BOSTON, MA
JUNE 23-26, 2015

Red Hat Gluster Storage performance

Manoj Pillai and Ben England
Performance Engineering
June 25, 2015

redhat.

New or improved features (in last year)

Erasure Coding

NFS-Ganesha

RDMA

Snapshots

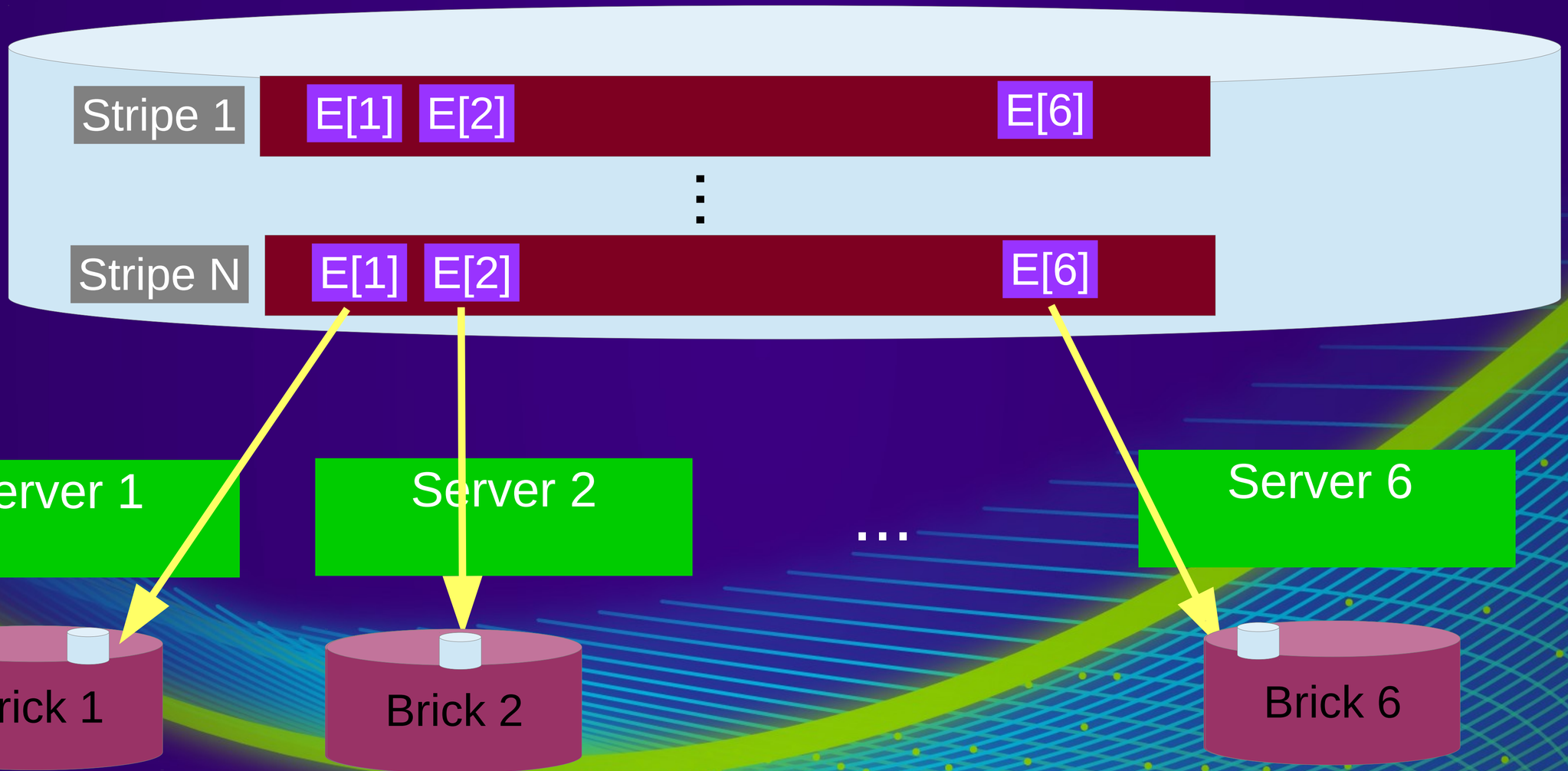
SSD support

Erasure Coding

"distributed software RAID"

- Alternative to RAID controllers or 3-way replication
- Cuts storage cost/TB, but computationally expensive
- Better Sequential Write performance for some workloads
- Roughly same sequential Read performance (depends on mountpoints)
- In RHGS 3.1 avoid Erasure Coding for pure-small-file or pure random I/O workloads
- Example use cases are archival, video capture

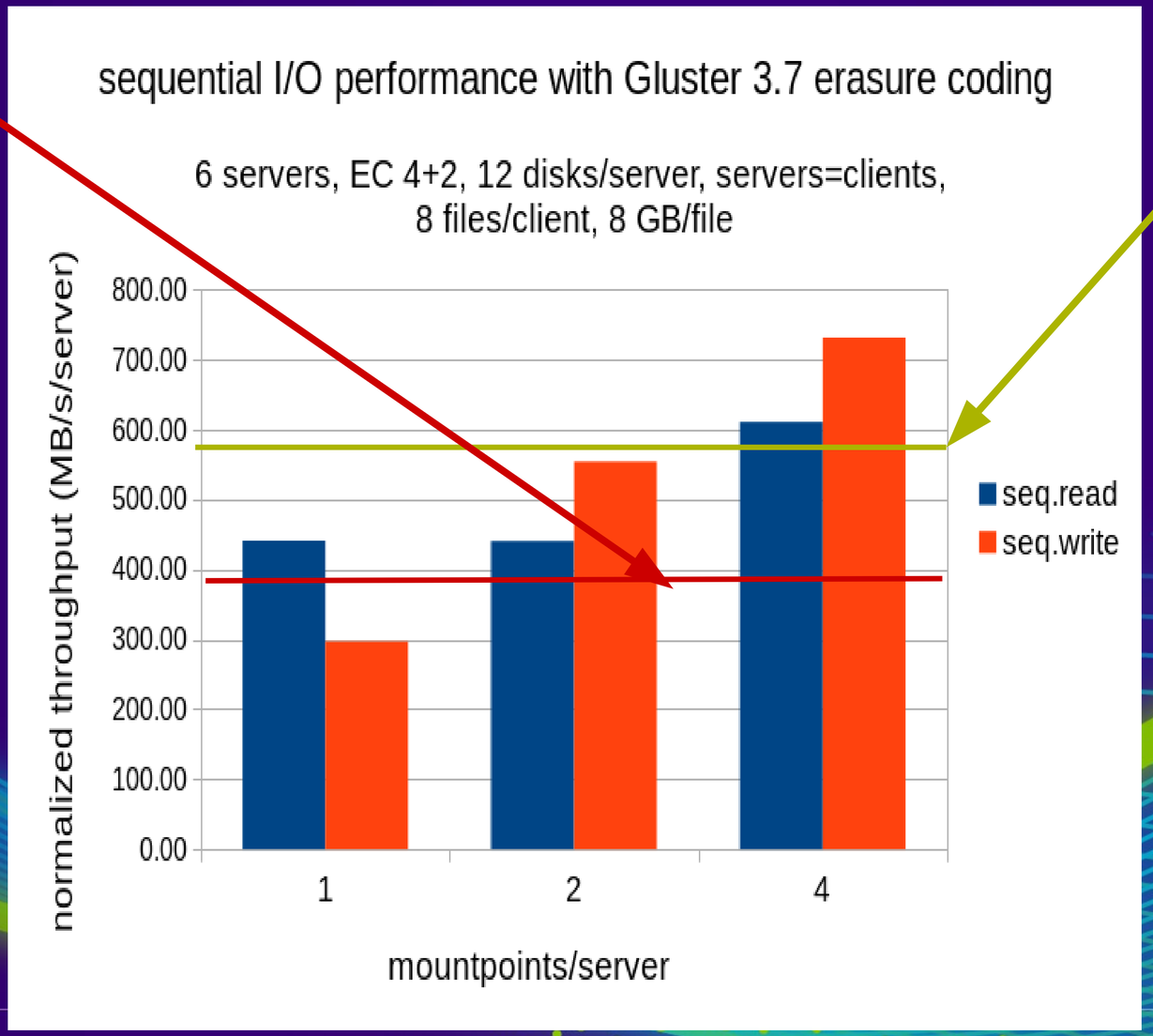
Disperse translator spreads EC stripes in file across hosts
Example: EC4+2



