Cray and Lustre

Charlie Carroll

Branislav Radovanovic

Cray Inc.

CUG 2006

Outline

- Past: Cray's history with CFS and Lustre
- Present: Where we stand today
- Future: Cray's plans for Lustre
- Lustre performance results

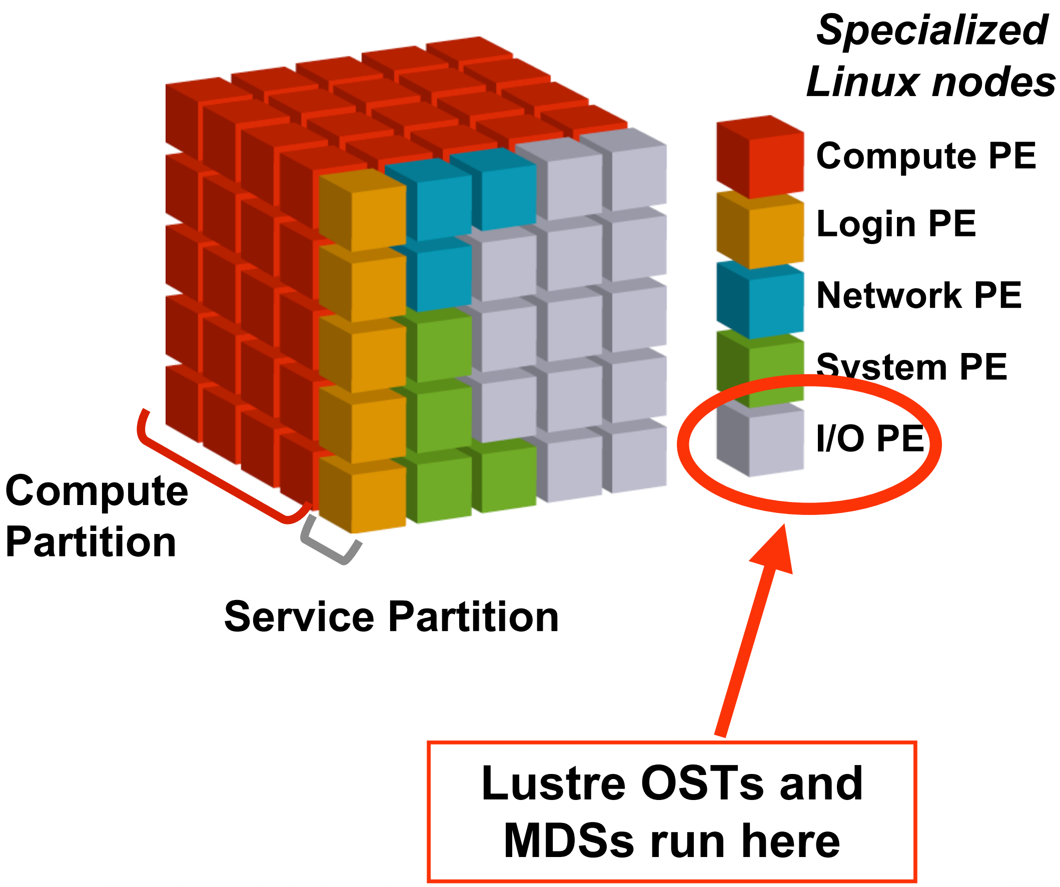
Cray and CFS History

- Red Storm was the first Cray system with Lustre; our relationship with CFS started in April 2003
- From Peter Braam's slides:
  - CFS founded in 2001
  - Shipping production code since 2003
  - Privately held, self-funded, US-based corporation
  - Profitable since Day One
  - More than 40 full-time engineers
  - Twenty percent of the Top 100 supercomputers (as of Nov. 2005)

Outline

- Past: Cray's history with CFS and Lustre
- Present: Where we stand today
- Future: Cray's plans for Lustre
- Lustre performance results

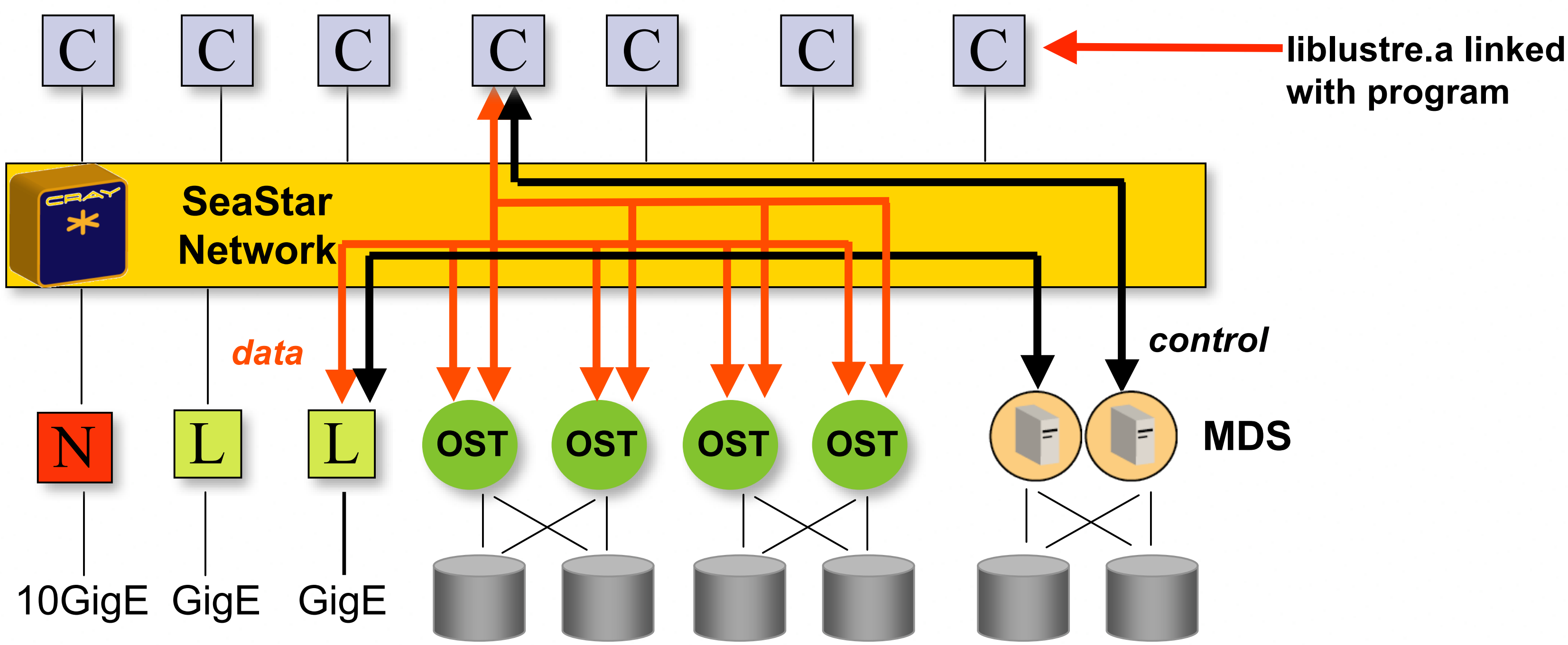
Cray XT3 Software Architecture



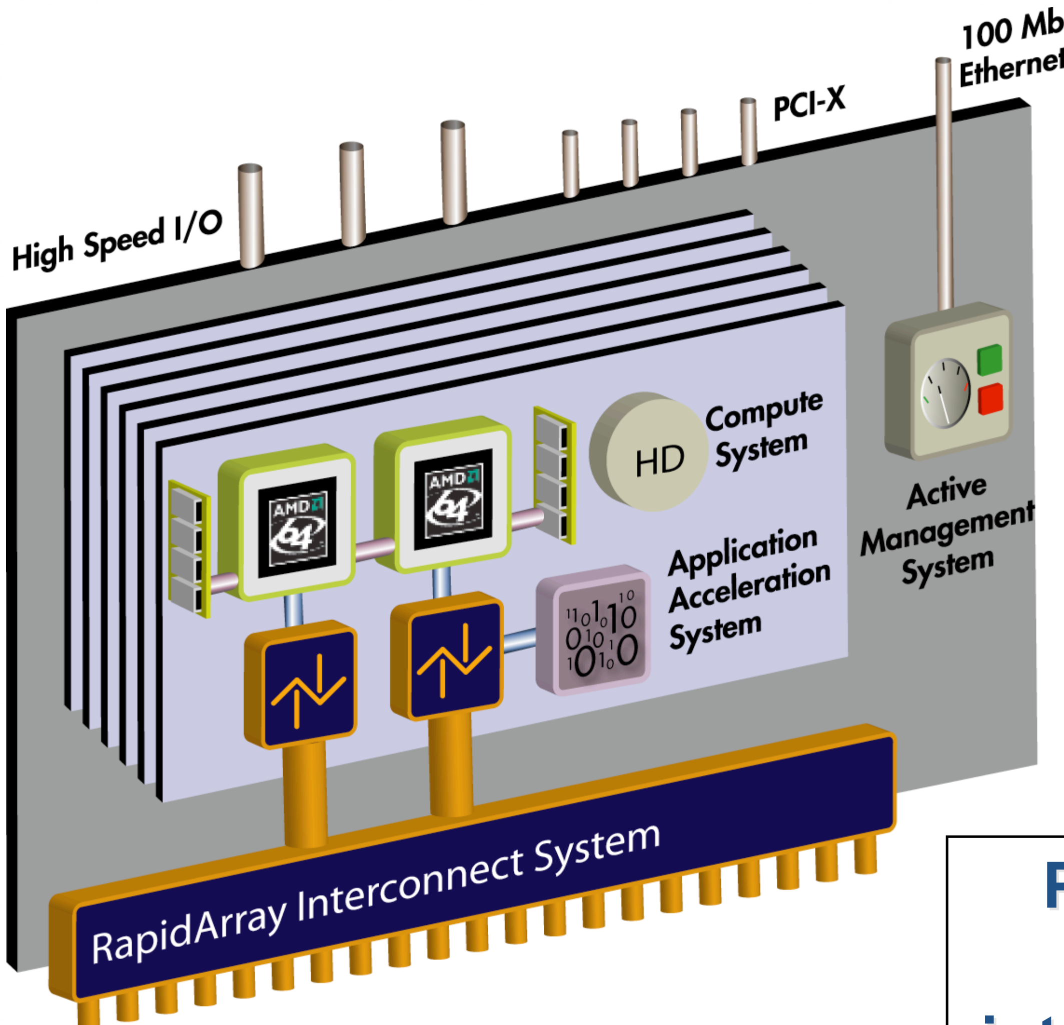
- Lightweight OS on compute PEs, full-featured Linux on Service PEs
- Service PEs specialize by function
- Software architecture eliminates OS "Jitter"
- Software architecture enables reproducible run times
- Large machines boot in under 30 minutes, including file system
- Job launch time is a few seconds on 1000s of PEs

Cray XT3 Lustre Architecture

- Highly scalable; more than 10,000 compute nodes
- XT3 compute clients run Catamount microkernel and use liblustre
- Portals networking over XT3 SeaStar interconnect
- Full Linux on service and I/O nodes



