RED HAT SUMMIT

10 YEARS *and counting*

SAN FRANCISCO | APRIL 14-17, 2014

Red Hat Storage (RHS) Performance

Ben England

Principal S/W Engr., Red Hat

redhat.

What this session covers

What's new with Red Hat Storage (perf.) since summit 2013?

How to monitor RHS performance

Future directions for RHS performance.

Related presentations

- other RHS presentations in summit 2014
  - http://www.redhat.com/summit/sessions/topics/red-hat-storage-server.html
- previous RHS perf presentations in 2012, 2013
  - http://rhsummit.files.wordpress.com/2013/07/england_th_0450_rhs_perf_practices-4_neependra.pdf
  - http://rhsummit.files.wordpress.com/2012/03/england-rhs-performance.pdf

Improvement since last year

Libgfapi – removes FUSE bottleneck for high-IOPS applications

FUSE – linux caching can now be utilized with fopen-keep-cache feature

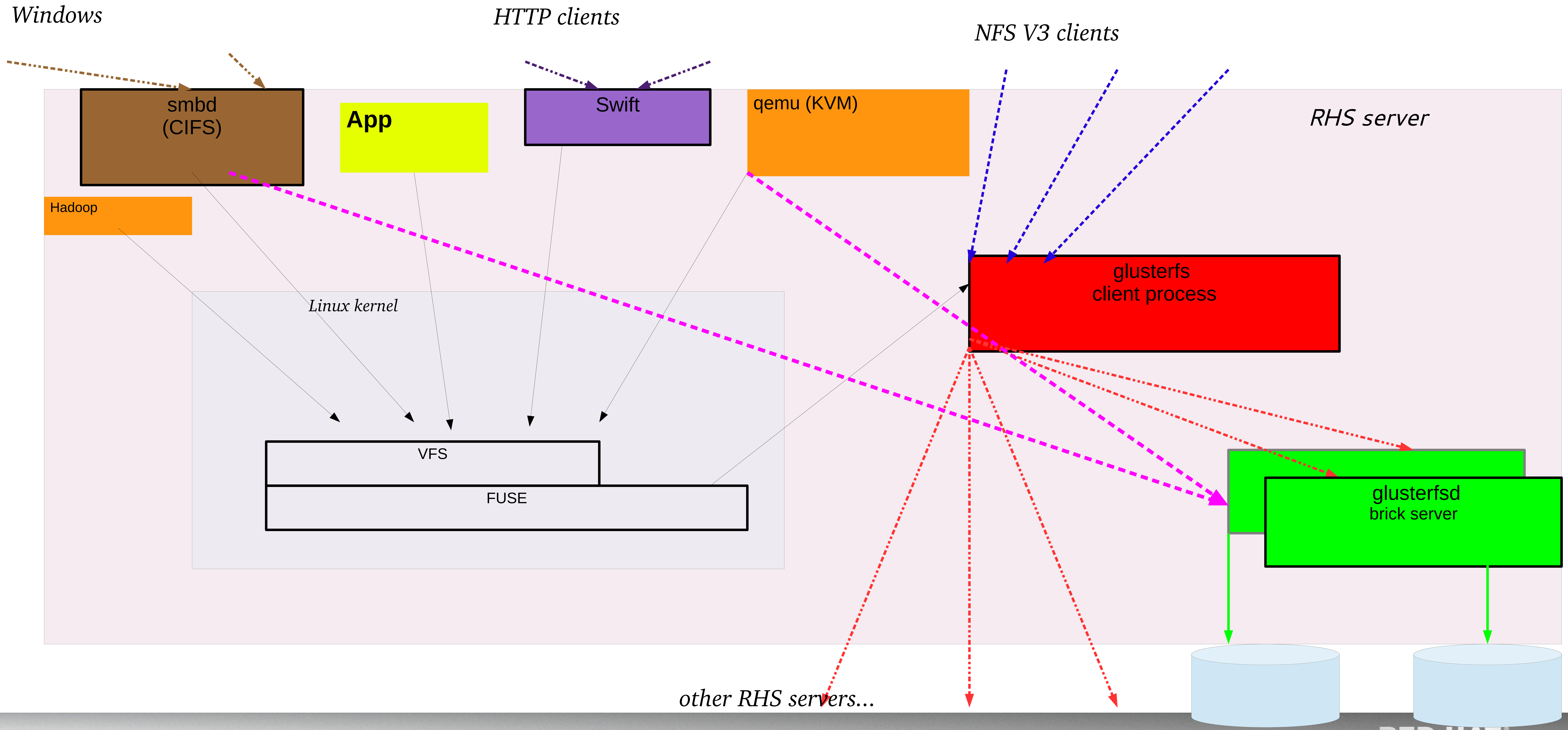
Brick configuration – alternatives for small-file, random-io-centric configurations

SSD – where it helps, configuration options

Swift - RHS converging with OpenStack IceHouse, is scalable and stable

NFS - large sequential writes more stable, 16x larger RPC size limit

Anatomy of Red Hat Storage (RHS) – libgfapi



Libgfapi development and significance

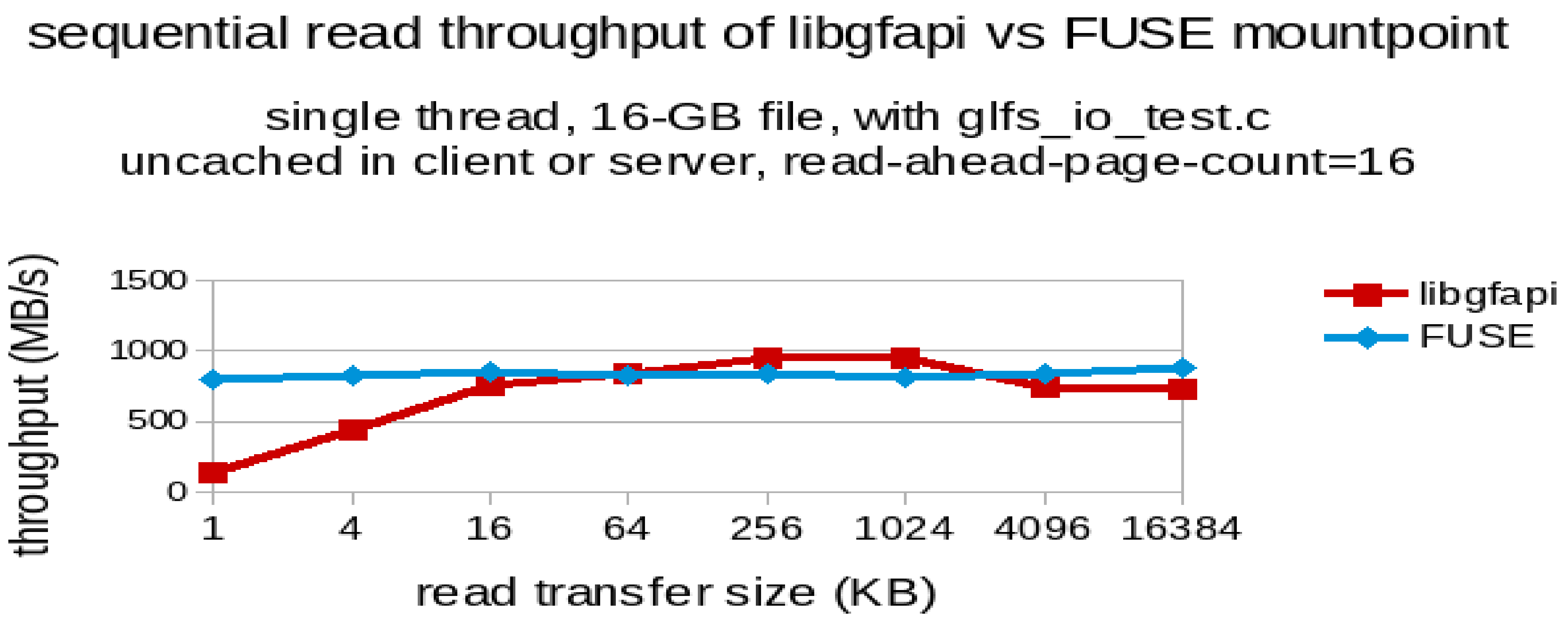
- Python and java bindings
- Integration with libvirt, SMB
  - Gluster performs better than Ceph for Cinder volumes (Principled Tech.)
- Lowered context switching overhead
- A distributed libgfapi benchmark available at:
  - https://github.com/bengland2/parallel-libgfapi
- Fio benchmark has libgfapi “engine”
  - https://github.com/rootfs/fio/blob/master/engines/glusterfs.c