EC large-file perf summary

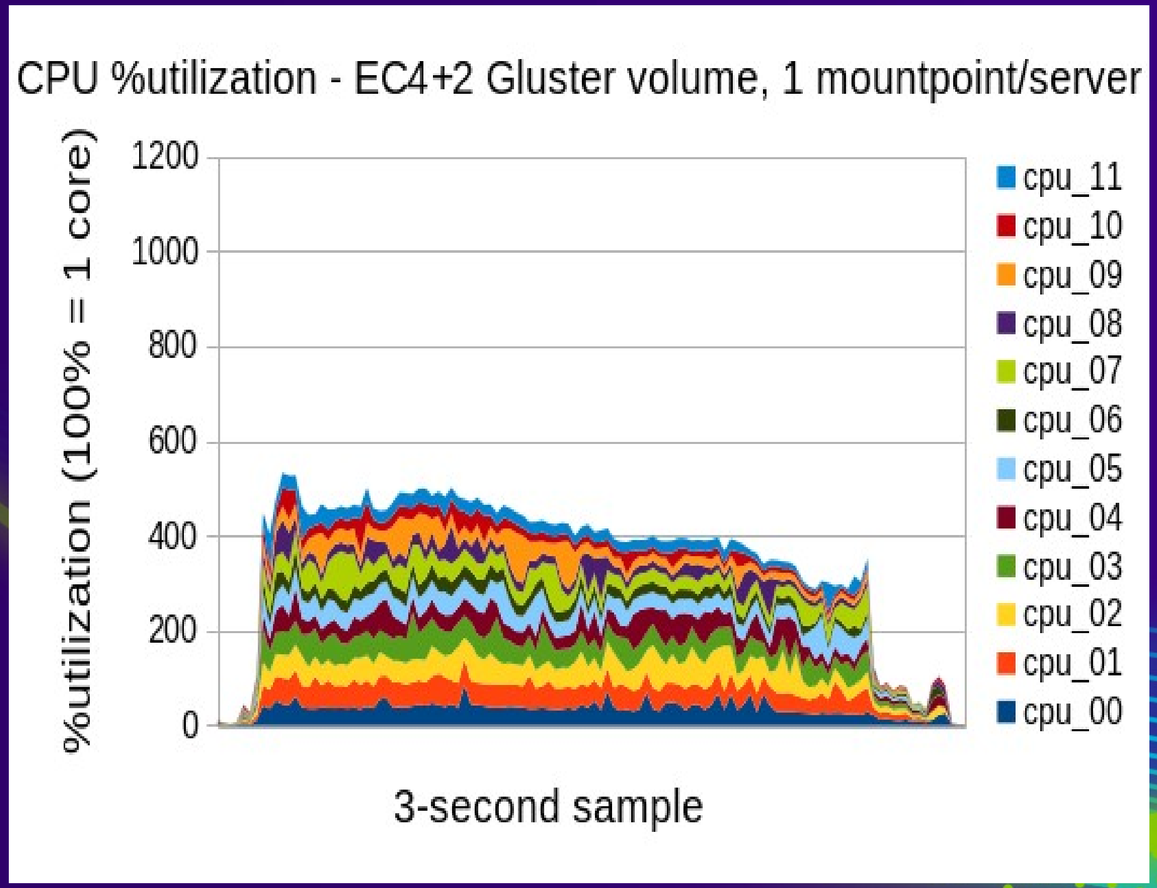
3-replica write limit

2-replica write limit

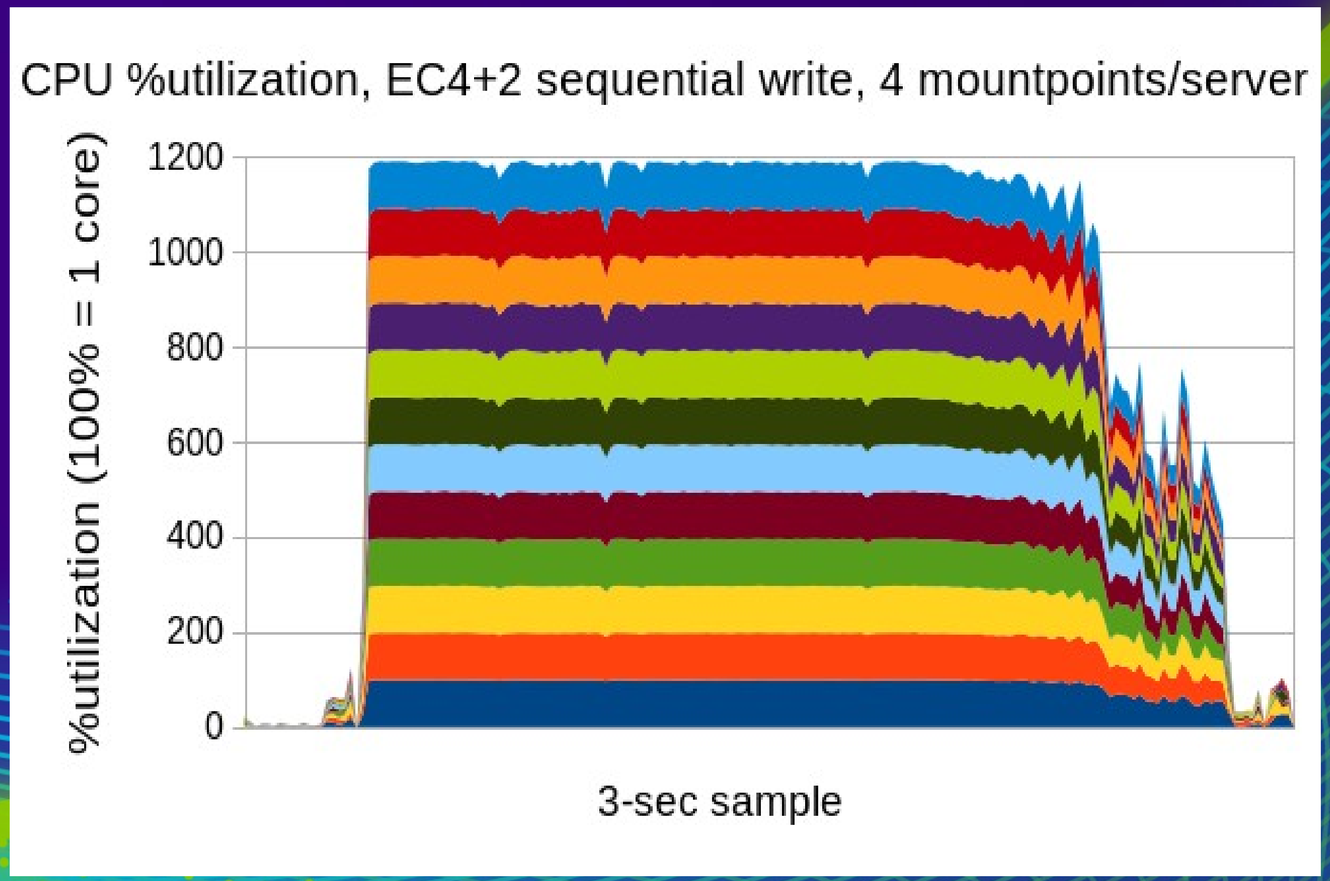


A tale of two mountpoints (per server)
stacked CPU utilization graph by CPU core, 12 cores
glusterfs hot thread limits 1 mountpoint's throughput

