DE LA RECHERCHE À L'INDUSTRIE

cea

www.cea.fr

EXPORTING LUSTRE WITH NFS-GANESHA

Philippe Deniel, Dominique Martinet
(philippe.deniel@cea.fr , dominique.martinet@cea.fr)

LAD2013 September 16-17, 2013

WHAT IS NFS-GANESHA

NFS-Ganesha was born at CEA/DAM in 2005

- Original need was to export HPSS over NFS
  - IBM stopped supporting this feature
  - The hpss_nfs daemon was really unreliable and with poor caching capabilities
- We designed something of our own in 4Q2004
  - We start coding in January 2005, once a design document had been written
  - Ganesha was designed with more than HPSS in mind

NFS-Ganesha is in production since early 2006

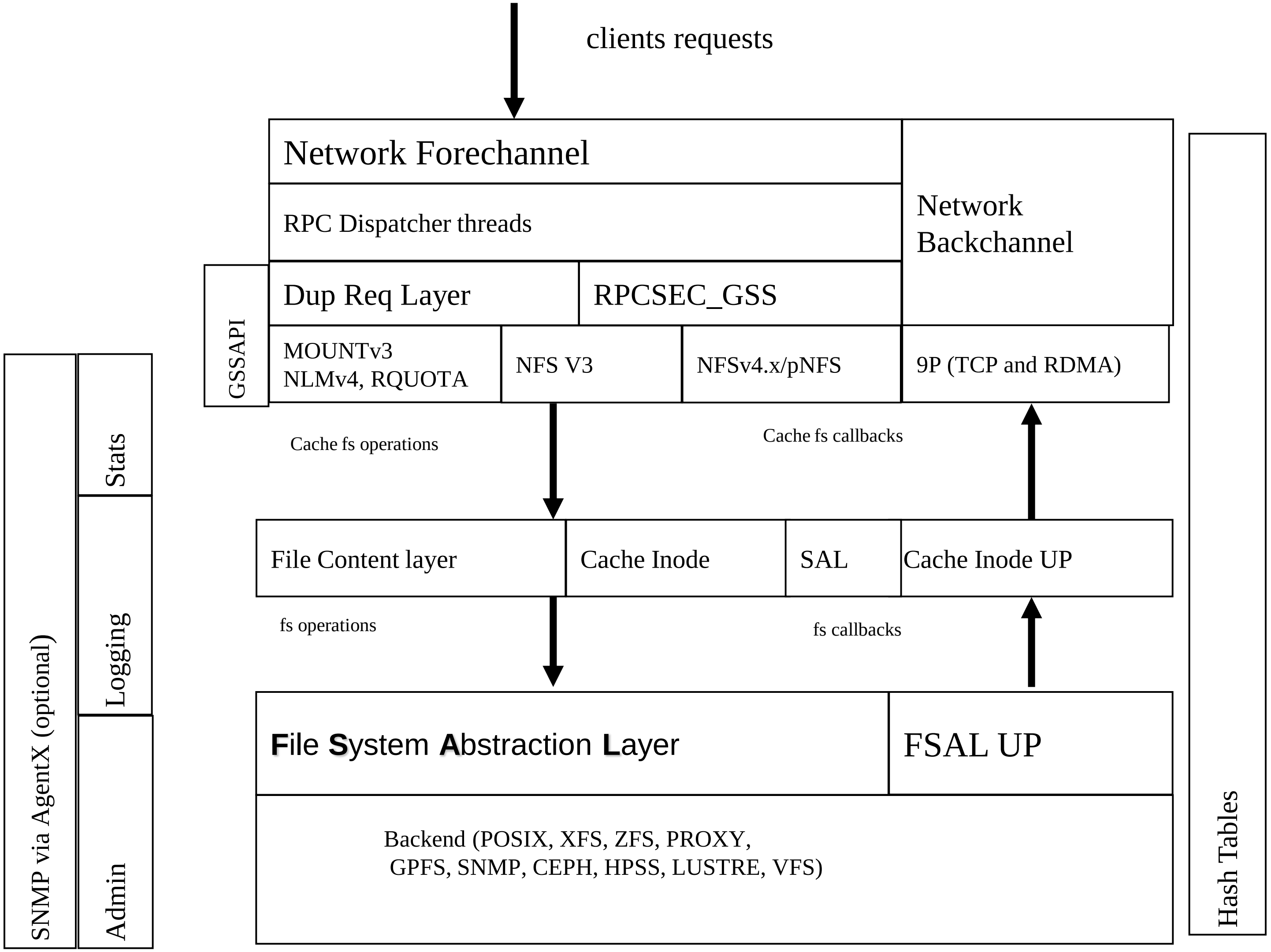
- First used to export HPSS to TERA10 system
- Used to export LUSTRE at TGCC in 2011, in front of CCRT's compute machines

FEATURES

NFS-Ganesha has known many evolutions. Currently it includes the following feature (non-exhaustive list)

- Supported protocols
  - NFSv3 and ancillary protocols (MOUNTDv3, NLMv4, client side of NSM)
    - NLMv4 implementation supports SHARE/UNSHARE used by Microsoft NFS client
  - NFSv4.0 (including lock support)
  - NFSv4.1 (including pNFS support)
  - 9p.2000L (with TCP and RDMA transport layers)
- Supported backends (known as FSAL : File System Abstraction Layer) are
  - CEPH
  - GPFS
  - HPSS
  - PROXY (operates as a NFSv4 client to turn Ganesha into a NFS PROXY)
  - LUSTRE 2.x
  - ZFS (content of a ZFS tank)
  - VFS (with kernel > 2.6.39. Makes it possible to export every FS managed by the kernel’s VFS)