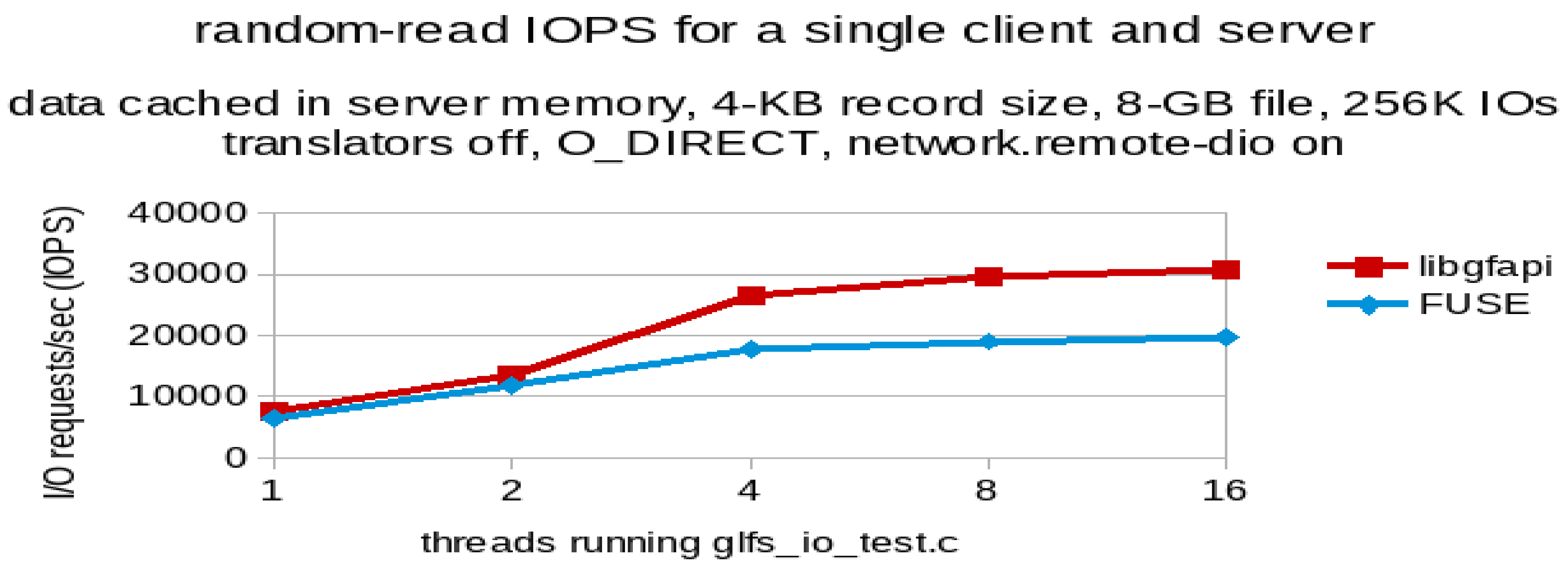
Libgfapi tips

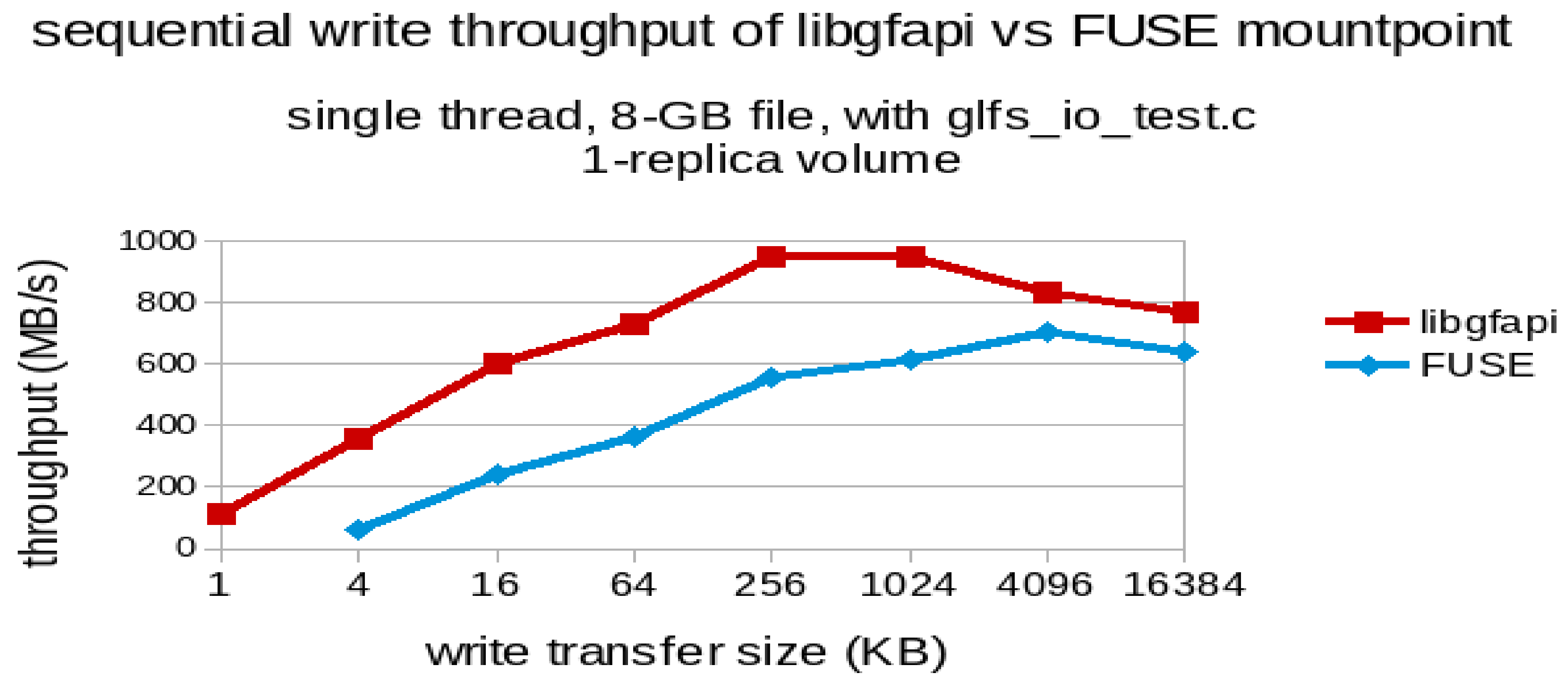
- For each Gluster volume:
  - gluster volume set your-volume allow-insecure on
  - Gluster volume set stat-prefetch on (if it is set to “off”)
- For each Gluster server:
  - Add “option rpc-auth-allow-insecure on” to /etc/glusterfs/glusterd.vol
  - Restart glusterd
- Watch TCP port consumption
  - On client: TCP ports = bricks/volume x libgfapi instances
  - On server: TCP ports = bricks/server x clients' libgfapi instances
  - Control number of bricks/server

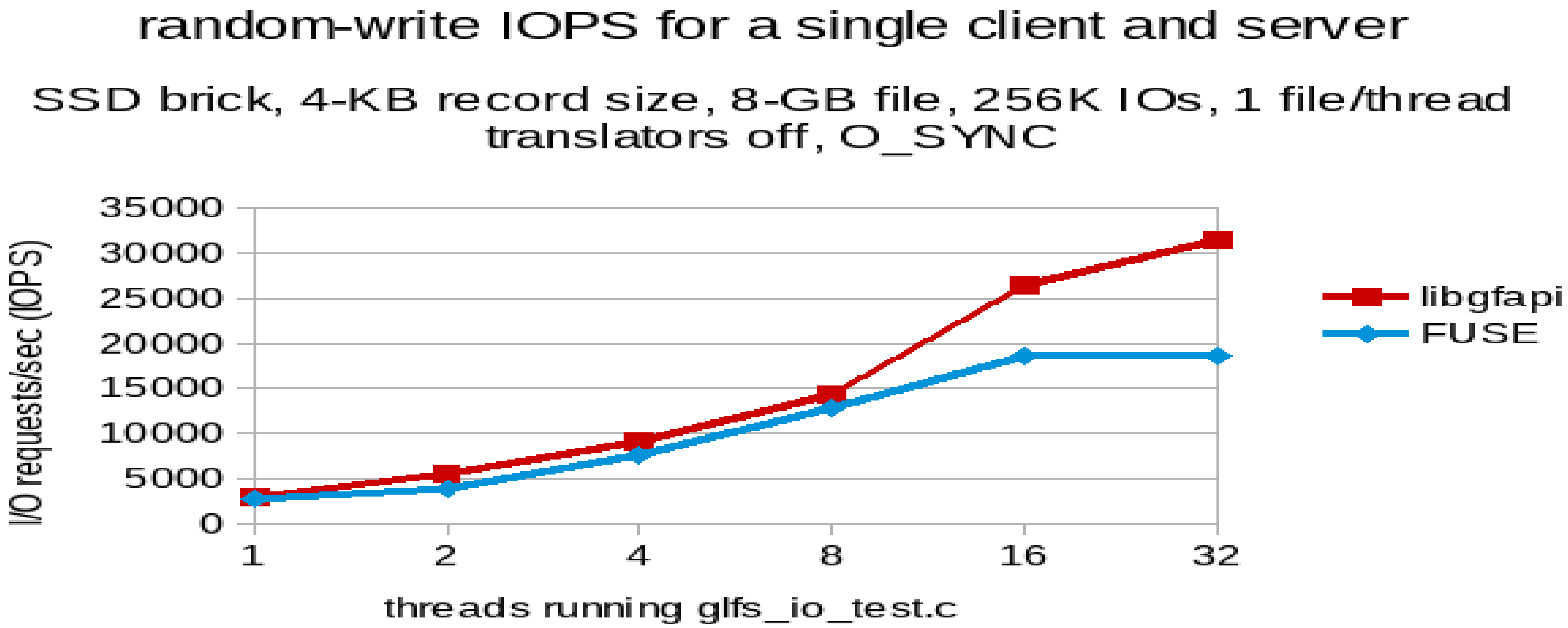
libgfapi - performance

- for single client & server with 40-Gbps IB connection, Nytro SSD caching

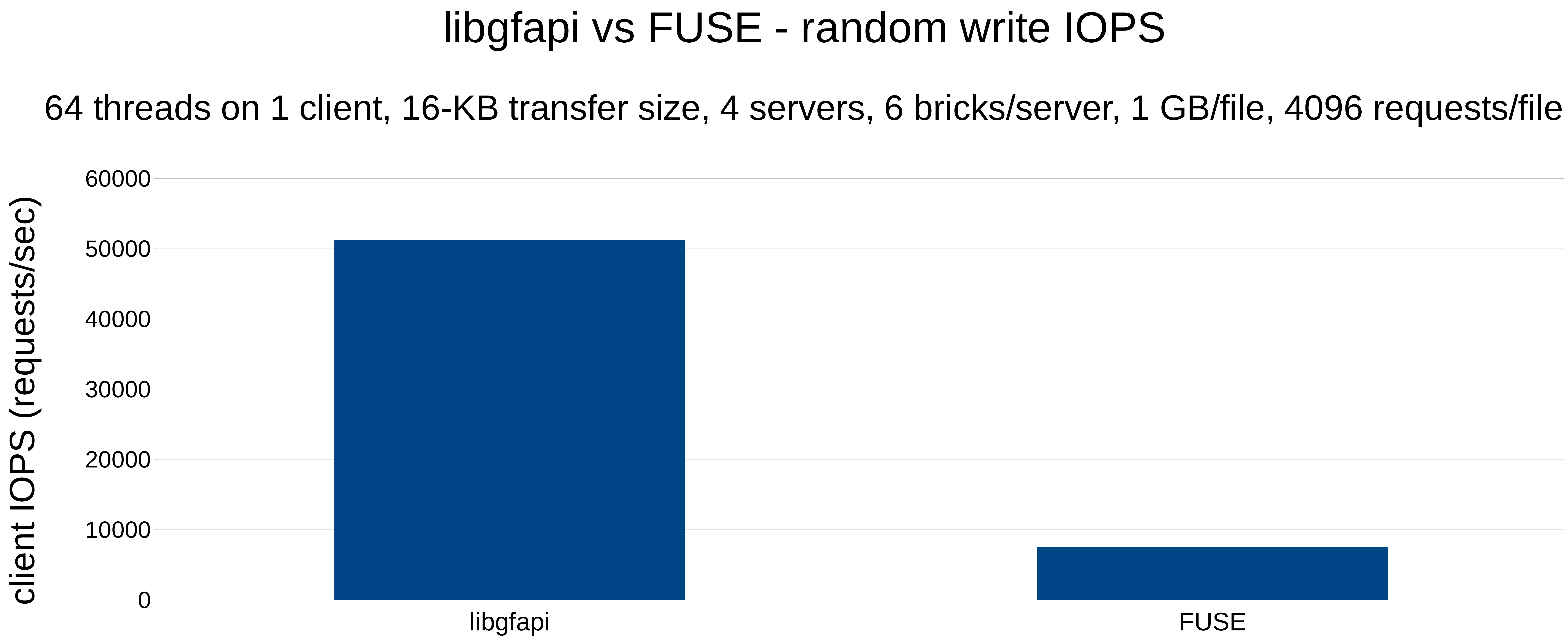








Libgfapi gets rid of client-side bottleneck

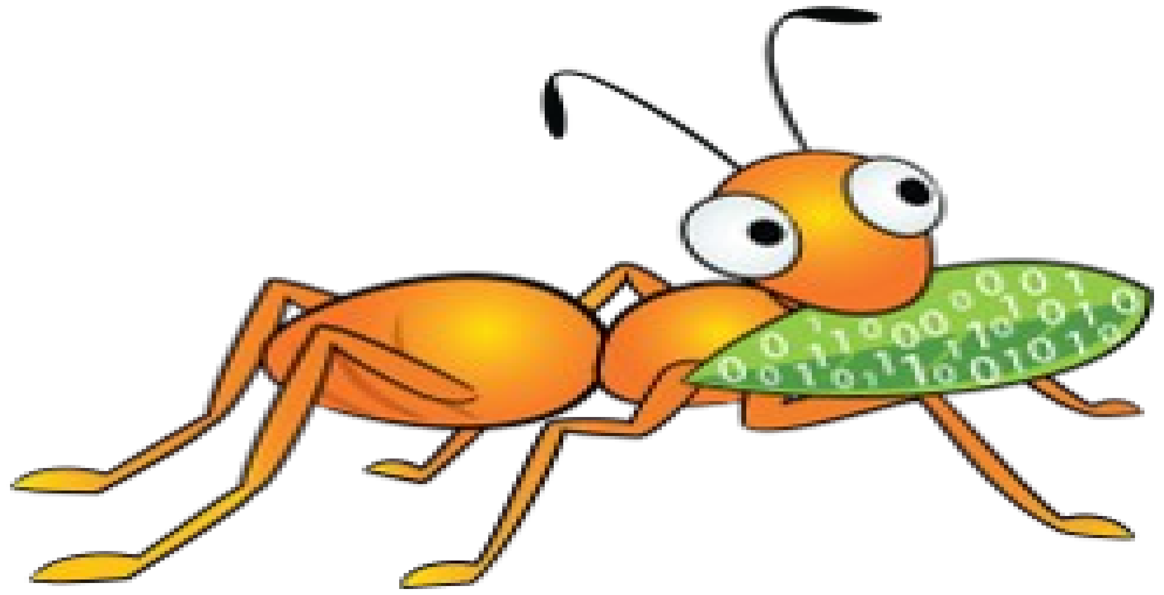


Libgfapi and small files

- 4 servers, 10-GbE jumbo frames, 2-socket westmere, 48 GB RAM, 12 7200-RPM drives, 6 RAID1 LUNs, RHS 2.1U2
- 1024 libgfapi processes spread across 4 servers
- Each process writes/reads 5000 1-MB files (1000 files/dir)
- Aggregate write rate: 155 MB/s/server
- Aggregate read rate: 1500 MB/s/server, 3000 disk IOPS/server
- On reads, disks are bottleneck!