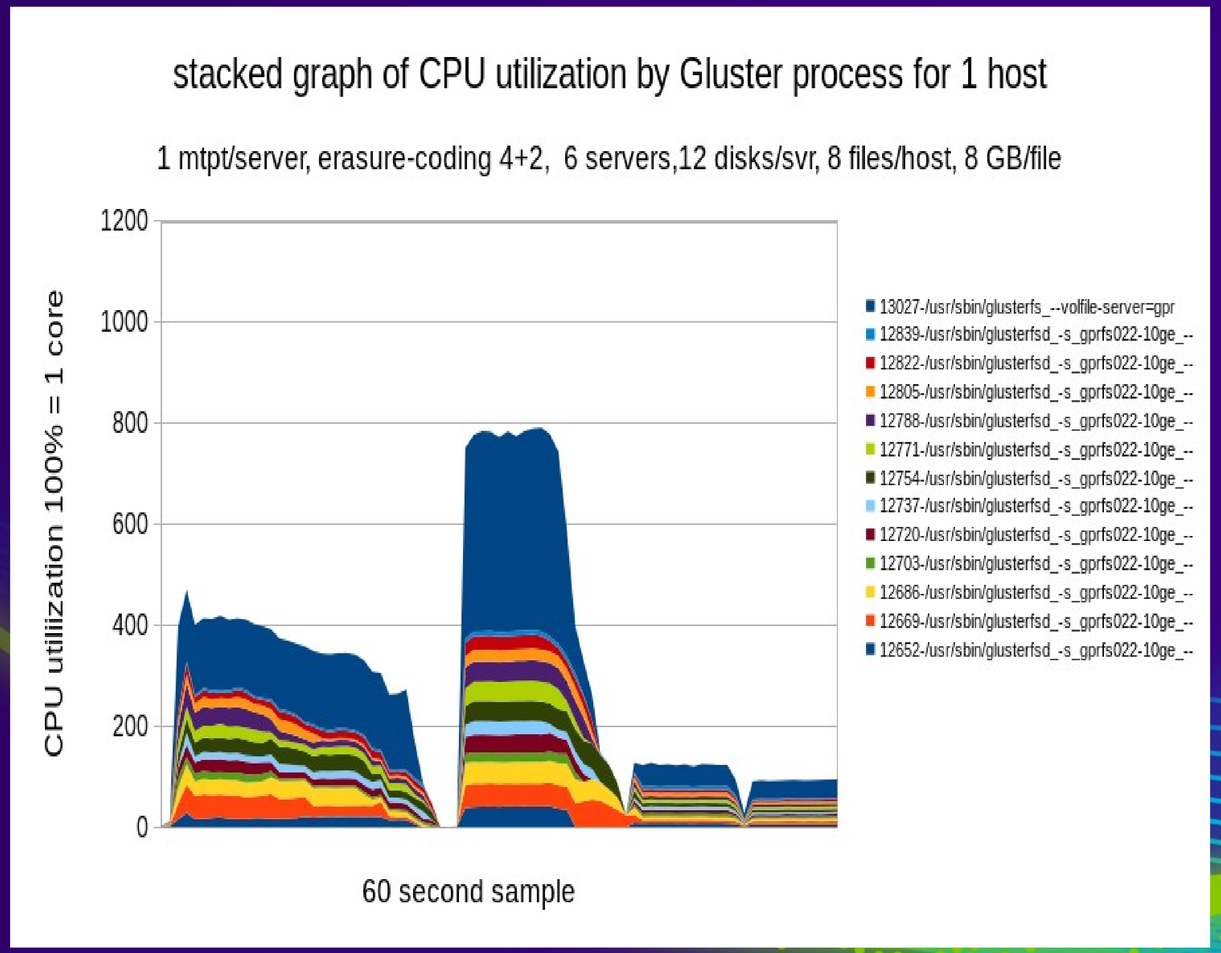
1 mountpoint per server

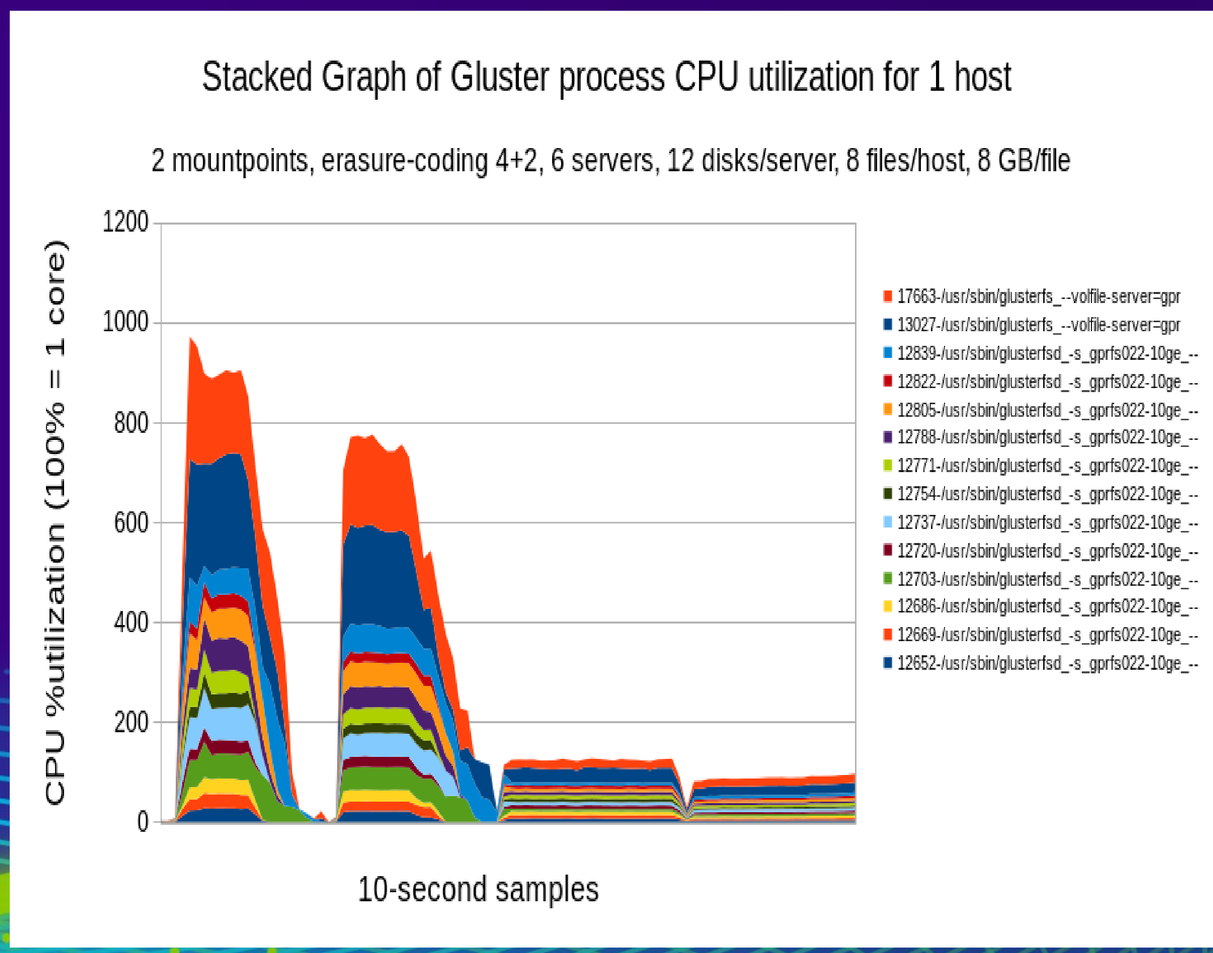


4 mountpoints per server



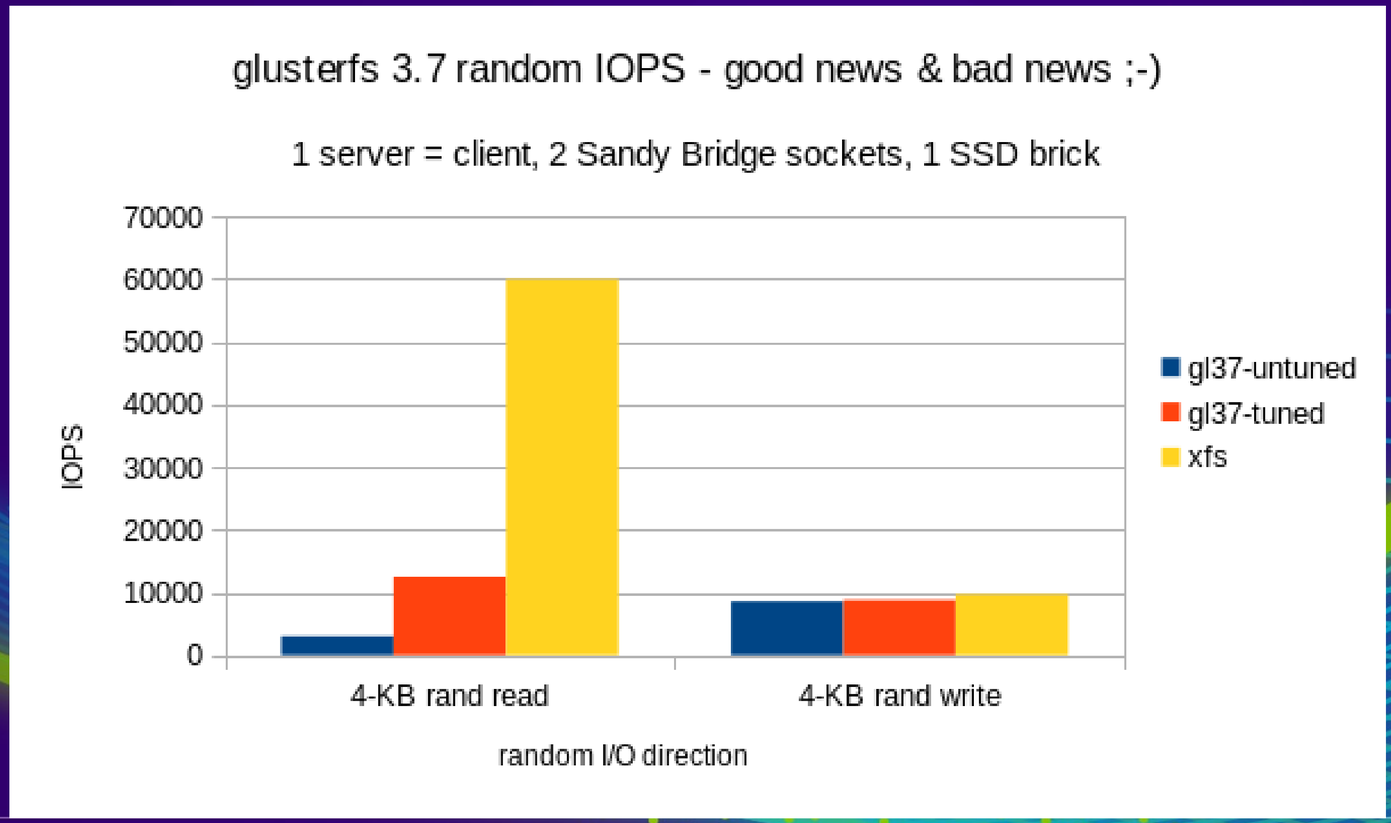
A tale of two mountpoints
why the difference in CPU utilization?