GANESHA'S ARCHITECTURE



clients requests

Network Forechannel

RPC Dispatcher threads

Network Backchannel

GSSAPI

Dup Req Layer

RPCSEC_GSS

MOUNTv3 NLMv4, RQUOTA

NFS V3

NFSv4.x/pNFS

9P (TCP and RDMA)

Cache fs operations

Cache fs callbacks

File Content layer

Cache Inode

SAL

Cache Inode UP

fs operations

fs callbacks

File **S**ystem **A**bstraction **L**ayer

FSAL UP

Backend (POSIX, XFS, ZFS, PROXY, GPFS, SNMP, CEPH, HPSS, LUSTRE, VFS)

SNMP via AgentX (optional)

Stats

Logging

Admin

Hash Tables

NFS-GANESHA COMMUNITY

NFS-Ganesha was released as free software on July 4th, 2007

- Available on https://github.com/nfs-ganesha/nfs-ganesha/
- NFS-Ganesha is available under the terms of the LGPLv3 license

A Community starts to develop

- CEA/DAM is still active in the development
  - manage FSAL_HPSS, FSAL_PROXY and FSAL_LUSTRE, 9P and RDMA based transport
- IBM became an active member of the community in late 2009
  - Ganesha is to be integrated in SONAS as NFS Gateway
  - IBM is in charge of FSAL_GPFS and SAL (states management layer)
- LinuxBox (a small company created by former CITI folks) joined the community in september 2010
  - They are very active on NFSv4.1 with focus on CEPH
- Panasas joined the community in May 2011
  - Ganesha is to be used as NFSv4.1/pNFS MDS in Panasas Product

MORE FOCUS NFS-GANESHA LUSTRE

FSAL_LUSTRE provides access to LUSTRE for NFS-Ganesha daemon

- FSALs are provided as a dynamic library to be dlopen-ed at startup by ganesha.nfsd daemon (in Ganesha 2.0)

- Based on a few LUSTRE features
  - Uses “.lustre/fid” special directory to access objects
  - Calls from liblustreapi
    - Fid2path
    - path2fid

- Provides access to xattr
  - Native feature in 9p2000.L and NFSv4.x
  - Makes use of “ghost directories” in NFSv3 and NFSv4 (Linux has no NFSv4 client support for extended attributes as Solaris does)

MORE FOCUS NFS-GANESHA LUSTRE

Future cool features for LUSTRE

- pNFS support (using file based layout) for FSAL_LUSTRE
  - Main discussion is about placing pNFS Data Servers correctly
  - It seems logical to place them closer as possible to OSSs, or even running on OSSs
    - The latest choice would make the translation from LUSTRE layout to pNFS layout easier
  - Memory pressure should be considered
    - pNFS/DS are rather stateless creatures (the states are managed by the pNFS/MDS)
    - Ganesha as pNFS/DS would be redesigned with reduced caches

- Use LUSTRE changelogs to implement “FSAL upcalls” (as GPFS does) to update caches as LUSTRE changes
  - Upcalls are trapped by a pool of Ganesha’s threads
    - Related cached inode is removed from the cache
  - Would make NFS-Ganesha caches coherent with LUSTRE
    - Would make Ganesha fully compliant with NFSv4.1 (as RFC5661 says)
  - Would help in clustering NFS-Ganesha server on top of LUSTRE

