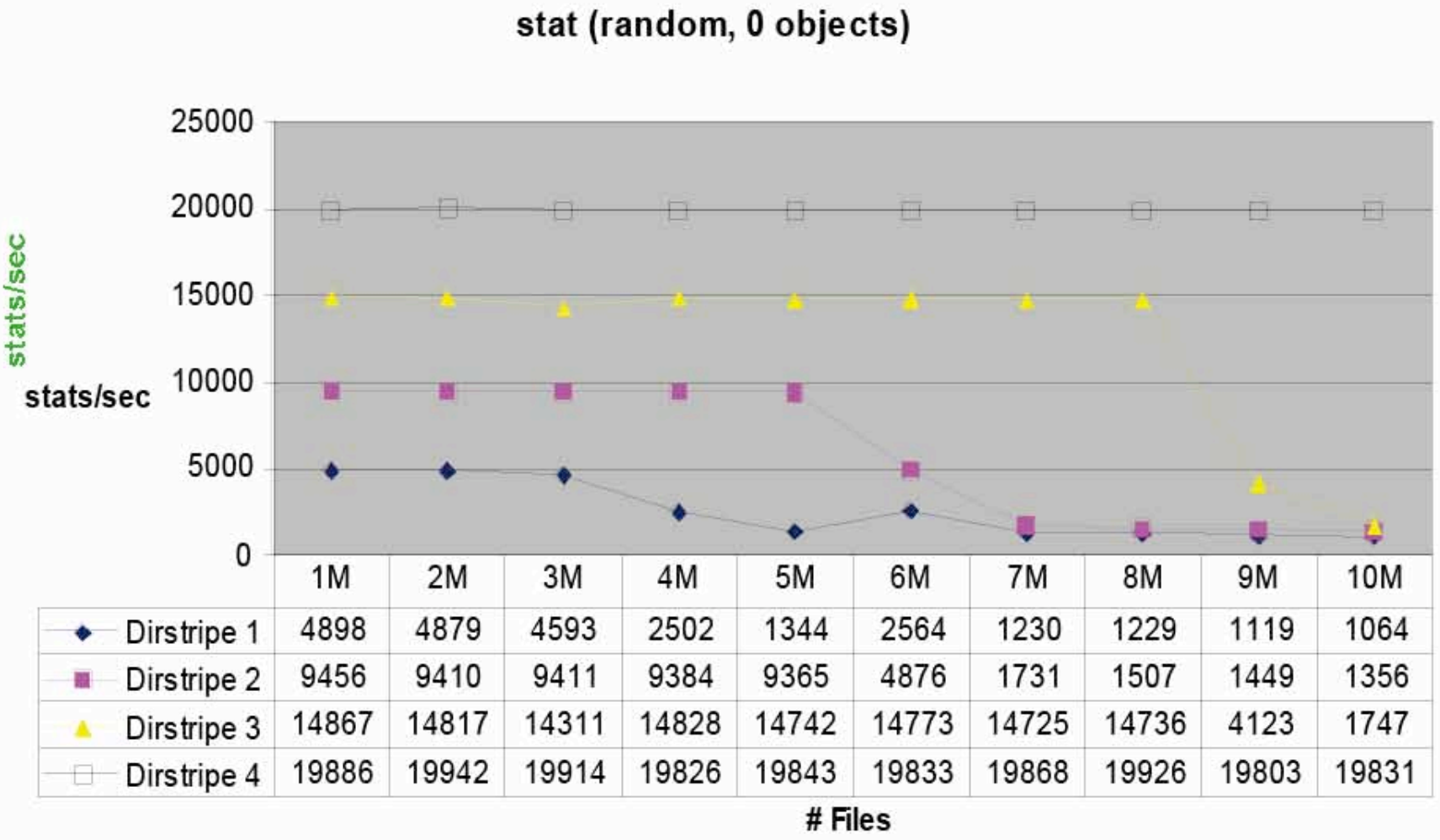
- **Directories sometimes placed on new server**
  - Different placement policies can decide where, e.g.:
    - Hash(name) determines server
    - Uid, client NID determines server



• MDS1 contains directory file with name
• MDS2 contains the inode associated with the name

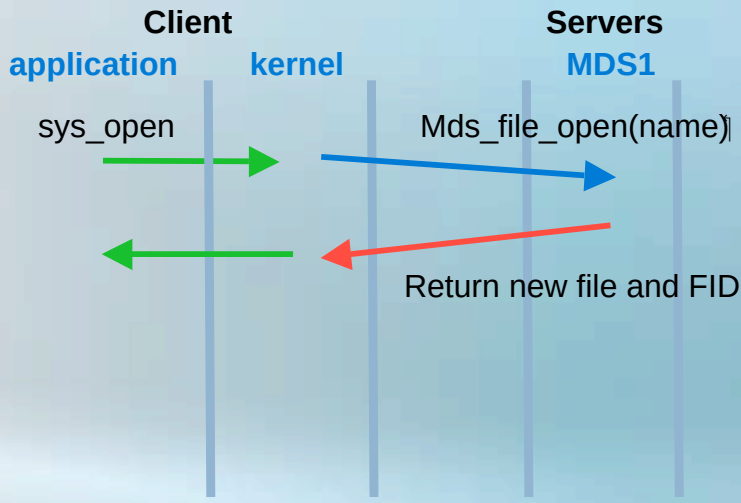
Clustered Metadata Directory Creation
• name and inode reside on different server
• client interacts with one MDS for updates

Scalability results - stat in large directories

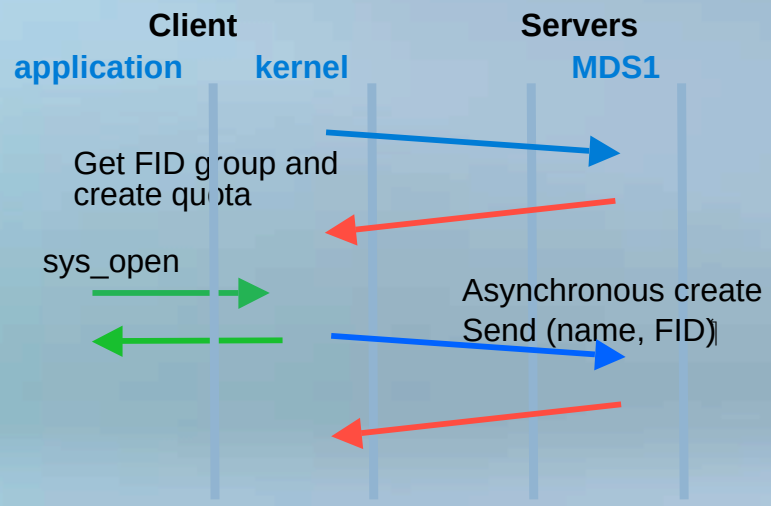


Metadata write back cache

- **A new file identifier (FID) system is coming**
  - Allows for full WB cache of metadata
  - Makes recovery significantly simpler
- **Illustration below shows a file creation**
- **Required for DARPA HPCS project**



Normal Synhronous File Creation
• server sends FID



Asynchronous file creation -
• Client has FID group & quota