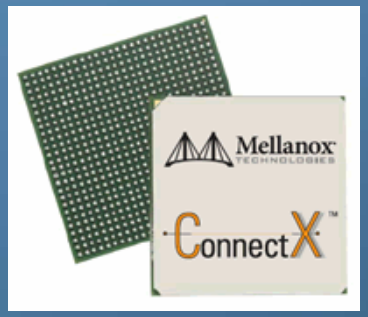
Z RESEARCH, Inc.

Commoditizing Supercomputing and Superstorage

Massive Distributed Storage over InfiniBand RDMA



What is GlusterFS?

GlusterFS is a Cluster File System that aggregates multiple storage bricks over InfiniBand RDMA into one large parallel network file system

❖ *GlusterFS* is *MORE* than making data available over a network or the organization of data on disk storage….

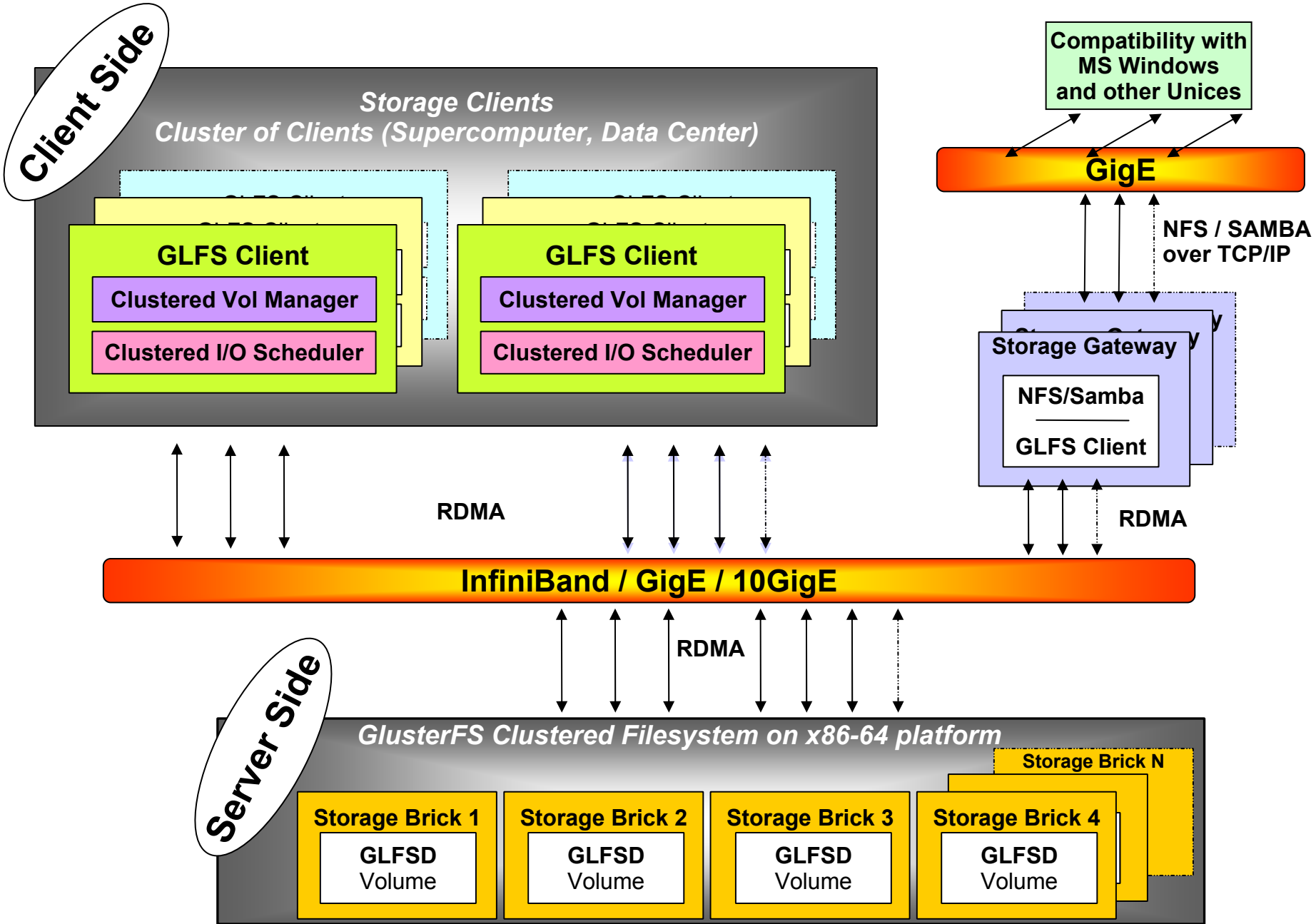
- Typical clustered file systems work to aggregate storage and provide unified views but….
  - scalability comes with increased cost, reduced reliability, difficult management, increased maintenance and recovery time….
  - limited reliability means volume sizes are kept small….
  - capacity and i/o performance can be limited…..

❖ *GlusterFS allows scaling of capacity and I/O using industry standard inexpensive modules!*

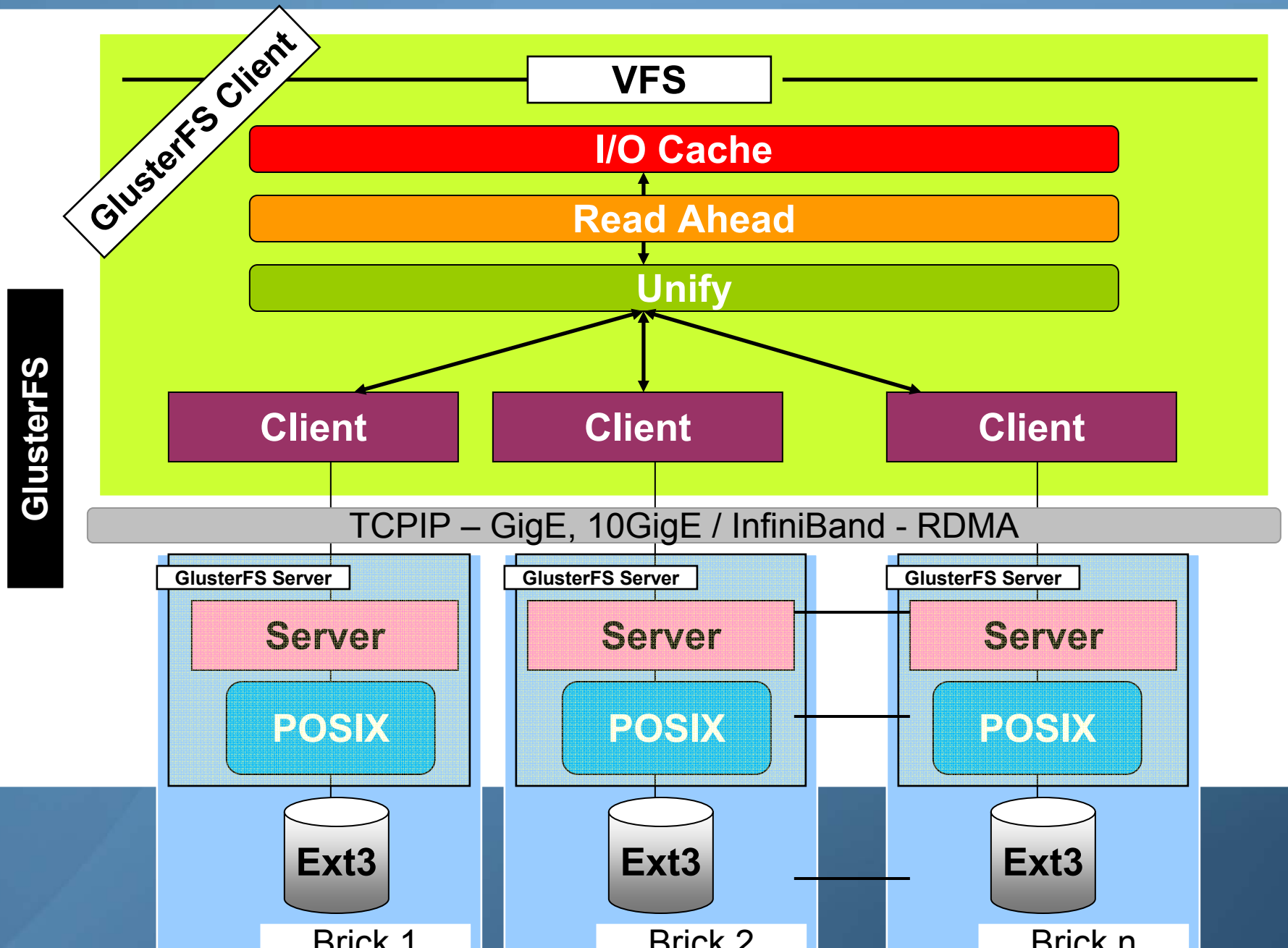
GlusterFS Features

1. Fully POSIX compliant!
2. Unified VFS!
3. More flexible volume management (stackable features)!
4. Application specific scheduling / load balancing
   - roundrobin; adaptive least usage; non-uniform file access (NUFA)!
5. Automatic file replication (AFR); Snapshot! and Undelete!
6. Striping for performance!
7. Self-heal! No fsck!!!!
8. Pluggable transport modules (IB verbs, IB-SDP)!
9. I/O accelerators - I/O threads, I/O cache, read ahead and write behind !
10. Policy driven - user group/directory level quotas , access control lists (ACL)