BENCH: HARDWARE AND SOFTWARE USED

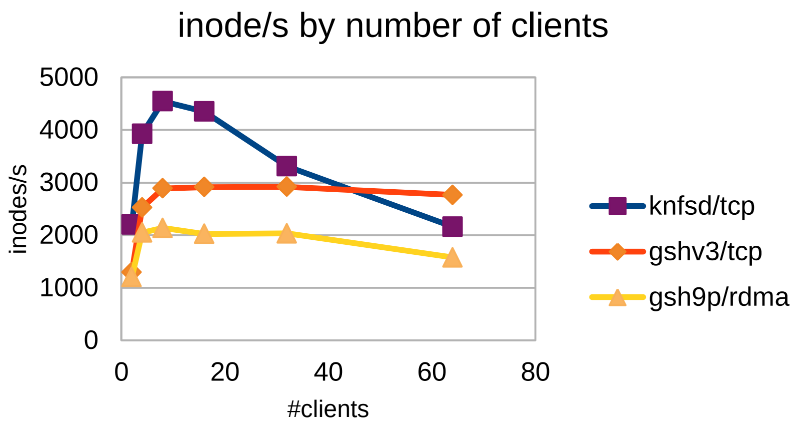
Details of benchmark configuration

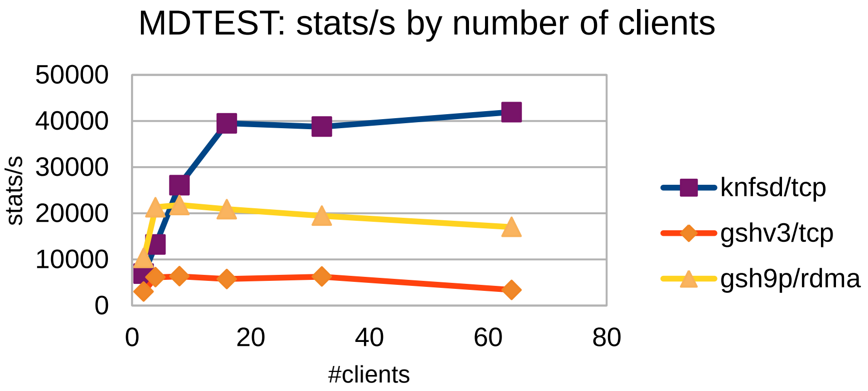
- Hardware
  - Clients are BULL B500 nodes
    - 4 sockets, Nehalem processors (8 cores)
    - 64 GB RAM
  - Lustre MDS and OSS
    - Bull MESCA S6030 nodes, 4 sockets Nehalem (8 cores) , 64 GB RAM
  - Network is Mellanox QDR Infiniband

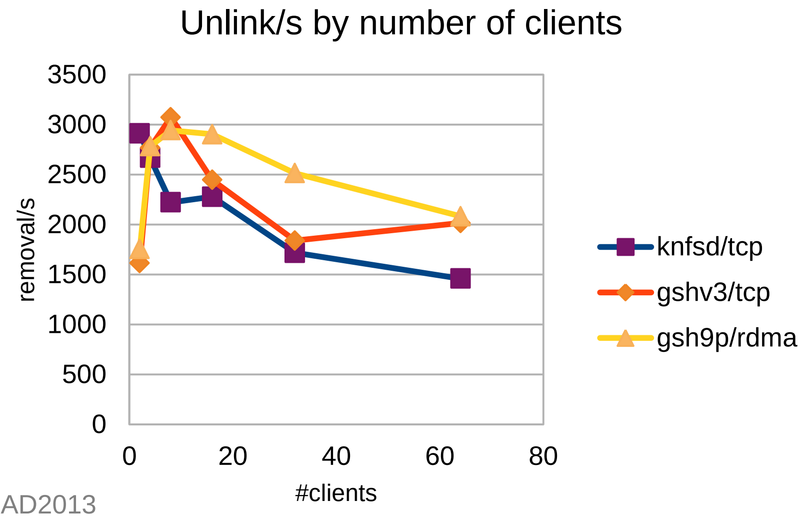
- Software
  - Lustre 2.1.4 sur BULL AE2.2 (based of EL6.1)
  - Clients are running BULL AE2.2
  - Ganesha pre-2.0-dev_42-40-gd3b8c25 (yes, that's a "git describe –long" ;-)) with mooshika-0.3.7-gb3e264a

BENCH : GANESHA VS KNFSD (METADATA 1/3)

RESULTS OF MDTEST: directory create/stats/rm



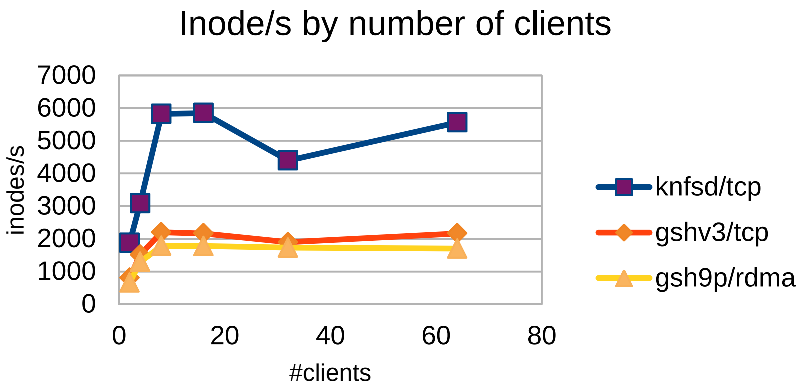


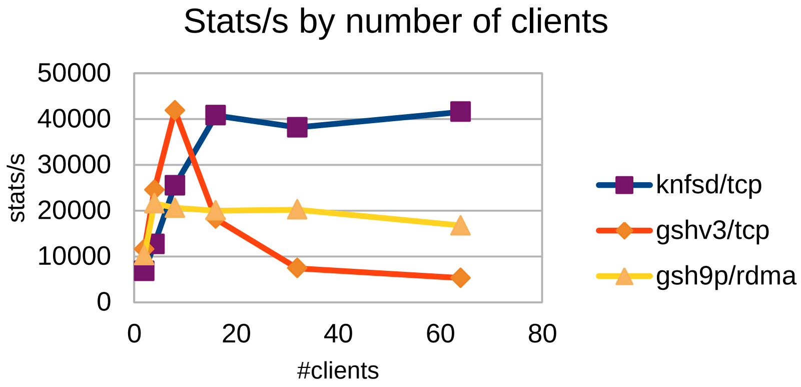


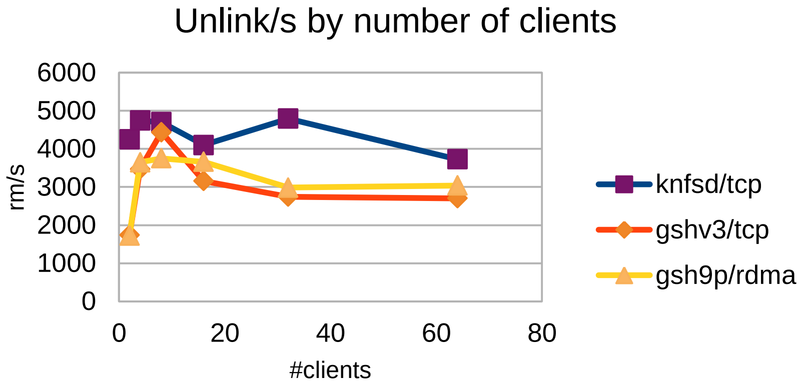
Knfsd is better than Ganesha on Directory metadata management, Especially on stats (possible cache effect)

