

Computer System Design and Administration

Topic 7. Network file system service: NFSv4



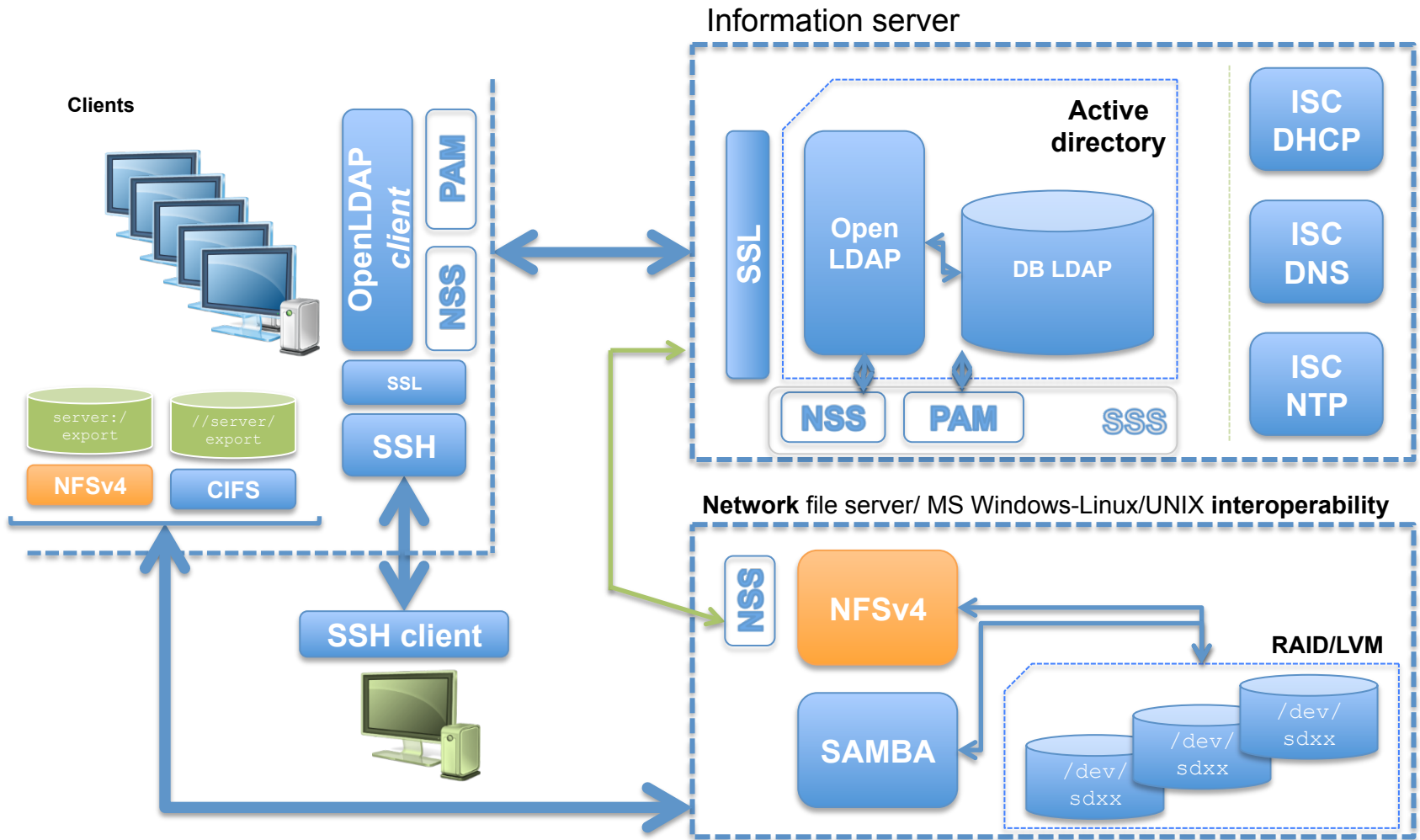
José Ángel Herrero Velasco

Department of Computer and
Electrical Engineering

This work is published under a License:

[Creative Commons BY-NC-SA 4.0](https://creativecommons.org/licenses/by-nc-sa/4.0/)

Secure information service: Puzzle

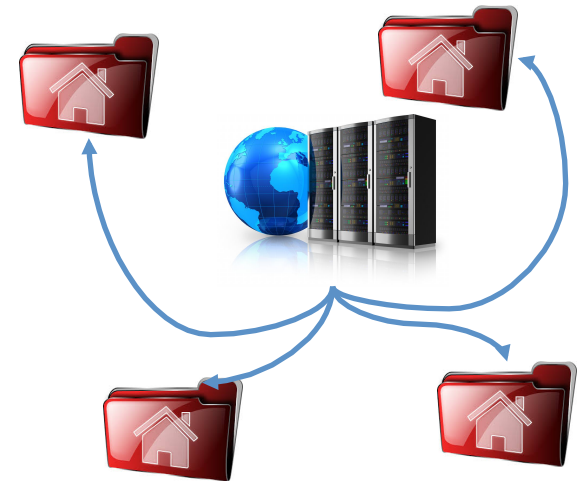


Target: ...breaking down the boundaries of local data

- Deployment and development of a *secure network file and resource system*, based on **NFS & SMB/CIFS (SAMBA)** technology:
 - **Network file system:**
 - It is about **centralizing** the storage of (mainly) user files from a computer environment.
 - Computational resources sharing system:
 - MS Windows – Linux/UNIX interoperability.