Cray XD1 System Architecture



Compute

- **12 AMD Opteron 32/64 bit, x86 processors**
- **High Performance Linux**

RapidArray Interconnect

- **12 communications processors**
- **1 Tb/s switch fabric**

Active Management

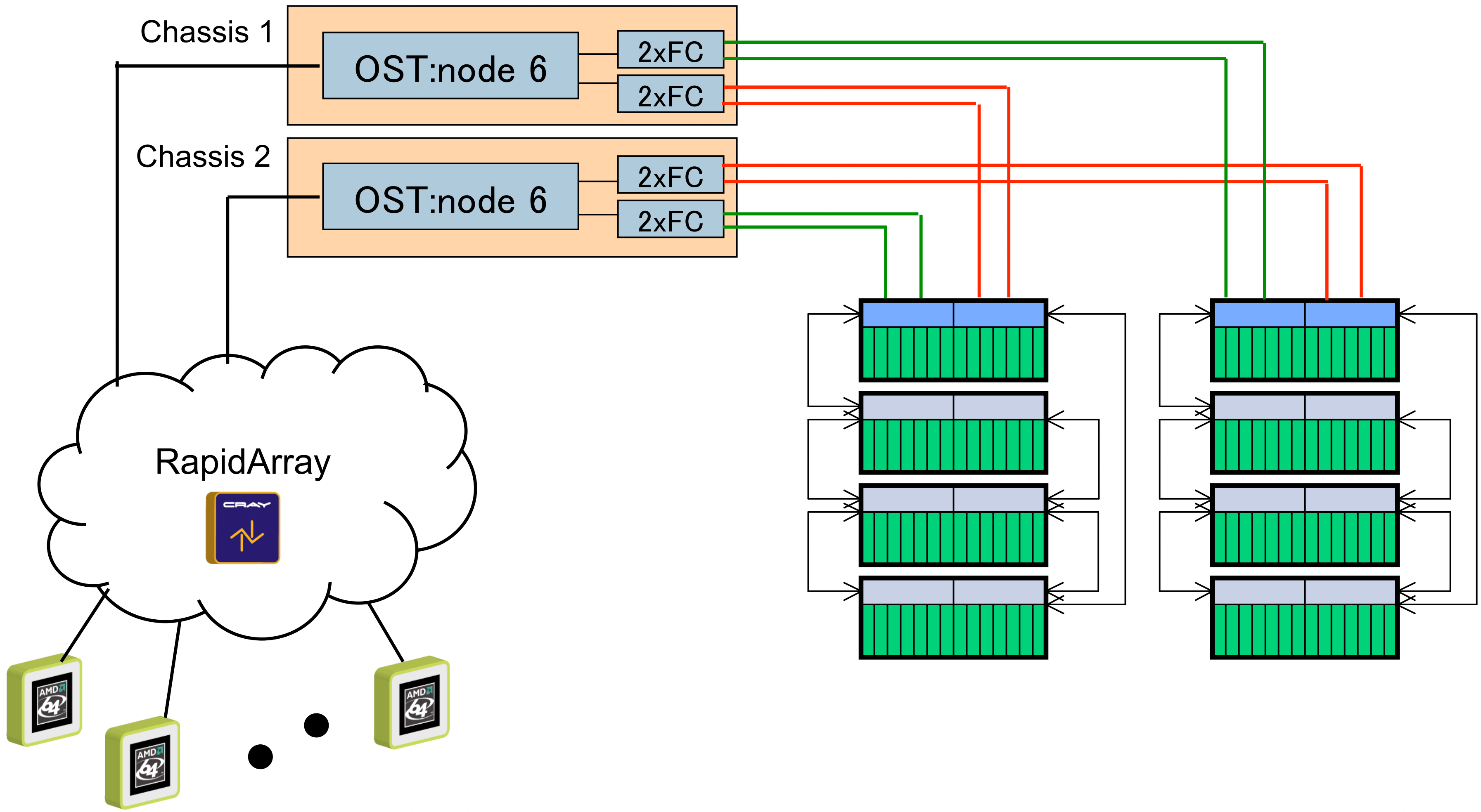
- **Dedicated processor**

Application Acceleration

- **6 co-processors**

Processors directly connected via integrated switch fabric

Cray XD1 Lustre Example



Sample of Cray Lustre Installations

- Rice University
- Naval Research Laboratory
- Oak Ridge National Laboratory
- Ohio Supercomputer Center
- Sandia National Laboratories
- University of Western Australia
- Pittsburgh Supercomputing Center
- Atomic Weapons Establishment
- Alabama Supercomputer Authority
- Swiss National Supercomputing Centre
- Maui High Performance Computing Center
- Sony Information Technologies Laboratories
- Japan Advanced Institute of Science and Technology
- US Army Engineer Research and Development Center

Installed on nearly 30 Cray XT3 and Cray XD1 systems worldwide

Experiences with Lustre

- When it works, it works very well
- But it is fragile
  - It sits at the top of the stack
  - If a lower-level component sneezes, Lustre gets violently ill
- It is difficult to diagnose
  - Find cause from symptoms is rarely straightforward
  - Hard to be sure that everything is healthy

Cray XT3 Lustre Experiences

- Lustre is an inherent part of the Cray XT3 system
- Cray XT3 systems started shipping in early 2005 and the full software stack needed time to mature
  - Lustre had to wait for all lower levels to stabilize
- Liblustre.a is not a standard client
  - New code for CFS
  - Catamount microkernel is very simple
- Cray XT3 systems scale to large sizes
  - Most production systems have 1000s of sockets
  - Most production systems have 30-60 OSTs

Cray XD1 Lustre experiences

- Lustre is optional for Cray XD1 systems, ~10%
- Cray XD1 Lustre started shipping in summer 2005
- Cray XD1 software stack is all Linux
  - Uses standard Lustre client on compute nodes
  - Designed for lower-end system – fewer options, easier to test
- Cray XD1 systems are generally small
  - Most systems with Lustre have 1-3 cabs (144-432 sockets)
  - Most systems with Lustre have 4-8 OSTs

Cray XT3-XD1 Comparison

	Cray XT3	Cray XD1
Environment	Linux and Catamount	standard Linux
Client	liblustre.a	standard Lustre
Scalability	highly	moderately

- Summary
  - Experiences with Lustre on the Cray XD1 were better than on the Cray XT3
  - Proves Lustre can be a powerful and useful product
  - Demonstrates that the Cray XT3 environment needs to be improved

Current Lustre Status

- Lustre is now perceived as stable on XT3 systems
  - Being used in production at major sites
  - Users trust they can write/read Lustre files
- Lustre has always been seen as stable on XD1s
- Lustre is supported by Cray and CFS developers
  - Over a dozen developers working on Lustre
  - Trained Field Service and SPS engineers
- Focus on stability and reliability for Cray XT3
  - Fixing bugs, particularly for scaling
  - Very few optimizations
  - Performance results to date have been acceptable

Outline

- Past: Cray's history with CFS and Lustre
- Present: Where we stand today
- Future: Cray's plans for Lustre
- Lustre performance results

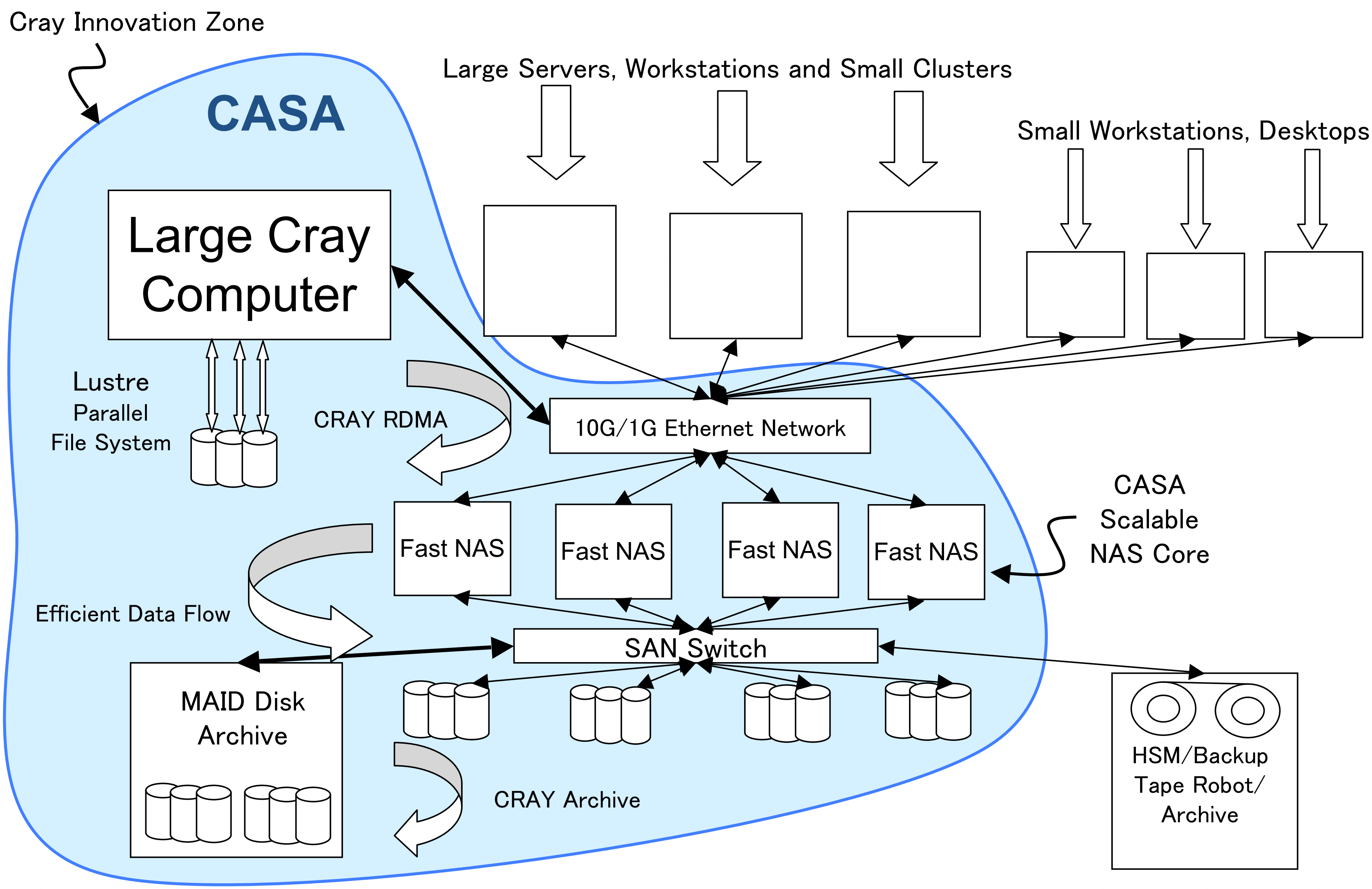
HPC Customer Storage Req'ts

- High-speed local storage for HPC systems
  - Very high speed and scalable to support applications performance
  - Mostly used for temporary, scratch data
- Site-wide data storage
  - Permanent files shared by multiple systems
  - Interoperability with installed environments
- Data management for permanent files
  - Protect permanent files
  - Backup, archival, HSM

Cray Storage Strategy

- Abandon hunt for the “holy grail” – the single solution that can solve all customers’ storage problems
- Instead pull together a set of tools to solve these problems
  - Find products excellent at meeting key requirements
  - Focused R&D with partners to strengthen products for HPC and with Cray systems

CRAY Advanced Storage Architecture (CASA)



Cray's File System Strategy

- Lustre is Cray's solution for high-performance local storage
  - Cray XT3 and Cray XD1 today
  - Future Cray systems under development

Current Priorities

- Reliability
- Troubleshooting
- Fragmentation
- Performance

Reliability

- Lustre is designed to support failover of the OST and MDS
  - Backup servers take over in case of a failure
  - New server connects to clients to replay cached or in-progress transactions that had not been committed to disk
  - Recovery transparent to applications
- Today Lustre failover works w/ Linux but not Catamount
  - Failover works successfully on the Cray XD1
  - Catamount clients are not guaranteed to reconnect to a server within any specific recovery window
- CFS is fixing bugs and reworking recovery code to support Catamount
  - Expect solution to be transparent to the majority of applications but some small percentage of running jobs may die
- After XT3 failover is functional, Cray will hook into CRMS to automate the failover process
  - Shut down the failed server and bring online the backup server without operator involvement