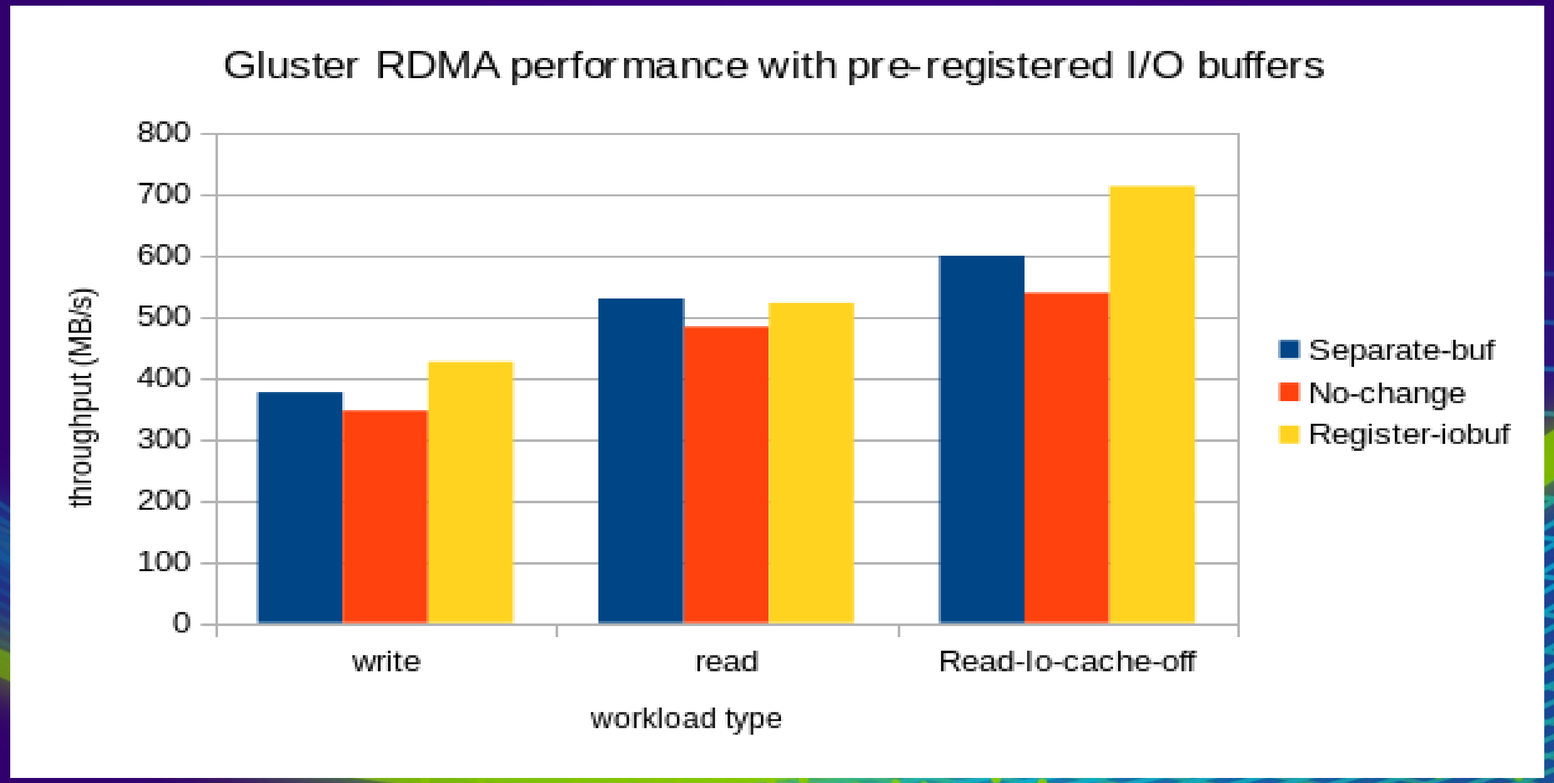
SSDs as bricks

- Multi-thread-epoll = multiple threads working in each client mountpoint and server brick
- Can be helpful for SSDs or any other high-IOPS workload
- glusterfs-3.7 with single 2-socket Sandy Bridge using 1 SAS SSD (SANDisk Lightning)
-



RDMA enhancements

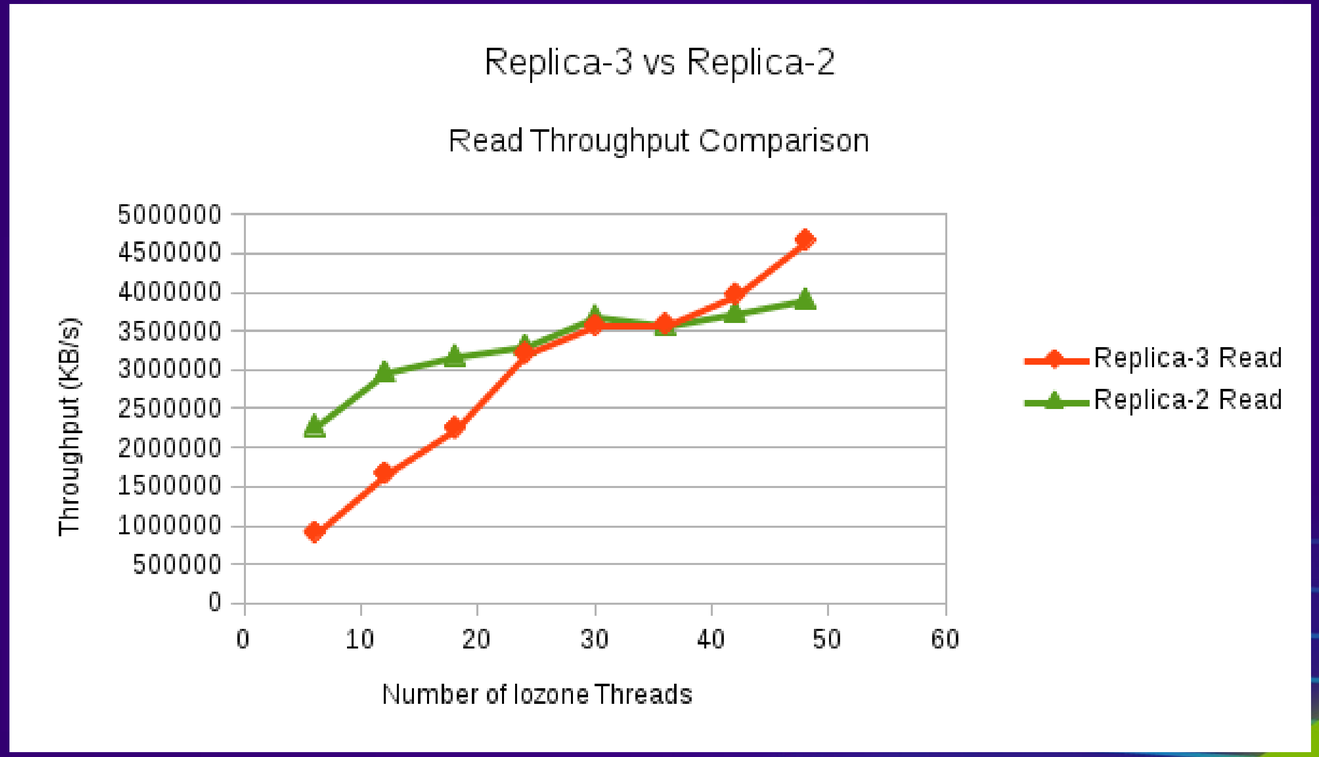
- Gluster has had RDMA in some form for a long time
- Gluster-3.6 added librdmacm support – broadens supported hardware
- By Gluster 3.7, memory pre-registration reduces latency