BENCH : GANESHA VS KNFSD (METADATA 2/3)

RESULTS OF MDTEST: files create/stats/rm



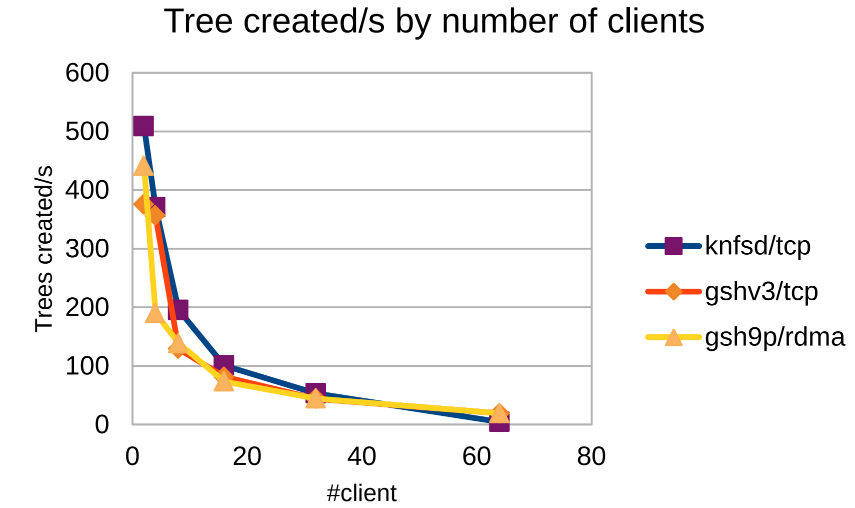




Knfsd is better than Ganesha on File metadata management, too

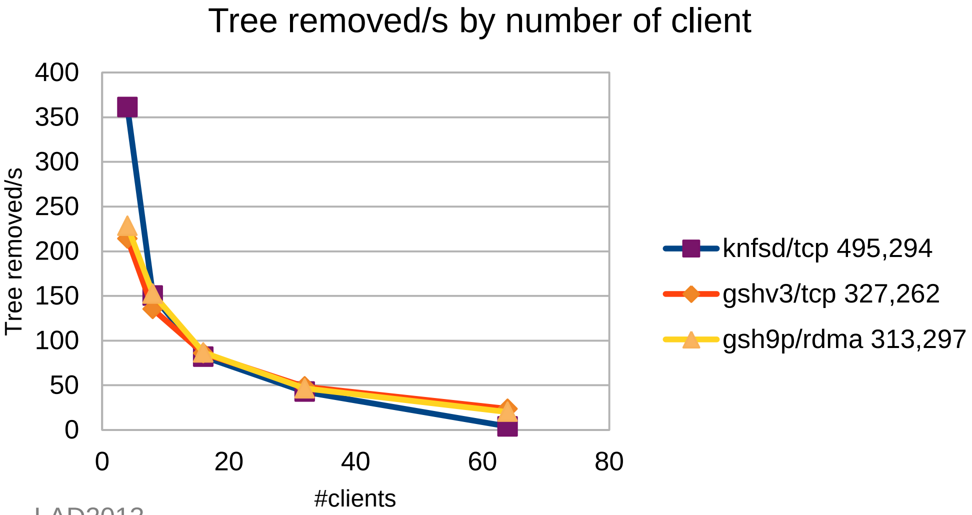
BENCH : GANESHA VS KNFSD (METADATA 2/3)