Troubleshooting

- Cray's highest development priority for CFS
  - Improved error messages
    - CFS has a near-term plan to revise error messages
  - Improved documentation
    - CFS is developing a complete Lustre manual
    - Some sections included in 1.4.6; completion planned for 1.6.0
  - Improved tools for file system logging, debugging, and monitoring
    - Know when something has gone wrong, where, and how to repair; detailed design TBD
- Top items from attendees of Lustre Users Group in April
  - Understandable and documented error messages; troubleshooting
  - OST stripe management: 1) pools; 2) join files; 3) background migration
  - Improved logging, debugging and diagnostic tools; NID logic; per-client stats

Fragmentation

- Over time, the ext3 file system underlying Lustre can become fragmented, degrading performance
  - Files are split into many small pieces, scattered across the disks
  - Exacerbated by Catamount clients, which do I/O synchronously
    - Linux clients coalesce writes and send to OSTs in 1 MB chunks
  - Severity should be lessened with XT3 1.4 with the ext3 mballoc (multiblock allocator) option
    - Upgrading to 1.4 won't help an already fragmented disk, defragmentation or reformat needed
- Addressing fragmentation
  - Cray is developing a utility to evaluate file system fragmentation
  - CFS is developing a defragmentation script
    - Might be slow; won't be effective on full file systems
  - Cray and CFS are evaluating whether problem remains with the XT3 1.4 release

Performance

- Lustre performance difficult to characterize with the XT3 1.3 release
  - Linux 2.4, disk fragmentation, etc.
  - Cray and various sites ran many performance tests with limited success and wide variability
- Cray will characterize Lustre performance with the XT3 1.4 release
  - CFS will help with fixes and tuning
  - CFS has committed to make XT3 Lustre “scream”

Outline

- Past: Cray's history with CFS and Lustre
- Present: Where we stand today
- Future: Cray's plans for Lustre
- Lustre performance results

Performance Results Outline

- Tips for Testing
- File Access Patterns
- Lustre Scaling
- Improving RAID-5 Performance

Striping

- Lustre has the flexibility to specify how a file is striped across OSTs
  - Default set when file system is made
  - User can specify with *lfs setstripe* [*dir* | *file*] ...
- Striping across multiple OSTs is useful when an application writes large, contiguous chunks of data
  - OSTs run in parallel, increasing I/O performance
- If the application isn’t writing large data, striping will hurt, not help
  - Don't stripe for small files
  - Don't stripe any more widely than you have to
  - Don't use small blocks per OST in a stripe

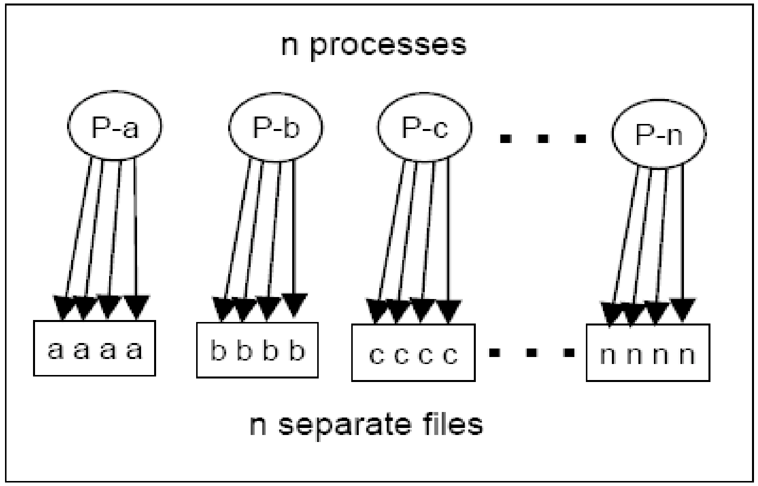
Debugging Messages

- Lustre debugging messages are controlled by a /proc variable
  - Turning off debug logs increases performance and repeatability
    - `% echo 0 > /proc/sys/portals/debug`
  - Turning off debug logs makes it difficult to diagnose problems if something goes wrong

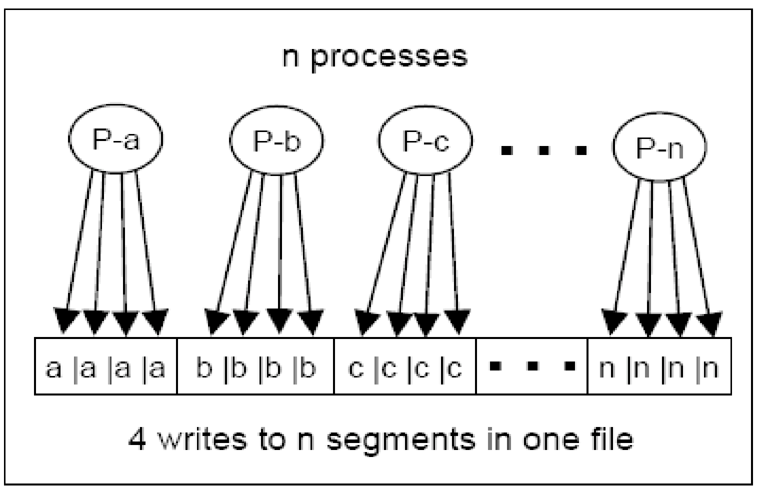
Performance Results Outline

- Tips for Testing
- File Access Patterns
- Lustre Scaling
- Improving RAID-5 Performance

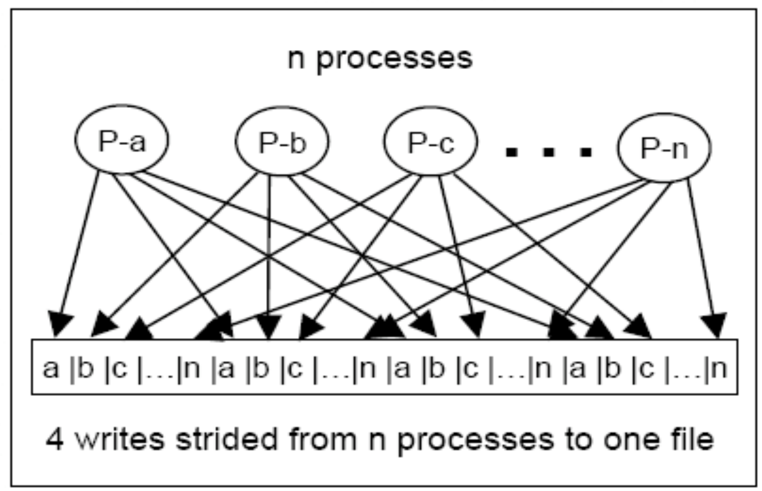
File Access Patterns



file per process



shared file : segmented access

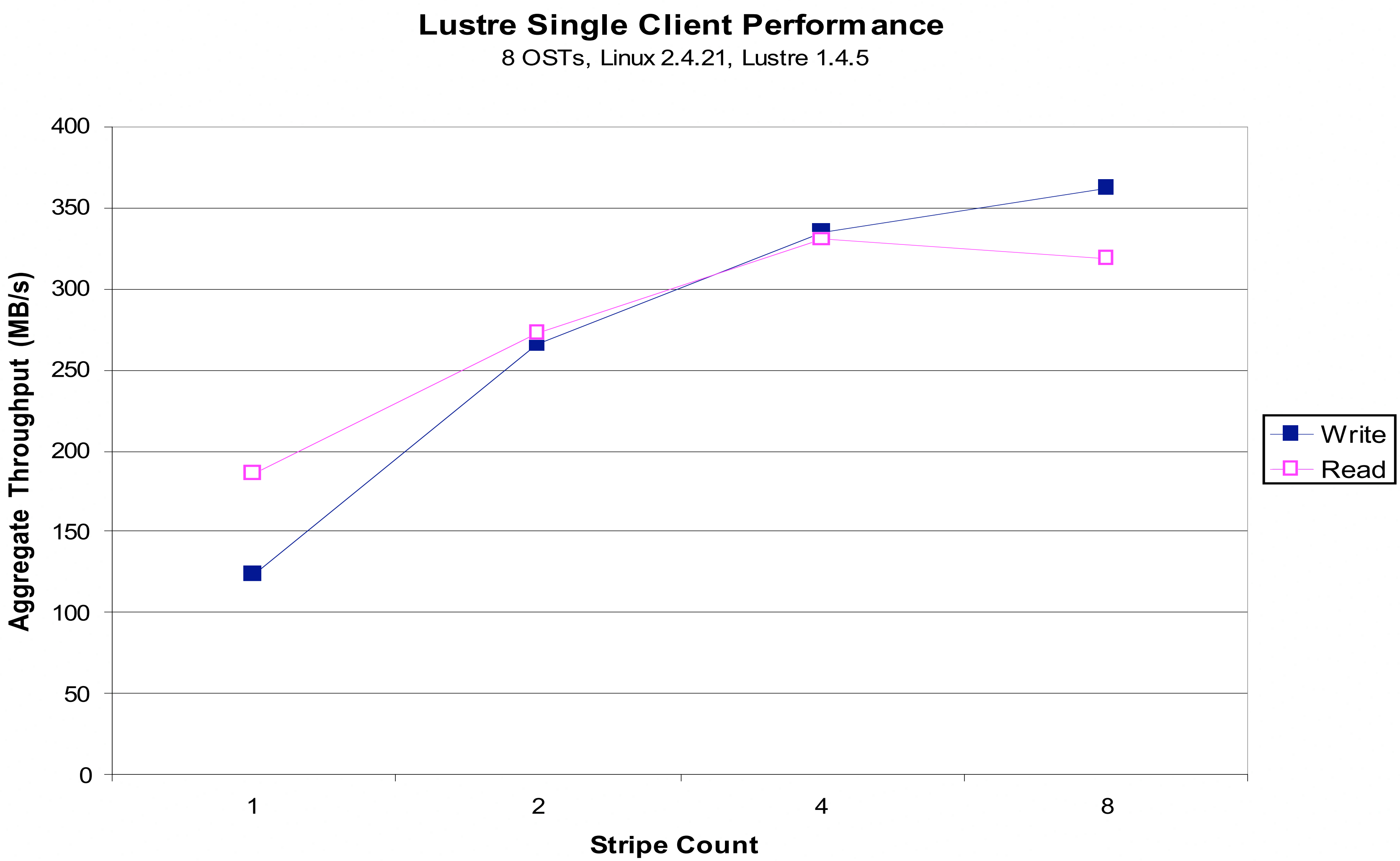


shared file : strided access

Access Pattern Testing

- Platform
  - XT3 with Compute Node Linux
  - Lustre v.1.4.5 running on Linux 2.4.21 sles9
  - DDN S2A8500 with 8 FC-2 ports
- Test
  - IOR 1.1.2.1
    - Use 1MB IOR Transfers to match size of Lustre RPC
  - Measured Variations
    - 1 client to multiple [1, 2, 4, 8] OSTs
    - Multiple [1, 2, 4, 8, 16] clients to 8 OSTs