GlusterFS Design



Stackable Design



GlusterFS Function - unify

Client View (unify/roundrobin)

../files/aaa
../files/bbb
../files/ccc

Server/Head Node 1

../files/aaa

Server/Head Node 2

../files/bbb

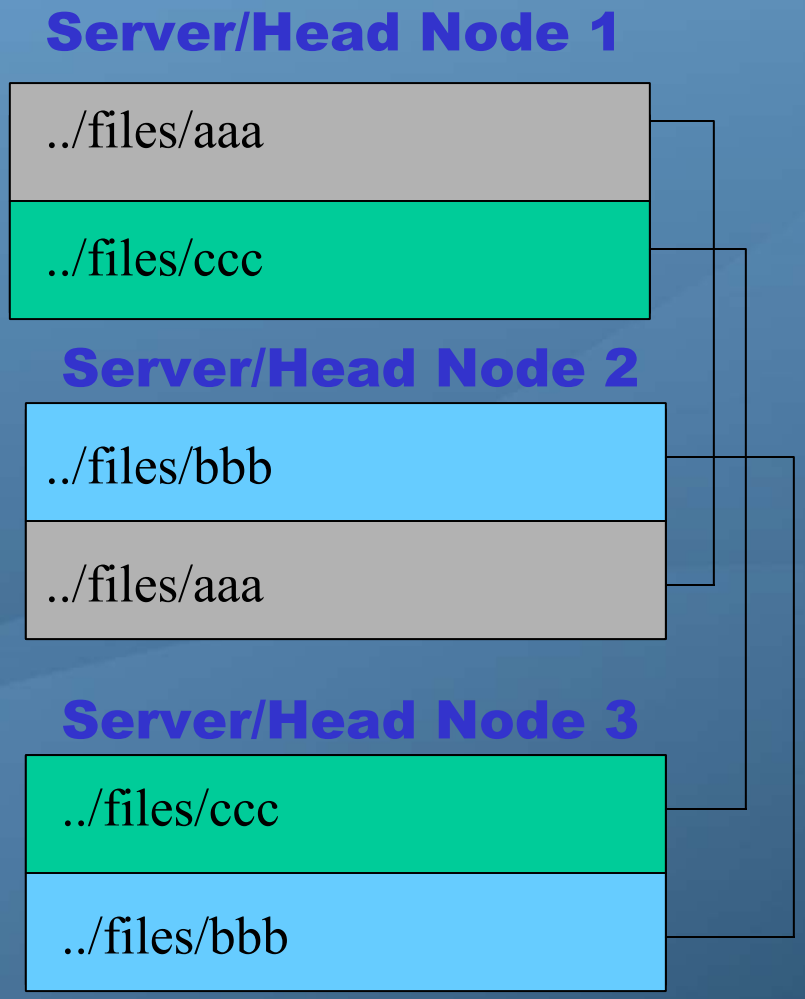
Server/Head Node 3

../files/ccc

GlusterFS Function – unify+AFR

Client View (unify/roundrobin+AFR)

../files/aaa
../files/bbb
../files/ccc



GlusterFS Function - stripe

Client View (stripe)

../files/aaa
../files/bbb
../files/ccc

Server/Head Node 1

../files/aaa
../files/bbb
../files/ccc

Server/Head Node 2

../files/aaa
../files/bbb
../files/ccc

Server/Head Node 3

../files/aaa
../files/bbb
../files/ccc

RESEARCH

I/O Scheduling

1. Round robin
2. Adaptive least usage (ALU)
3. NUFA
4. Random
5. Custom

```
volume bricks
  type cluster/unify
  subvolumes ss1c ss2c ss3c ss4c
  option scheduler alu
  option alu.limits.min-free-disk  60GB
  option alu.limits.max-open-files 10000
  option alu.order disk-usage:read-usage:write-usage:open-files-usage:disk-speed-usage
  option alu.disk-usage.entry-threshold 2GB         # Units in KB, MB and GB are allowed
  option alu.disk-usage.exit-threshold  60MB        # Units in KB, MB and GB are allowed
  option alu.open-files-usage.entry-threshold 1024
  option alu.open-files-usage.exit-threshold 32
  option alu.stat-refresh.interval 10sec
end-volume
```

Benchmarks

GlusterFS Throughput & Scaling Benchmarks

Benchmark Environment

Method: Multiple 'dd' of varying blocks are read and written from multiple clients simultaneously.

GlusterFS Brick Configuration (16 bricks)
Processor - Dual Intel(R) Xeon(R) CPU 5160 @ 3.00GHz
RAM - 8GB FB-DIMM
Linux Kernel - 2.6.18-5+em64t+ofed111 (Debian)
Disk - SATA-II 500GB
HCA - Mellanox MHGS18-XT/S InfiniBand HCA

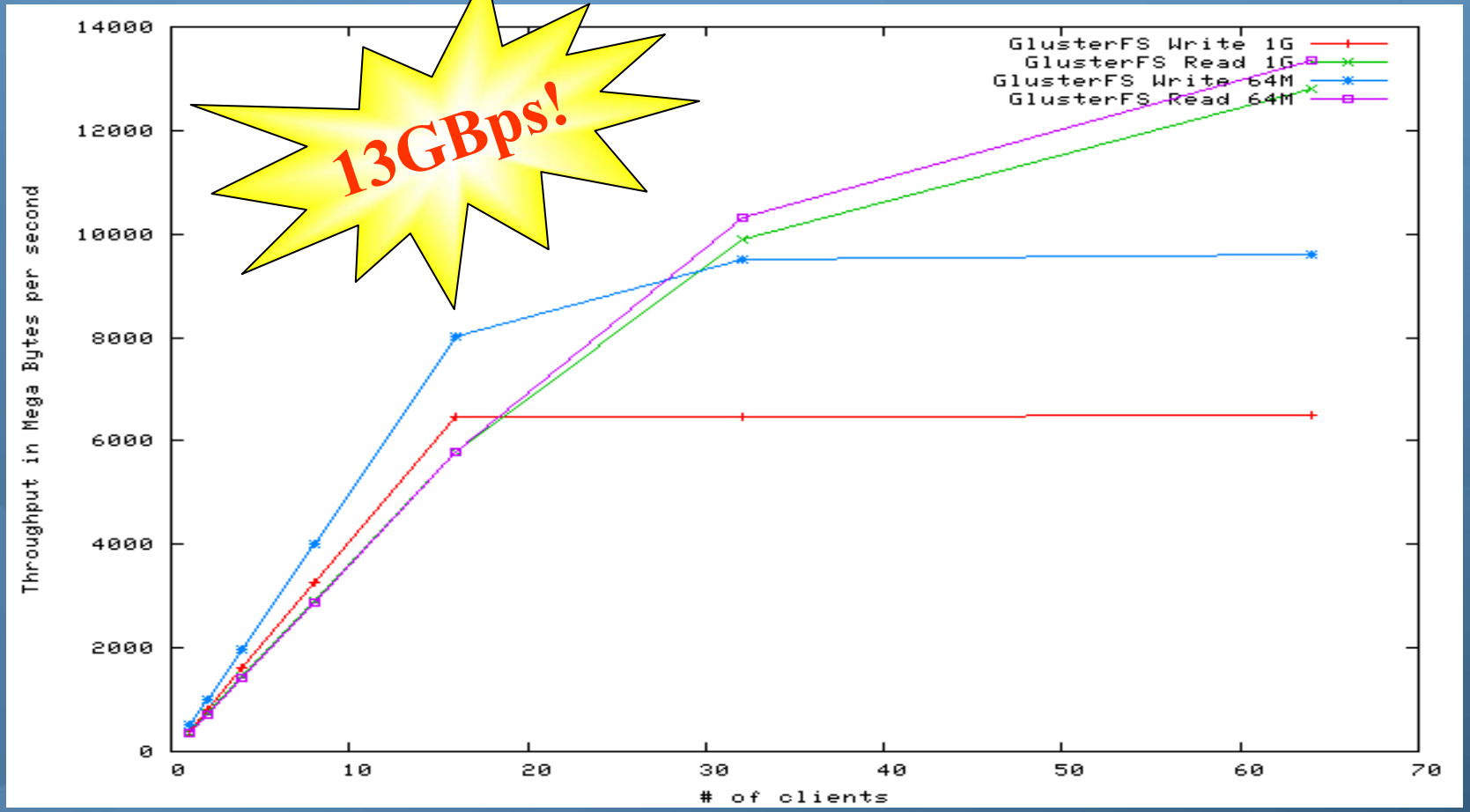
Client Configuration (64 clients)
RAM - 4GB DDR2 (533 Mhz)
Processor - Single Intel(R) Pentium(R) D CPU 3.40GHz
Linux Kernel - 2.6.18-5+em64t+ofed111 (Debian)
Disk - SATA-II 500GB
HCA - Mellanox MHGS18-XT/S InfiniBand HCA