JBOD Support

- RHGS has traditionally used H/W RAID for brick storage, with replica-2 protection.
- RHGS 3.0.4 has JBOD+replica-3 support.
- H/W RAID problems:
  - Proprietary interfaces for managing h/w RAID
  - Performance impact with many concurrent streams
- JBOD+replica-3 shortcomings:
  - Each file on one disk, low throughput for serial workloads
  - Large number of bricks in the volume; problematic for some workloads
- JBOD+replica-3 expands the set of workloads that RHGS can handle well
  - Best for highly-concurrent, large-file read workloads

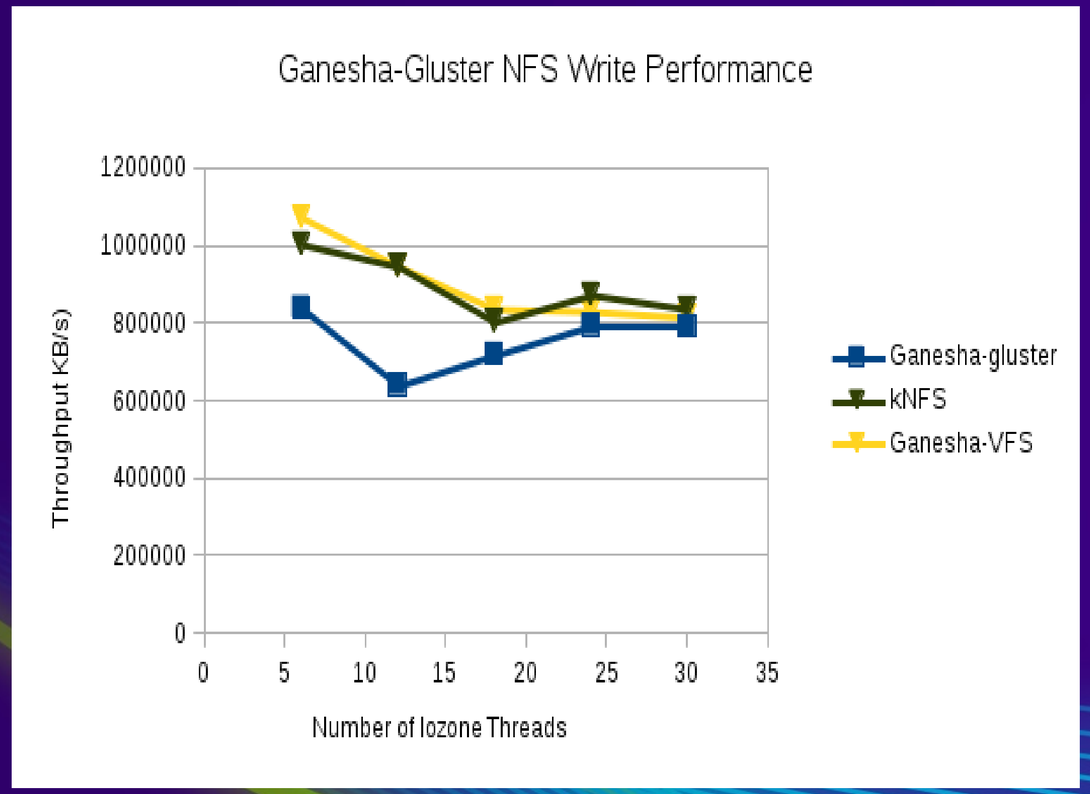
- JBOD+replica-3 outperforms RAID-6+replica-2 at higher thread counts
  - For large-file workload

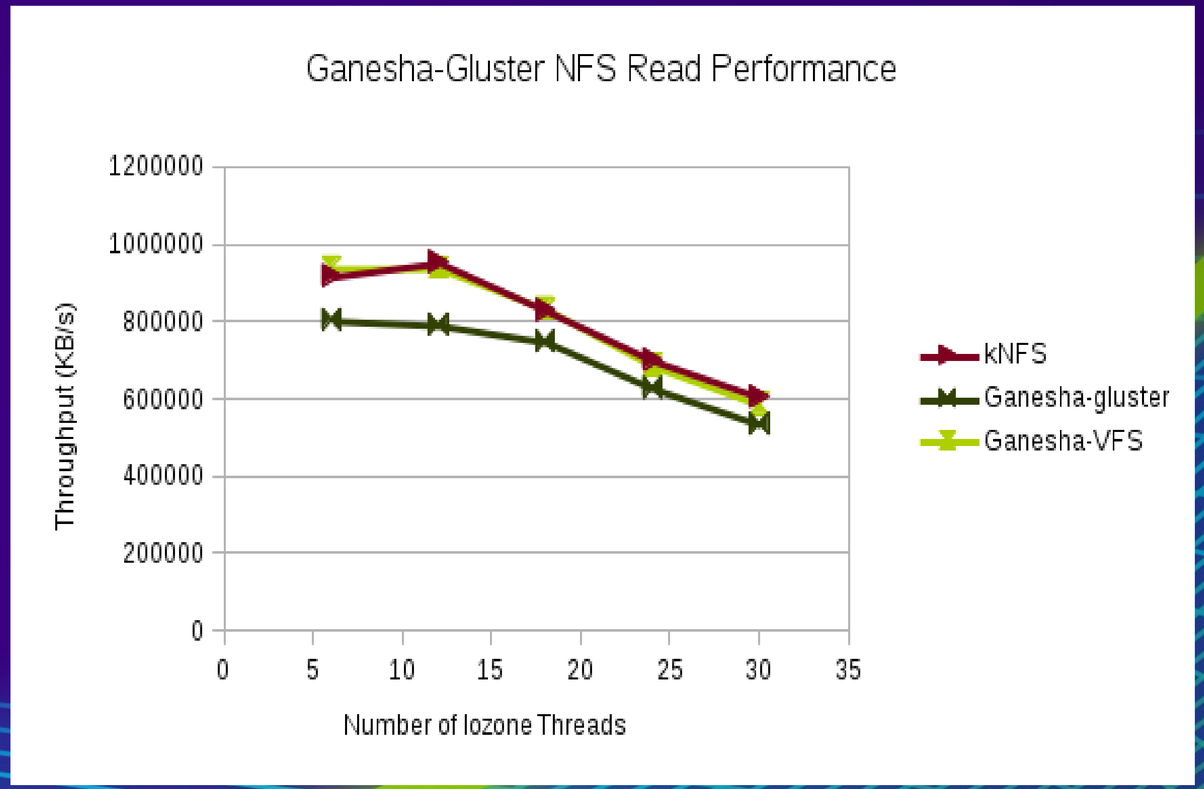


NFS-Ganesha

- NFSv3 access to RHGS volumes supported so far with gluster native NFS server
- NFS-Ganesha integration with FSAL-gluster expands supported access protocols
  - NFSv3 – has been in Technology Preview
  - NFSv4, NFSv4.1, pNFS
- Access path uses libgfapi, avoids FUSE

Ganesha-Gluster vs Ganesha-VFS vs Kernel-NFS





Snapshots

- Based on device-mapper thin-provisioned snapshots
  - Simplified space management for snapshots
  - Allow large number of snapshots without performance degradation
- Required change from traditional LV to thin LV for RHGS brick storage
  - Performance impact? Typically 10-15% for large file sequential read as a result of fragmentation
- Snapshot performance impact
  - Mainly due to writes to “shared” blocks, copy-on-write triggered on first write to a region after snapshot
  - Independent of number of snapshots in existence

Improved rebalancing

- Rebalancing lets you add/remove hardware from an online Gluster volume
- Important for scalability, redeployment of hardware resources
- Existing algorithm had shortcomings
  - Did not work well for small files
  - Was not parallel enough
  - No throttle
- New algorithm solves these problems
  - Executes in parallel on all bricks
  - Gives you control over number of concurrent I/O requests/brick