Distributed file systems

- A **distributed file system** is a suite of logic *structures, methods* and (remote) *procedures* that are managed through a group of **services**:
 - Takes charge of **storage, setup, recovery, naming, sharing** and **security** of data:
 - Data is stored in files that can be located on **different servers**.
 - Enables the client host *processes* to access files in a transparent and efficient way:
 - “Centralized” view of the data (files) as a **unique data volume**.
- **Advantage:**
 - Any user/app is able to access its files from anywhere.
 - Provides a **software programming interface** that hides the location details of data (files).
 - “**Simple**” management (centralized storage):
 - All data on one (or group) server.
 - Provides **security** mechanisms:
 - ACL system.
 - Disk quotas.
 - **Kerberos** integration.
 - “**Failover capability**” (*metadata, files...*).
 - Improves **reliability**, with regard to local systems:
 - Using RAID devices on server.
 - Replication techniques.
 - High **performance**, despite network:
 - Using *cache* on clients.



Distributed file systems

• Requirements:

- Transparency.
- Concurrency.
- Performance.
- Replication.
- Heterogeneity.
- Consistency.
- Fault tolerance.
- Security.

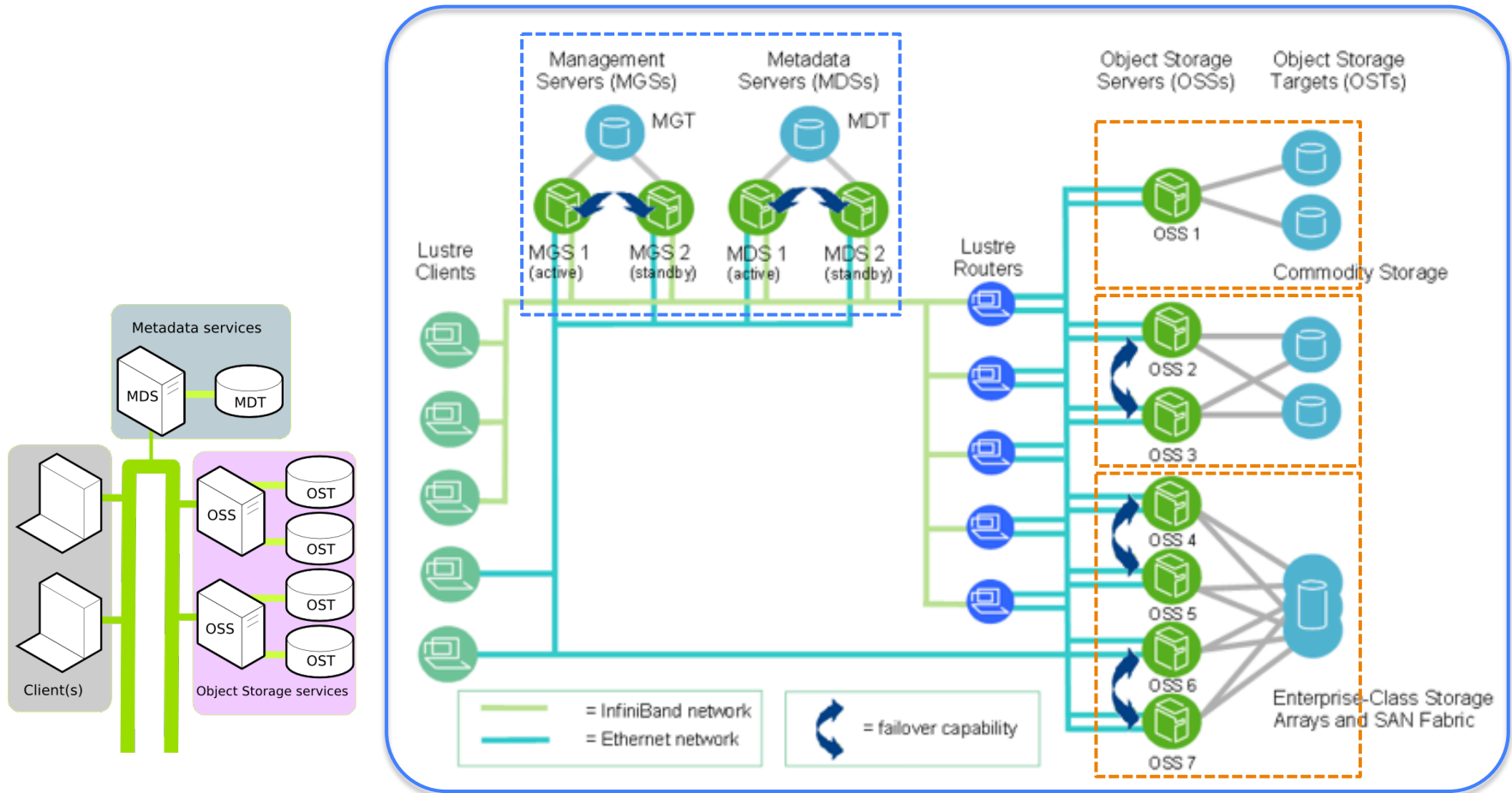
• Classification:

- *DFS and Clustered file systems:*
 - AFS, ClusterFS, **GlusterFS**, Google file system...
 - **HDFS** (Hadoop).
 - **Lustre**.
 - *Microsoft DFS*.
- *Cloud file systems:*
 - **Dropbox**, Oracle Cloud file system, SCFS...
- *Parallel file systems:*
 - **PVFS2**, orangeFS...
- *P2P file systems:*
 - Wuala, PAST...
- *Network file systems:*
 - NFS, pNFS...