Interconnect Switch: Voltaire port InfiniBand Switch (14U)

GlusterFS version *1.3.pre0-BENKI*

GlusterFS Performance

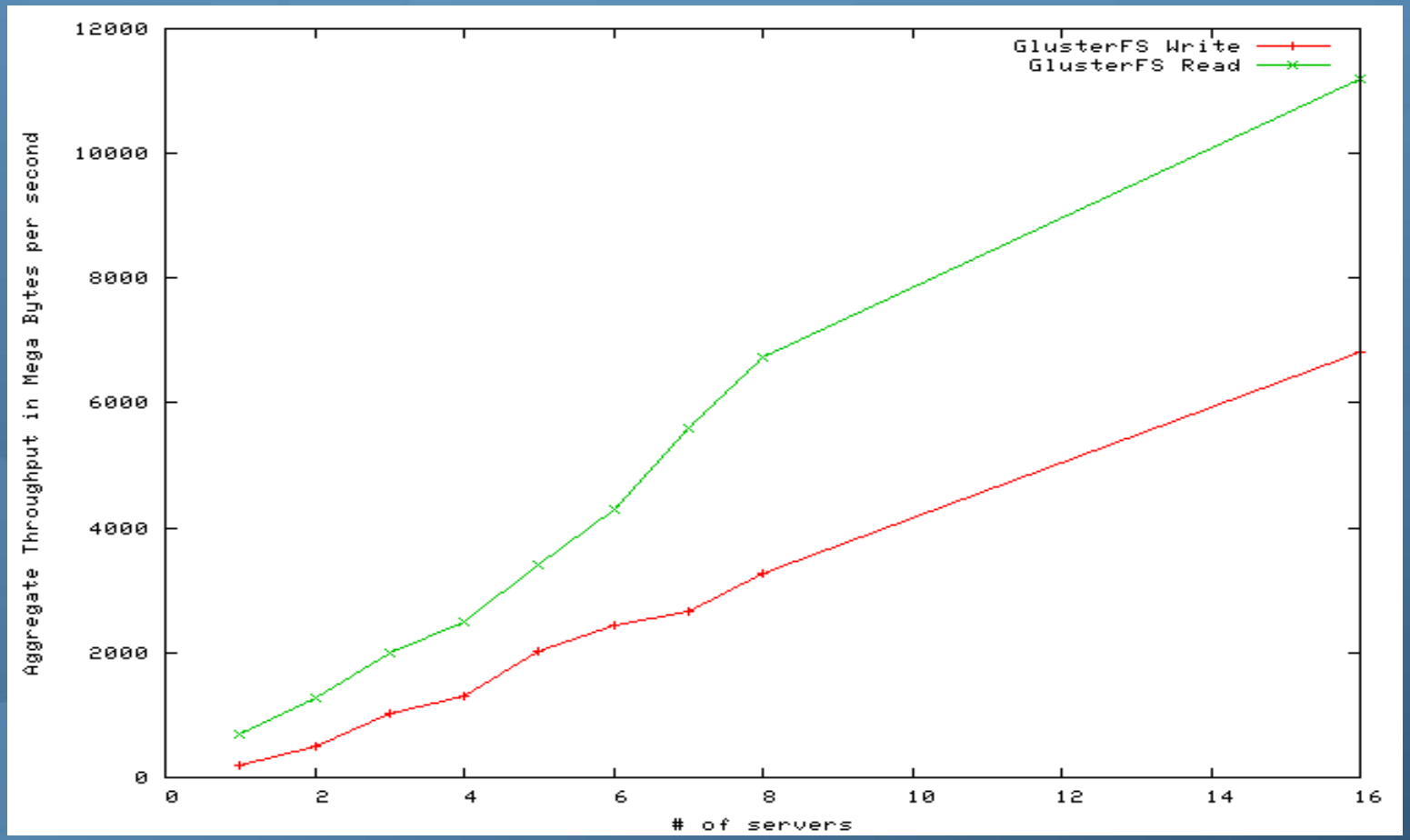
ggregated I/O Benchmark on 16 bricks(servers) and 64 clients over IB Verbs transpor



- Peak aggregated read throughput was 13 GBps.
- After a particular threshold, write performance plateaus because of disk I/O bottleneck.
- System memory greater than the peak load will ensure best possible performance.
- ib-verbs transport driver is about 30% faster than ib-sdp transport driver.

Scalability

Performance improves when the number of bricks are increased



Throughput increases with corresponding increased in servers from 1 to 16

GlusterFS Value Proposition

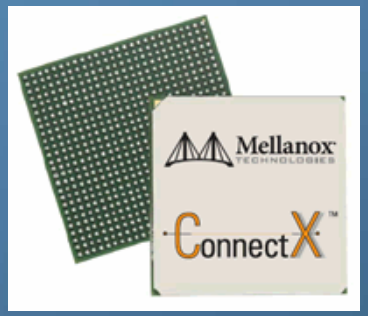
- A single solution for 10's of Terabytes to Petabytes
- No single point of failure – completely distributed - no centralized meta-data
- Non-stop Storage – can withstand hardware failures, sel healing, snap-shots
- **Data easily recovered even without GlusterFS**
- Customizable schedulers
- User Friendly - Installs and upgrades in minutes
- Operating system agnostic!
- Extremely cost effective – deployed on any x86-64 hardware!

RESEARCH

http://www.zresearch.com

http://www.gluster.org

Thank You!