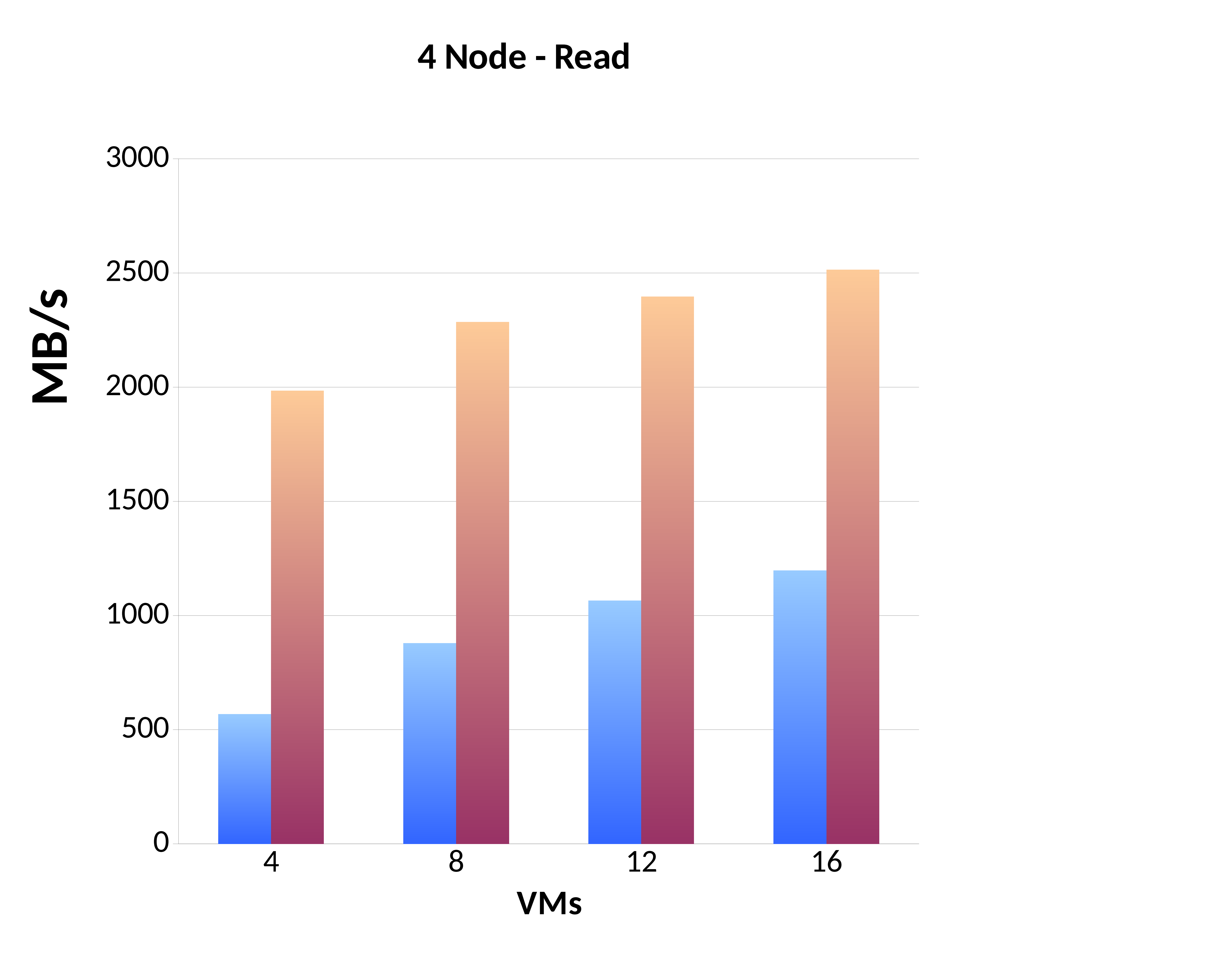
Libgfapi vs Ceph for Openstack Cinder

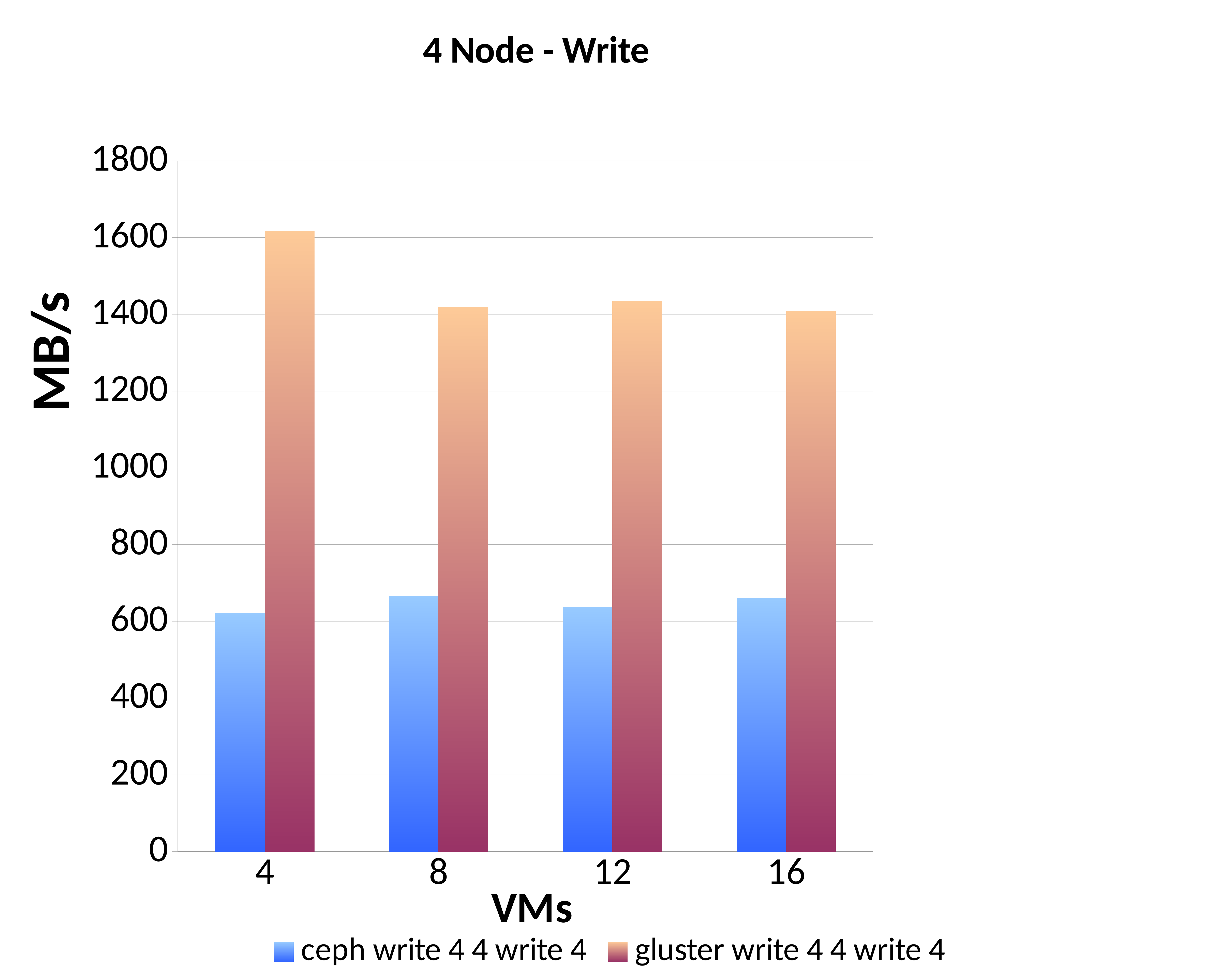
- Principled Technologies compared for Havana RDO Cinder
- Results show Gluster outperformed Ceph significantly in some cases
  - Sequential large-file I/O
  - Small-file I/O (also sequential)



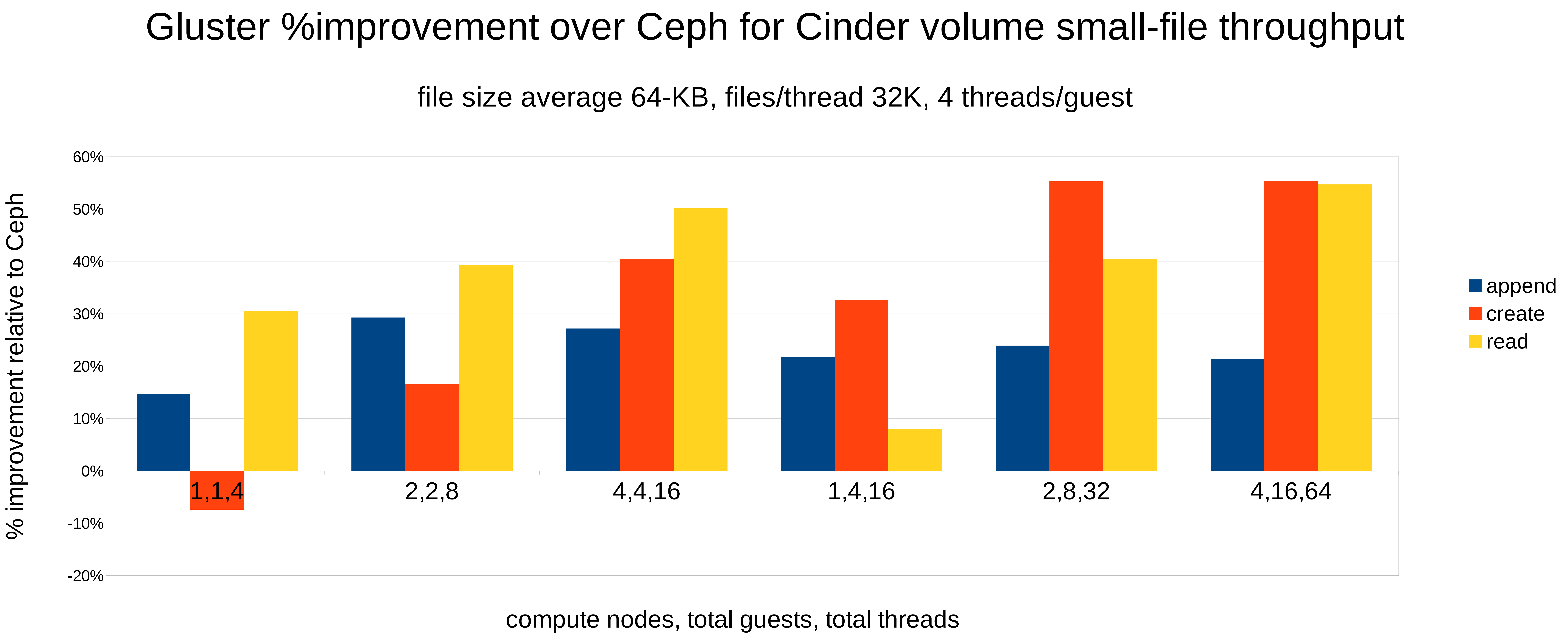
- Ceph outperformed Gluster for pure-random-I/O case, why?
  - RAID6 appears to be the root cause, saw high block device avg wait

