NASA Lab 10 - Network File System (NFS)

2024/04/28 TA 杜冠勳

Outline

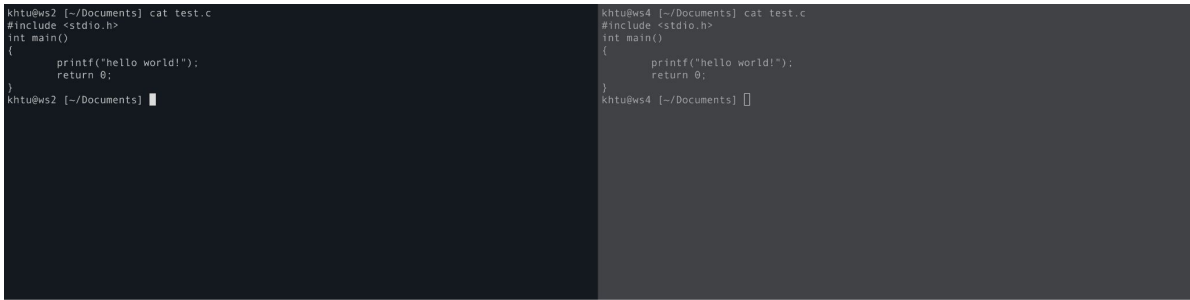
- What Is NFS?
- NFS in Real-World Scenarios
- How NFS Works - VFS ⇢ RPC ⇢ XDR
- Deep Dive Architecture: From Block Devices to nfsd
- Key Configuration of NFS
- LAB Time

What is NFS?

- NFS was **created by Sun Microsystems in the early 1980s** , as an open protocol for sharing files over a network.
- **Makes a remote folder look like a local one** – you can mount a remote folder onto your own computer, making it behave just like a local directory.
- No need to manually copy files and modify – you can directly read, write, and modify remote data.

NFS in Real-World Scenarios

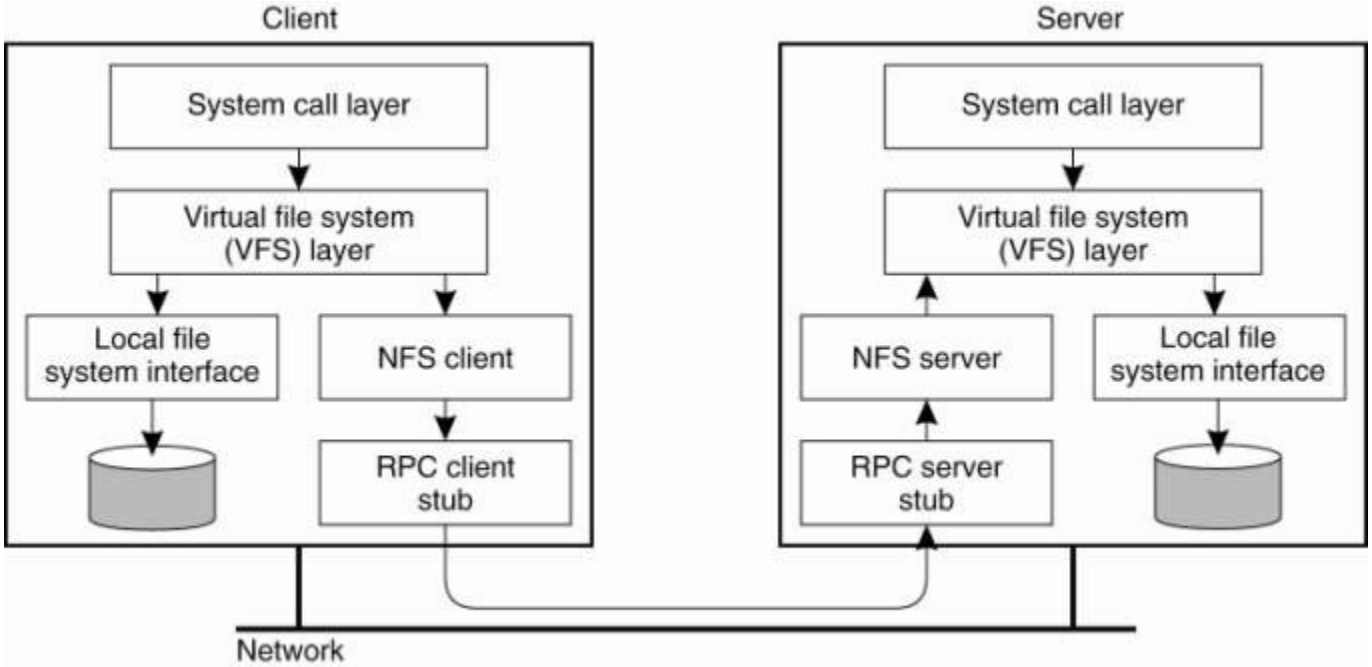
- In CSIE workstation, every workstation mounts the same home folder over NFS.
- Instant sync: NFS keeps all machines in step automatically.
- Whenever multiple machines need to see and edit the same files as if they were local — labs, HPC clusters, media studios — NFS is still the simplest, time-tested way to make it happen.