Backup Slides

GlusterFS Vs Lustre Benchmark

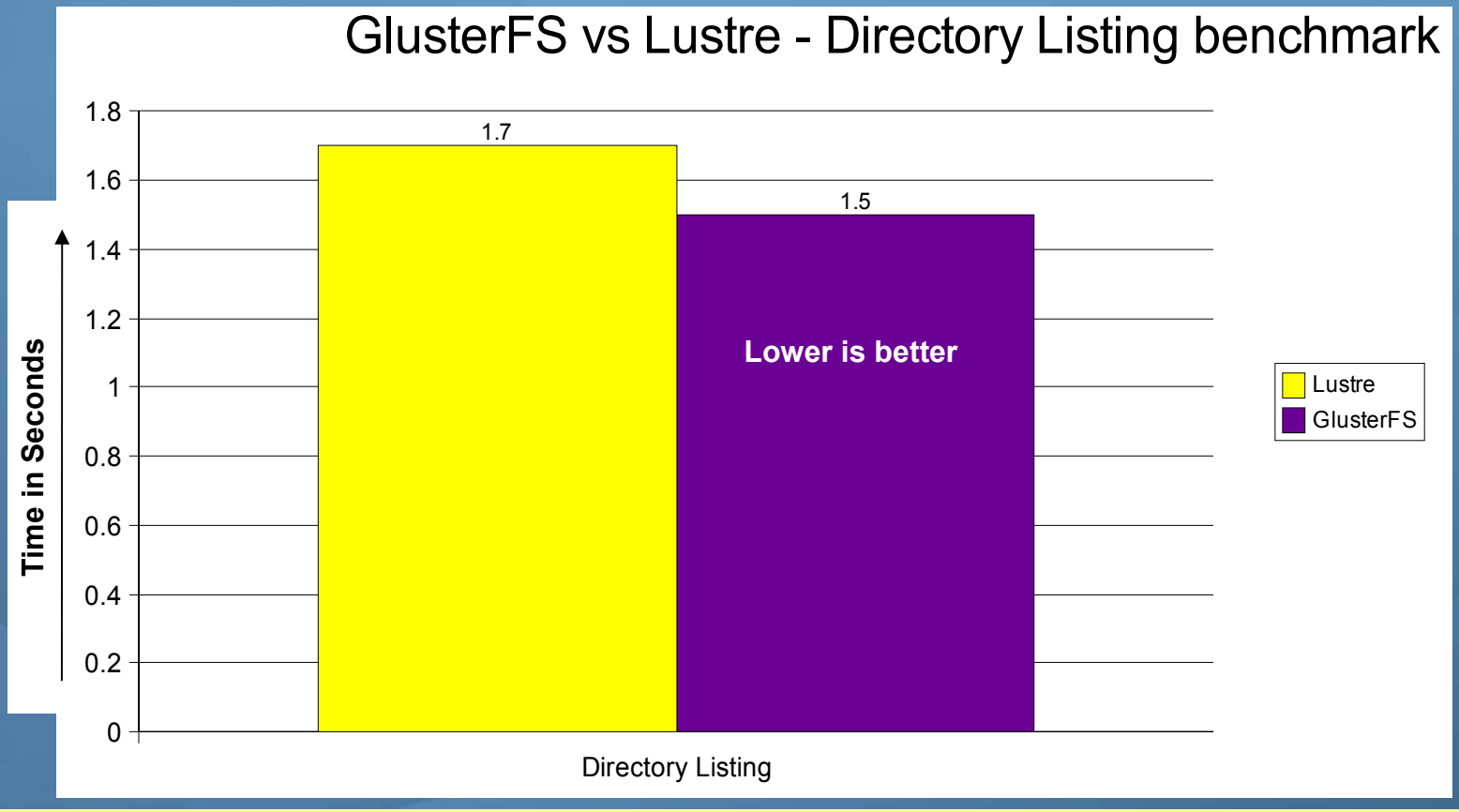
Benchmark Environment

Brick Config (10 bricks)
Processor - 2 x AMD Dual-Core Opteron™ Model 275 processors
RAM - 6 GB
Interconnect - InfiniBand 20 Gb/s - Mellanox MT25208 InfiniHost III Ex
Hard disk - Western Digital Corp. WD1200JB-00REA0, ATA DISK drive

Client Config (20 clients)
Processor - 2 x AMD Dual-Core Opteron™ Model 275 processors
RAM - 6 GB
Interconnect - InfiniBand 20 Gb/s - Mellanox MT25208 InfiniHost III Ex
Hard disk - Western Digital Corp. WD1200JB-00REA0, ATA DISK drive

Software Version
Operation System - Redhat Enterprise GNU/Linux 4 (Update 3)
Linux version - 2.6.9-42
Lustre version - 1.4.9.1
GlusterFS version - 1.3-pre2.3

Directory Listing Benchmark

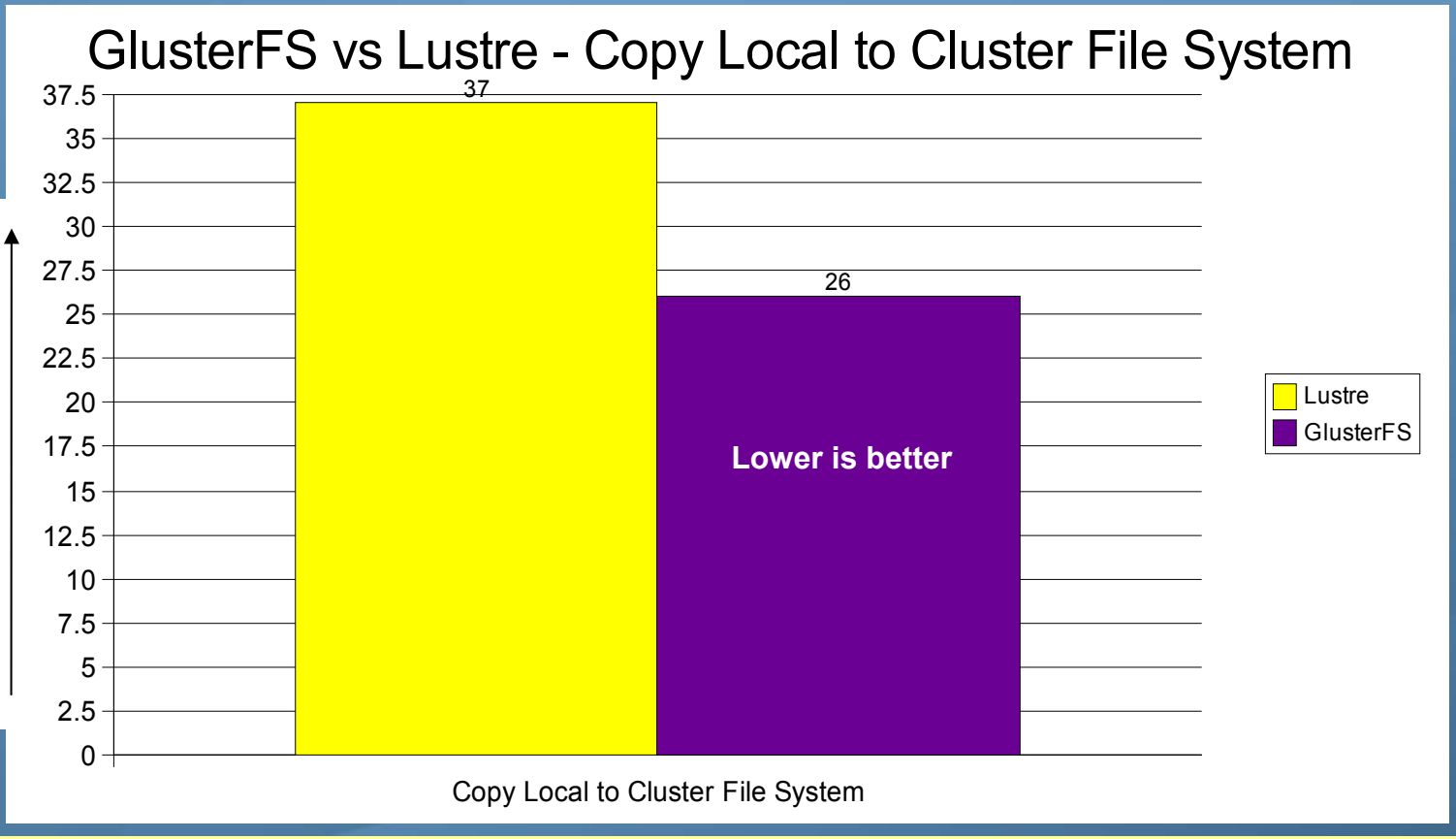


$ find /mnt/glusterfs

"find" command navigates across the directory tree structure and prints them to console. In this case, there were thirteen thousand binary files.

Note: Commands are same for both GlusterFS and Lustre, except the directory part.