Distributed file systems

Lustre

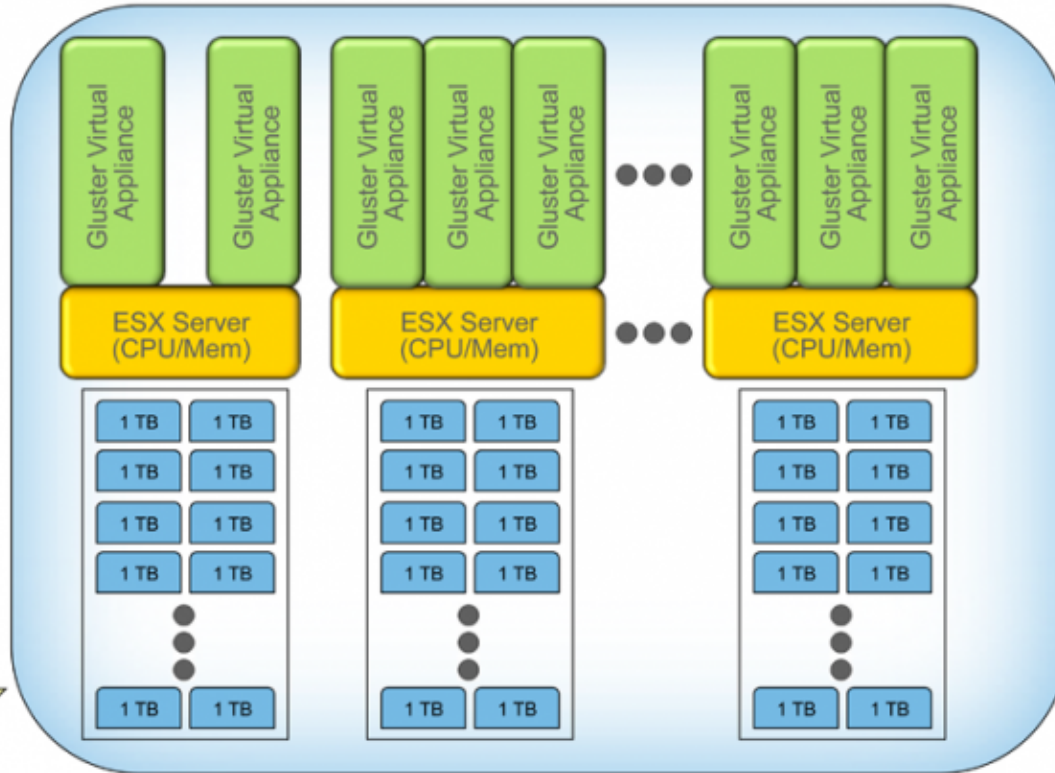


Distributed file systems

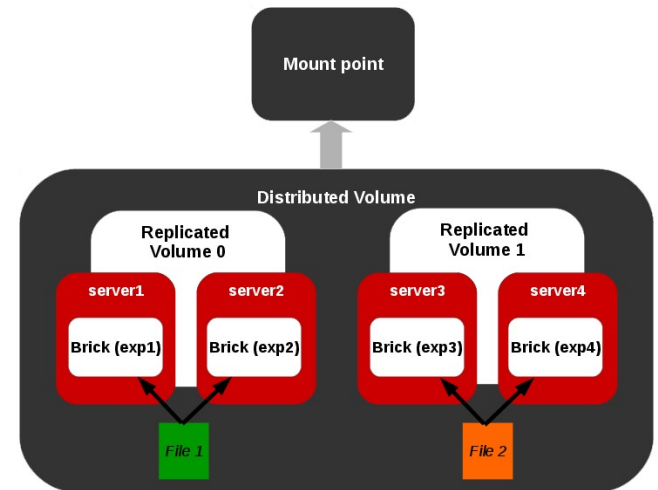
GlusterFS

Scale Out Performance & Availability →

Scale Out Capacity ↓



FUSE



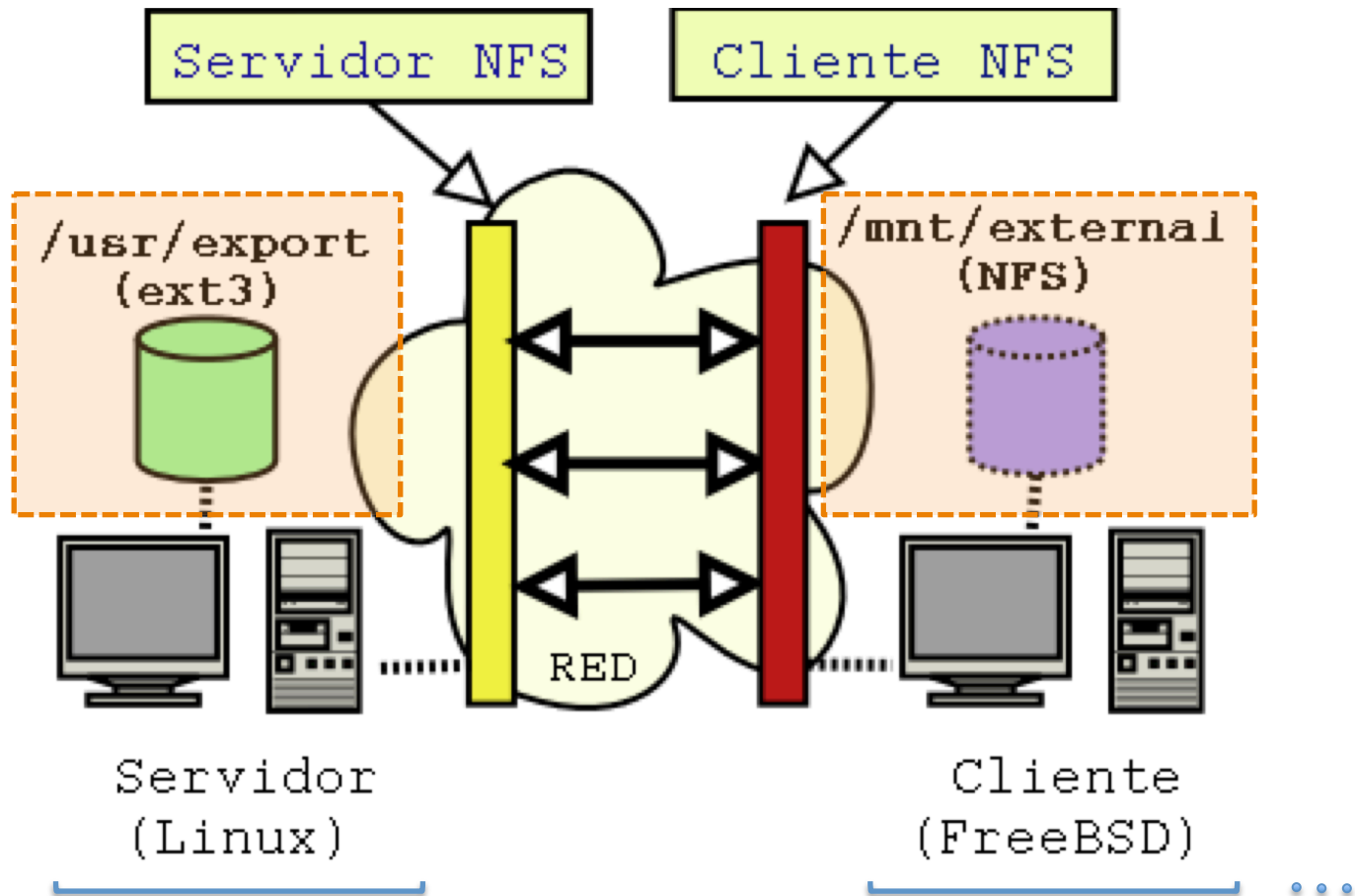
Source: <http://www.tecmint.com>.