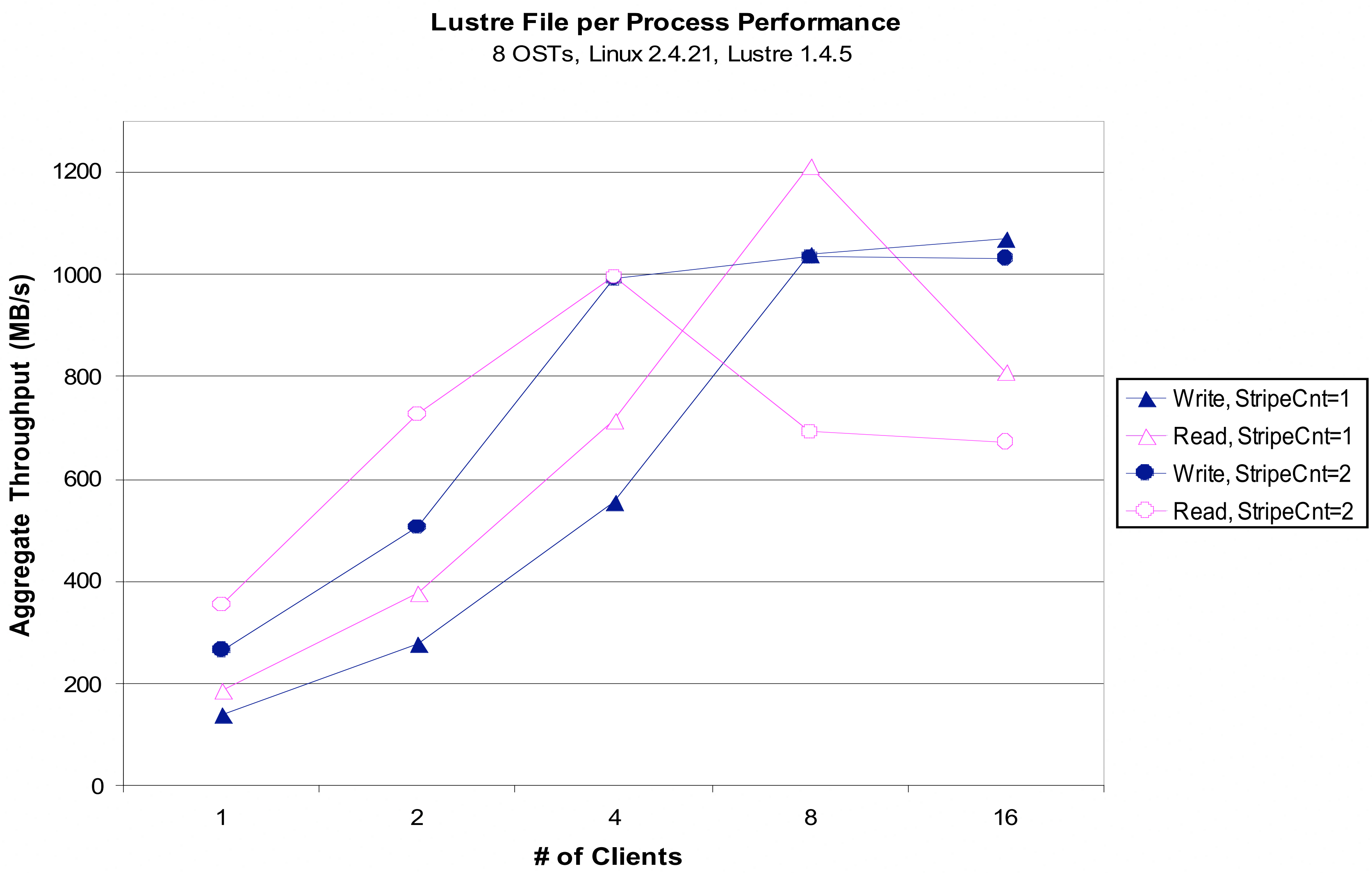
Single Process & Stripe Count



Discussion: Stripe Count

- For a single client
  - Striping a file across more OSTs increases throughput
  - The incremental improvement in performance diminishes for each increase in stripe count
    - The relative increase from one to two OSTs is greater than the change from two to four or four to eight.
- With a stripe count of 8, the single client is writing to all 8 OSTs and the throughput from the client is at a maximum of ~360 MB/s write and ~320 MB/s read

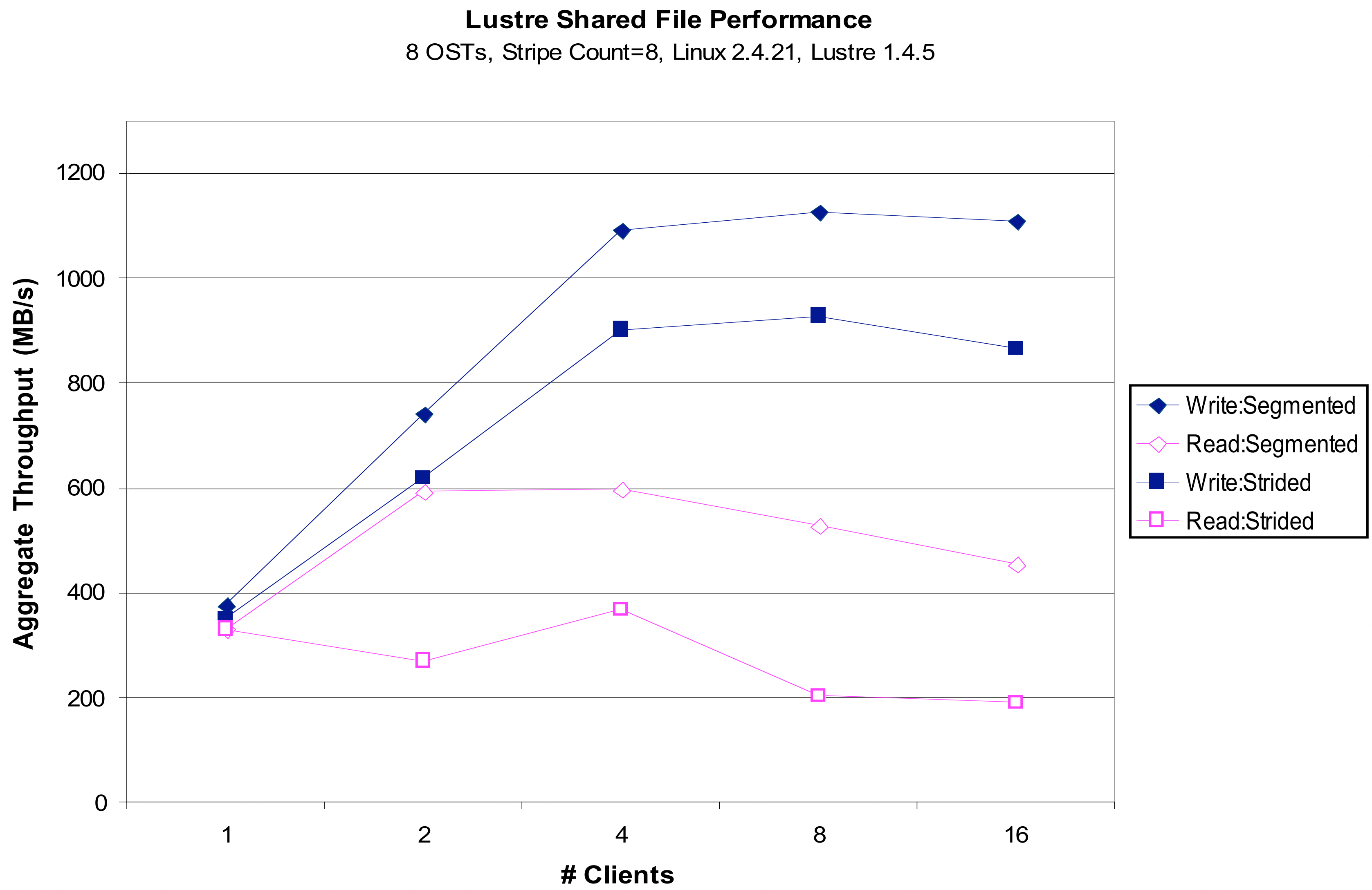
File per Process & Stripe Count



Discussion: File per Process

- Files with the wider stripe count reach the maximum throughput with fewer clients
  - Maximum throughput occurs when (# of clients)*(stripe count) equals (# OSTs)
- Wider stripe count does not provide more performance with higher client count
- Need to test with more clients to learn more about aggregate performance when we exceed this number of clients
  - At this point, read performance drops off while write performance levels off

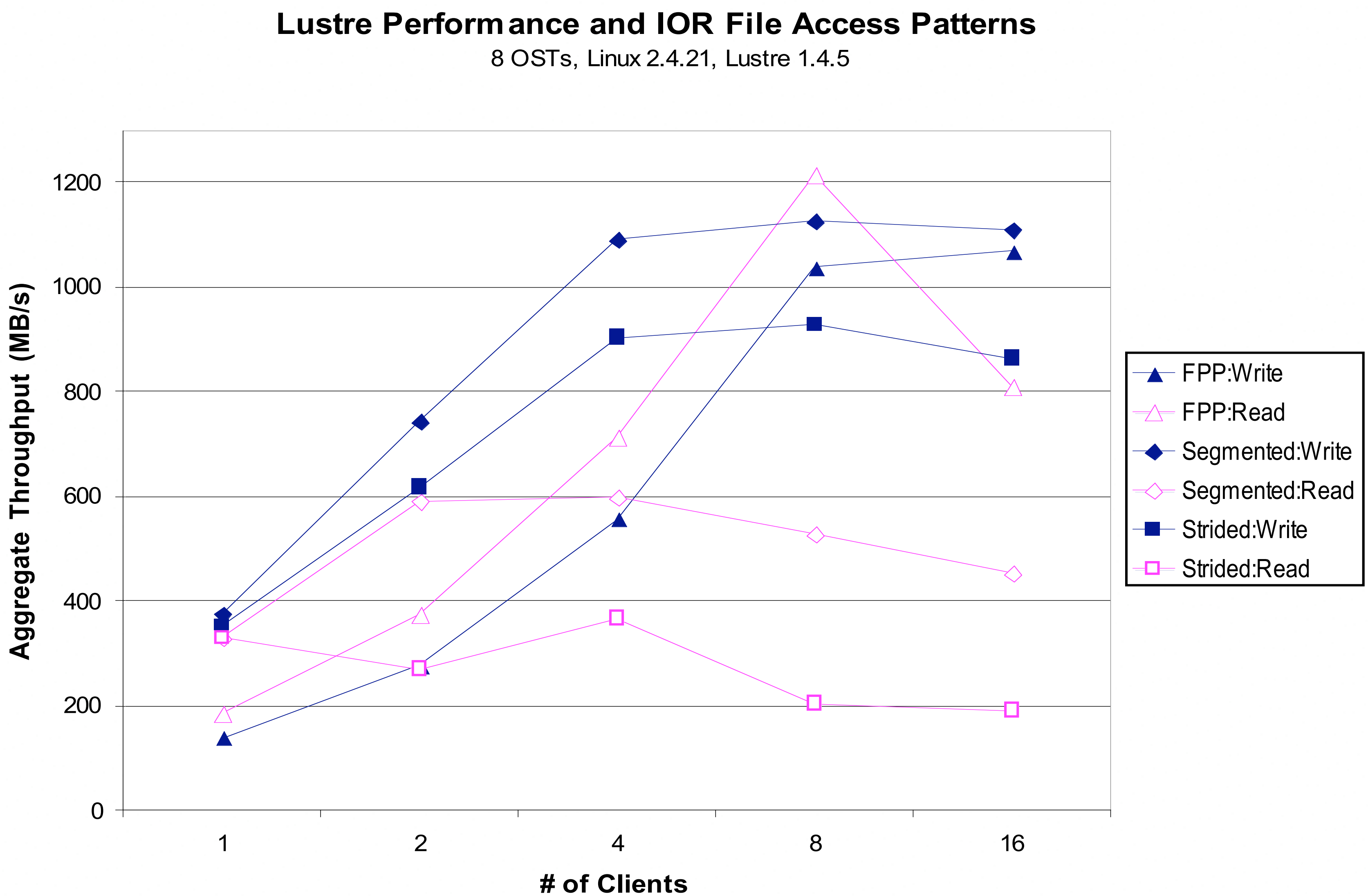
Shared File Performance



Discussion: Shared File

- Shared files with a segmented access pattern support greater read and write rates than shared files with a strided access pattern
- The poor read performance for both types of shared file is not understood, though we suspect the poor I/O architecture of the Linux 2.4 kernel is a factor. We expect significant improvements when we repeat the tests on XT3 SIO nodes running a Linux 2.6 kernel