Gluster libgfapi vs Ceph for OpenStack Cinder – large files example: 4 servers, 4-16 guests, 4 threads/guest, 64-KB file size



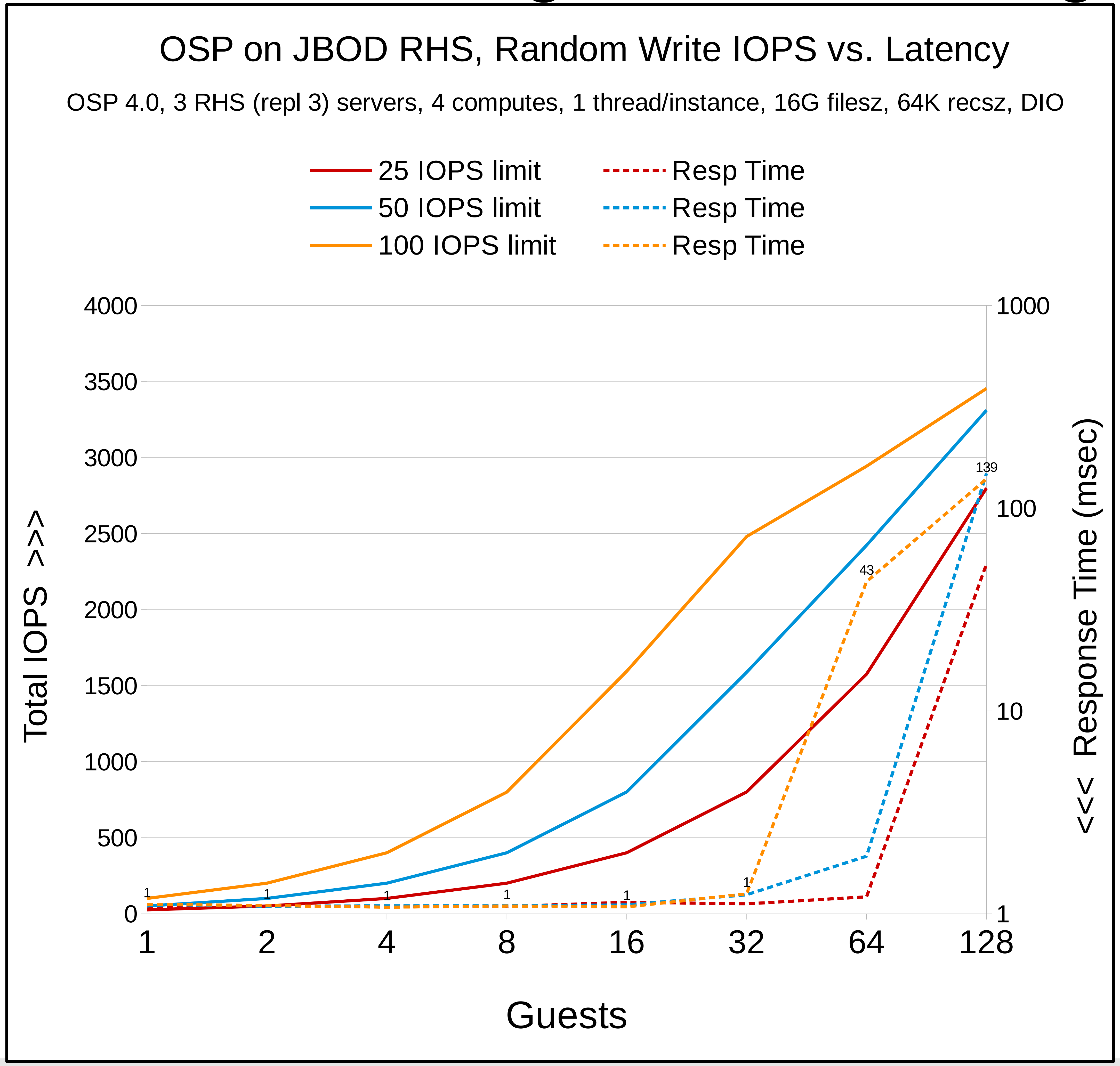


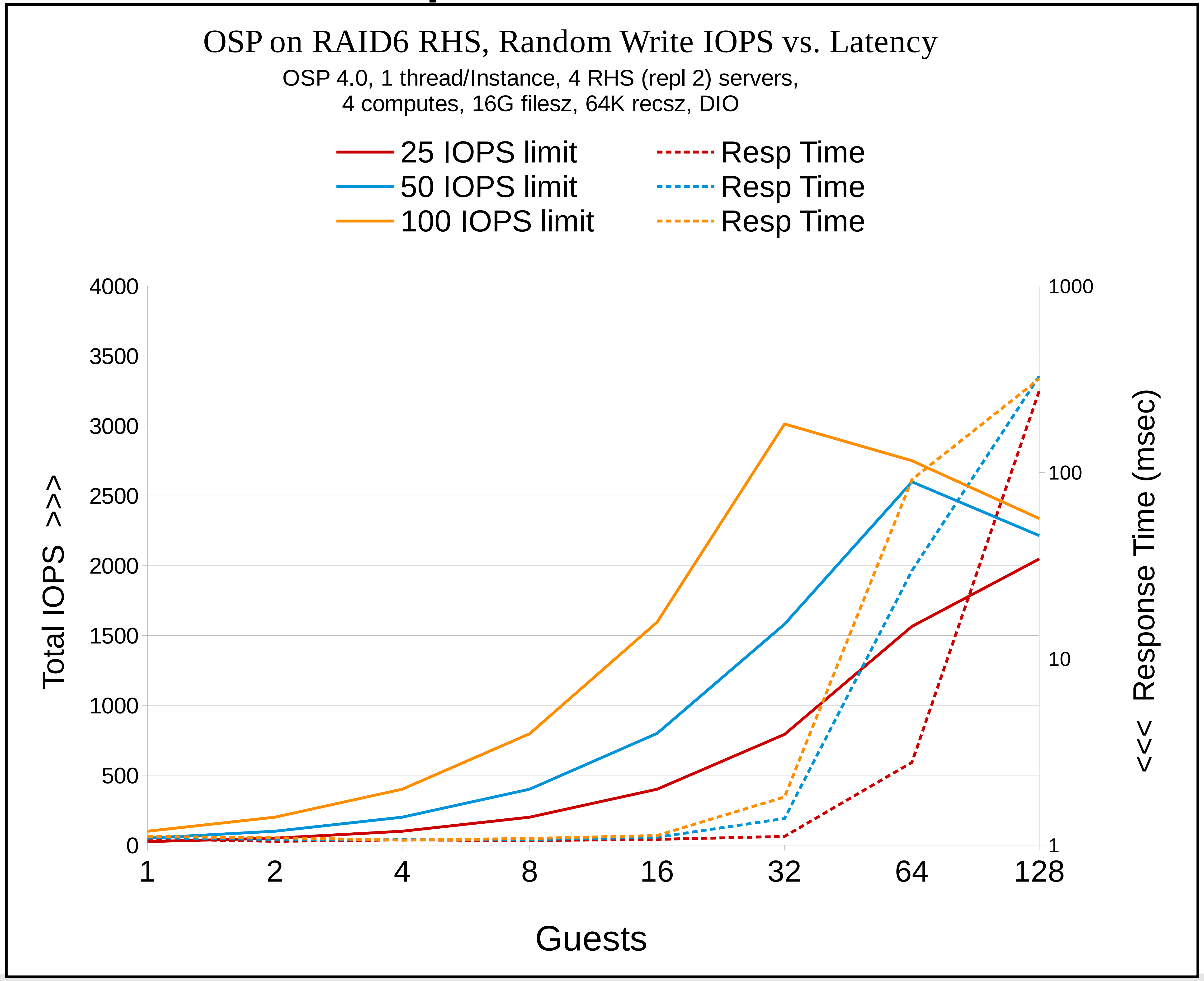
Libgfapi with OpenStack Cinder – small files



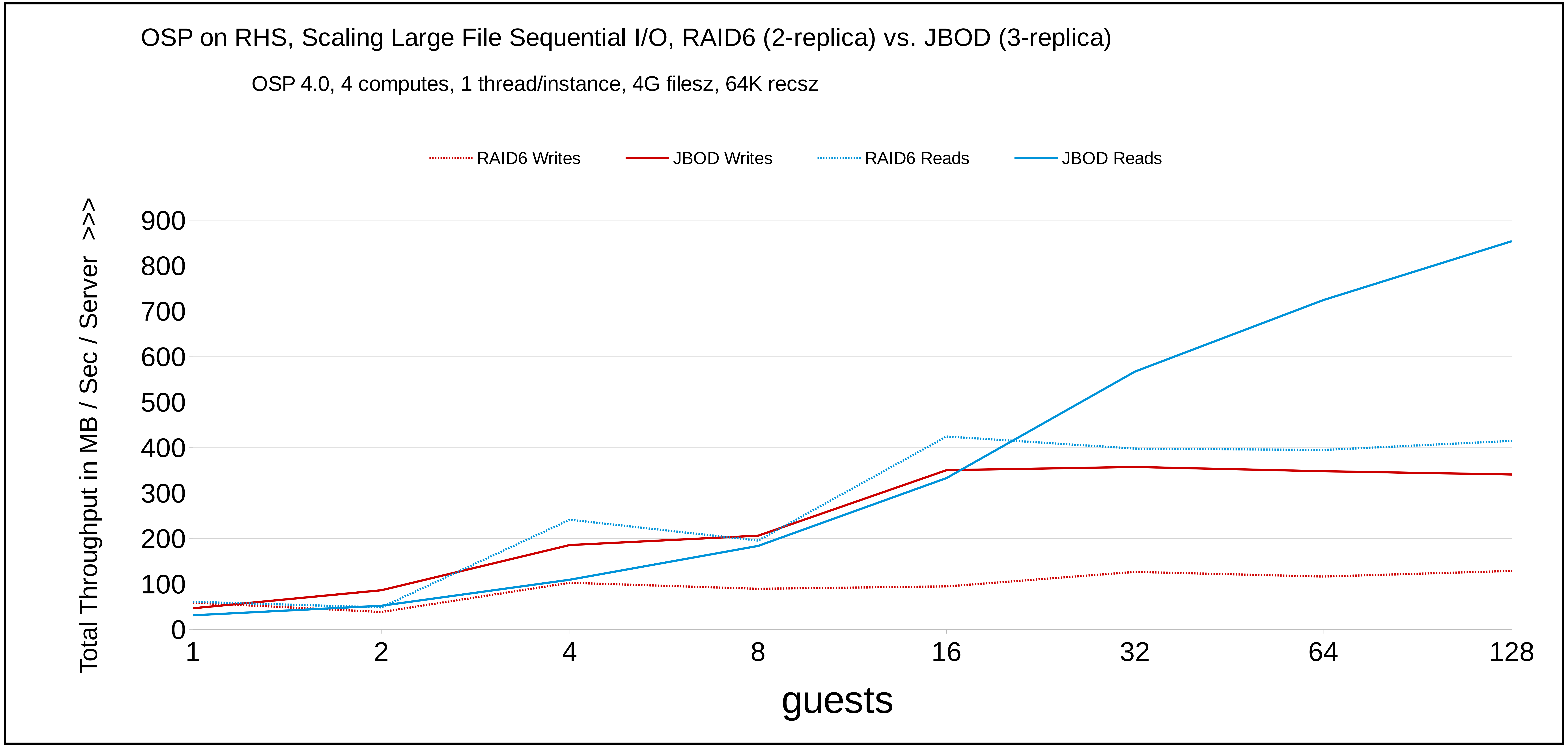
Random I/O on RHEL OSP with RHS Cinder volumes