RESULTS OF MDTEST: files create/stats/rm

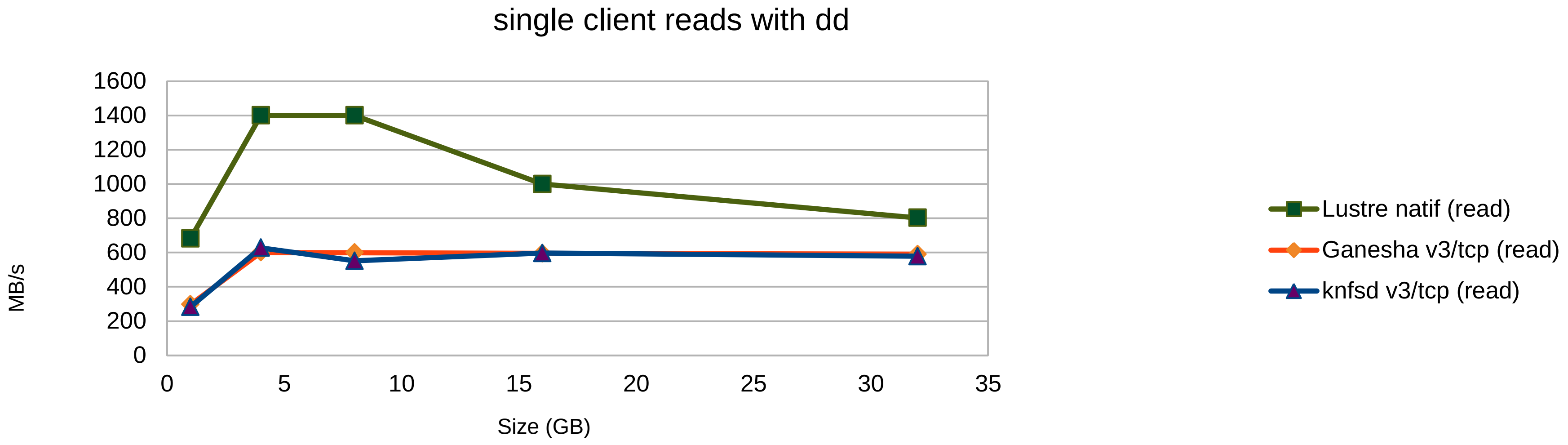


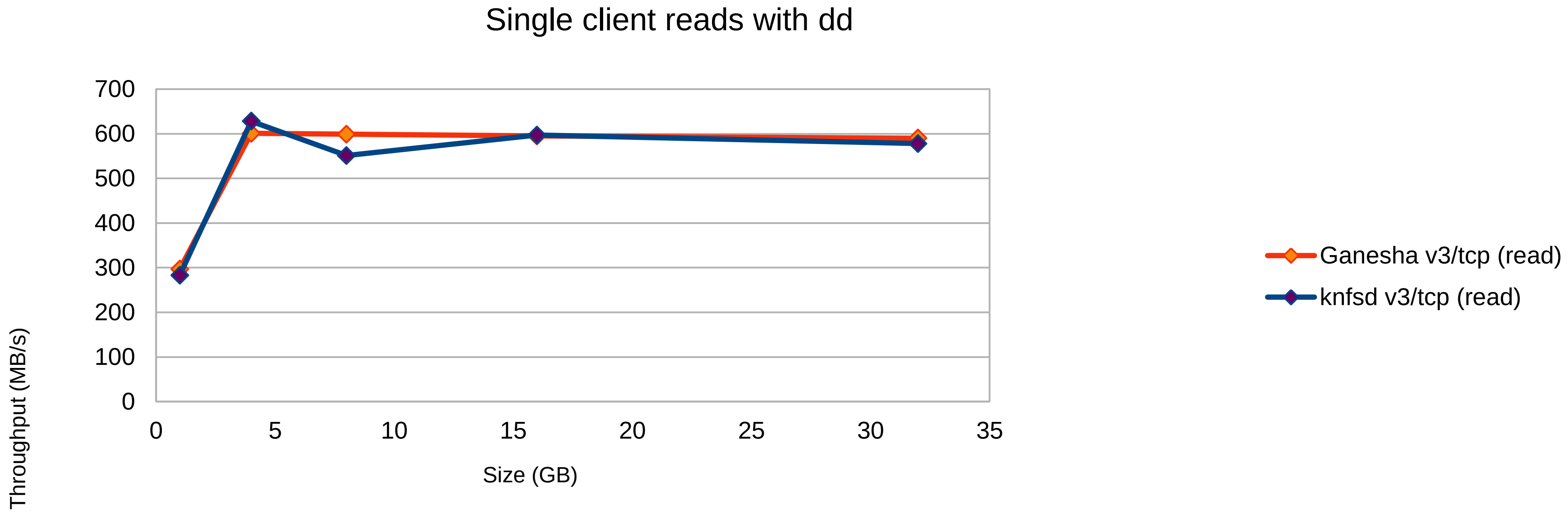
Knfsd and Ganesha have similar performances on tree operations