Copy Local to Cluster File System

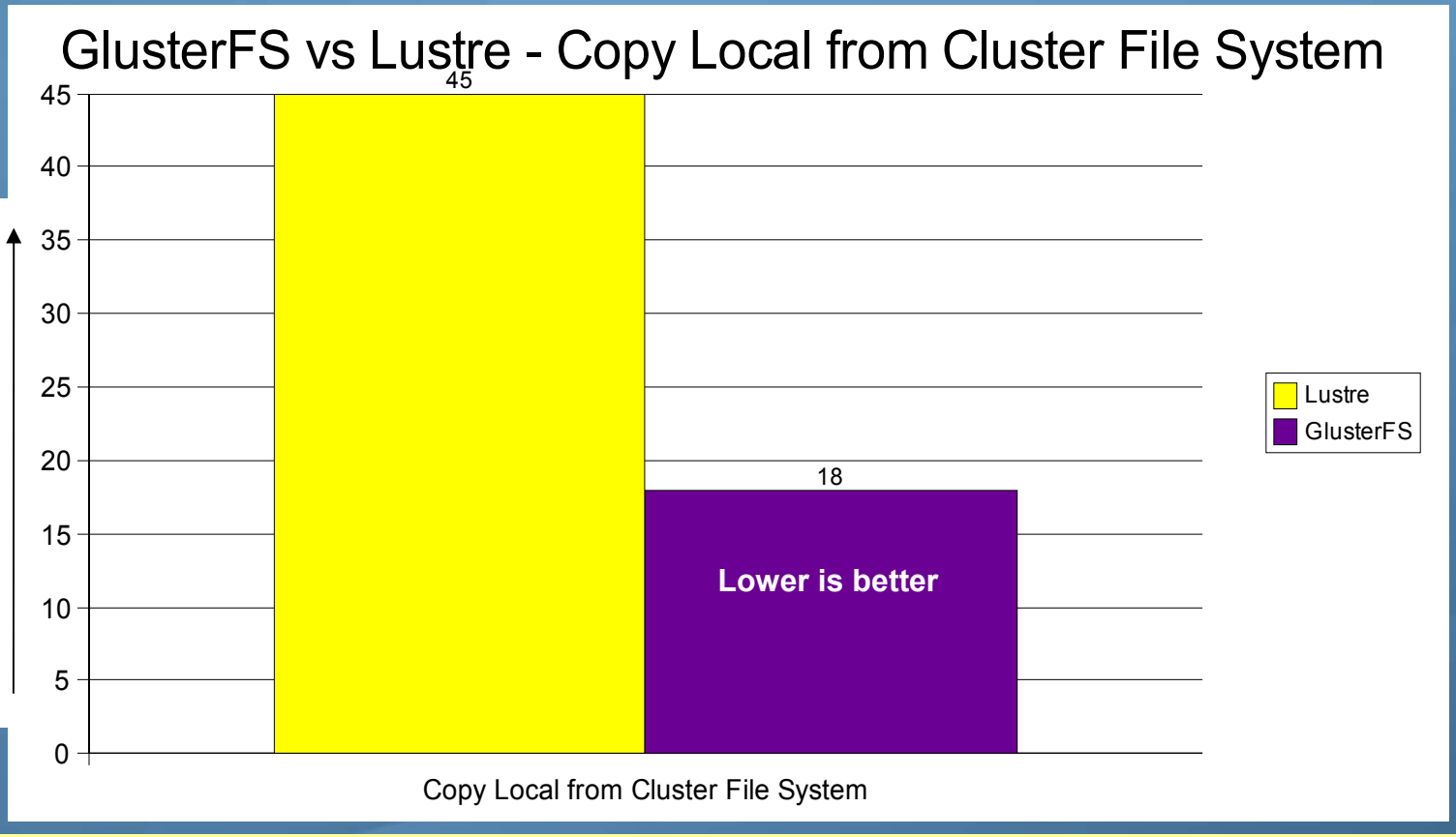


$ cp -r /local/* /mnt/glusterfs/

cp utility is used to copy files and directories.

Copy 12039 files (595 MB) were copied into the cluster file system.

Copy Local from Cluster File System

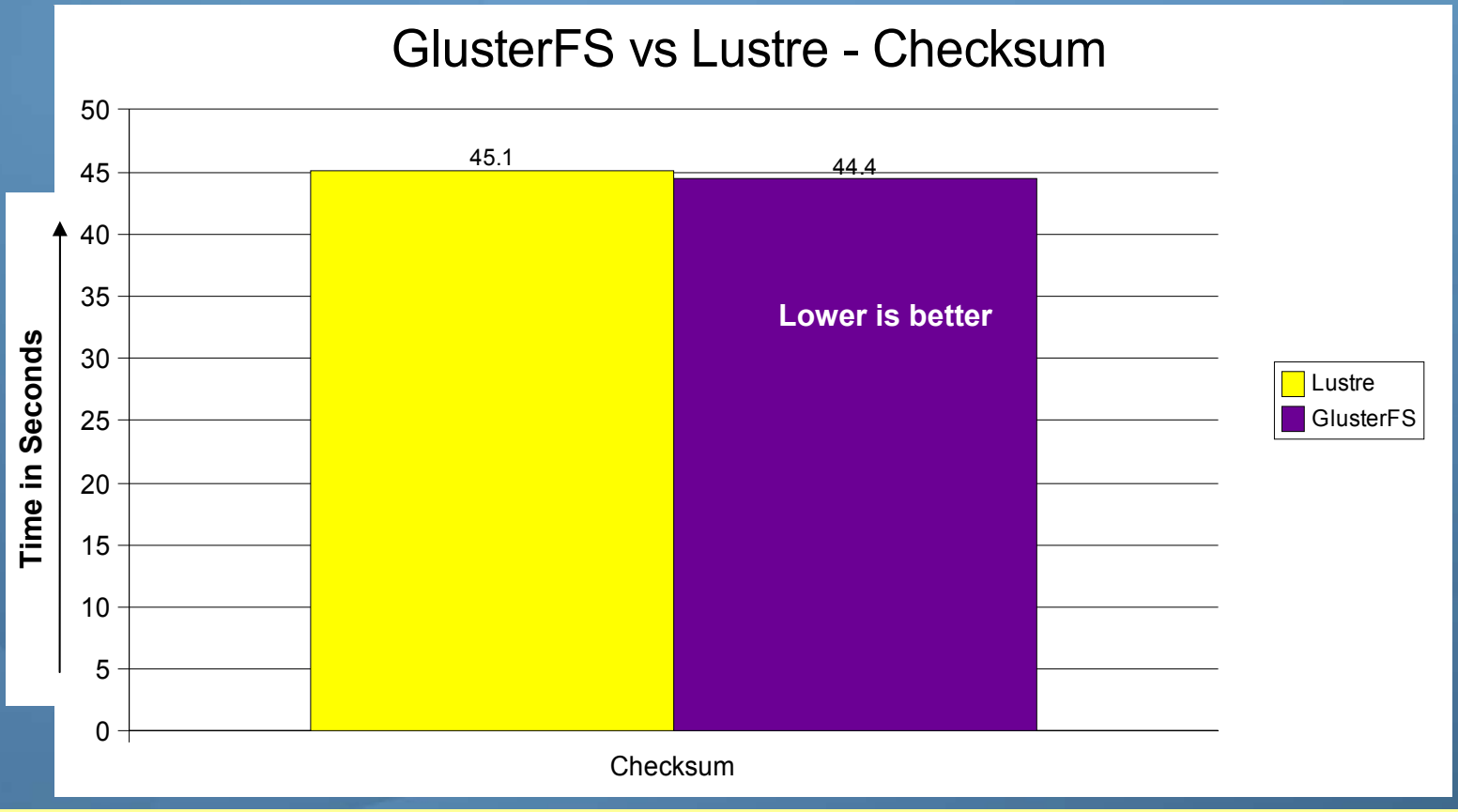


$ cp -r /mnt/glusterfs/ /local/*

cp utility is used to copy files and directories.

Copy 12039 files (595 MB) were copied from the cluster file system.