- Rate-limited guests for sizing, measured response time



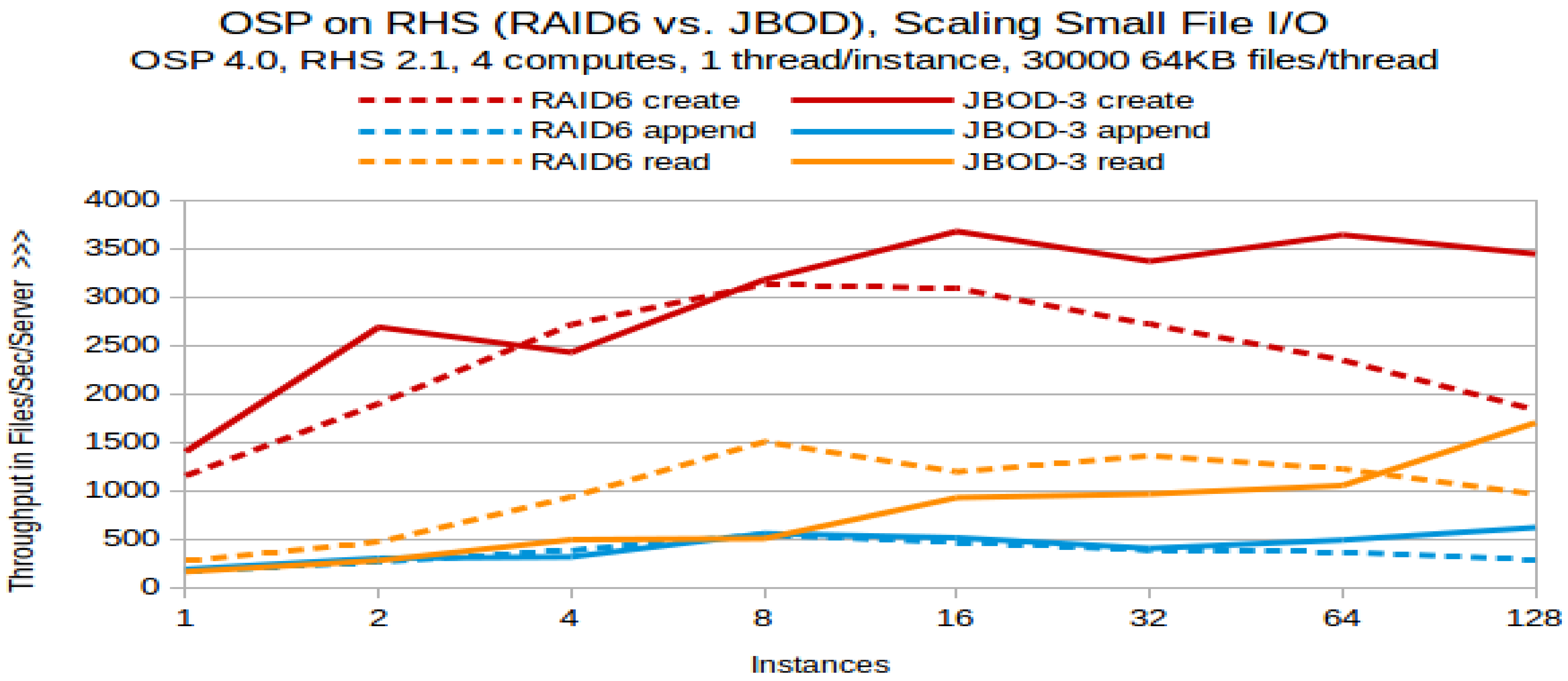


RHS libgfapi: Sequential I/O to Cinder volumes



Cinder volumes on libgfapi: small files

- Increase in concurrency on RAID6 decreases system throughput



Brick configuration alternatives – metrics

How do you evaluate optimality of storage with both Gluster replication and RAID?

- Disks/LUN –
  - more bricks = more CPU power
- Physical:usable TB – what fraction of physical space is available?
- Disk loss tolerance – how many disks can you lose without data loss?
- Physical:logical write IOPS –
  - how many physical disk IOPS per user IOP? Both Gluster replication and RAID affect this

Brick configuration – how do alternatives look?

Small-file workload, 8 clients, 4 threads/client, 100K files/thread, 64 KB/file
2 servers in all cases except in *JBOD 3way* case

RAID type	disks/LUN	physical:usable TB ratio	loss tolerance (disks)	Logical:physical random IO ratio	create Files/sec	read Files/sec
RAID6	12	2.4	4	12	234	676
JBOD	1	2.0	1	2	975	874
RAID1	2	4.0	2	4	670	773
RAID10	12	4.0	2	4	308	631
JBOD 3way	1	3.0	2	3	500	393?

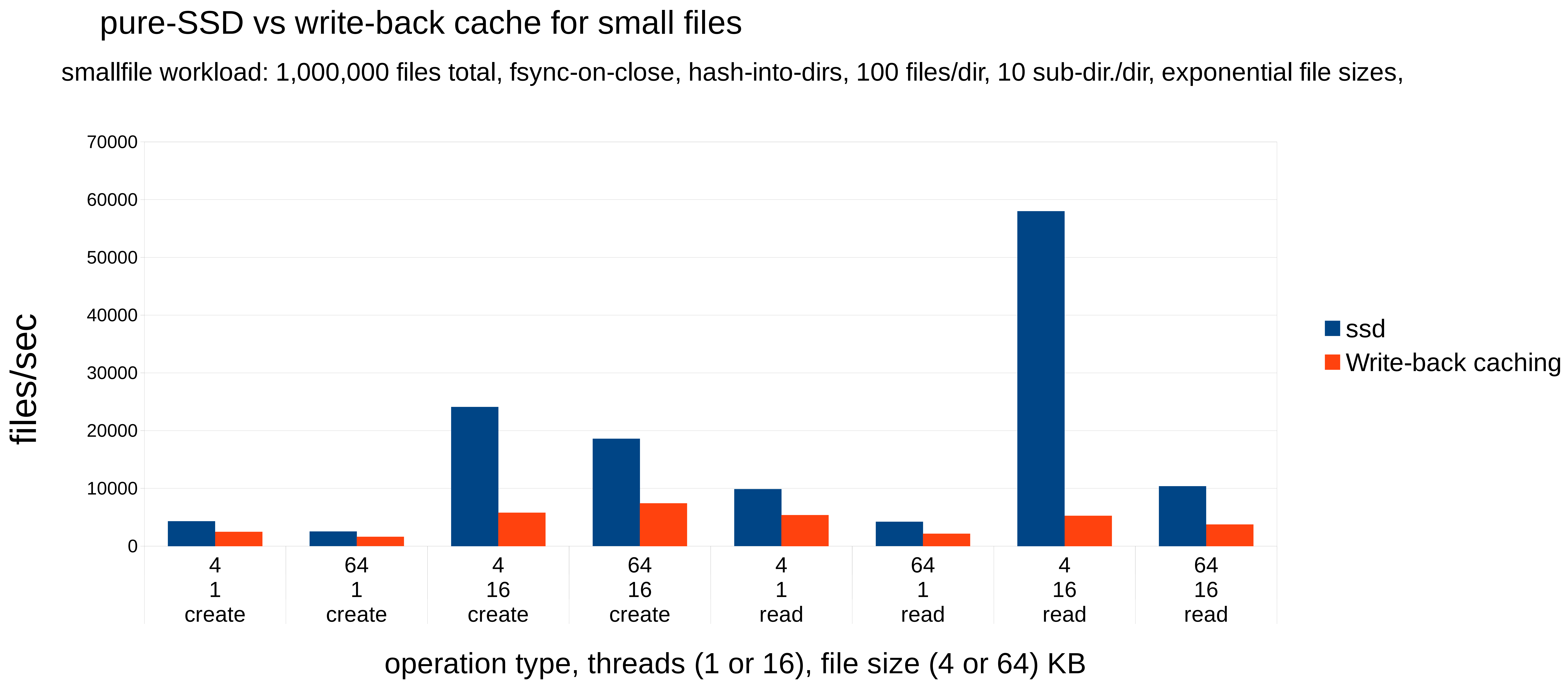
Brick config. – relevance of SSDs

- Where SSDs don't matter:
  - Sequential large-file read/write – RAID6 drive near 1 GB/sec throughput
  - Reads cacheable by Linux – RAM faster than SSD
- Where SSDs do matter:
  - Random read working set > RAM, < SSD
  - Random writes
  - Concurrent sequential file writes are slightly random at block device
  - Small-file workload random at block device
  - metadata

Brick configuration – 3 ways to utilize SSDs

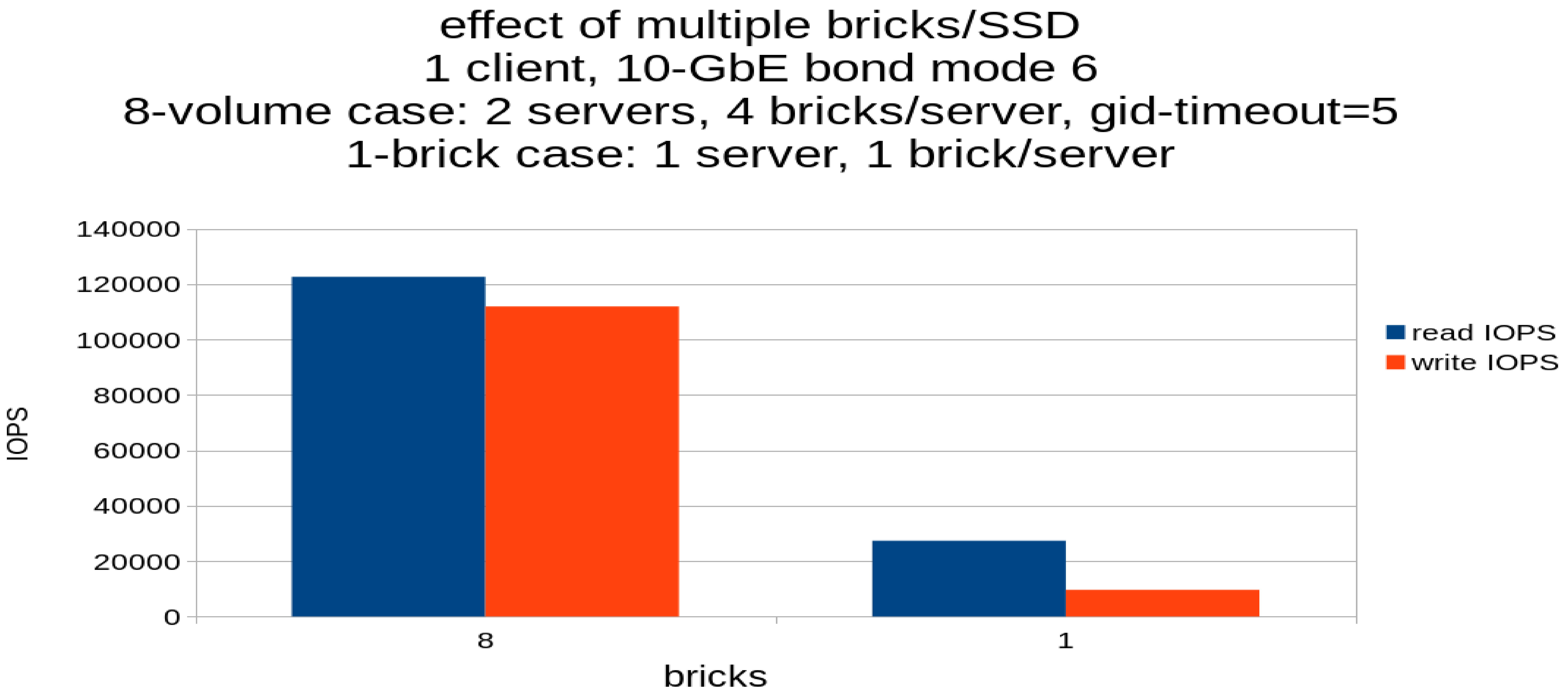
- SSDs as brick
  - Configuration guideline – multiple bricks/SSD
- RAID controller supplies SSD caching layer (example: LSI Nytro)
  - Transparent to RHEL, Gluster, but no control over policy
- Linux-based SSD caching layer – dm-cache (RHEL7)
  - Greater control over caching policy, etc.
  - Not supported on RHEL6 -> Not supported in Denali (RHS 3.0)

