NFS-Ganesha,
a NFS server in User Space

Philippe Deniel (philippe.deniel@cea.fr)

The needs behind NFS-Ganesha

- Provide a NFS Interface to services that are accessible in User Space and that provide data organized as “tree namespaces”
- Provide an efficient caching device for data and metadata
- Provide a scalable product, with as few bottlenecks as possible
- Provide generic support to various namespaces

NFS-Ganesha and the TERA compute center

- NFS-Ganesha is used via NFSv3 in a supercomputing environment, in front of compute clusters since 01/2006
  - About 5000 different clients (TERA10 compute cluster + TERA100 compute cluster)
  - heavy load (parallel MPI based simulation code)
- The daemon is very stable and scales well on new machines
  - Takes benefit of multi-cores/multi-sockets systems
  - Takes benefit of system with a large amount of memory
- Plan to step to NFSv4 with krb5i authentication
- Plan to use pNFS with NFS-ganesha by the end of 2012 on production systems

Supported protocols

- Currently, NFS-Ganesha supports
  - NFSv2 / MOUNTv1
  - NFSv3 / MOUNTv3 / NLMv4
  - RQUOTAv1 and RQUOTAv2
  - NFSv4.0
  - NFSv4.1
    - pNFS's layout file model is supported
    - Other layouts to be coming soon (OSD layout used with exofs)

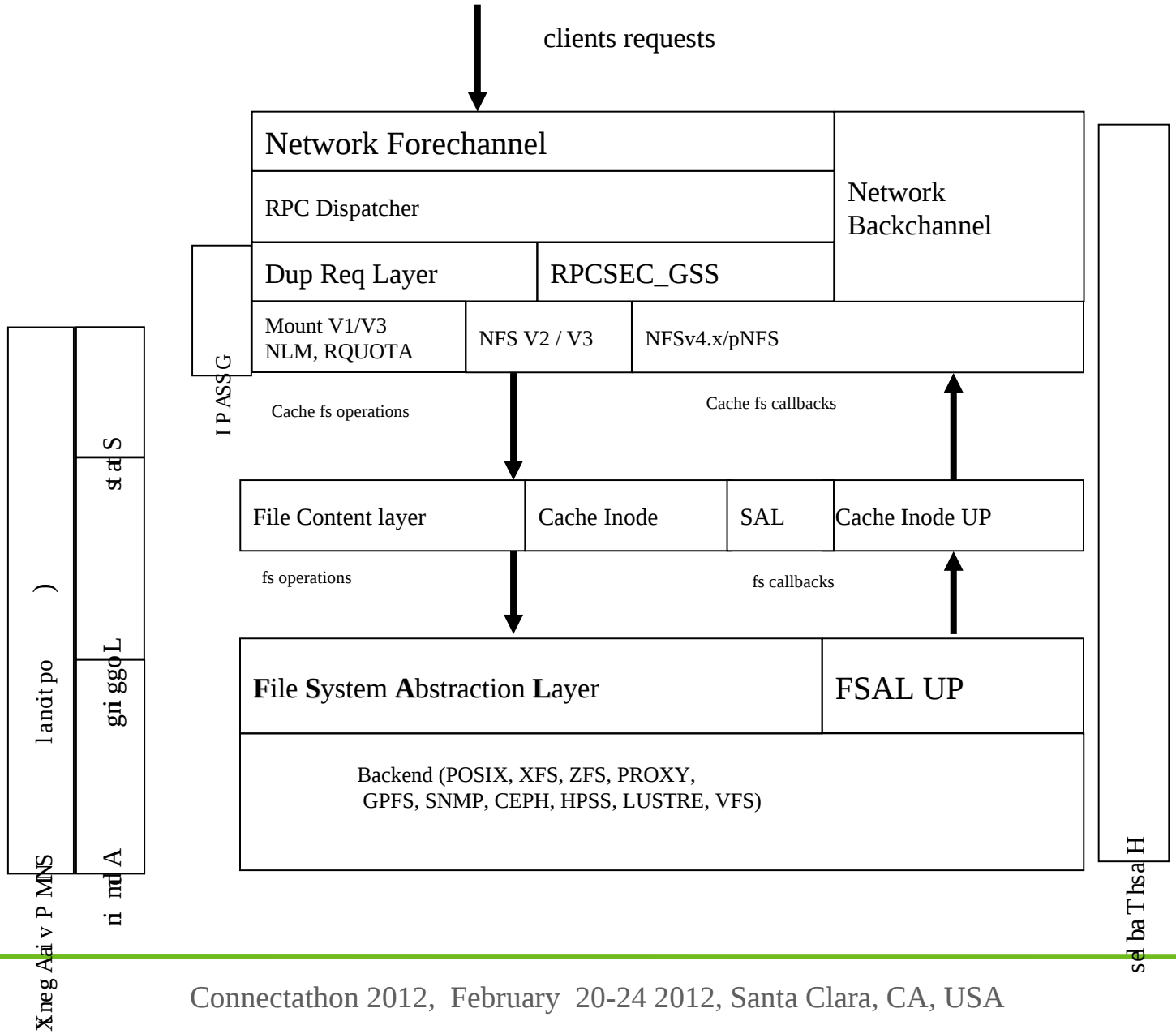
User Space is a nice place ;-)