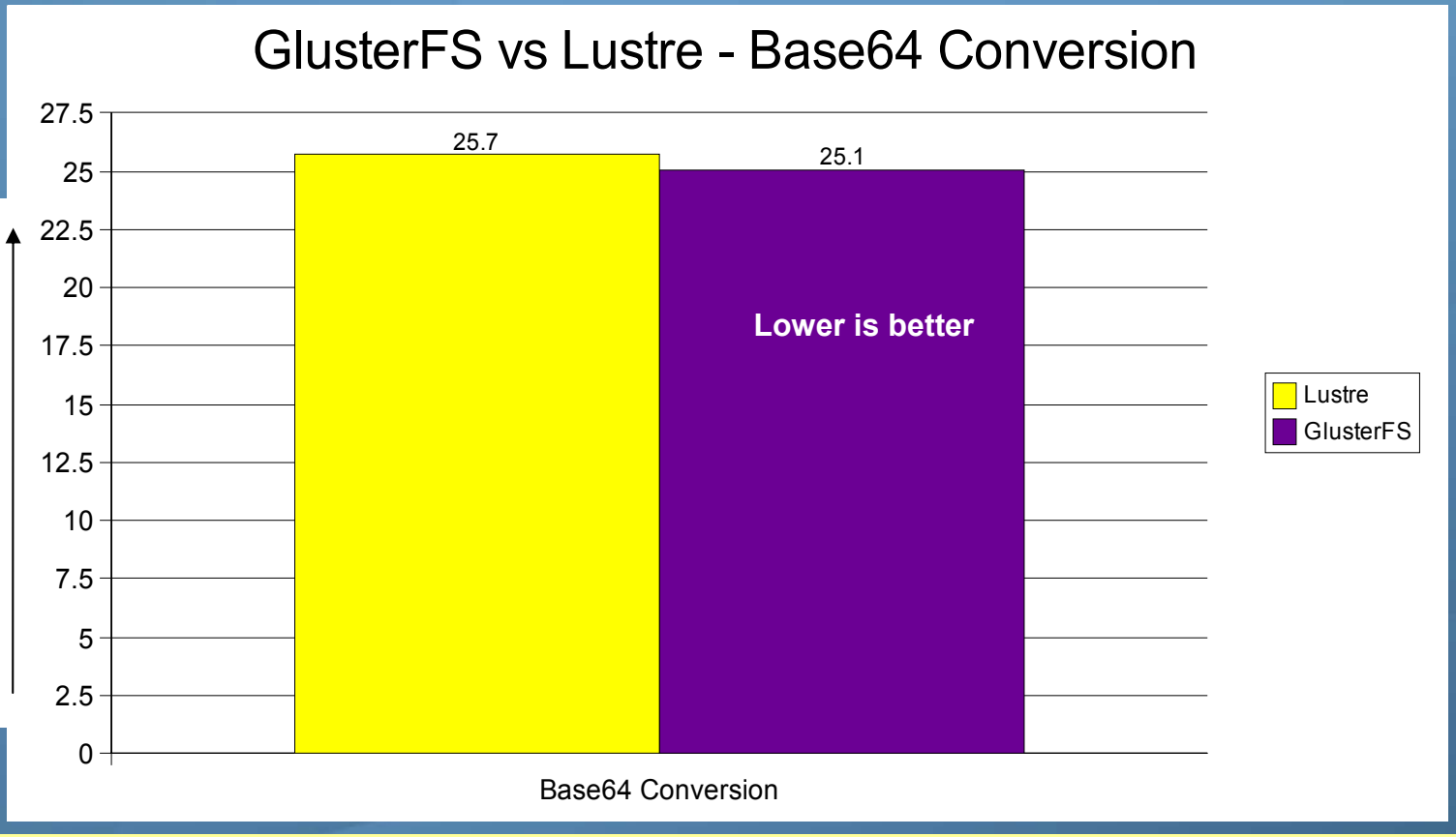
Checksum



Perform md5sum calculation for all files across your file system. In this case, there were thirteen thousand binary files.

$ find . -type f -exec md5sum {} \;

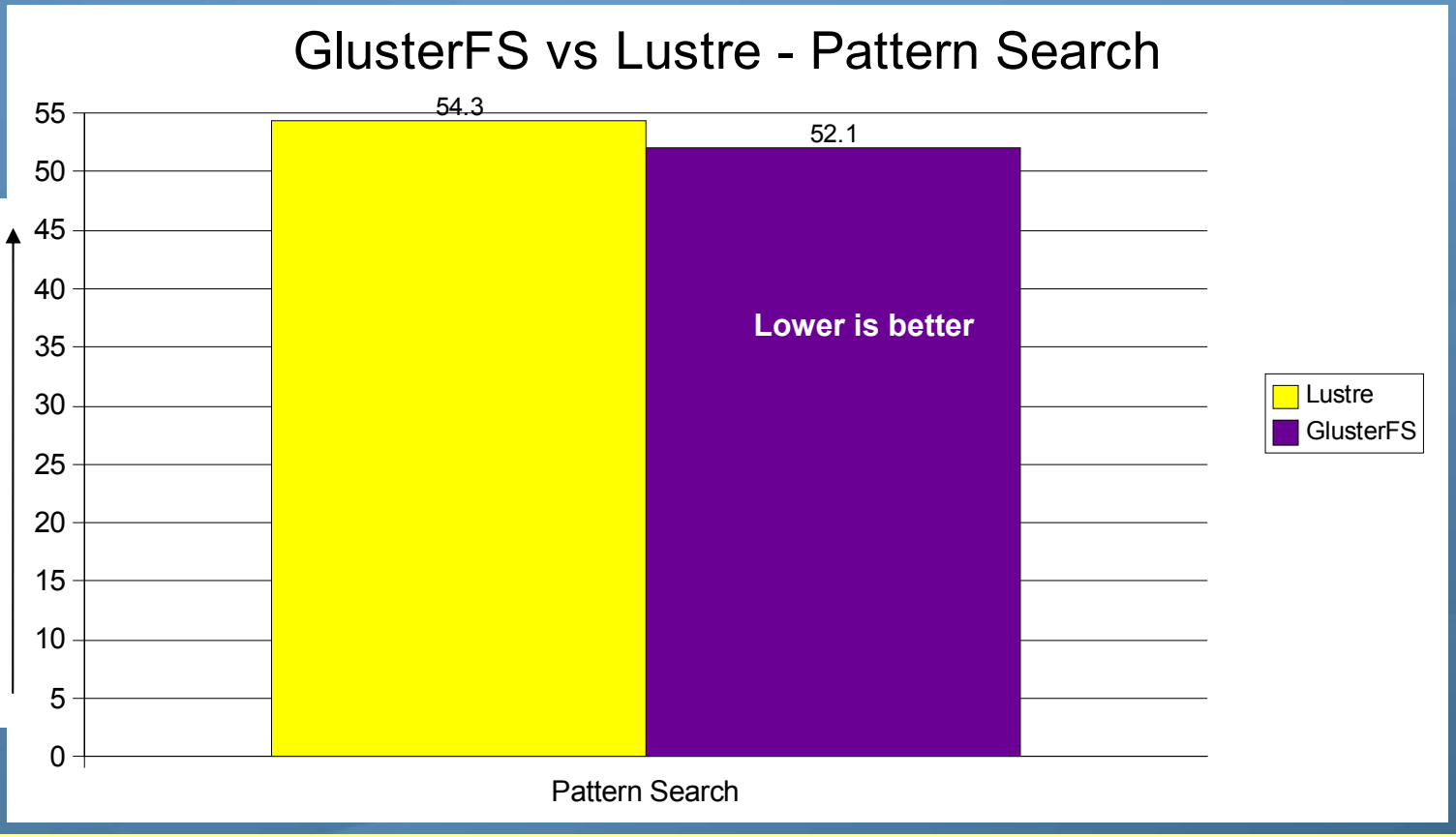
Base64 Conversion



Base64 is an algorithm for encoding binary to ASCII and vice-versa. This benchmark was performed on a 640 MB binary file.