Example of SSD performance gain

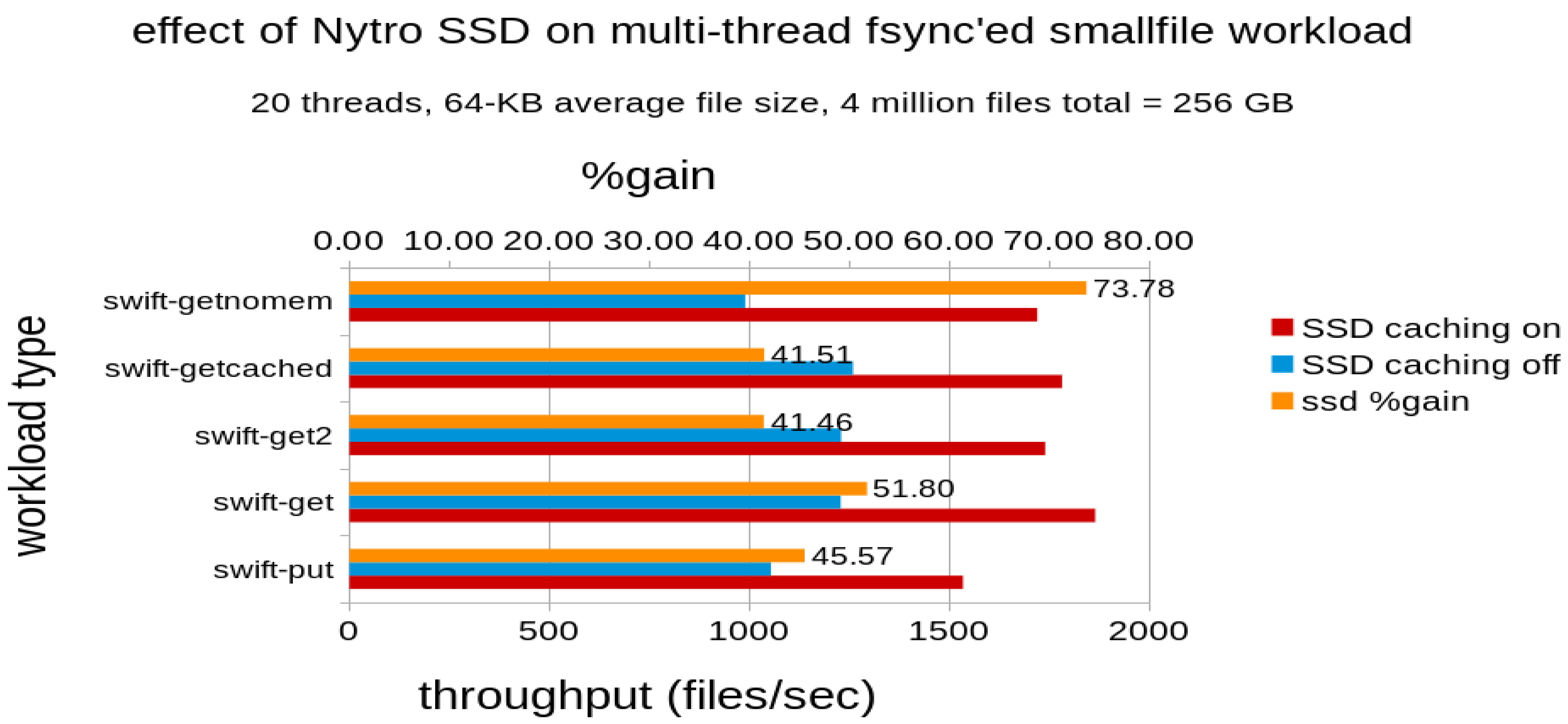


SSDs: Multiple bricks per SSD

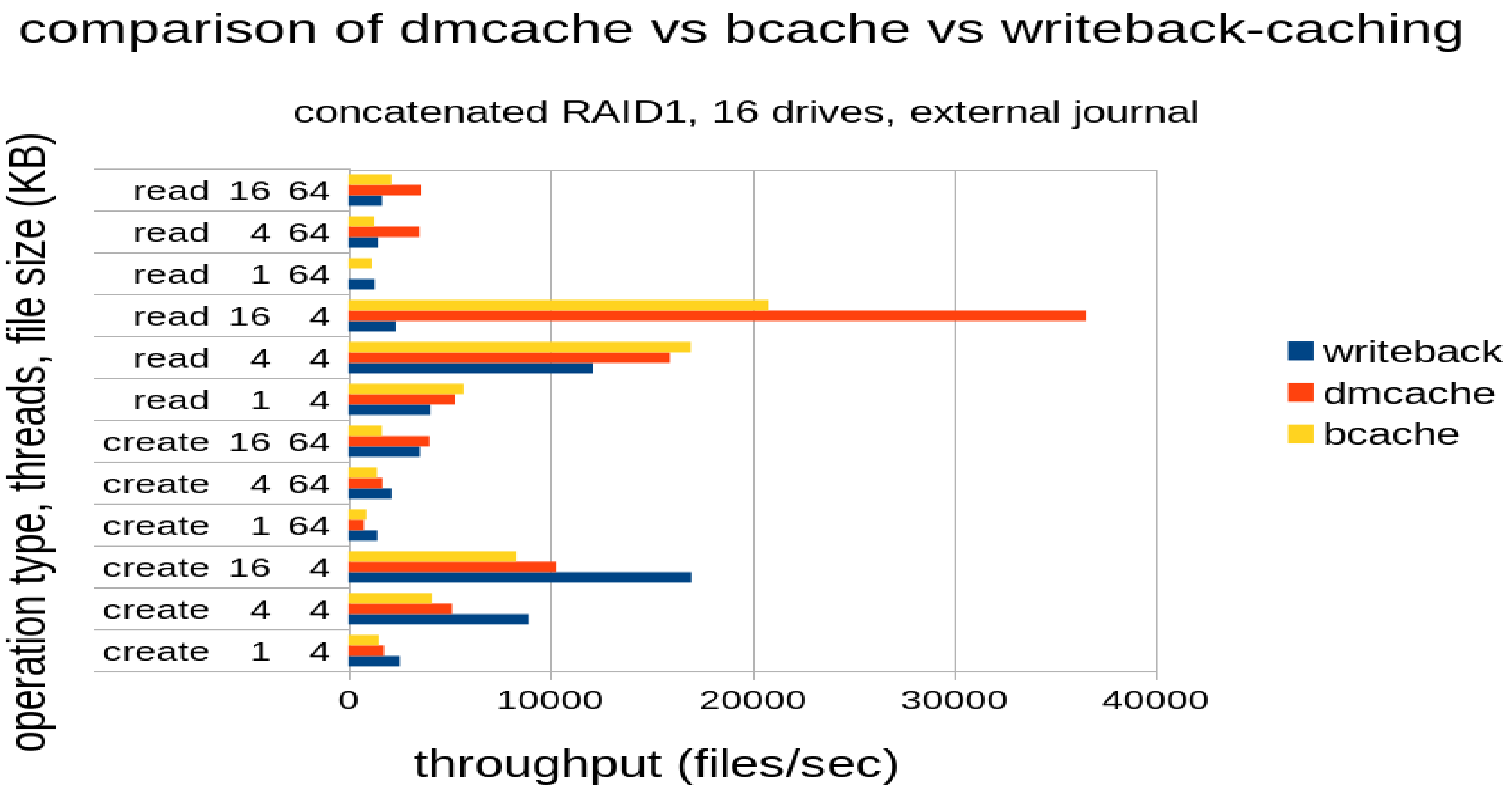
distribute CPU load across glusterfsd brick server processes
workload here: random I/O



SSD caching by LSI Nytro RAID controller



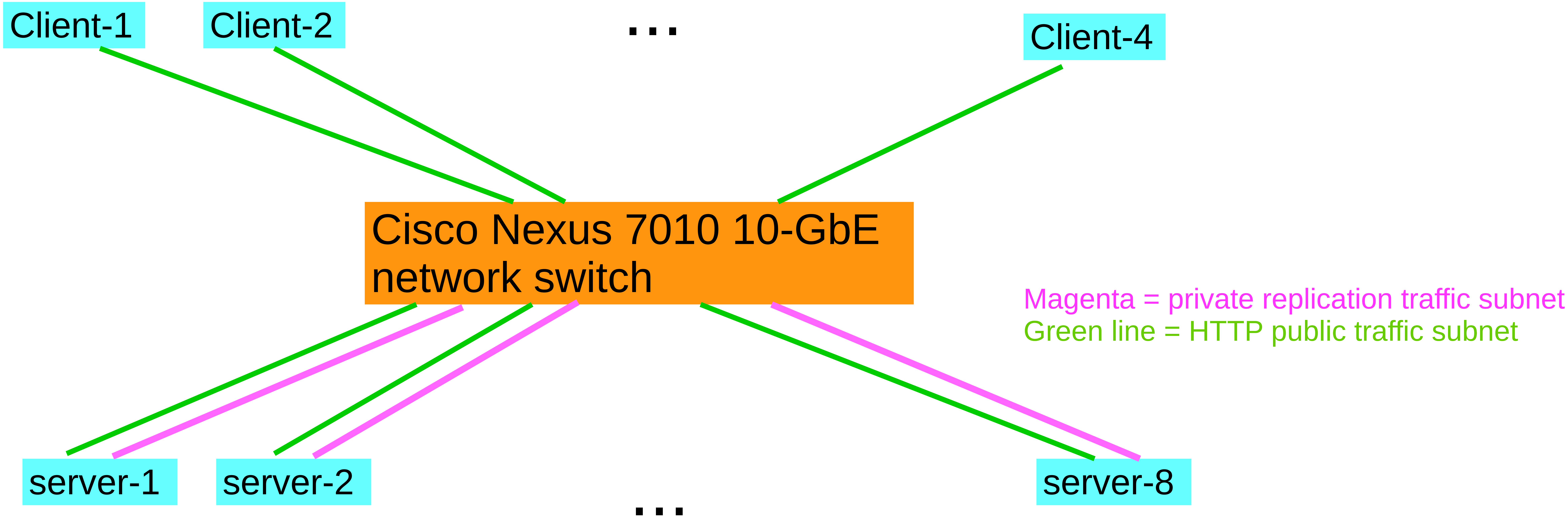
dm-cache in RHEL7 can help with small-file reads



SSD conclusions

- Can add SSD to Gluster volumes today in one of above ways
  - Use more SSD than server RAM to see benefits
  - Don't really need for sequential large-file I/O workloads
- Each of these 3 alternatives has weaknesses
  - SSD brick – very expensive, too many bricks required in Gluster volume
  - Firmware caching - very conservative in using SSD blocks
  - Dm-cache – RHEL7-only, not really writeback caching
- Could Gluster become SSD-aware?
  - Could reduce latency of metadata access such as change logs, .glusterfs, journal, etc.