Network file systems

- Main **gaps** with distributed systems:
 - Provide a client-server service to **export local file systems** through TCP/IP networks:
 - ext2/ext3/ext4, xfs, raiserfs...
 - Store data (files) on a unique server:
 - Implications:
 - Performance.
 - Management.
 - Security.
 - Fault tolerance.
- The most representative example of network file systems is **NFS**:
 - **NFS: Network File System.**

NFS: Network File System

NFS



NFS: Network File System

• Definition:

→ **Breaking down** the boundaries of local data:

- Set of rules (protocol) that enables **sharing** local file systems over TCP/IP networks in a transparent way.
- NFS is a *special (unusual)* distributed file system type.
- NFS is **built over other protocols and mechanisms**:
 - Uses the **Sun RPC mechanism** and Sun eXternal Data Representation (XDR) standard:
 - **RMI: the *object-oriented programming*.**
 - Defined as a set of *remote procedures*.
 - Protocol is ***stateless*** (initially):
 - **Each procedure call contains *all the information necessary to complete the call*.**
 - **Server maintains no “between call” information.**

NFS: Network File System

- **History and protocol development:**

- Developed by SUN Micro. in 1984:

- To be used on *diskless* hosts.
 - Along with protocols:
 - **DHCP, TFTP, BOOTP (?!?!).**

- Nowadays, NFS is available for every UNIX/Linux distribution:

- Including MS Windows systems.

- **Versions:**

- **NFSv1** (1984): used only for in-house experimental environments (not published).
 - **NFSv2** (1989): RFC 1094:
 - **Over UDP; Virtual File System interface.**
 - **Only allows the first 2 GB of a file to be read due to 32 bits limitations.**
 - **NFSv3** (1995): RFC 1813:
 - **Support for 64-bit file sizes and offsets; asynchronous writes on the server; file handles long file names.**
 - **And many more improvements.**
 - **NFSv4** (2000): RFC 3530:
 - **Includes significant security enhancements and...**
 - » Performance.
 - » Server maintains client state.
 - » Multi-Component Messages → Less Network traffic.
 - » Mandates strong security architecture.
 - » Elimination of 'side-car' protocols → No **rpc.statd** or In-kernel **lockd**.
 - » Only port 2049.

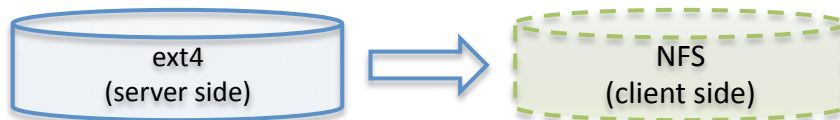
→ Compatible versions:

nfsv2 ↔ nfsv3 ↔ nfsv4.

NFS: The user's perspective

- From the **user's perspective**, there aren't any differences between a *local* file system and a *remote* file system:

- Local device (Ext4) → Remote device (NFS).



- **Transparency in usability:**

- On clients, data (files) is accessed through a directory (**mount point**):
 - Like the rest of local devices.
- Penalty on performance may even be minimal compared with local devices:
 - Disk and network technology.
 - Performance will be heavily dependent on:
 - **Number of client hosts accessing simultaneously:**
 - » (Very important issue!!!).
 - **Network technology and design (server side).**

- **Uniform vision of user files:**

- Users have files available from any client host in the network:
 - P.e. → \$HOME.
- Handy and easy.

NFS: The sysadmin's perspective

- **Centralized data:**
 - All data (files) housed on one server:
 - This can be good and bad at the same time ← “PoF”.
 - Make service **management** easier:
 - System managers happier.
 - Management tasks can be concentrated.
- Storage system **optimization**:
 - Regarding to **security** and management.
 - Disks → **RAID/LVM** devices...
 - Backup process.
- **Isolation**:
 - Disks are actually located on servers that are not accessed by users:
 - Dedicated (private) networks.
- **Fault tolerant** (*partially*):
 - **Advantage**:
 - Critical faults on clients are less “critical” faults.
 - Data (files) is not located on clients.
 - **Disadvantage**:
 - If the server goes “down” → Clients can’t access data (files).
 - Although you can deploy “**high ability**” measures (*Failover*).