File Access Comparison



Discussion: File Access Patterns

- In general, write throughput increases until the number of clients equals the number of OSTs
- File per Process (FPP) and Shared-Segmented reach comparable maximum write throughput
- File per Process has higher read rates than either shared file access pattern

Summary: File Access Patterns

- Comparing our results with the results Livermore Labs reported in their paper at the Mass Storage Conference, both studies show that
  - Write performance is generally greater than read performance, especially for the shared file access patterns.
  - Shared-Segmented access patterns are faster than the Shared-Strided access pattern
- But, our results show that
  - File per Process reads scaled more closely to File per Process writes
  - For low client count, File per Process reads were even faster than File per Process writes
- The maximum throughput from the RAID is ~1.5 GB/s
  - Even though limited to 16 clients, with the File per Process access pattern, we were able to achieve ~1.1 GB/s for both reads and writes
  - Our preliminary results do not include enough clients to show the maximum aggregate throughput through the file system

Performance Results Outline

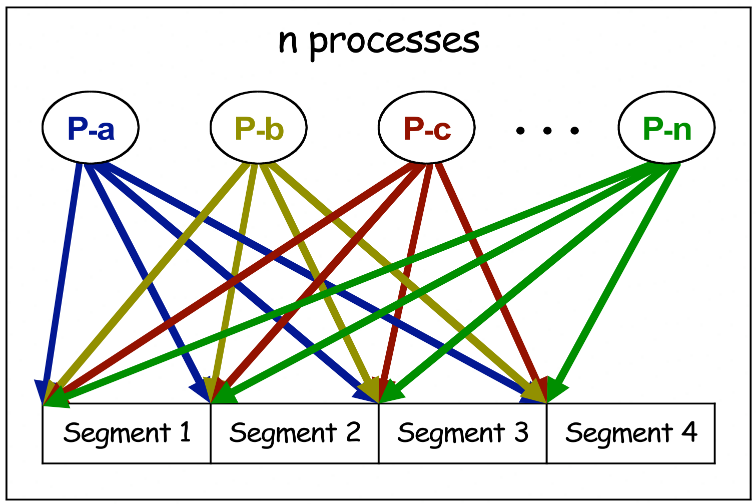
- Tips for Testing
- File Access Patterns
- Lustre Scaling
- Improving RAID-5 Performance

lmdd (Lmbench) I/O Benchmark

- lmdd developed by L. McVoy, SGI and C. Staelin, HP
  - Test and methodology described in
  http://www.usenix.org/publications/library/proceedings/sd96/full_papers/mcvoy.pdf
- Measured Variations
  - Multiple Catamount clients [1, 2, 4, … 128, 256] to multiple OSTs [1, 2, 4, 8, 11]
- File Access Pattern - Minimize HDD Latencies
  - Shared-Segmented, Overwriting the Segments
- Use 4kB to 16MB lmdd Requests
  - 1MB, Max. Size of Lustre Client RPC

File Access Pattern

- Test tool *lmdd* with Sequential I/O to a shared file
  - Target file composed of four shared segments
  - Each process overwrites or reads these same four file segments



- This shared-segmented file access pattern effectively removes the storage controller from the Lustre performance tests
  - Minimizes HDD Head Movement on reads and writes
  - Maximizes RAID Cache Hits on Read

Test Environment

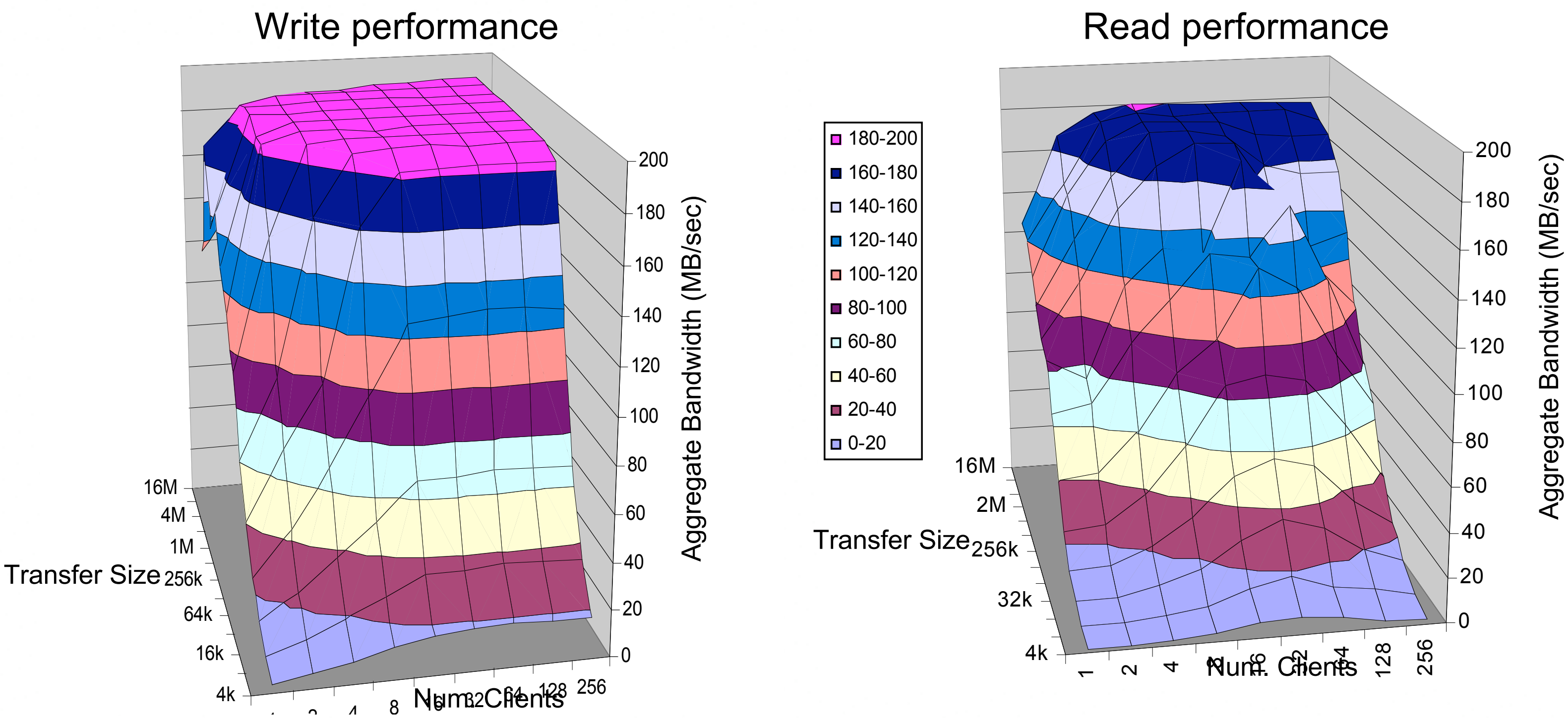
- Test
  - One Shared lmdd File for all Processes

```
lmdd of=/lus/nid00008/lmdd.cat/lm1 bs=1048576 count=5120
lmdd if=/lus/nid00008/lmdd.cat/lm1 bs=1048576 count=5120
```

    - stripe size = 1 MB
    - stripe count = 1, … 11 OSTs
    - request size = 4 kB, … 16 MB
  - Limit the test to 256 clients for all Lustre OSTs
- Platform
  - Cray XT3 with a RAID 3 and RAID 5 systems
    - Catamount Lustre client
- Versions
  - Linux 2.4.21 (XT3 1.3.10)
  - Lustre and liblustre.a 1.4.5

One OST, 1–256 Clients

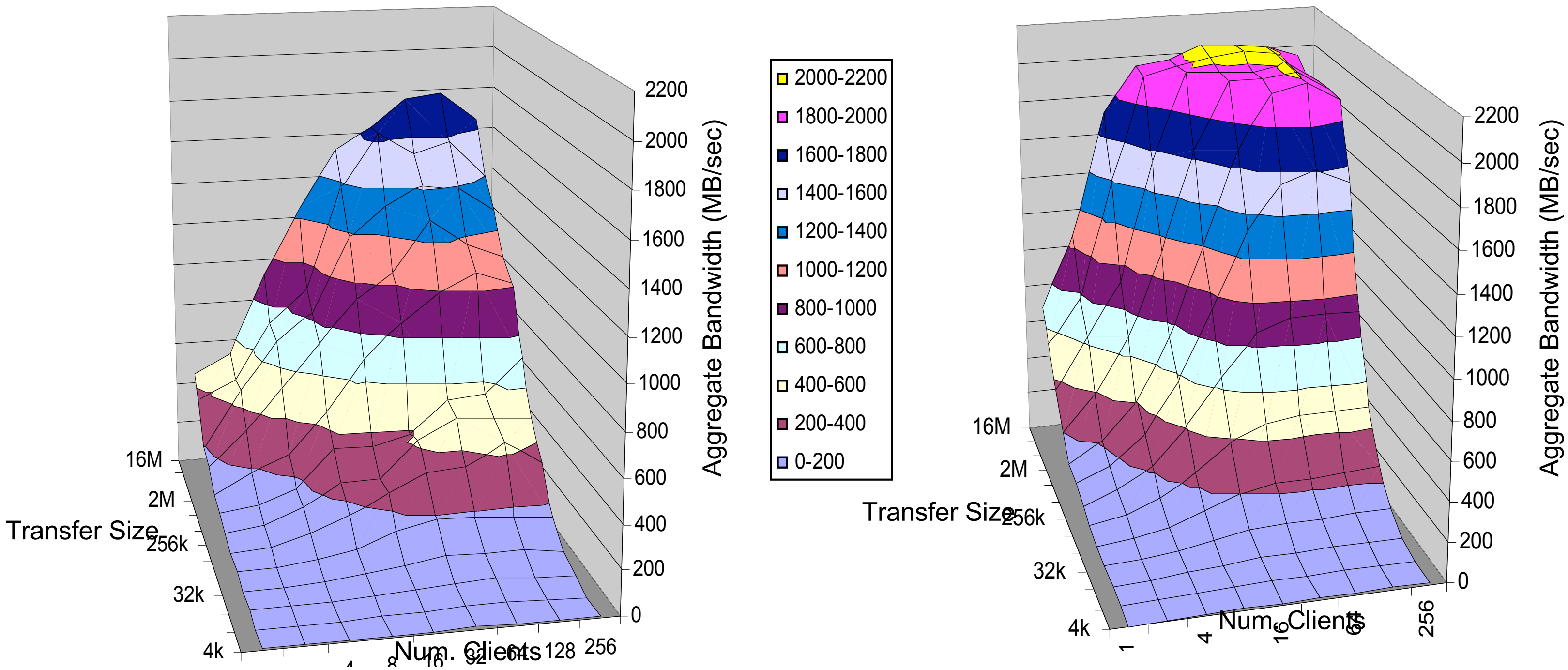
- Imdd, Sequential I/O to a Shared File
  - Stripe Count = 1, Stripe Size = 1MB
  - RAID 3