Best practices for sizing, install, administration

Configurations to avoid with Gluster (today)

- Super-large RAID volumes (e.g. RAID60)
- – example: RAID60 with 2 striped RAID6 12-disk components
- – Single glusterfsd process serving a large number of disks
  – recommend separate RAID LUNs instead
- JBOD configuration with very large server count
- – Gluster directories are still spread across every brick
- – with JBOD, that means every disk!
- – 64 servers x 36 disks/server = ~2300 bricks
- – recommendation: use RAID6 bricks of 12 disks each
- – even then, 64x3 = 192 bricks, still not ideal for anything but large files

Test methodology

- How well does RHGS work for your use-case?
- Some benchmarking tools:
  - Use tools with a distributed mode, so multiple clients can put load on servers
  - Iozone (large-file sequential workloads), smallfile benchmark, fio (better than iozone for random i/o testing.
- Beyond micro-benchmarking
  - SPECsfs2014 provides approximation to some real-life workloads
  - Being used internally
  - Requires license
- SPECsfs2014 provides mixed-workload generation in different flavors
    - VDA (video data acquisition), VDI (virtual desktop infrastructure), SWBUILD (software build)

Application filesystem usage patterns to avoid with Gluster

- Single-threaded application – one-file-at-a-time processing
- – uses only small fraction (1 DHT subvolume) of Gluster hardware

  Tiny files – cheap on local filesystems, expensive on distributed filesystems
- Small directories
- – creation/deletion/read/rename/metadata-change cost x brick count!
- – ***large file:directory ratio*** not bad as of glusterfs-3.7
- Using repeated directory scanning to synchronize processes on different clients
- – Gluster 3.6 (RHS 3.0.4) does not yet invalidate metadata cache on clients

Initial Data ingest

- Problem: applications often have previous data, must load Gluster volume
- Typical methods are excruciatingly slow (see lower right!)
- – Example: single mountpoint, rsync -ravu
- Solutions:
  - - for large files on glusterfs, use largest xfer size
  - - copy multiple subdirectories in parallel
  - - multiple mountpoints per client
  - - multiple clients
  - - mount option "gid-timeout=5"
  - - for glusterfs, increase client.event-threads to 8

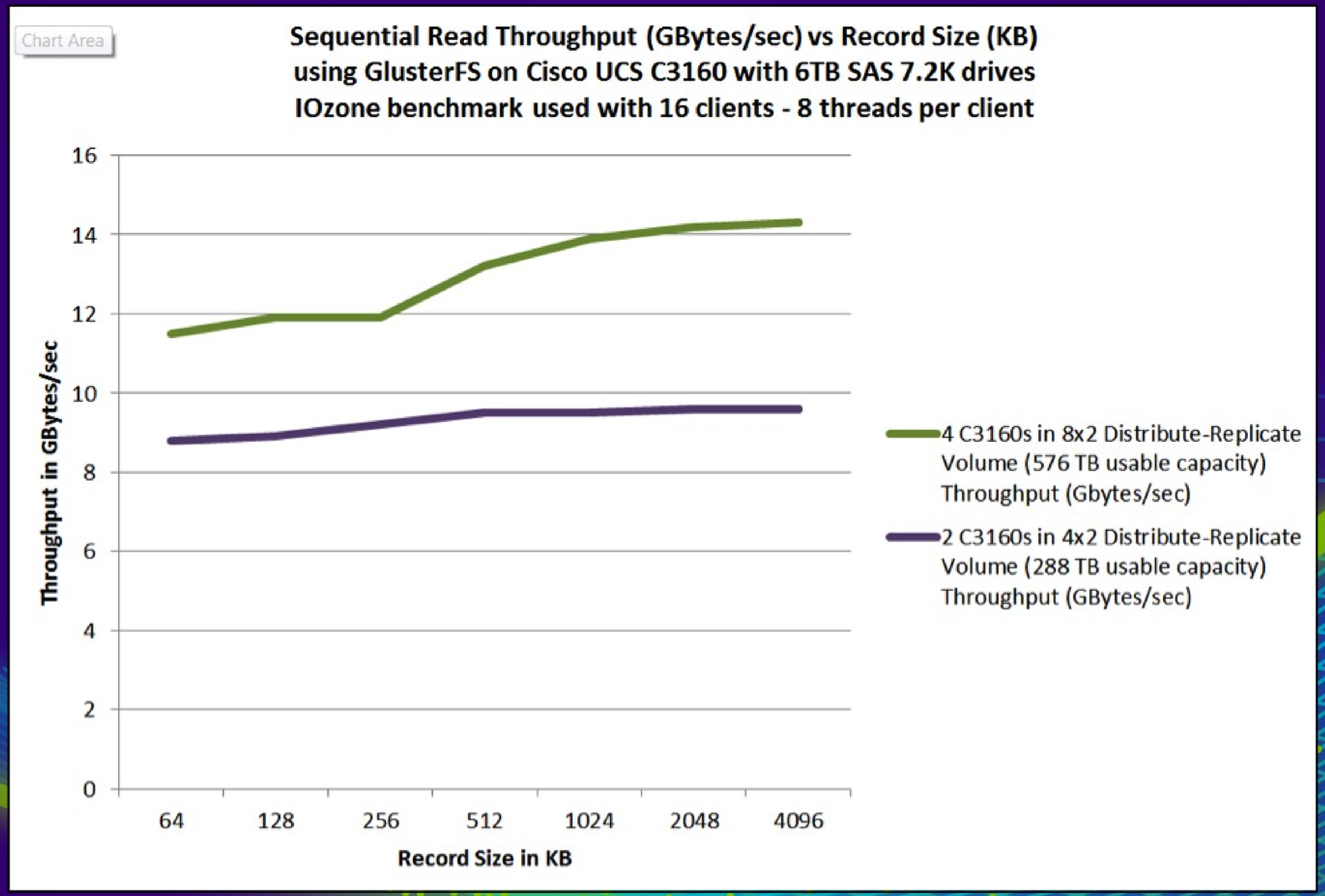
SSDs as bricks

- Avoid use of storage controller WB cache
- separate volume for SSD
- Check “top -H”, look for hot glusterfsd threads on server with SSDs
- Gluster tuning for SSDs: server.event-threads > 2
- SAS SSD:
  - Sequential I/O: relatively low sequential write transfer rate
  - Random I/O: avoids seek overhead, good IOPS
  - Scaling: more SAS slots => greater TB/host, high aggregate IOPS
- PCI:
  - Sequential I/O: much higher transfer rate since shorter data path
    Random I/O: lowest latency yields highest IOPS
  - Scaling: more expensive, aggregate IOPS limited by PCI slots