- Running in User Space makes many things easier.
  - Security Managers (like Kerberos) reside in User Space, they can be accessed directly via GSSAPI (no need for “rpc_pipefs”)
  - ID mappers (NIS, LDAP) reside in User Space, they can be accessed directly (the daemon is linked with libnfsidmap)
  - Less constraints for memory allocation than in kernel space, managing huge pieces of memory is easy
  - When developing in User Space, you won't (usually) crash the kernel

Modular architecture

- Some layers make the « core » of the architecture:
  - RPC Layer : implements ONC/RPCv2 and RPCSEC_GSS (based on libtirpc)
  - FSAL : File System Abstraction Layer, provides an API to generically address the exported namespace
  - Cache Inode: manages the metadata cache for FSAL. It is designed to scale to millions of entries
  - File Content Cache: manages the data cache for the FSAL entries
  - Log management: one API for internal logging (in files) and remote logging (via syslog API)
  - Memory allocation and management: NFS-Ganesha do not rely on malloc/free but allocates a big chunck of memory at startup and manages it on its own. This is done by using pre-allocated structures and blocks of memory managed as “Buddy Blocks”
  - Hash Table: provides a Red-Black Tree based hash table to provide associative addressing to internal structure. This layer is widely used to build various cache, including the metadata cache.
  - FSAL UP / Cache Inode UP: provides the daemon with a way to be notified by the FSAL that changes have been made to the underlying FS outside Ganesha. These information is used to invalidate or update the Cache Inode.
- Each layer has a well defined API. It can be modified (e.g. to implement a different cache policy) without impacting the other modules

The modules within NFS-Ganesha



FSAL: File System Abstraction Layer

- The FSAL provides a namespace independent API, used to address the namespace in upper layers
- Its semantic is close to the NFSv4 one, but allows easy implementation of NFSv2/NFSv3
- Handle based API (lookup, readdir, …)
- Implements namespaces' specific authentication mechanisms.
- FSAL module is the one and only place where the native namespace's API is used

Available FSALs (1/3)

- FSAL/PROXY : API calls are implemented as NFSv4 client requests, turning NFS-Ganesha into a NFSv4 Proxy
- FSAL/LUSTRE: provides access to a LUSTRE filesystem
- FSAL/XFS: provides access to a XFS filesystem
- FSAL/ZFS : this FSAL derives from the fuse based implementation of ZFS. Links to FUSE were removed to make ZFS running totally in User Space and being access via NFS-Ganesha (all in User Space)
- FSAL/FUSE : this FSAL makes it possible to bind any “FUSE ready” project with NFS-Ganesha to generate a user space NFS daemon capable of exporting the related namespace (see Ben Martin’s article at http://www.linux.com/archive/feature/153789)