How NFS Works – VFS ⇢ RPC ⇢ XDR

Layers	Descriptions
Virtual File System (VFS)	VFS (Virtual File System) is an API layer inside the Linux kernel. It lets different file systems be accessed in the same way using standard APIs like `open()`, `read()`, and `write()`.
Remote Procedure Call (RPC)	RPC allows us to run programs or functions on one computer from another computer. RPC is like controlling a machine remotely, asking it to do something for you and then sending the result back to you.
External Data Representation (XDR)	XDR defines a standard format for sending data over the network. It ensures things like numbers, strings, and permissions are encoded in a consistent way, so that machines with different CPU architectures can understand each other's data.

How NFS Works - Workflow



Ref: https://csis.pace.edu/~marchese/CS865/Lectures/Chap11/Chapter11.htm

Low Level Structure - Deep Dive Architecture

- Block Devices
  a. Hard drives, SSDs, RAID sets, LVM volumes, or ZFS pools.
  b. Think of them as rows of numbered storage blocks; they know nothing about files.
- File System
  a. Layers structure onto those blocks (e.g., ext4, XFS).
  b. Tracks file metadata in inodes.
  c. Uses journaling to protect data after crashes or power loss.
  d. Enforces UID/GID permissions.

Low Level Structure - Deep Dive Architecture

- Virtual File System (VFS)
  a. A kernel "translation layer" that offers one set of calls—`open()`, `read()`, `write()` — for all file systems, local or remote. Apps never need to know what storage lies underneath.
- NFS Protocol
  a. The kernel `nfsd` service listens on **port 2049** .
  b. Remote clients send RPC messages like READ, WRITE, LOOKUP.
  c. `nfsd` translates each request into VFS calls, runs them on the server's local file system, then wraps the result back into an RPC reply.
  d. From the client's viewpoint, the remote folder behaves just like a local one.

Comparison - NFS v2, v3, v4

Version	What It Does	Key Improvements
NFS v2	Basic file sharing over the network	Simple and widely supported, but limited to 2GB files and uses UDP only
NFS v3	Adds better performance and larger file support	Supports files >2GB, uses TCP/UDP, adds async writes for speed
NFS v4	Secure, modern, and internet-ready	Built-in security (ACL…), works over a single port 2049, supports file locking and stateful protocol

Key Configuration of NFS

Option	Description
`rw` / `ro`	`rw` = the share is writable ‖ `ro` = read-only
`sync` / `async`	`sync` = write to disk before replying ‖ `async` = reply first, flush later
`no_subtree_check`	Skip extra checks on deep folder trees → faster exports
`root_squash` / `no_root_squash`	`root_squash` maps client UID 0 to nobody (blocks remote root) ‖ `no_root_squash` lets client root act as root
`fsid=0`	Marks this path as the pseudo-root required by NFS v4