$ base64 --encode big-file big-file.base64

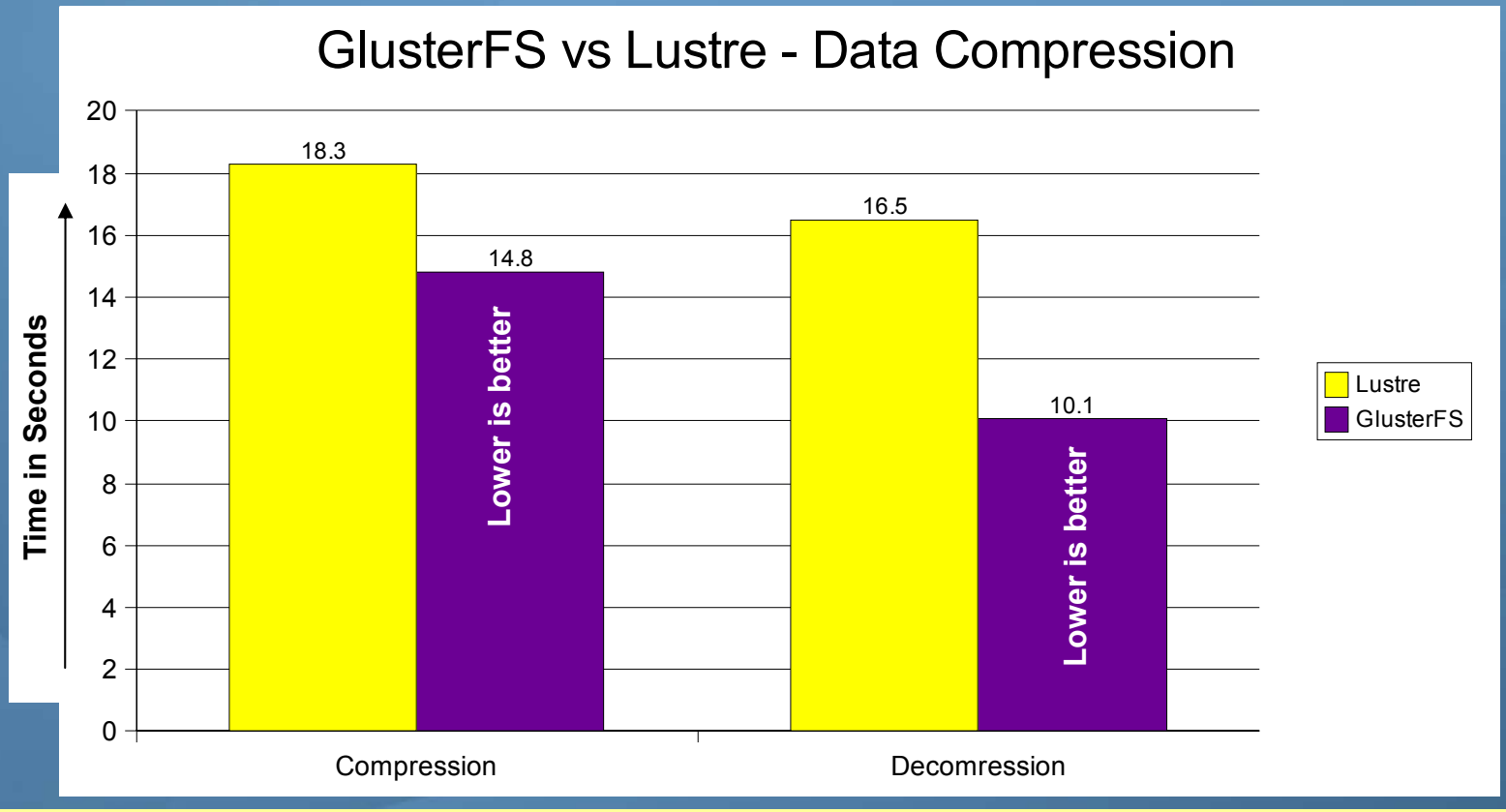
Pattern Search



grep utility searches for a PATTERN on a file and prints the matching lines to console. This benchmark used 1GB ASCII BASE-64 file.

$ grep GNU big-file.base64

Data Compression

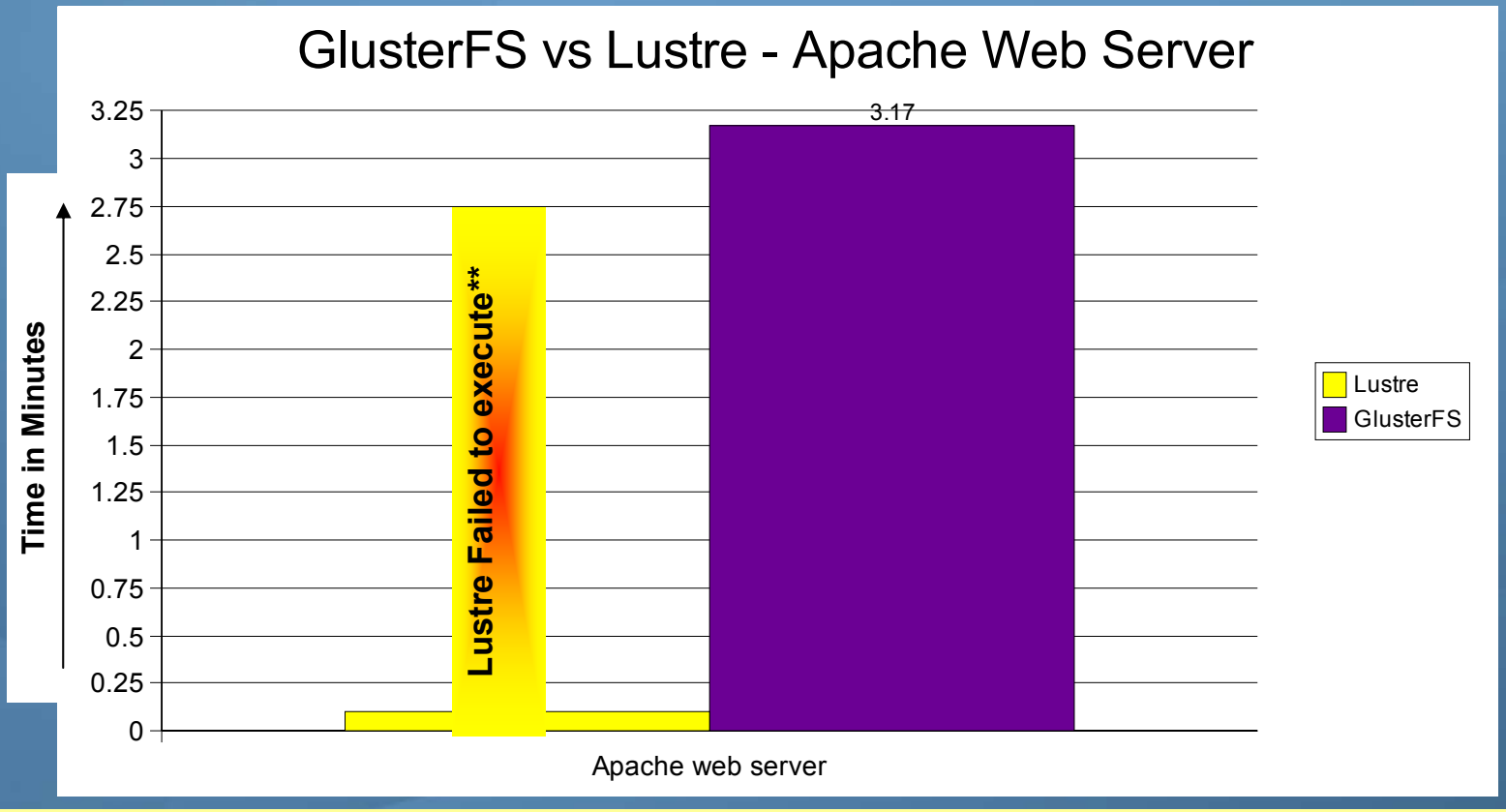


GNU gzip utility compresses files using Lempel-Ziv coding.

This benchmark was performed on 1GB TAR binary file.

$ gzip big-file.tar
$ gunzip big-file.tar.gz

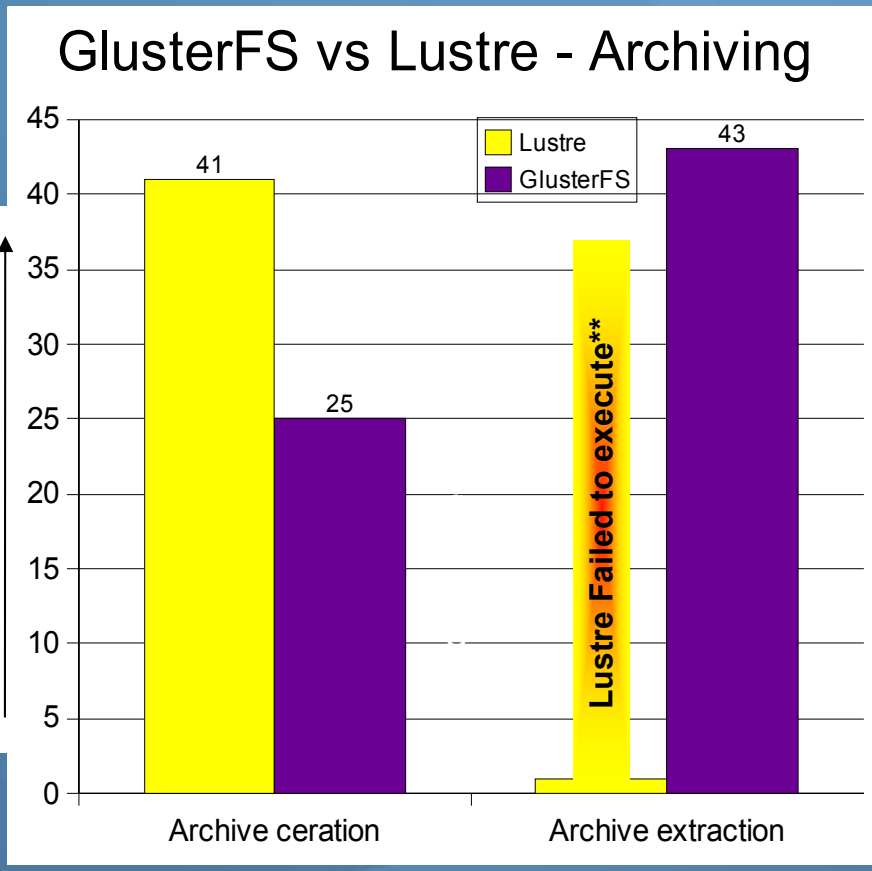
Apache Web Server



Apache served 12039 files (595 MB) over HTTP protocol. wget client fetched the files recursively.

****Lustre failed after downloading 33 MB out of 585 MB in 11 mins.**

Archiving



Archive Creation

tar utility is used for archiving filesystem data.
$ tar czf benchmark.tar.gz /mnt/glusterfs
'tar utility created an archive of 12039 files (595 MB) served through GlusterFS.