High-speed networking > 10 Gbps

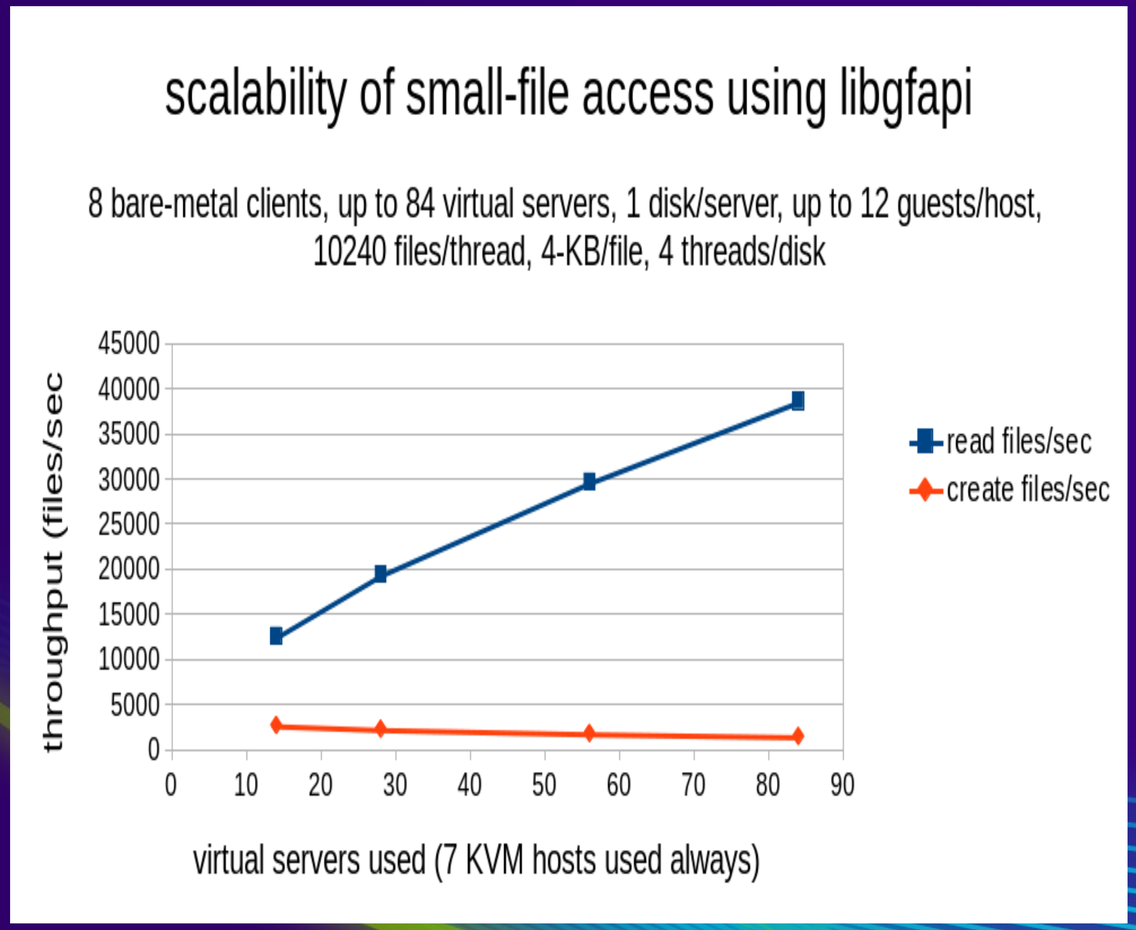
- Don't need RDMA for 10-Gbps network, better with >= 40 Gbps
- Infiniband alternative to RDMA – ipoib
- - Jumbo Frames (MTU=65520) – all switches must support
- - “connected mode”
- - TCP will get you to about ½ – ¾ 40-Gbps line speed
- 10-GbE bonding – see gluster.org how-to
- - default bonding mode 0 – don't use it
- - best modes are 2 (balance-xor), 4 (802.3ad), 6 (balance-alb)
- FUSE (glusterfs mountpoints) –
  - No 40-Gbps line speed from one mountpoint
  - Servers don't run FUSE => best with multiple clients/server
- NFS+SMB servers use libgfapi, no FUSE overhead

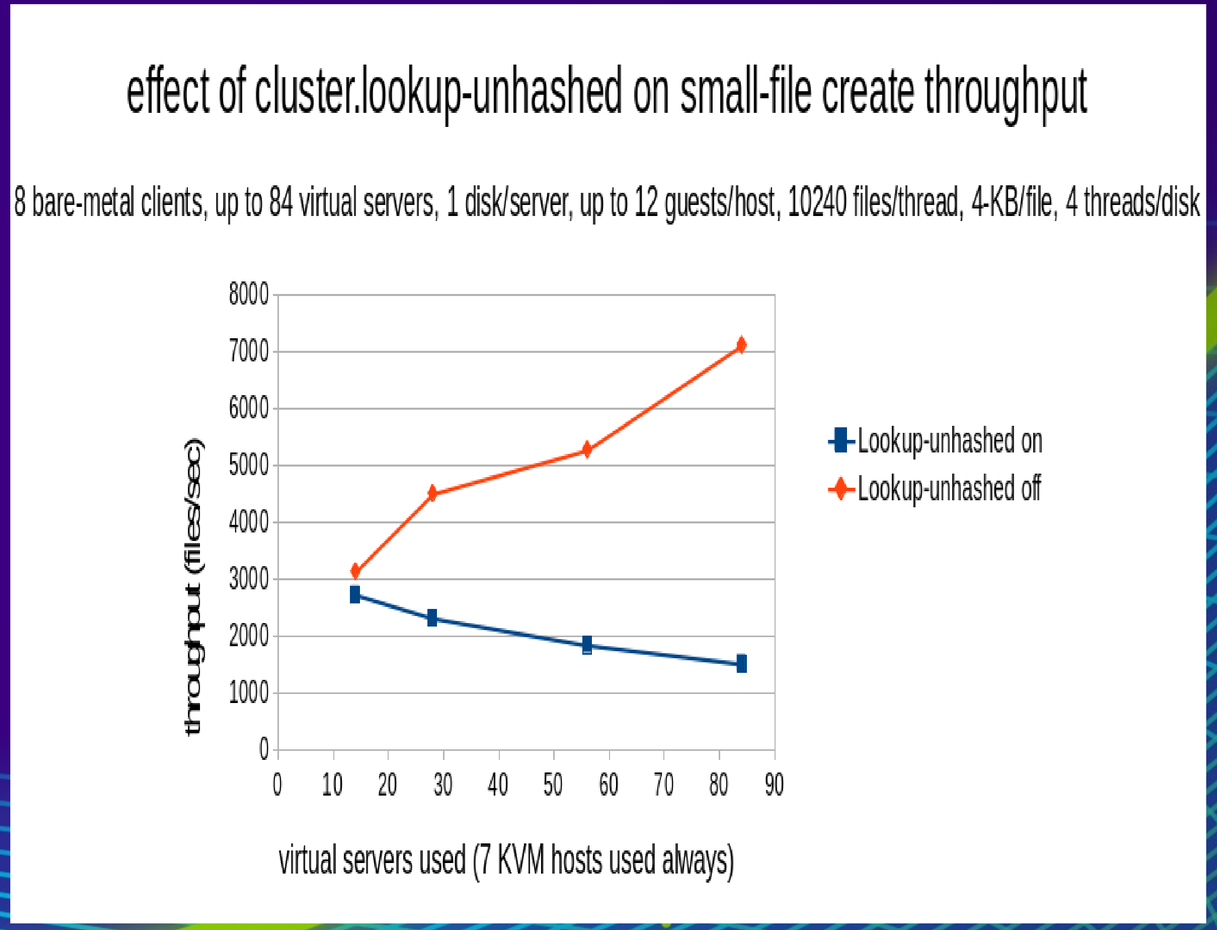
Networking – Putting it all together



Features coming soon
To a Gluster volume near you
(i.e. glusterfs-3.7 and later)