Available FSALs (2/3)

- FSAL/GPFS : FSAL developed by IBM to provide backend to the GPFS filesystem
- FSAL/CEPH : FSAL developed by the LinuxBox company to provide backend to the CEPH filesystem
- FSAL/VFS: in linux 2.6.39 it becomes possible to “open by handle” on VFS managed filesystem (as in XFS and GPFS). This will make it possible to have a FSAL_VFS working in a similar way to FSAL_XFS and FSAL_GPFS
- FSAL_HPSS: “historical” FSAL used to access the namespace of the HPSS namespace (HPSS is an HSM from IBM Government Systems, massively used in the HPC community)
- FSALs can be compiled in both a static and a dynamic way. In this later case, the nfs-ganesha daemon “dlopens” the shared library as it starts.

Available FSALs (3/3)

- Coming soon (2Q2012)
  - FSAL/EXOFS : will provide access to the EXOFS filesystem. This is a requirement for implementing pNFS/OSD2
    - pNFS OSD layout will be part of this feature
  - FSAL API extension
    - Designed to manage FS specific feature (snapshots, non POSIX attributes (generation number, creation time, version), ...)

What is specific to Ganesha

- NFS-Ganesha provides access to xattr in both NFSv3 and NFSv4,
  - Regular use of OP4_OPENATTR for client who uses it (Solaris)
  - Use of '.xattr.d' ghost directory in NFSv3 and NFSv4
    - If a file dir/foo exists, then its attributes are accessed in dir/.xattr.d.foo
  - Example (on top of FSAL_HPSS)

```
# ls .xattr.d.file/
bitfile_id  class_of_service  ns_handle  storage_levels  type
# cat .xattr.d.file/storage_levels
Level 0 (disk) : 0 bytes
Level 1 (tape): 209715200 bytes
# cat .xattr.d.file/class_of_service
18
```

  - Extended attributes can be set/modify/remove as files as well

Things to be completed…

- NFSv4 delegation support
- NFSv4.1 backchannels (under development)
- Complete pNFS implementations
  - OSD2 support scheduled with FSAL_EXOFS
  - New LUSTRE layout ?
- Miscellaneous NFSv4.1 features
  - Directory delegations (when supported by the client)
- Asynchronous metadata management
- RPC/RDMA support

Open Source Collaborations

- Since early 2010, several companies started to collaborate to NFS-Ganesha
  - IBM was the first company to come in early 2010
  - LinuxBox (Ann Harbor, MI, US) joined later in 4Q2010
  - Panasas joined the community in 2011
- OpenSource development and collaboration is a great chance for the project.
- Help is always welcome. If you are interested in NFS-Ganesha, just join us :-)

What you need to remember about this topic ;-)

- NFS-Ganesha is a NFS server. It supports NFSv2, NFSv3, NFSv4 and NFSv4.1 (with pNFS)
- NFS-Ganesha runs fully in User Space
- NFS-Ganesha is designed to be generic via FSAL
- NFS-Ganesha scales on the hardware
- NFS-Ganesha has several backends
- NFS-Ganesha uses huge caches (up to tens of millions of entries)
- NFS-Ganesha has a very layered architecture
- NFS-Ganesha is massively multi-threaded
- OpenSource community is very active.

Where to download NFS-Ganesha?

- NFS-Ganesha is available under the terms of the LGPLv3 license
- NFS-Ganesha is hosted on SourceForge:
  - Project homepage
    - http://nfs-ganesha.sourceforge.net
  - Download page
    - http://sourceforge.net/projects/nfs-ganesha/files
  - Mailing lists
    - nfs-ganesha-devel@lists.sourceforge.net
    - nfs-ganesha-support@lists.sourceforge.net
    - nfs-ganesha-announce@lists.sourceforge.net

Git access

- git clone git@github.com:phdeniel/nfs-ganesha.git
- git clone http://phdeniel@github.com/phdeniel/nfs-ganesha.git