NFSv4: Main features

- **Basics:**

- **Uniform namespaces:**
 - → 90% “Root space” (kernel).
- “In Kernel lock” and “mount” protocols have been integrated to NFSv4 *core*:
 - portmapper, rpc.mountd, rpc.lockd & rpc.statd → **KERNEL Linux**.
- **Statefulness & Sessions:**
 - *Read/write* operations use **state description**.
- Compound operations.
- **Caching** → Directory & File Delegations.
- *Sparse File Support*.
- Space Reservation.
- **Firewalls** (NAT) compatibility.
- Disk **quotas** (user/group) support.
- It enables exporting many FS through a simple root (*fsid = 0*).
- Files export is based on **virtual file system (VFS) abstraction** → Transparency.

- **Security:**

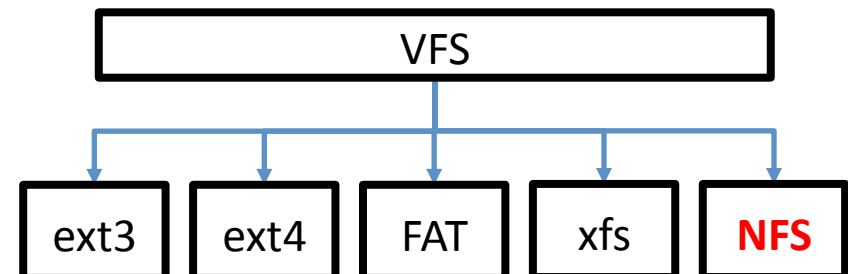
- Only port → **2049 (TCP)**.
- Kerberos support (GSS).
- **ACLs** (Access control lists) support.

- **Performance:**

- **Multiplatform** (client & sever side):
 - Good performance on *low bandwidth* networks.
- **Strongly dependent on:**
 - Disk and network technology.
 - **Number of clients** (*writes*).

- **Management:**

- Centralized service.
- No maintenance.



NFSv4: Main features

• Architecture:

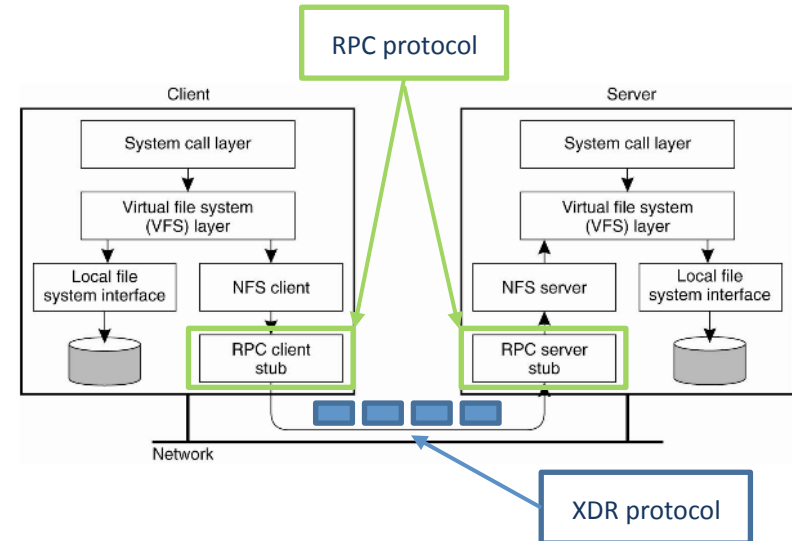
– Client/server architecture:

- Server:
 - **Current exports will work seamlessly:**
» `/etc/exports`.
- Clients:
 - **A mount configuration file.**

• Network *sharing* file service:

– Common process (services):

- `nfs`:
 - **Starts RPC processes needed to resolve NFS file requests.**
- `nfslock`:
 - **Optional service.**
 - **RPC process control to lock files requested by clients.**
 - **→ Unused on NFSv4 (deprecated) → Functionality integrated in Linux Kernel.**
- `portmap`:
 - **Linux RPC service: RPC management (server side).**
 - **It configures the RPC connections for each RPC request.**
 - **→ Unused on NFSv4 (deprecated) → Functionality integrated in Linux Kernel.**



NFSv4: Main features

- NFSv4 RPC daemons (UNIX process):

- Server side:

- `rpc.nfsd`:
 - NFS server process.
 - The main functionality is handled by the *nfsd* kernel module.
 - Implements the user level part of the NFS service: sockets and threads the kernel should listen to and use.
- `rpc.rquotad`:
 - Quota manager for “remote” users on NFS server.
 - It also allows setting of quotas on NFS mounted filesystem.
 - The daemon is started by NFS service on server side. Clients don't need these kinds of configurations.
- `rpc.idmapd` (both server and client side):
 - The daemon provides to server and clients the *upcalls* to translate the name (`user@dominio`) and UIDs/GIDs on the system in NFSv4.
- `rpc.svcgssd`:
 - The daemon provides to server the transport mechanisms to the Kerberos authentication process (NFSv4).
 - It enables *authentication* between server and clients (*host level*).

- Client side:

- `rpc.idmapd` (both server and client side):
 - The daemon provides to server and clients the *upcalls* to translate the name (`user@dominio`) and UIDs/GIDs on the system in NFSv4.
- `rpc.gssd`:
 - The daemon provides to clients the transport mechanisms to the Kerberos authentication process (NFSv4).
 - It enables *authentication* between server and clients.

NFSv4: Security features

• Preliminary considerations:

– /etc/exports file:

- Beware of **syntax errors** and **blanks**:

- It establishes **which** local file systems are exported, by **whom** and export **restrictions**:

» /home atc.unican.es(rw)

» /home atc.unican.es (rw)

Blanks

- Beware of **wildcard** and **meta-characters** (“*”).

- Beware of “**no_root_squash**” option:

- **Distinction between “root” user on server side and “root” user on client side.**
- **nfsnobody.**

– More options:

- **secure:**

- **This option requires that requests originate on an Internet port less than IPPORT_RESERVED:**
- » < 1024.