Swift – example test configuration - TBS

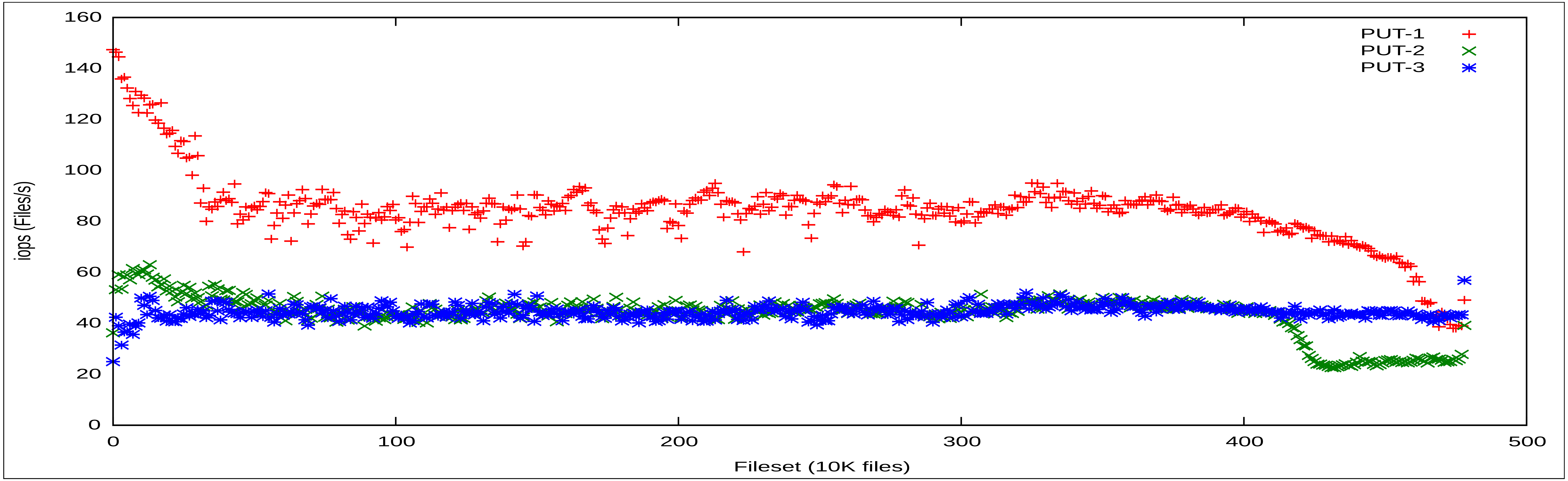


Swift-on-RHS: large objects

- No tuning except 10-GbE, 2 NICs, MTU=9000 (jumbo frames), rhs-high-throughput tuned profile
- 150-MB average object size, 4 clients, 8 servers
- writes: HTTP 1.6 GB/sec = 400 MB/sec/client, 30% net. util
- reads: HTTP 2.8 GB/s = 700 MB/s/client, 60% net. util.

Swift-on-RHS: small objects

- Catalyst workload: 28.68 million objects, PUT and GET
- 8 physical clients, 64 thread/client, 1 container/client, file size from 5-KB up to 30 MB, average 30 KB.



RHS performance monitoring

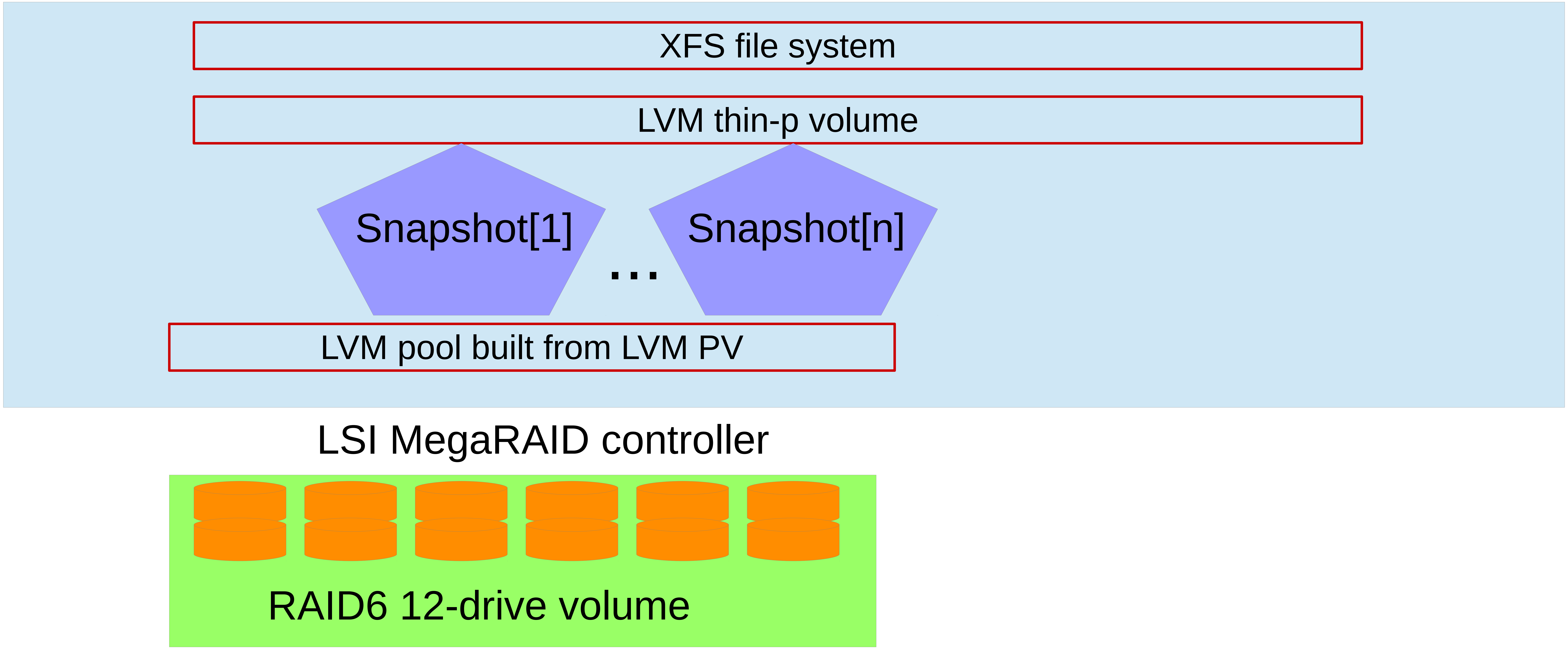
- Start with network
  - Are there hot spots on the servers or clients?
- Gluster volume profile your-volume [start | info]
  - Perf. Copilot plugin
- Gluster volume top your-volume [read | write | opendir | readdir]
- Perf-copilot demo

Recorded demo of RHS perf.

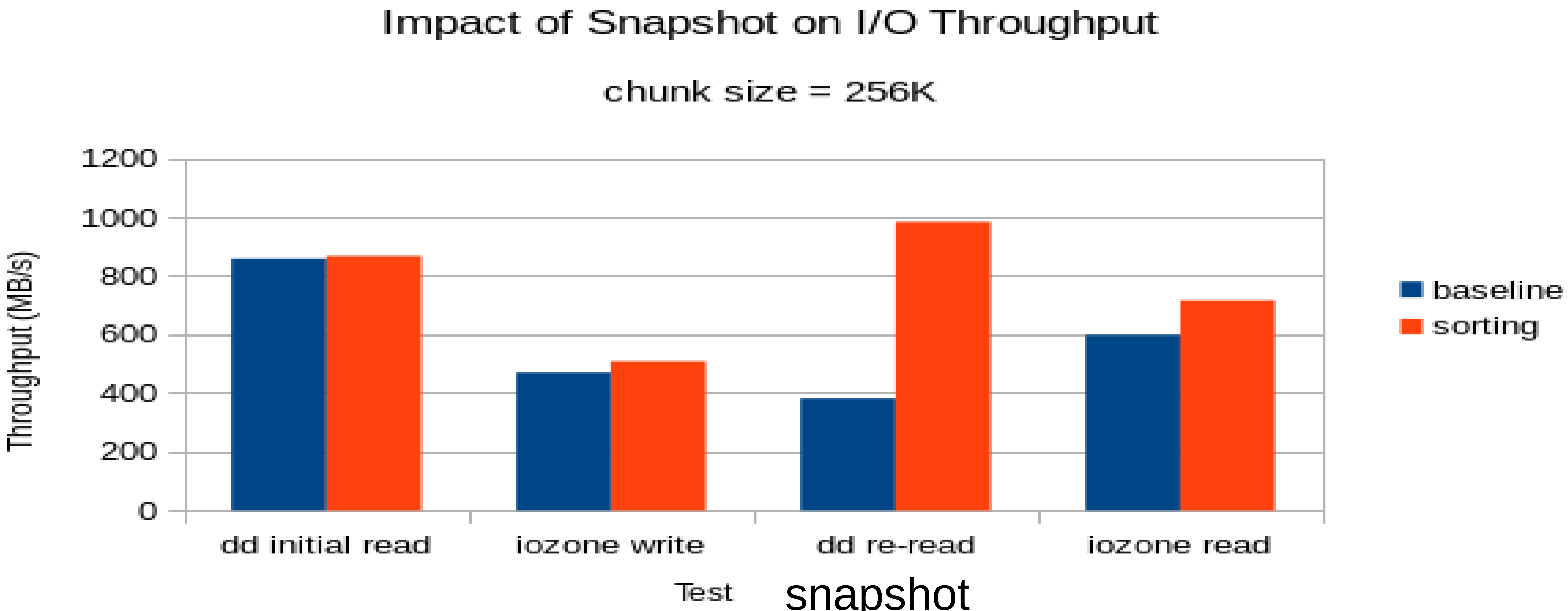
Future (upstream) directions

- Thin Provisioning (LVM) – Allows Gluster volumes with different users and policies to easily co-exist on same physical storage
- Snapshot (LVM) – For online backup/geo-replication of live files
- Small-file – optimizations in system calls
- Erasure coding – lowering cost/TB and write traffic on network
- Scalability – larger server and brick counts in a volume
- RDMA+libgfapi – bypassing kernel for fast path
- NFS-Ganesha and pNFS – how it complements glusterfs

LVM thin-p + snapshot testing

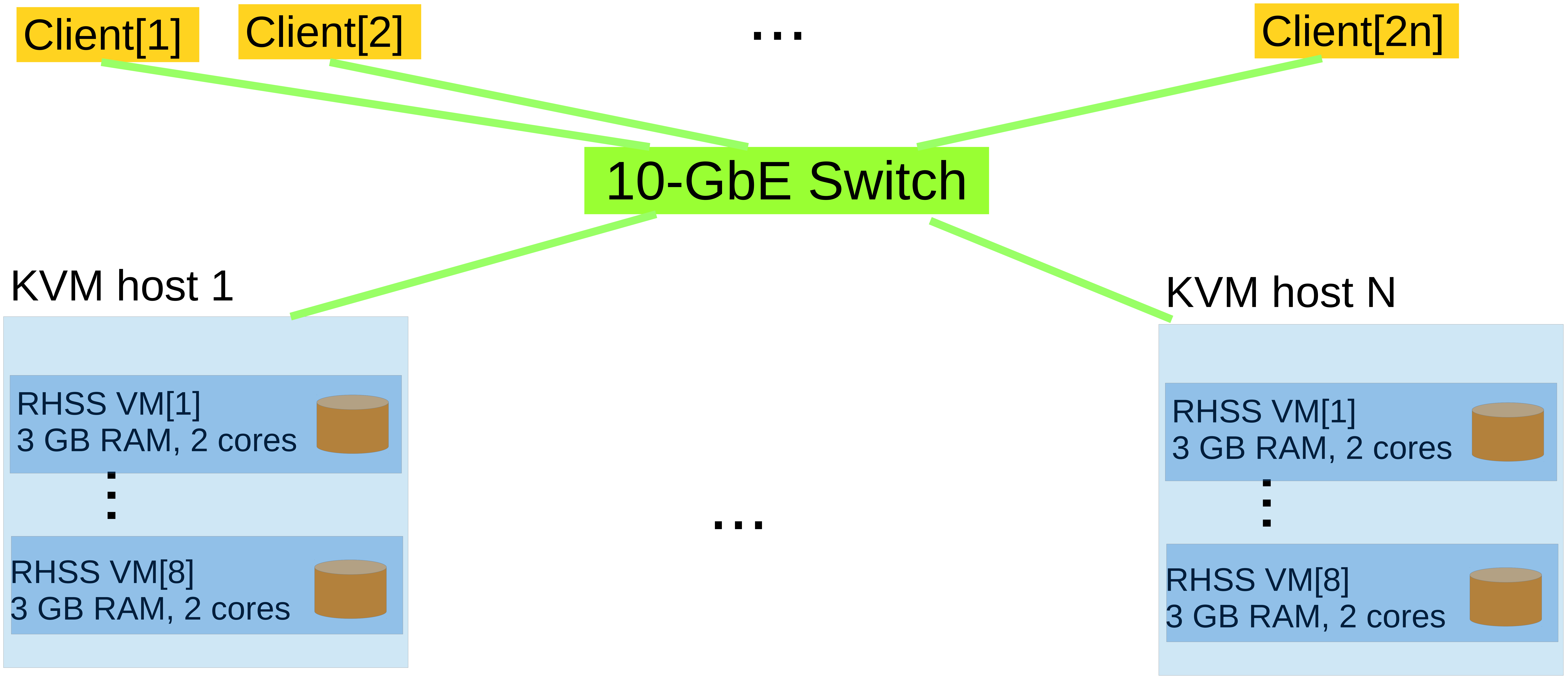


avoiding LVM snapshot fragmentation
LVM enhancement: bio-sorting on writes
testing just XFS over LVM thin-provisioned volume