- Theoretical bandwidth to a single OST is 200 MB/sec
  - Achieved bandwidth: reads - 188 MB/sec, writes - 180 MB/sec
  - Two to four clients easily can saturate an OST

Eleven OSTs, 1–256 Clients

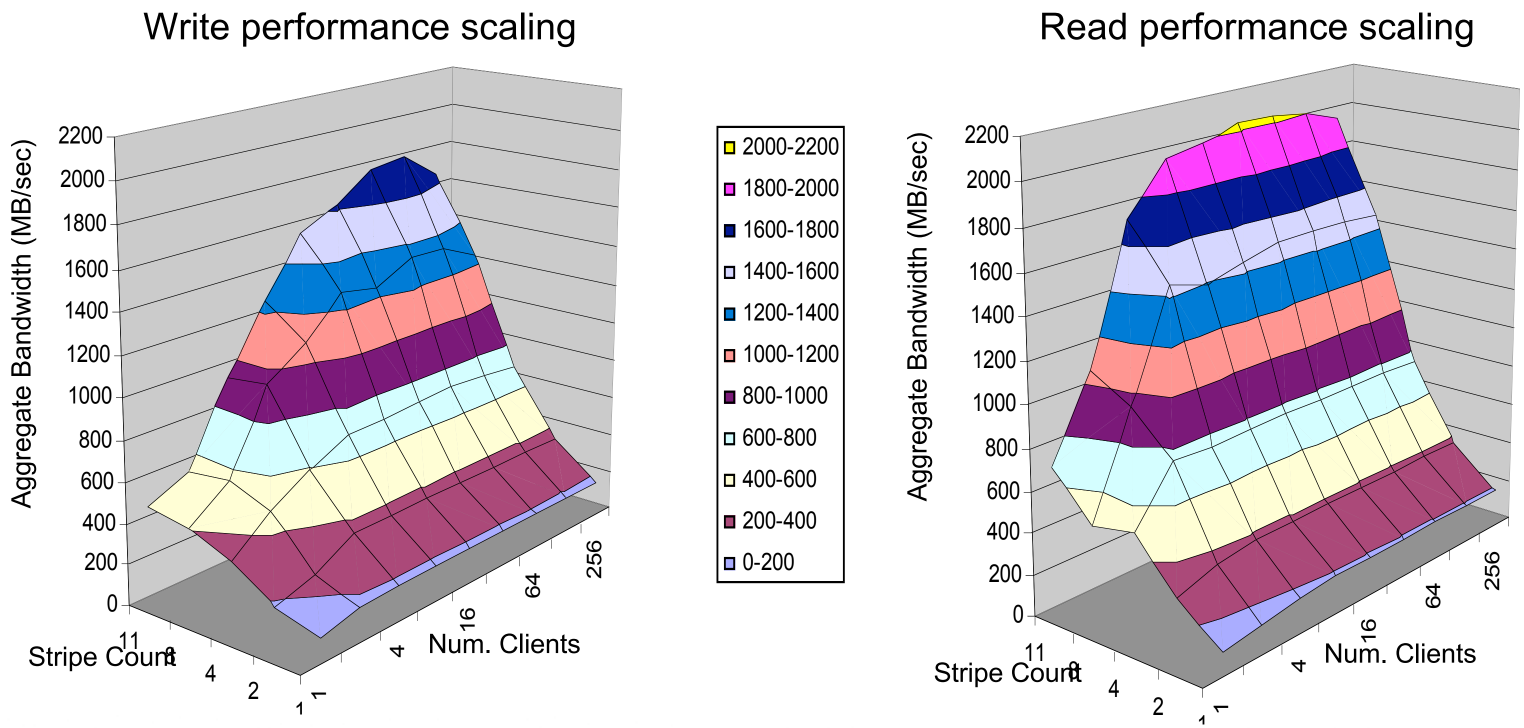
- lmdd, Sequential I/O to a Shared File
  - Stripe Count = 11, Stripe Size = 1MB
  - RAID 3



- Theoretical bandwidth to single 11 OSTs is 2200 MB/sec
  - Achieved bandwidth: reads - 2.03 GB /sec, writes - 1.78 GB /sec

Scaling 1-11 OSTs, 1–256 Clients

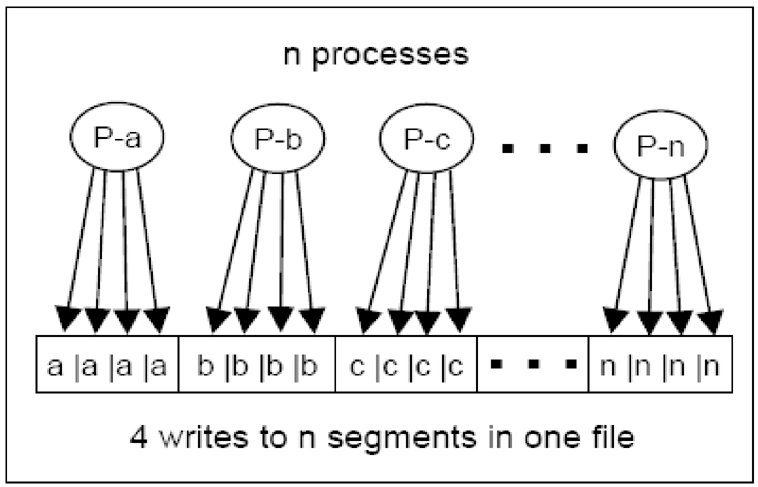
- Imdd, Sequential I/O to a Shared File
  - Stripe Count = 1 – 11 OSTs, Stripe Size = 1MB
  - I/O request size 16MB



- Our results, for sequential I/O, show that
  - Aggregate Lustre performance scales well with the number of OSTs
  - To realize the full benefit of scaling, need large number of clients

File Access Pattern

- Test tool *IOR* with Sequential I/O to a shared file
  - Each process writes or reads its own segment in the shared file



shared file : segmented access

- This shared-segmented file access pattern shows the performance of Lustre combined with the storage controller

Test Environment

- Test
  - One Shared IOR File for all Processes (segmented access)

```
IOR -a POSIX -w -k -o /lus/IORfile -i 1 -t 1m -b 256g -T 90
IOR -a POSIX -r -o /lus/IORfile -i 1 -t 1m -b 256g -T 90
```

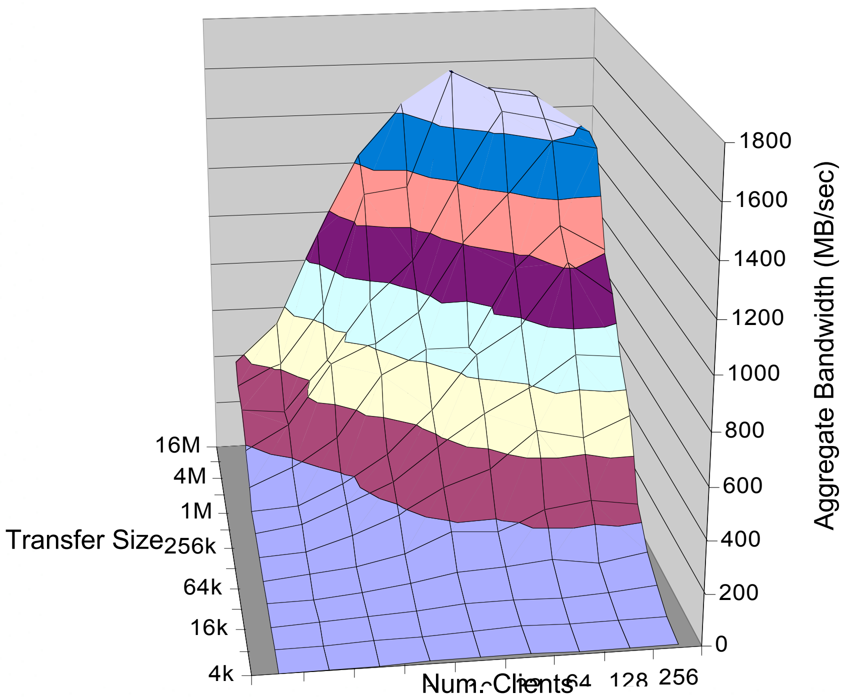
    - stripe size = 11 OSTs
    - request size = 4kB, …16MB
  - Limit the test to 256 clients for all Lustre OSTs
- Platform
  - Cray XT3 with a RAID 3 system
    - Catamount Lustre client
- Versions
  - Linux 2.4.21 (XT3 1.3.10)
  - Lustre 1.4.5
  - IOR 1.1.2.1

The Supercomputer Company

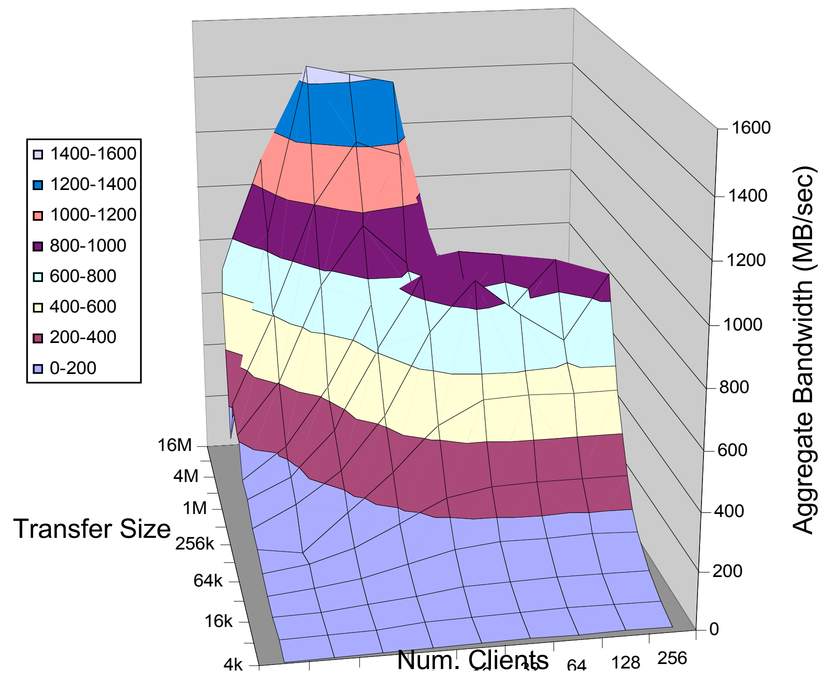
Eleven OSTs, 1–256 Clients

- IOR, multiple sequential I/O streams to a Shared File
  - Stripe Count = 11 OSTs, Stripe Size = 1MB
  - RAID 3

Write performance



Read performance



- The poor cache hit ratio significantly affects read performance
  - Peak bandwidth for reads - 1.46 GB/sec, writes 1.60 GB/sec
  - For large number of clients reads degraded to 600 - 800 MB/sec from the 1.46 GB/sec peak bandwidth due to read cache misses