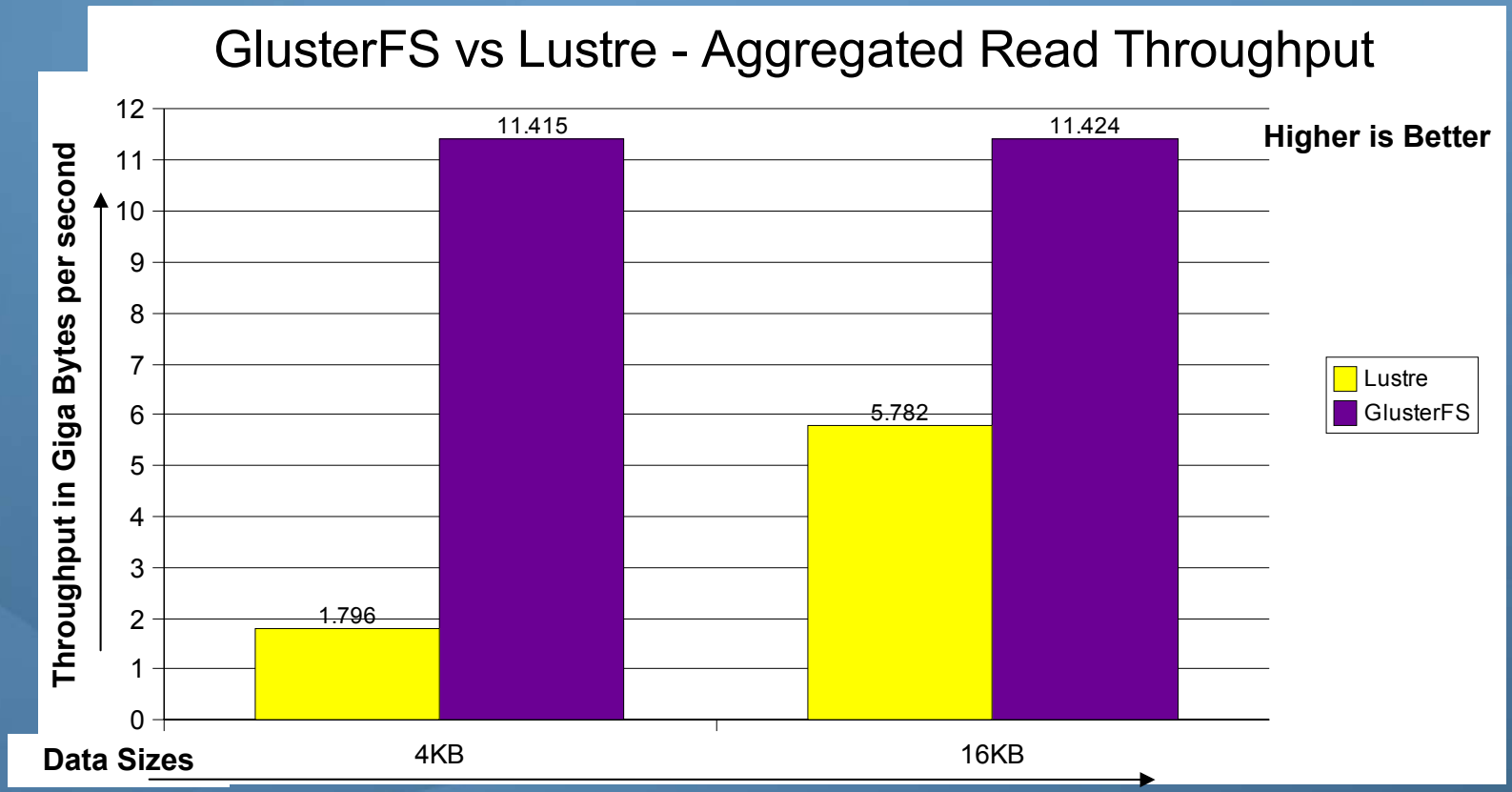
Archive Extraction

'tar utility extracted the archive on to GlusterFS filesystem.
$ tar xzf benchmark.tar.gz

**Lustre Falied to Execute:-

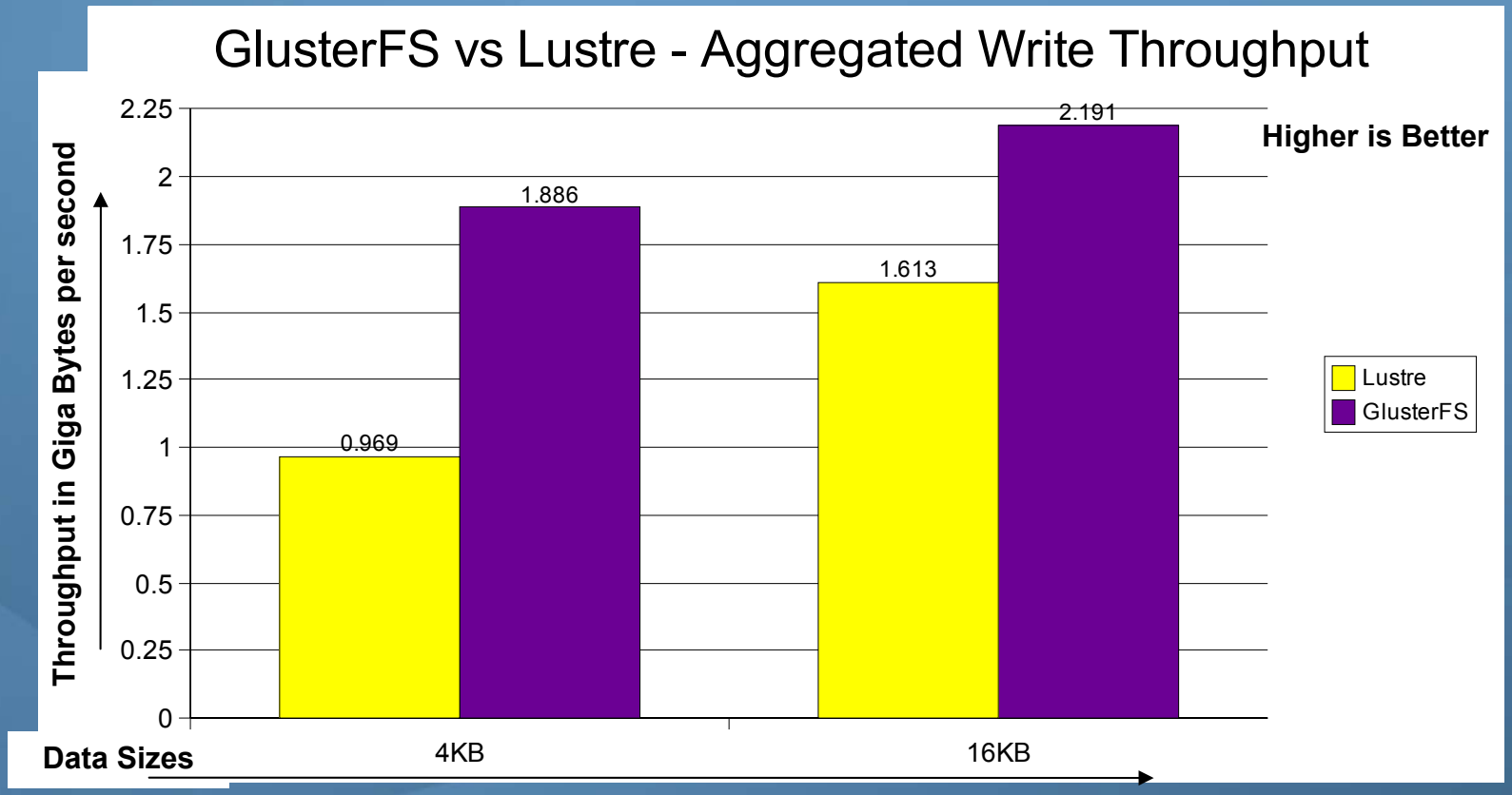
Tar extraction failed under Lustre with no space left on the devic error.
Disk-free (df -h) utility revealed lot of free space. It appears like I/O scheduler performed a poor job in distributing the data evenly
→tar: lib/libtdsS.so.1: Cannot create symlink to `libtdsS.so.1.0.0': No space left on device
→tar: lib/libg.a: Cannot open: No space left on device
→tar: Error exit delayed from previous errors

Aggregated Read Throughput



Multiple dd utility were executed simultaneously with different block sizes to read from GlusterFS filesystem.

Aggregated Write Throughput



Multiple dd utility were executed simultaneously with different block sizes to write to GlusterFS Filesystem.