- **ro/rw:**

- **Read only; allows only read requests on the exported NFS volume.**
- **Read/Write; allows both read and write requests on the exported NFS volume.**

- **noaccess:**

- **It is used to deny access of PATHs in exported file systems:**
- » P.e.: export /home directory and deny access to /home/jherrero directory.

- **nohide:**

- **It makes an exported FS child “not hidden” if you mount its exported FS parent.**

- **crossmnt:**

- **It makes it possible for clients to move from the filesystem marked with *crossmnt* to exported filesystems mounted on it.**

NFSv4: Security features

- **Network level** security in NFS processes (*daemons*):

- **Services:**

- NFSv2/NFSv3:

- `portmap`:
 - » Dynamic port assignment.
- `mountd`:
 - » Remote file systems mount.
- `rquotad`:
 - » Disk quotas management on NFS exported systems (remote).

- NFSv4:

- `nfsd` (2049 TCP port), `rpc.rquotad` (944 TCP port).
- NFS service processes.

- **How?:**

- **TCP Wrappers** (`inetd/xinetd` – `tcpd`):

- `/etc/hosts.allow`:
 - » Host (network) list which are allowed to access.
- `/etc/hosts.deny`:
 - » Host (network) list of those NOT allowed to access and use such services.

- **IPtables:**

- **It enables us to configure a software TCP/IP firewall.**
- **We can take control of TCP/IP (input/output) access through the server.**

- **Policy (good practices):**

- **At first, DENY every host and ALLOW them on demand...**

NFSv4: Security features

- **User level** security in **NFS processes:**
 - mount and remote access**
 - Once the exported NFS file system is mounted in “**RW**” mode on client, the protection of each exported file will be depend on its UNIX permissions (**owner and access rights**):
 - If *two* users, from *two* different client hosts, share the same UID and access the same NFS directory, then both of them will be able to modify the files.
 - Also, any local “root” user could modify files owned by other local users by using the “**su**” command (no password).
 - **Recommendations:**
 - Use the “`root_squash`” option (*default*):
 - It changes the “root” ID of any NFS client host to “nobody” → Local “root” as regular user.
 - Use the next options:
 - **all_squash:**
 - » Every user as *nobody* user (UID).
 - **squash_uids and squash_gids:**
 - » A subset of users or groups UID(s) or GID(s) as *nobody* user (UID).
 - **anonuid and anongid:**
 - » Only one user or group as *nobody* user (UID).
 - Use a unique **users (UIDs) space** for NFS:
 - **Centralized information service: LDAP.**
 - Use authentication mechanisms → **Kerberos.**

NFSv4: Security features

- **File access level** security in **NFS clients**:

- **NFSv4 Access control lists: ACLs:**

- Useful for *privilege separation*.
- Provides *semantics* richer than UNIX mechanisms for NFS file access:
 - **Read/write/execute <> Owner/Group/Other.**

- **Support:**

- NFSv2 y NFSv3:
 - **ACLs POSIX.**
- NFSv4:
 - **ACL own mechanism integrated.**
- None of the filesystems which the linux server exports **support** NFSv4 ACLs:
 - **Read** <http://wiki.linux-nfs.org/wiki/index.php/ACLs>.

- NFSv4 ACLs are **“default-deny”**:

- If a permission is not explicitly granted by an Allow ACE, it is **denied**.

- **ACEs mechanism:**

ACEs: Access Control Entry

- **Types (4):**
 - **A (allow), D (deny), U (Audit), L (Alarm).**
- **Flags:**
 - **Group → g.**
 - **Inheritance → d, f, n, i.**
 - **Administrative → S, F.**
- **Principals:**
 - **The owner :: OWNER@.**
 - **A user :: [principal]@[domain].**
 - **The owner group :: GROUP@.**
 - **Through a privilege mask :: EVERYONE@.**
- **Permissions:**
 - **14 bits are user to access control.**

```
[access type]:[flags]:[principal]:[permissions]
[access type] → A (allow) / D (deny) ... L / U
[flags] → group, inheritance, and administrative
           g (group) / d (directory-inherit) / n (directory-inherit) / i ...
[principal] → username / group (flags)
[permissions] → NFSv4 ACLs

$ nfs4_setfacl -a "A:fdi:usel@localdomain:rwaDxtTcCy" file
```



NFSv4: Security features

| ID | Access |
|----------|---|
| r | read-data (files) / list-directory (directories). |
| w | write-data (files) / create-file (directories). |
| a | append-data (files) / create-subdirectory (directories). |
| x | execute (files) / change-directory (directories). |
| d | delete - delete the file/directory. |
| D | delete-child - remove a file or subdirectory from within the given directory. |
| t | read-attributes - read the attributes of the file/directory. |
| T | write-attributes - write the attributes of the file/directory. |
| n | read-named-attributes - read the named attributes of the file/directory. |
| N | write-named-attributes - write the named attributes of the file/directory. |
| c | read-ACL - read the file/directory NFSv4 ACL. |
| C | write-ACL - write the file/directory NFSv4 ACL. |
| o | write-owner - change ownership of the file/directory. |
| Y | synchronize - allow clients to use synchronous I/O with the server. |

NFSv4: Disk quotas

- **Data growth level security:**
disk quotas:

In the **same way** as local file systems.

- `rpc.rquotad` daemon.
- Controls the *growth* of the exported NFS file systems on client hosts:
 - Limits the amount of **disk space** that can be used:
 - **Disk size (bytes).**
 - **Number of files.**
 - For each **user** and/or **group**.
 - NFS manages *independently* one **disk quota system** for each exported file system.
- It enables **2 limits**:
 - *Hard*:
 - **It can never be exceeded.**
 - *Soft*:
 - **It can be exceeded for a "certain" time.**
- + grace period:
 - **Period of time during which users may violate their "soft" quota.**
 - **If it is exceeded, the soft quota becomes a hard quota.**
 - **This period is restarted when the quota is returned (soft):**
 - » We remove files.