Summary: Lustre Scaling

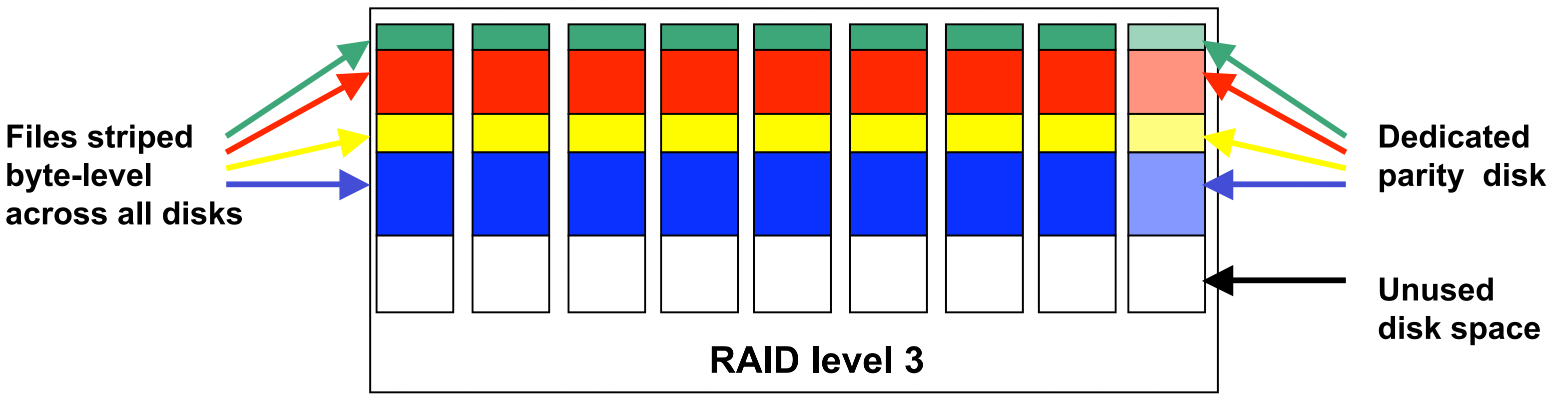
- By itself, the Lustre file system scales very well
  - Adding more OSTs improves aggregate performance
  - Striping across multiple OSTs yields significant I/O performance improvements for large I/O request sizes (greater than #OSTs x 1MB) to huge shared files
  - Large number of clients (more than 16 per OST) are needed to take the full advantage of the high stripe count
- But for large numbers of clients, caching policies of the back-end RAID system begins to impact performance
  - Read performance is impacted the most
  - In our tests, we believe write-back cache masked disk drive mechanical latencies
- The performance testing was done using Catamount Lustre client and Lustre server running 2.4.21 Linux kernel
  - We expect improved results in future tests with the 2.6 Linux kernel in the OS release 1.4

Performance Results Outline

- Tips for Testing
- File Access Patterns
- Lustre Scaling
- Improving RAID-5 Performance

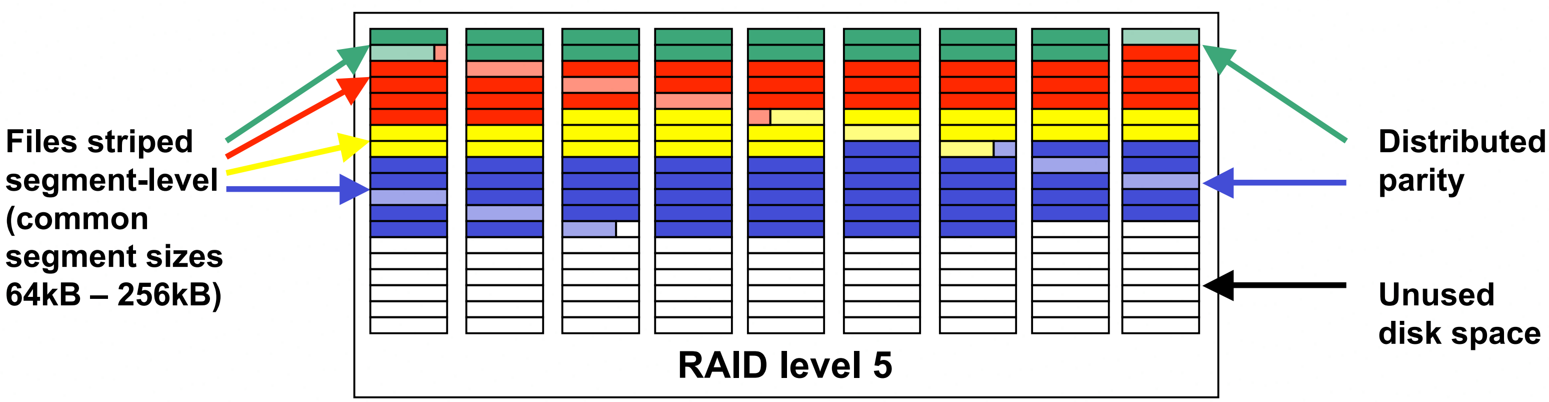
RAID-3

- RAID Level 3 – (RAID 3)
  - Uses byte-level striping across data disks with dedicated parity drive
  - Hardware accelerator to calculate parity for writes and reads
  - Exact number of bytes sent in each stripe depends on the particular implementation (settings)
  - RAID 3 uses dedicated parity disk
  - Any single disk failure in the array can be tolerated (data recalculated using parity)



RAID-5

- RAID Level 5 – (RAID 5)
  - Uses block-level striping with distributed parity
  - Hardware accelerator to calculate parity for writes and reads
  - Exact number of bytes sent in each stripe could be selected
  - Distributed parity algorithm, writes data and parity blocks across all the drives in the array
  - Any single disk failure in the array can be tolerated (data recalculated using parity)

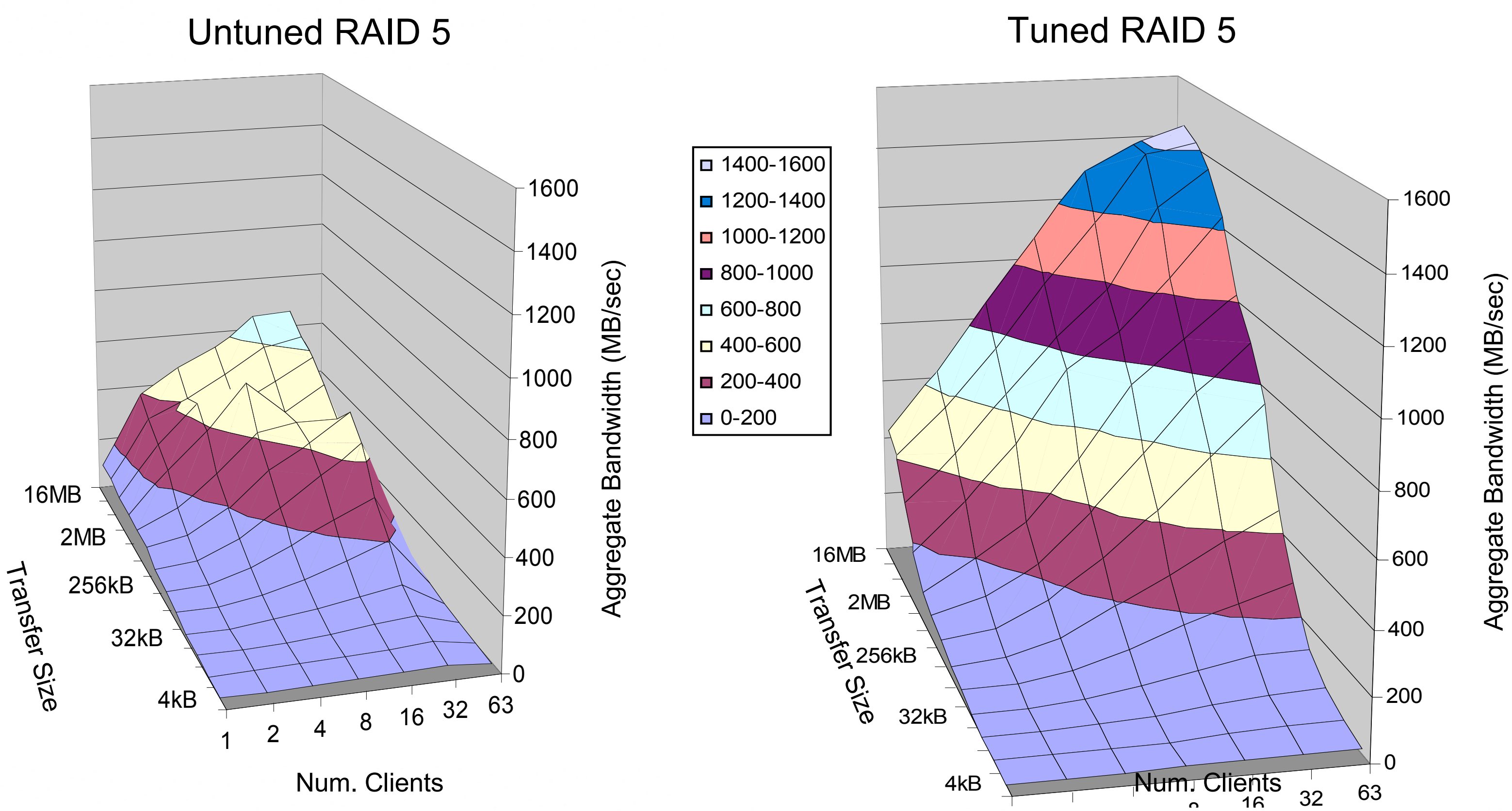


RAID-3 *vs* RAID-5

- Generally believed that RAID-3 performance is better than RAID-5 for Lustre
  - RAID 3 is commonly used for the applications using large files requiring high transfer rates
  - RAID 5 is usually preferred for transaction processing, relational database, and general purpose applications
- But RAID-3 requires all 8 disks to synchronize results during reads, thus RAID-5 should be able to outperform RAID-3
- Investigate RAID-5 configuration to improve performance

RAID-5 untuned *vs* tuned: Writes

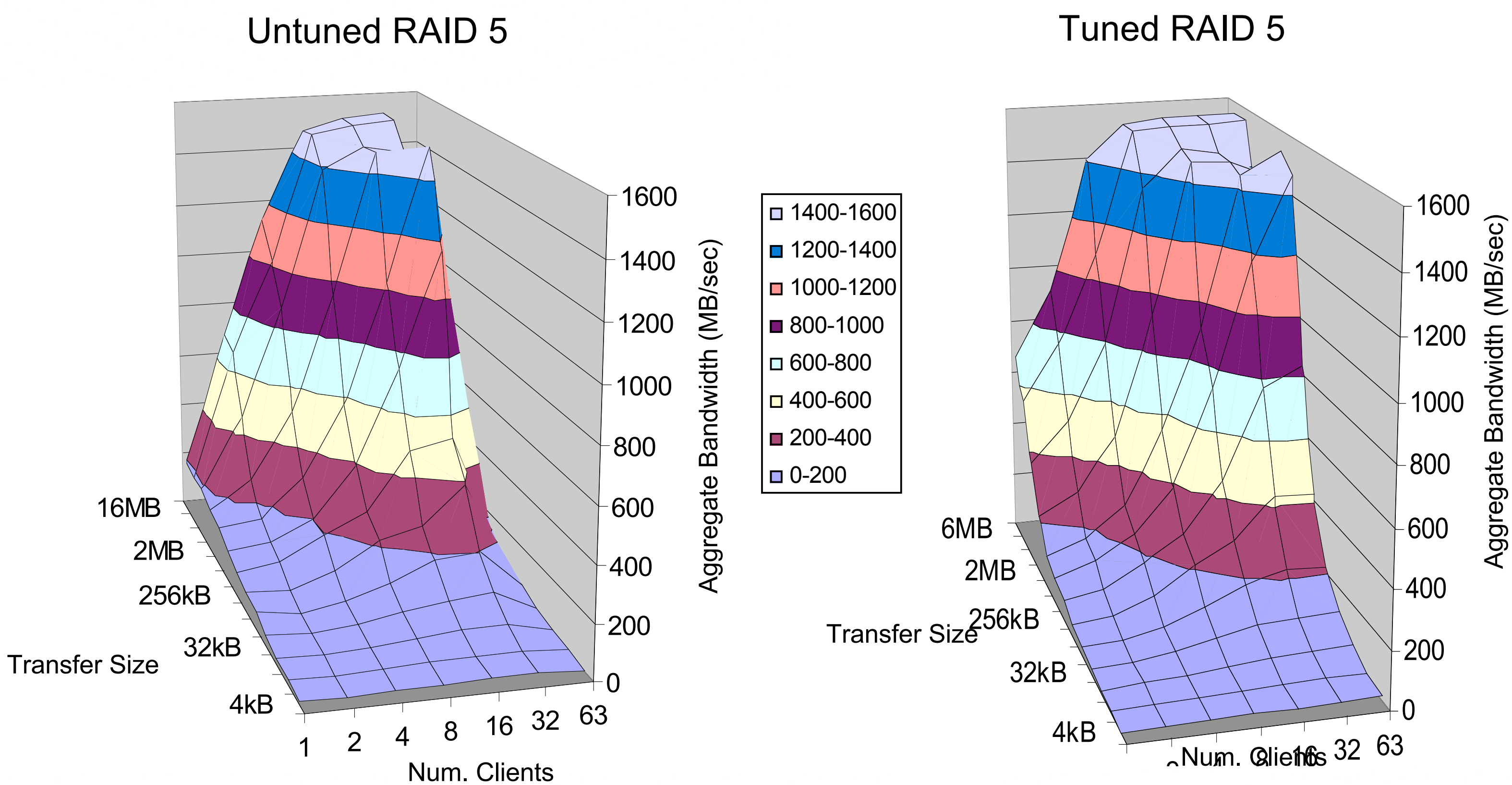
- lmdd, Sequential I/O to a Shared File
  - Stripe Count = 8 OSTs, Stripe Size = 1MB
  - Write requests 4kB, … 16MB