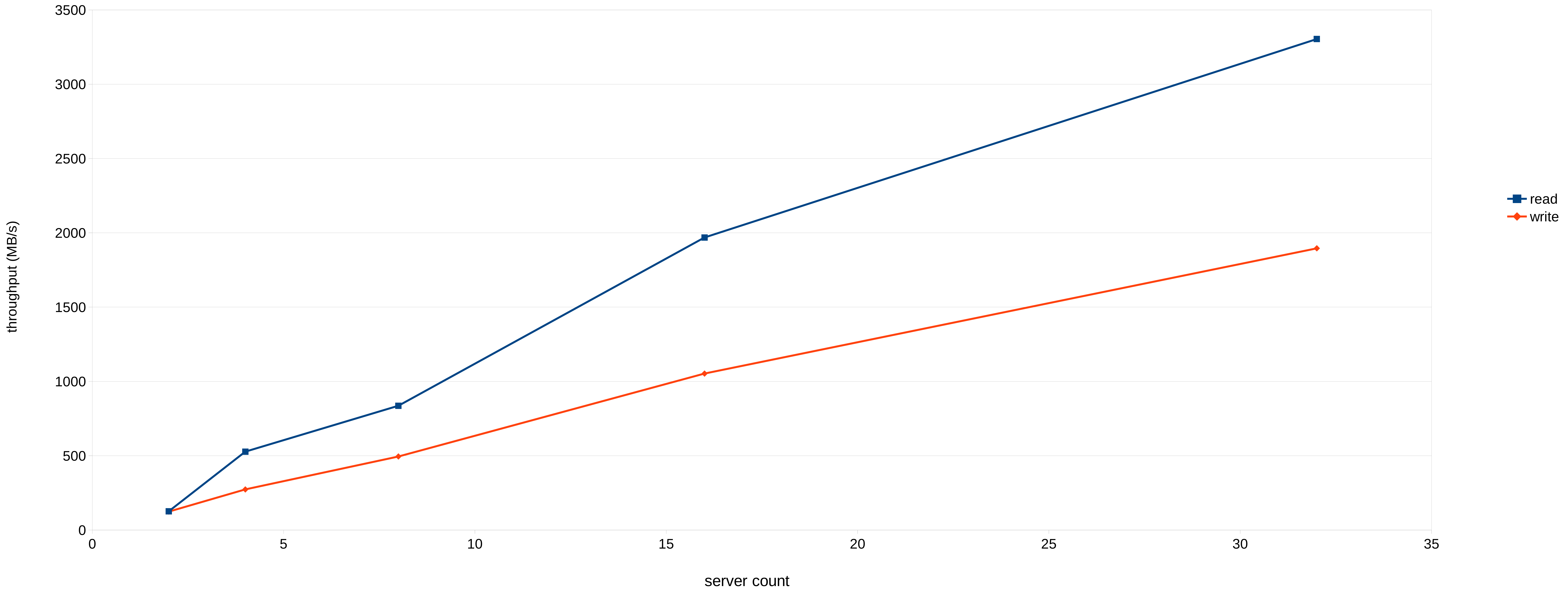
snapshot

Scalability using virtual RHS servers



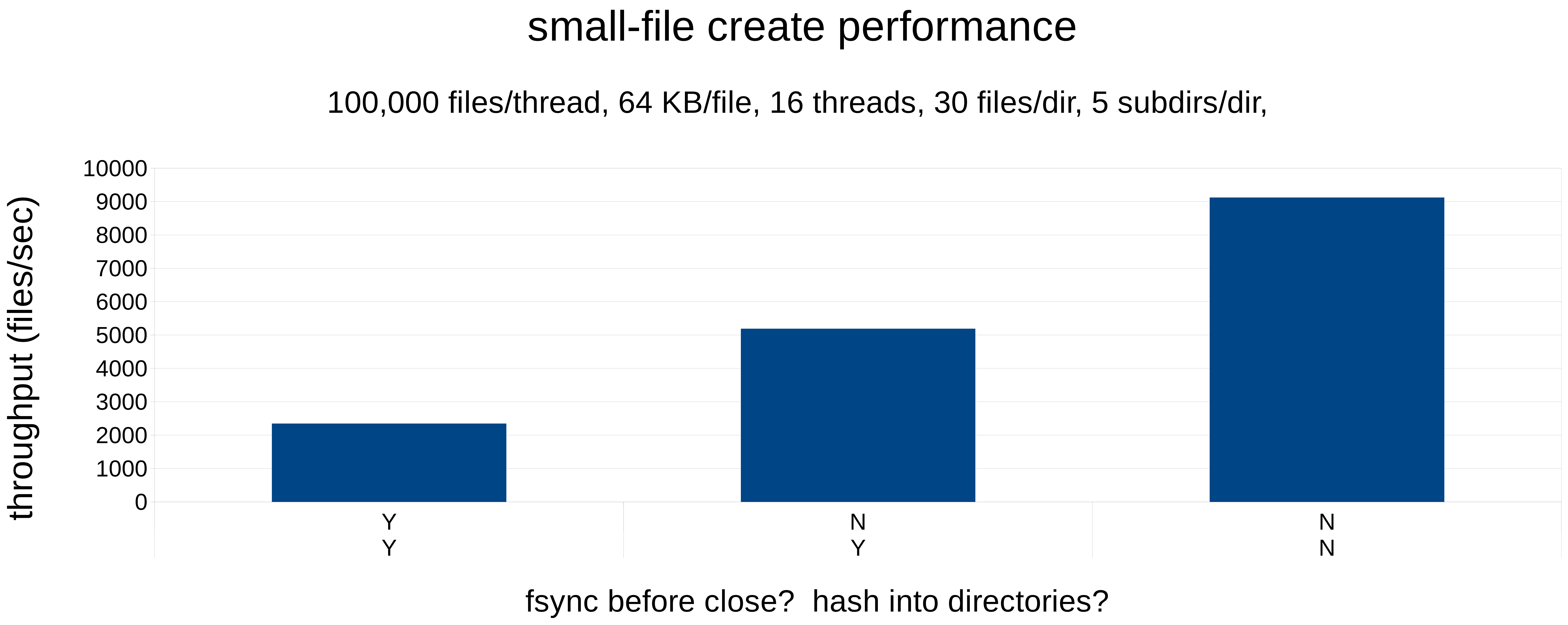
scaling of throughput with virtual Gluster servers for sequential multi-stream I/O

Glusterfs, up to 8 bare-metal clients, up to 4 bare-metal KVM hosts, up to 8 VMs/host,
2-replica volume, read-hash-mode 2, 1 brick/VM, RA 4096 KB on /dev/vdb, deadline sch.
2 iozone threads/server, 4 GB/thread, threads spread evenly across 8 clients.



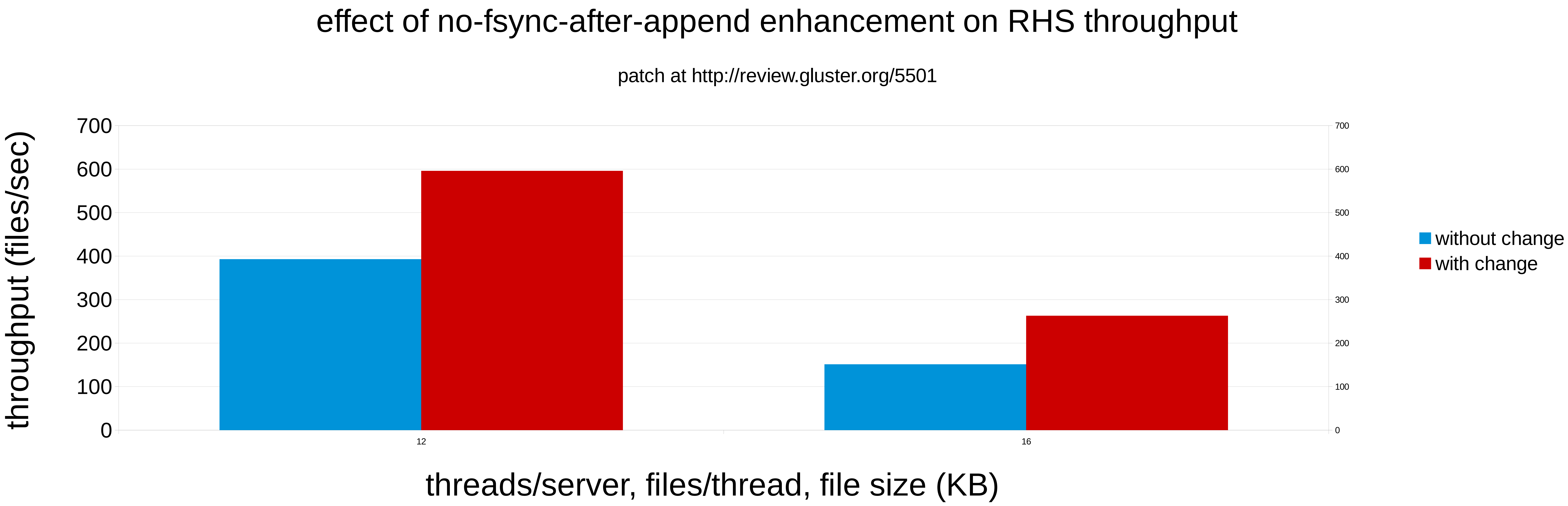
Optimizing small-file creates

Test Done with dm-cache on RHEL7 with XFS filesystem, no Gluster, 180 GB total data



Avoiding FSYNC fop in AFR
http://review.gluster.org/#/c/5501/, in gluster 3.5

- Reverses loss in small-file performance from RHS 2.0 -> 2.1



Check Out Other Red Hat Storage Activities at The Summit

- Enter the raffle to win tickets for a $500 gift card or trip to LegoLand!
  - Entry cards available in all storage sessions - the more you attend, the more chances you have to win!
- Talk to Storage Experts:
  - Red Hat Booth (# 211)
    - Infrastructure
    - Infrastructure-as-a-Service
- Storage Partner Solutions Booth (# 605)
- Upstream Gluster projects
  - Developer Lounge

Follow us on Twitter, Facebook: @RedHatStorage