NFSv4: Service installation

- **Previous:**

- **Recommendations:**

- Server must be physically safe.
- Use NFSv4 when the computational environment provides:
 - **Local DNS service.**
 - **Local NTP service:**
 - » Kerberos (auth. mechanism for NFSv4) is very sensitive to time sync.
 - **Local/remote KDC (kerberos) services:**
 - » If you want a very strong auth. mechanism.
- Use **RAID** (5,6) mechanisms in your disk devices (store backend):
 - **Hardware/software.**
 - **+ Security and + performance.**

- **Checking:**

- **NFS support in our kernel:**

- **We need NFS support (kernel):**
 - » Otherwise we should re-compile kernel.
- **Take a look at the Kernel config file:**
 - » /boot/config-3.2.*:

```
CONFIG_NFS_FS=m o y
CONFIG_NFS_V3=m o y
CONFIG_NFS_V4=m o y
CONFIG_NFSD=m o y
CONFIG_NFSD_V3=m o y
CONFIG_NFSD_V4=m o y
```

NFSv4: Service installation

- NFS service and tools installation:

| | | |
|--------------------|---|---|
| Server side | { | <pre>\$ apt-get update \$ apt-get install nfs-kernel-server nfs-common nfs4-acl-tools \$ apt-get install quota</pre> |
| Client side | { | <pre>\$ apt-get install nfs-common nfs4-acl-tools</pre> |

NFSv4: Service configuration

- **Previous** configuration (**server side**):

```
$ vi /etc/fstab
```

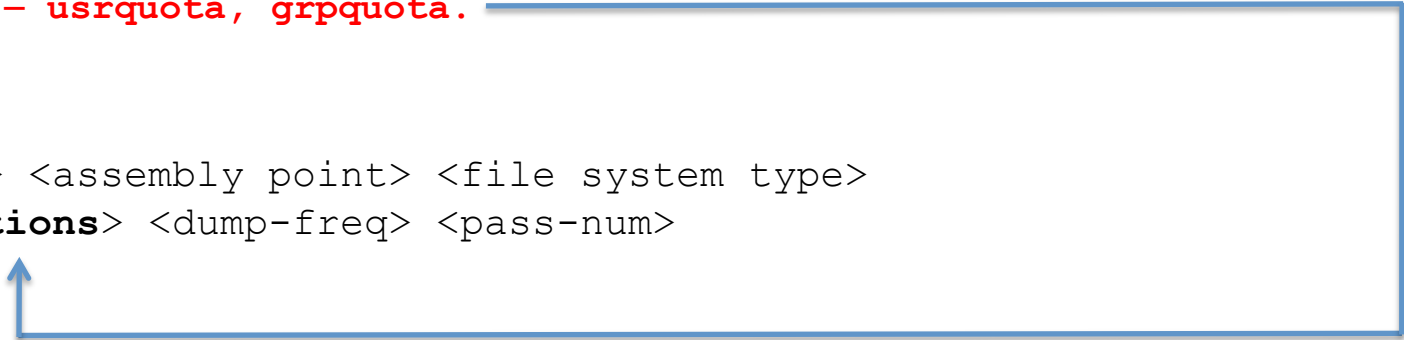
- **Features:**

– Previously, we must have the **local file systems** (*backend*) ready for their NFS export:

- **ext4, xfs, raiserfs...**
- The (NFS) **disk quota** must be enabled now:
 - **Over the local file systems.**
 - **Later, `rpc.quotad` daemon will be in charge of the client quota management.**
 - **`usrquota`, `grpquota`.**

- **Format:**

```
<device> <assembly point> <file system type>  
<options> <dump-freq> <pass-num>
```



NFSv4: Service configuration

- **Service** configuration (**server side**):

```
$ vi /etc/exports
```

- **Features:**

- NFSv4 needs to be able to **identify** each filesystem that it exports:

- NFS can define the **“root”** directory concept → *fsid = root* or *fsid = 0*:
 - **In terms of NFS exported file.**
 - **It enables exporting more than one local FS through only one “root” directory.**
- Other filesystems can be identified with a small integer.

- **Format:**

```
/directorio_a_exportar/ <Filtros de seguridad>([opciones])
```

NFSv4: Service configuration

- <security filters>:
 - hostname; That host is the only host allowed for using the resource.
 - ipaddr/mascara; Hosts (Subnet) allowed.
 - @GruposNIS → *Deprecated*.
 - gss/krb5x; KERBEROS security options:
 - **sys**: No crypt.
 - **krb5**: only kerberos 5 auth.
 - **krb5i**: krb5 and integrity (*checksum*).
 - **krb5p**: krb5i and privacy (top security).

- [options]:
 - **fsid** Identifier of exported file system. If *fsid = root* or *fsid = 0* then the FS is distinguished as the **root**.
 - **ro** read-only export.
 - **rw** read and write export.
 - **rw=hostname1,...** “rw” export mode for this list only (“ro” export mode for the rest).
 - **root_squash** Restrict the local root permissions on the remote disk:
 - “Root” users on clients are considered a regular user: “nobody”.
 - *Default*.
 - **no_root_squash** No restrictions for client “root” user... :
 - Dangerous!!!
 - **wdelay**: It enables delaying write operations over the NFS server. *Operation groups*:
 - Improves performance.
 - **sync/async writes**:
 - Worst performance (sync).
 - **acl**:
 - It enables the *access control lists*.

Take care using **wildcards** int (*,?).