Ganesha becomes slightly better as the number of client increases

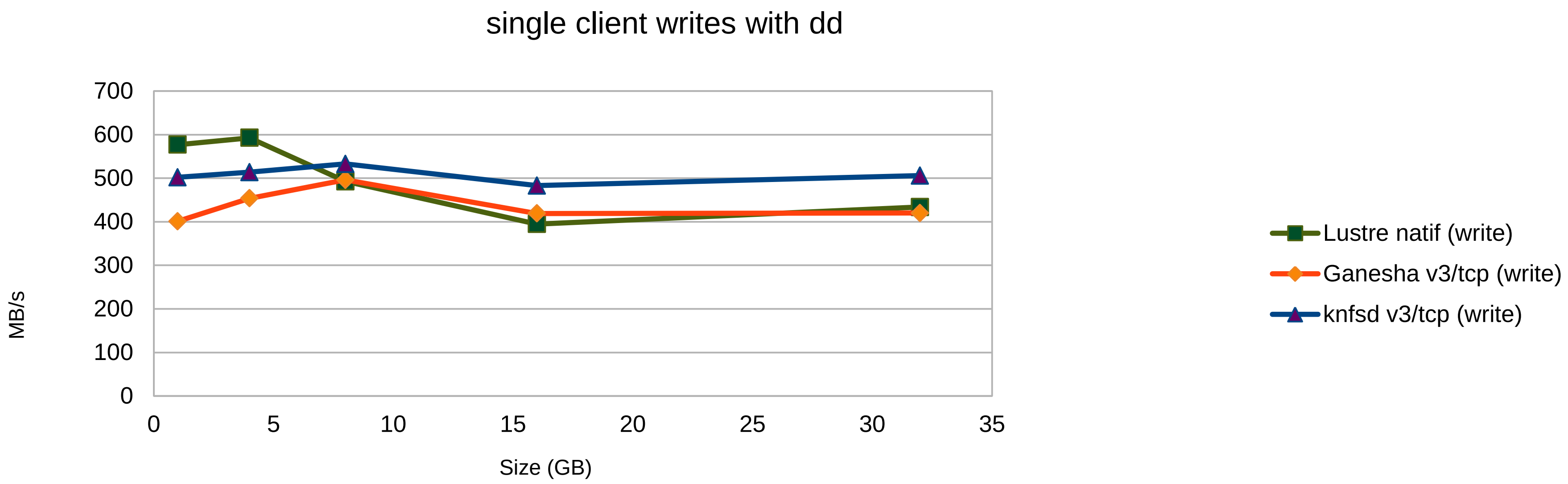


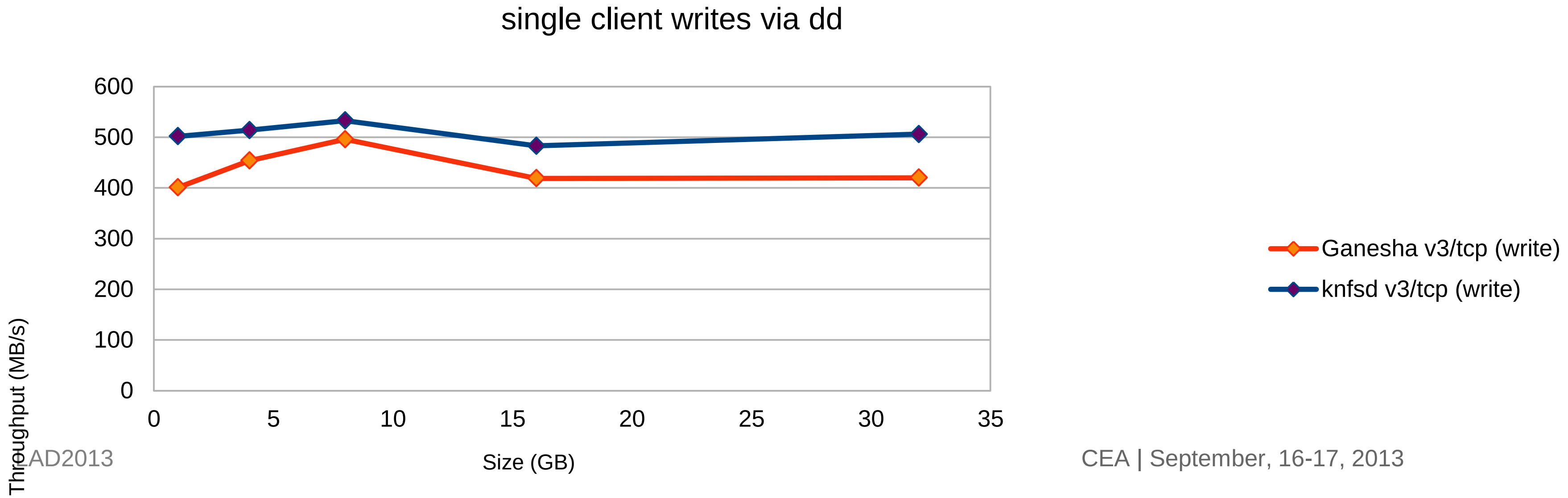
BENCH : GANESHA VS KNFSD (DD READ)