Lookup-unhashed fix



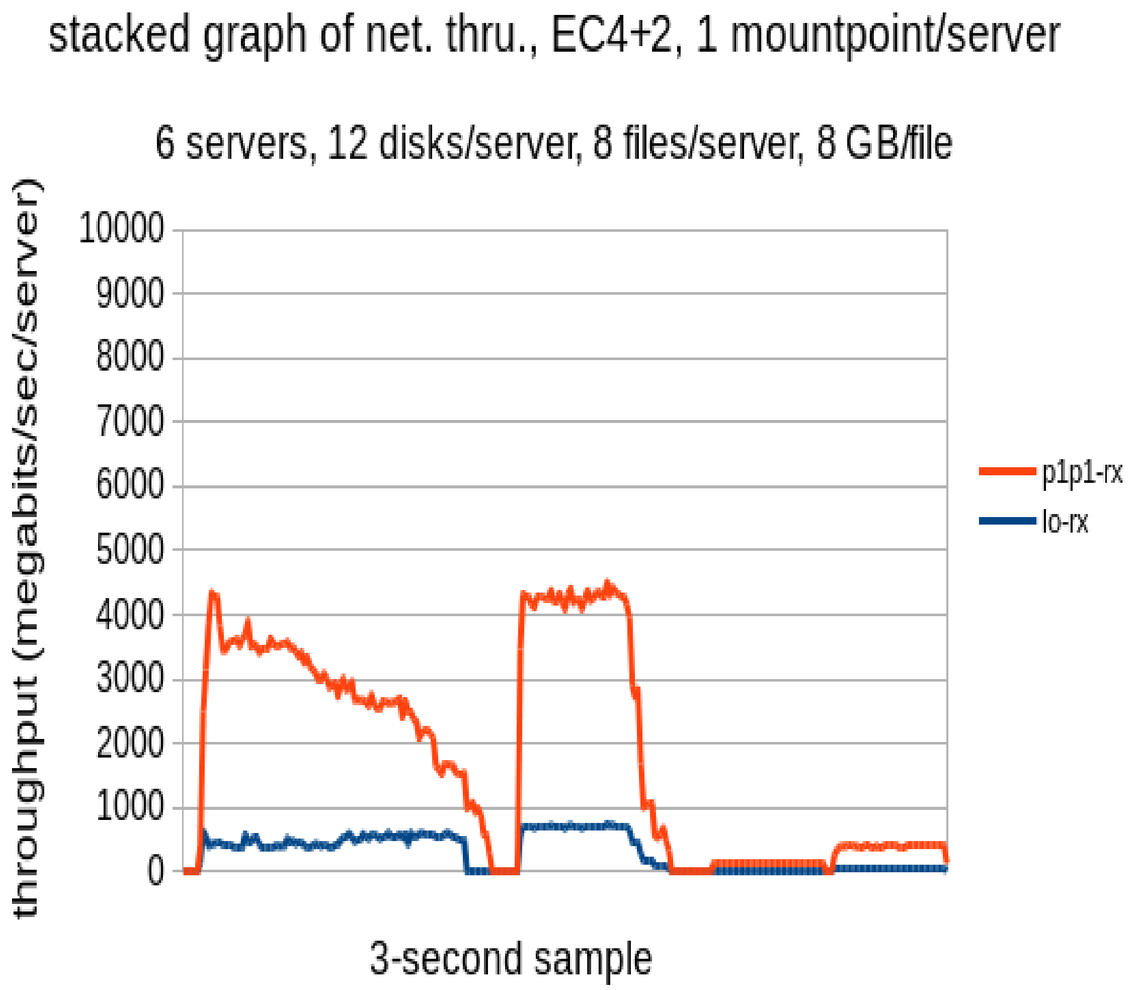


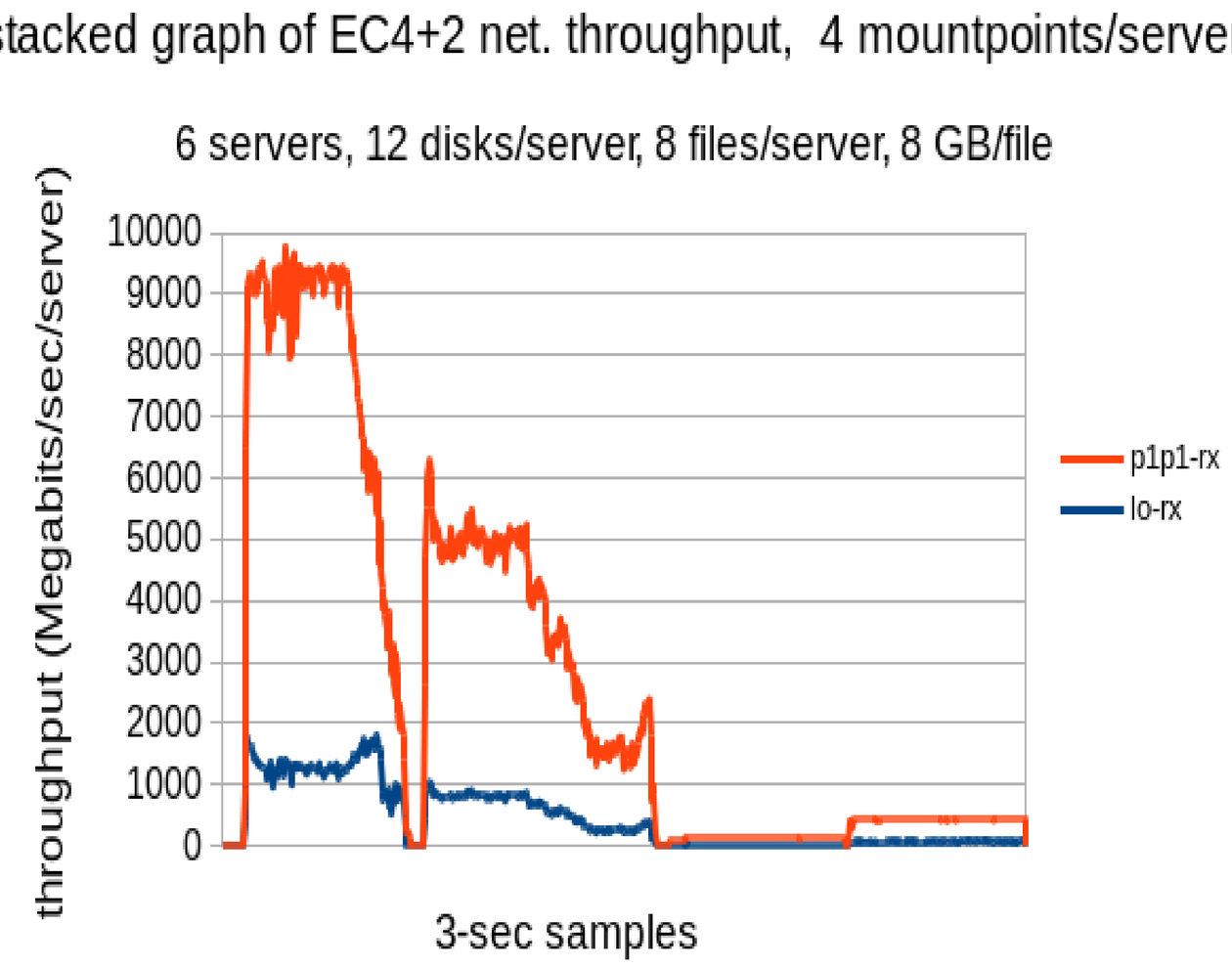
LEARN. NETWORK.
EXPERIENCE OPEN SOURCE.

Bitrot detection – in glusterfs-3.7 = RHS 3.1

- Provides greater durability for Gluster data (JBOD)
- Protects against silent loss of data
- Requires signature on replica recording original checksum
- Requires periodic scan to verify data still matches checksum
- Need more data on cost of the scan
- TBS – DIAGRAMS, ANY DATA?

A tale of two mountpoints - sequential write performance
And the result... drum roll....

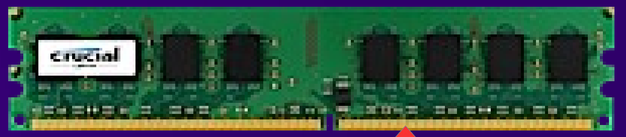


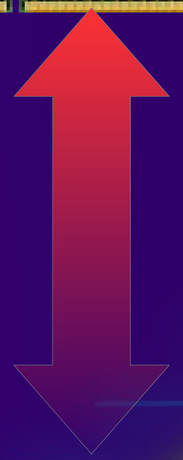


Balancing storage and networking performance

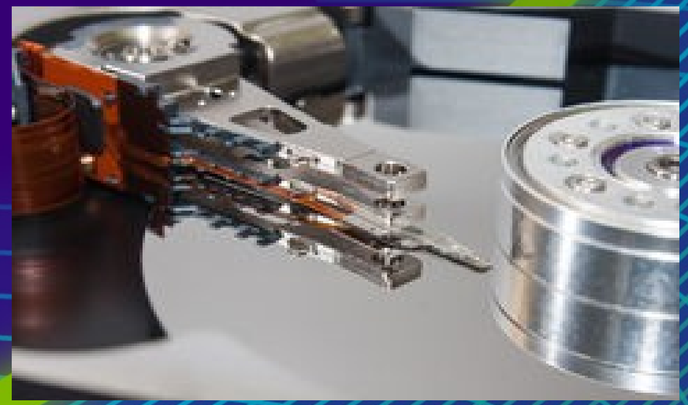
- Based on workload
  - Transactional or small-file workloads
    - don't need > 10 Gbps
    - Need lots of IOPS (e.g. SSD)
  - Large-file sequential workloads (e.g. video capture)
    - Don't need so many IOPS
    - Need network bandwidth
  - When in doubt, add more networking, cost < storage

Cache tiering





- Goal: performance of SSD with cost/TB of spinning rust
- Savings from Erasure Coding can pay for SSD!
- Definition: Gluster tiered volume consists of two subvolumes:
- - “hot” tier: sub-volume low capacity, high performance
- - “cold” tier: sub-volume – high capacity, low performance
- -- promotion policy: migrates data from cold tier to hot tier
- -- demotion policy: migrates data from hot tier to cold tier
- - new files are written to hot tier initially unless hot tier is full



perf enhancements
unless otherwise stated,
UNDER CONSIDERATION, NOT IMPLEMENTED

- Lookup-unhashed=auto in Glusterfs 3.7 today, in RHGS 3.1 soon
  - Eliminates LOOKUP per brick during file creation, etc.
- JBOD support – Glusterfs 4.0 – DHT V2 intended to eliminate spread of directories across all bricks
- Sharding – spread file across more bricks (like Ceph, HDFS)
- Erasure Coding – Intel instruction support, symmetric encoding, bigger chunk size
- Parallel utilities – examples are parallel-untar.py and parallel-rm-rf.py
- Better client-side caching – cache invalidation starting in glusterfs-3.7
- YOU CAN HELP DECIDE! Express interest and opinion on this