NFSv4: Service check & start up

- **Checking** the devices exported by NFS:
(**server side**) → /etc/exports:
 - `$ exportfs -v`
- **Applying** the NFS configuration setting:
(**server side**) → /etc/exports:
 - `$ exportfs -ra.`
 - You can restart the NFS service:
 - `$ service nfs-kernel-server restart.`

NFSv4: Server (daemons) configuration

- **Server side** configuration:
(*main* configuration file)

```
$ vi /etc/default/nfs-kernel-server
```

– Options:

- `RPCNFSDCOUNT=<num process>`:
 - **Number of NFS server instances (process).**
- `RPCMOUNTDOPTS=""`:
 - **`rpc.mountd` daemon options.**
 - **If a firewall is running on server, you must fix the NFS port here:**
 - » → `--port xxxxx`.
 - **For more details:**
 - » `$man rpc.mountd`.
 - **It is not necessary for NFSv4.**
- `NEED_SVCGSSD=<yes/no>`:
 - **To start (or not) the `rpc.svcgssd` daemon.**
 - **It is necessary to integrate with Kerberos (NFSv4).**
- `RPCSVCGSSDOPTS=""`:
 - **`rpc.svcgssd` daemon options.**
 - **For more details:**
 - » `$man rpc.svcgssd`.

NFSv4: Server (daemons) configuration

- **Server side** configuration:

(*common* NFS configuration: **Server and client side**)

```
$ vi /etc/default/nfs-common
```

- **Options:**

- `NEED_STATD=<yes/no>`:
 - **To start (or not) `statd` daemon.**
 - **It is not necessary for NFSv4.**
- `STATDOPTS=""`:
 - **`rpc.statd` daemon run options.**
 - **For more details:**
 - » `$man rpc.statd.`
- `NEED_IDMAPD=<yes/no>`:
 - **To start (or not) `rpc.idmapd` daemon.**
 - **It is necessary to integrate with Kerberos (NFSv4).**
- `NEED_GSSD=<yes/no>`:
 - **To start (or not) `daemon rpc.gssd`.**
 - **It is necessary to integrate with Kerberos (NFSv4).**

} Client side too!!

NFSv4: Server (daemons) configuration

- **Server side** configuration:

(**rpc.idmapd** daemon)

```
$ vi /etc/idmapd.conf:
```

– **Options:**

- [General]:

– Domain = <Network domain>

Both Server and client side.



(**rpc.rquotad** daemon)

```
$ vi /etc/default/quota:
```

– **Options:**

- [General]:

– RPCRQUOTADOPTS="-p <n° PORT>"

NFSv4: Client side configuration

- **Client side** configuration:

```
$ vi /etc/default/nfs-common
```

- It uses the same options as the server...

```
$ vi /etc/fstab
```

- **System (core) config** for automatic permanent mounting (*on boot*):

- **Local** device.

- **Remote** devices (NFS...).

- When the system *boots*, the specified devices are mounted.
- When the system *goes down*, the specified devices are unmounted.

- For NFS, there will be another entry:

- *Similarly* to the local devices.

```
$ mount -a.
```

- **Syntax:**

```
servidor:/<directorio exportado> /<directorio local> nfs4 (opciones) 0 0
```

dump fck



NFSv4: Client side configuration

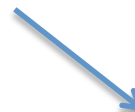
• Options:

```
server:/<exported directory> /<local directory> nfs4 (opciones) 0 0
```

- **rw**: read/write filesystem.
- **ro**: read-only filesystem. Remote NFS clients can't modify the filesystem.
- **hard**: applications using files stored on an NFS will always wait if the server goes down. User cannot terminate the process unless the option **intr** is set.
- **soft**: applications using files stored on an NFS will wait a specified time (using the **timeo** option) if the server goes down, and after that, will throw an error.
- **intr**: allows user interruption of processes waiting for a NFS request.
- **noexec**: disables execution of binaries or scripts on an NFS share.

Remote directory NOT *root*
(*fsid = 0*).

Local (*client side*) directory.



• Manual mounting:

```
$ mount -t nfs4 -o acl <nombre servidor>:<dir_s> <dir_d>
```

AutoFS: Dinamic mounting on demand

- When many hosts share an NFS data store, the `/etc/fstab` *static mechanism* can become a problem:
 - On a change in NFS config → Change every `/etc/fstab` file for each host:
 - Each one with different formats.
 - If NFS server goes “down”... **chaos is guaranteed.**
- **Automount** *simplifies* it on the client side and reduces the *workload* server:
 - On client side, NFS (remote) device is only mounted by user/apps **when necessary**:
 - Mounting **on demand**.
 - Allows *automounters* to specify alternative servers when the first one is “down”:
 - Fault tolerant.

Backup servers.



AutoFS: Installation and config

- **Installation** (“root”):

```
$ apt-get update
$ apt-get install autofs
```

- **Configuration:**

(Example for /home):

```
$ vi /etc/auto.master
  - Sets up in autofs where the config file is for each entry.
  - /home /etc/auto.mi_servidor --timeout=60.

$ vi /etc/auto.mi_servidor
  - * -fstype=nfs4,acl <server name>:<directory>/&.
```

 NFSv4 main options.

NFSv4: Checking and management

- **ACLs:**
 - **Getting the ACLs of a specific object:**

```
$ nfs4_getfacl <file|folder>
```
 - **Setting the ACLs of a specific object:**

```
$ nfs4_setfacl -a "A::user1@localdomain:rwDxtTcCy" <file|folder>
$ nfs4_editfacl < file|folder >
```

- **Disk quotas:**
 - **Monetarization (samples):**

```
$ repquota -a:
• Executed on server by "root".

$ quota -v:
• Executed on clients by "regular" users.
```
 - **Edition/setting of quotas:**

```
$ edquota <user|group>:
• Executed on server by "root".
```

- **Management (checking):**

```
$ nfsstat:
• List NFS statistics (server).

$ nfsiostat:
• Emulate iostat for NFS mount points using /proc/self/mountstats (clients).

$ rpcinfo -p:
• Report RPC information (server/clients).

$ showmount -a:
• Show mount information for an NFS server (server).

$ exportfs -r:
• Maintain table of exported NFS file systems (server).
```