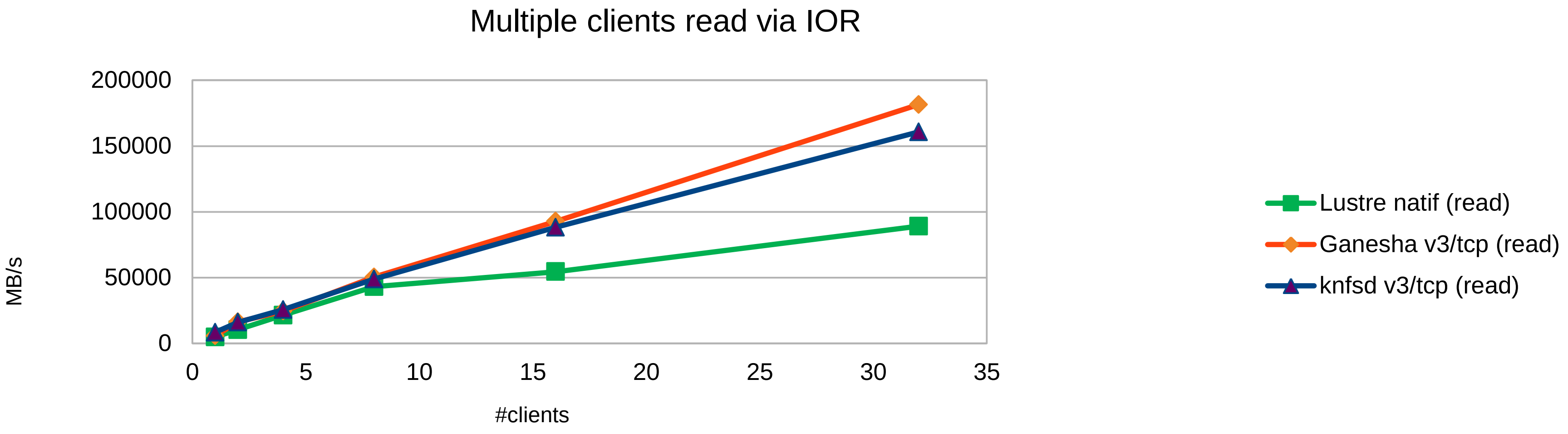


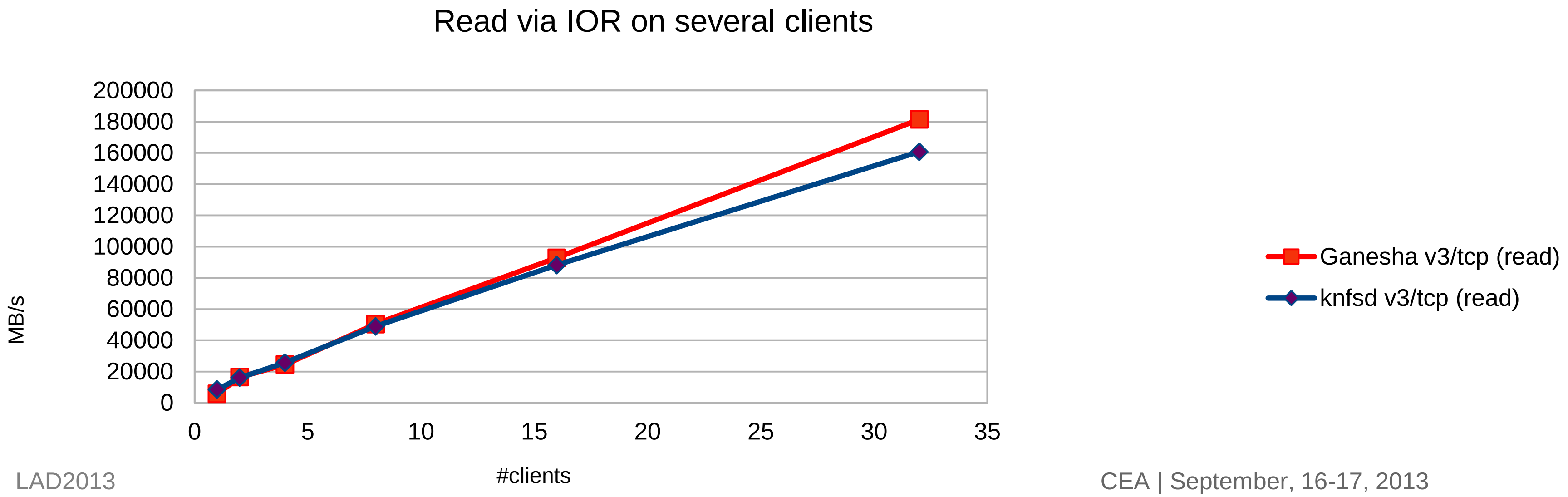
BENCH : GANESHA VS KNFSD (DD WRITE)