•Achieved bandwidth, RAID 5 – 683 MB/sec vs. RAID 5 - 1.45 GB/sec

RAID-5 untuned *vs* tuned: Reads

- Imdd, Sequential I/O to a Shared File
  - Stripe Count = 8 OSTs, Stripe Size = 1MB
  - Read requests 4kB, … 16MB



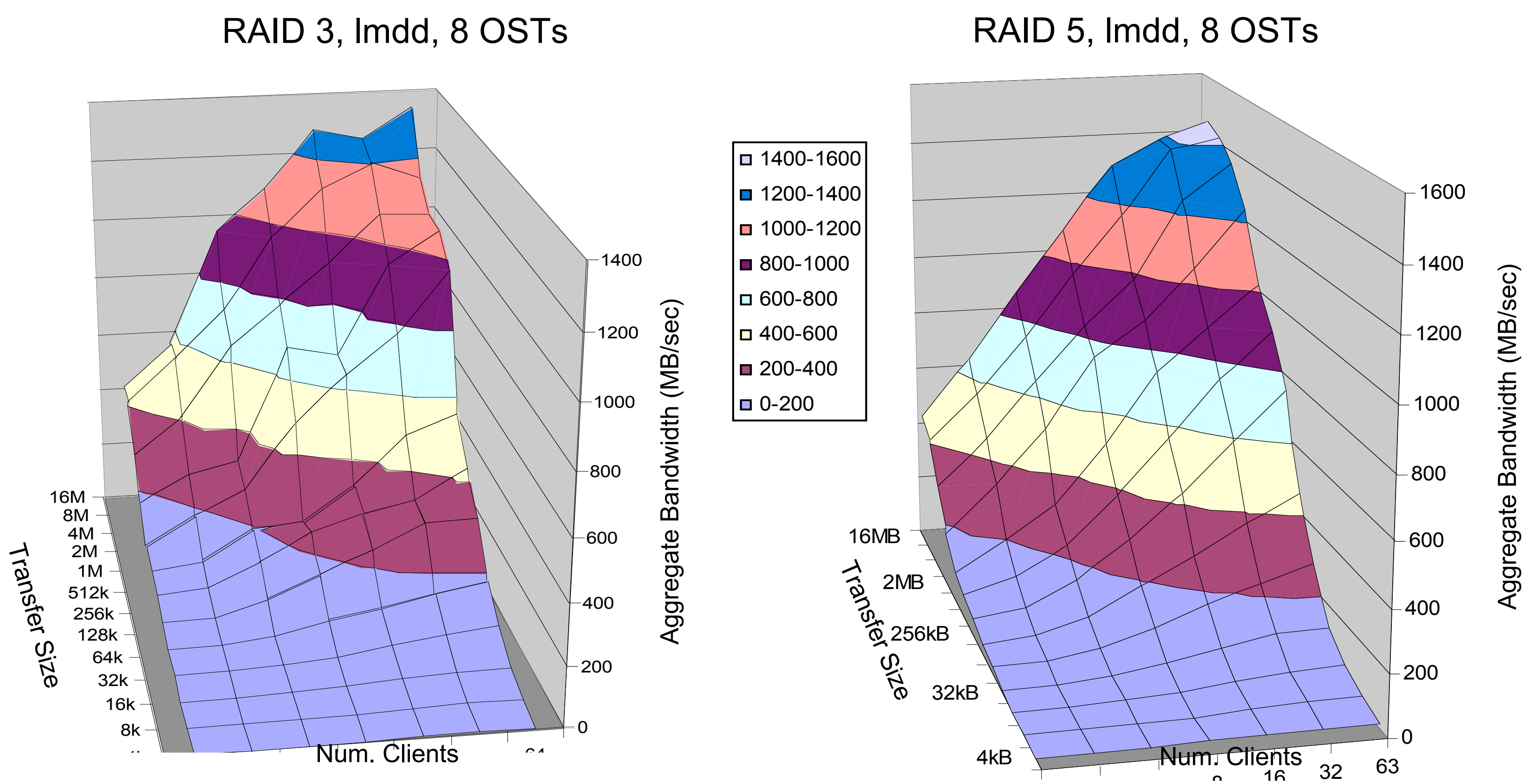
- Achieved bandwidth, RAID 5 – 1.53 GB/sec vs. RAID 5 - 1.55 GB/sec

Discussion: untuned *vs* tuned RAID5

- The following parameters have been tuned
  - Write-back cache size big enough (> 750 MB)
  - Cache high/low watermark should not be set too low or high
  - Cache segment alignment might need to match data layout on the physical disks
- Settings are critical for the RAID 5 performance
  - Factory default settings might not yield the best performance
- Overall I/O performance depends on how well back-end RAID could be tuned for the specific I/O pattern

RAID-3 *vs* tuned RAID-5: Writes

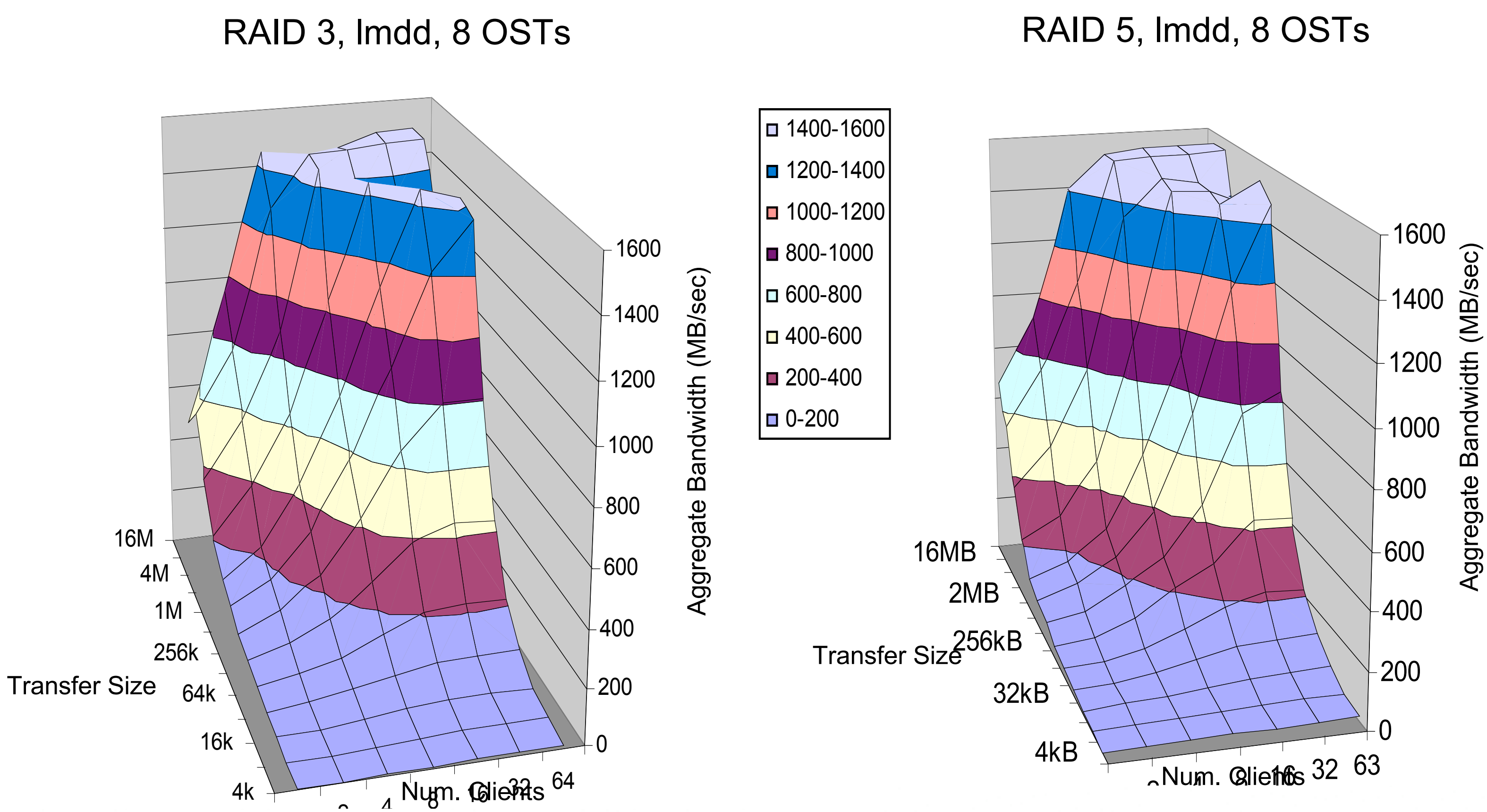
- lmdd, Sequential I/O to a Shared File
  - Stripe Count = 8 OSTs, Stripe Size = 1MB
  - Write requests 4kB, … 16MB



- Achieved bandwidth, RAID 3 - 1.34 GB/sec vs. RAID 5 - 1.45 GB/sec

RAID-3 *vs* RAID-5: Reads

- lmdd, Sequential I/O to a Shared File
  - Stripe Count = 8 OSTs, Stripe Size = 1MB
  - Read requests 4kB, … 16MB



- Achieved bandwidth, RAID 3 - 1.49 GB/sec vs. RAID 5 - 1.55 GB/sec

Discussion: RAID3 *vs* RAID5

- Properly tuned RAID 5 system can outperform RAID 3 for both reads and writes
- Preliminary test results show that RAID 5 could outperform RAID 3 for random I/O too. However, further testing needs to be done to get the complete picture
- Overall I/O performance depends on how well back-end RAID could be tuned for the specific I/O pattern

Summary

- Our results, for sequential I/O, show that
  - Aggregate Lustre performance scales well with the number of OSTs
  - RAID mechanical latencies are the most critical factor for overall I/O performance
  - Properly tuned RAID 5 could outperform RAID 3 for both reads and writes
- I/O that may be sequential to the application can appear random to the RAID controller when there are multiple independent, simultaneous sequential I/Os
  - Overwhelming the cache in this way leads to performance degradation in both reads and writes; however, reads are affected much more than writes.
  - RAID has to be carefully tuned to maximize cache hits for reads

Questions?