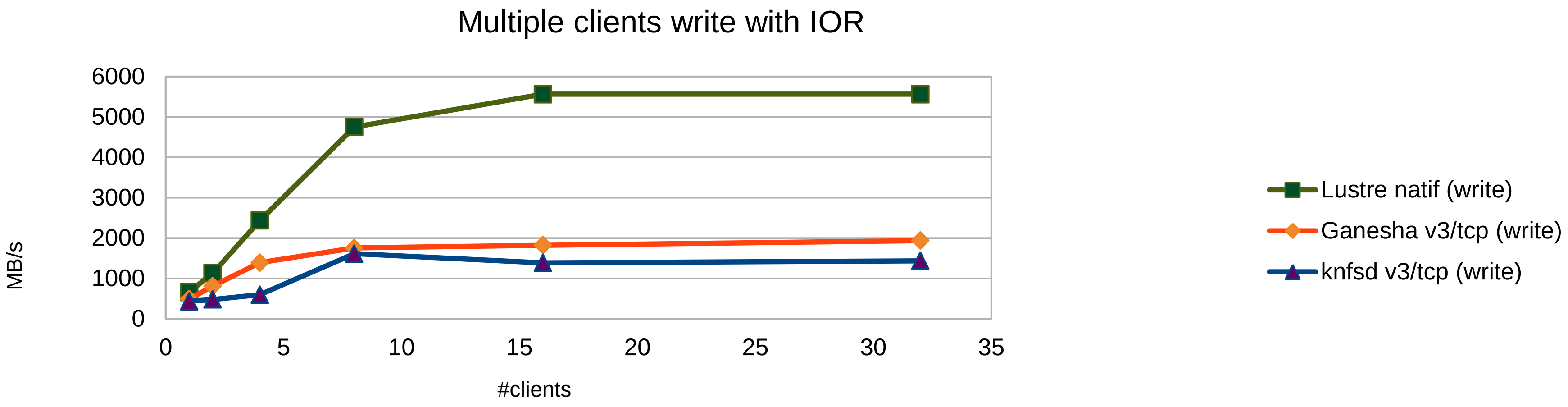


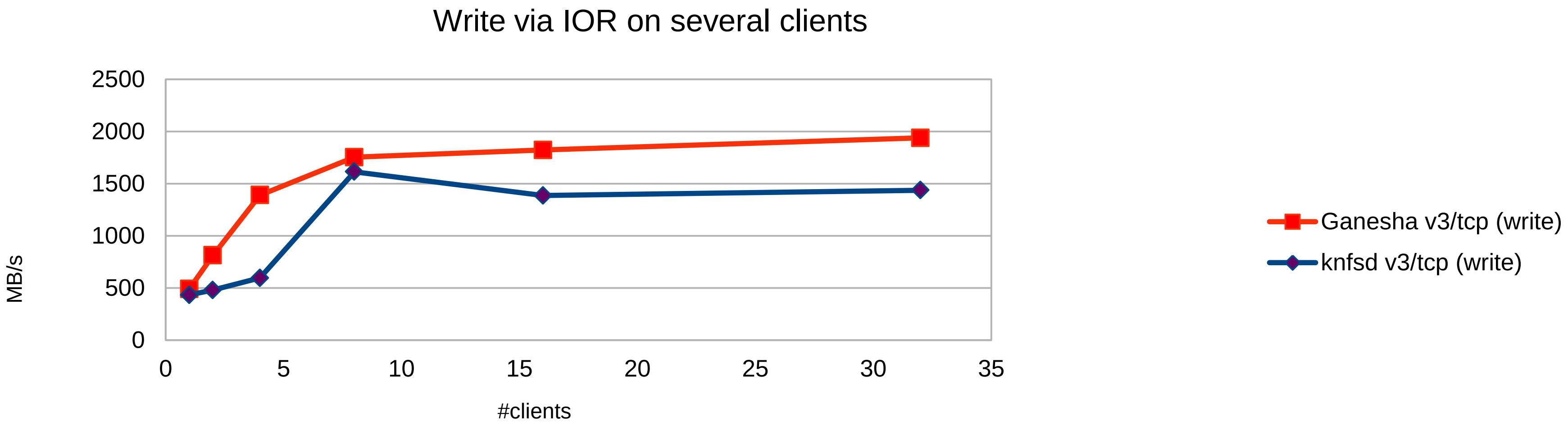
BENCH : GANESHA VS KNFSD (IOR READ)





BENCH : GANESHA VS KNFSD (IOR WRITE)





COMMENTS ABOUT IO BENCHMARKS

Ganesha and knfsd have similar single client performances

- Knfsd is faster on write (about 7% better)
- Ganesha is faster on read (about 3% better)

Read operations are strongly impacted by

  - Lustre’s caches
  - NFS client caches

Ganesha is interesting in clustered environment

- Ganesha’s performances are about 30% better than knfsd when multiple clients do write operations on the same server
- Read operations suffer from by huge cache effects
  - Read operations (with both server) are faster than LUSTRE reads!!!!!!!
  - Both Ganesha and knfsd behave the same way

COMMENTS ABOUT METADATA BENCHMARKS

Ganesha accesses objects “by fid”

- NFS file handles carries the lustre FID for the related object
  - Ganesha builds the related path in /mountpoint/.lustre
  - Ganesha then uses this “fid path” to access the object
- The knfsd is in the kernel space but Ganesha is in user space.
  - Information is to be moved from kernel space to Ganesha
- Lustre seems to behave differently as object are accessed by path or by FID
  - Any comment in the room ? Feedback is wanted on this point.
- Both Ganesha and knfsd run on a single client
  - Their performances will never exceed those of a single client
  - Using pNFS will break this bottleneck
- A single client in Lustre 2.1 suffers from “single shared file” issue as multiple access are done to a single file with direct impact to NFS performances
  - See LU1666, LU1669, LU2481 (mostly fixed in 2.1.5)

FEEDBACK FROM PRODUCTION CASES

Ganesha is used in production at CEA

- Ganesha exports HPSS namespace (metadata only) on TERA and TGCC
- Ganesha exports LUSTRE (full rw access) on TGCC
  - Part of the compute machine used an obsolete kernel (no LUSTRE)
  - NFSv3 was used as a fallback
  - Ganesha was providing NFS shares in RW
  - We know Ganesha can be used in HPC environment : we did use it
- What about crashes ?
  - Ganesha resides in the user space
  - NFSv3 is a stateless protocol
  - NFSv4.x has client recovery features
  - If the daemon crashes… just restart it and continue working
- Big issue related to knfsd
  - Depending on some access patterns, knfsd could generate lbugs
  - If knfsd crashes, it crashes the whole node and you need to reboot

AS A CONCLUSION

- Ganesha’s development is continuing
  - More NFSv4.x feature including more acl support and delegation support
  - More pNFS for LUSTRE
  - LUSTRE changelogs to implement Upcalls for FSAL_LUSTRE
  - Support for NFS/RDMA
    - Ganesha already have RDMA support for 9p2000.L

- Ganesha is stable enough to be used in production

- Ganesha keeps good performances against many clients

- User Space is a nice place
  - Easy access to many services (kerberos, idmapper, dns, …)
  - Make it easy to build a sandbox
  - It’s easier to update a daemon than a kernel element

- Security
  - Ganesha has efficient NFS/krb5 support via RPCSEC_GSS
  - We will make Ganesha capable of running as a non-root user
    - service will be restricted to NFSv4.x and 9p2000.L

QUESTIONS ?