Lab Time

Setup Two VMs

- After running following commands, you should see the VNC port for two VMs.
  - VM Account: `nasa`
  - VM Password: `nasa2025`

```
ssh <YOUR_STUDENT_ID>@nasaws[1-3].csie.ntu.edu.tw

mkdir /tmp2/<YOUR STUDENT ID> # If you haven't create one.

cp /tmp2/lab10/run-vm.sh /tmp2/<YOUR_STUDENT_ID>

cd /tmp2/<YOUR STUDENT ID>

chmod +x run-vm.sh

bash run-vm.sh <YOUR_STUDENT_ID> /tmp2/lab10/nasalab10.qcow2

# You can use tmux ls to see session name
tmux a -t <SESSION_NAME>
```

Set Network for VMs

```
# On your machine, do port forwarding
ssh -L <Local Port 1>:localhost:<Remote VM 1 VNC Port>
<username>@nasaws[1-3].csie.ntu.edu.tw

ssh -L <Local Port 2>:localhost:<Remote VM 2 VNC Port>
<username>@nasaws[1-3].csie.ntu.edu.tw

# using VNC Viewer for both VMs
sudo ip link set ens3 up
sudo ip link set ens4 up

sudo dhclient -v ens3
sudo dhclient -v ens4

ip a # VM1 and VM2 will use ens4's IP to connect

ping <another_VM_ens4_IP>

# Check Network Connection of VM
ping 8.8.8.8
```

(Optional) Use SSH to Login

```
echo '--- Launching VM1 (ID: b10505044-1) ---'
[khtu@nasa-ws4 lab10]$ echo '--- Launching VM1 (ID: b10505044-1) ---'
--- Launching VM1 (ID: b10505044-1) ---
[khtu@nasa-ws4 lab10]$ echo 'Base Image: /tmp2/lab10/nasalab10.qcow2'
Base Image: /tmp2/lab10/nasalab10.qcow2
[khtu@nasa-ws4 lab10]$ echo 'Overlay Disk: /tmp2/khtu/b10505044_nasalab10/vm1_overlay.qcow2'
Overlay Disk: /tmp2/khtu/b10505044_nasalab10/vm1_overlay.qcow2
[khtu@nasa-ws4 lab10]$ echo 'Host SSH Forward Port: 61070'
Host SSH Forward Port: 61070
[khtu@nasa-ws4 lab10]$ echo 'Host VNC Port: 51949 (Connect via 127.0.0.1)'
Host VNC Port: 51949 (Connect via 127.0.0.1)
```

```
echo '--- Launching VM2 (ID: b10505044-2) ---'
[khtu@nasa-ws4 lab10]$ echo '--- Launching VM2 (ID: b10505044-2) ---'
--- Launching VM2 (ID: b10505044-2) ---
[khtu@nasa-ws4 lab10]$ echo 'Base Image: /tmp2/lab10/nasalab10.qcow2'
Base Image: /tmp2/lab10/nasalab10.qcow2
[khtu@nasa-ws4 lab10]$ echo 'Overlay Disk: /tmp2/khtu/b10505044_nasalab10/vm2_overlay.qcow2'
Overlay Disk: /tmp2/khtu/b10505044_nasalab10/vm2_overlay.qcow2
[khtu@nasa-ws4 lab10]$ echo 'Host SSH Forward Port: 51936'
Host SSH Forward Port: 51936
[khtu@nasa-ws4 lab10]$ echo 'Host VNC Port: 58753 (Connect via 127.0.0.1)'
Host VNC Port: 58753 (Connect via 127.0.0.1)
```

```
# On nasa workstation
ssh -p <SSH Forward Port1> nasa@127.0.0.1 # VM 1

ssh -p <SSH Forward Port2> nasa@127.0.0.1 # VM 2
```

On Server VM

```
sudo apt update && sudo apt install nfs-kernel-server

sudo mkdir -p /srv/nfs/share

# Use sudo to modify /etc/exports, remember to replace <CLIENT_IP> with your own
CLIENT VM IP
/srv/nfs        <CLIENT_IP>(rw,sync,fsid=0,crossmnt,no_subtree_check)
/srv/nfs/share  <CLIENT_IP>(rw,sync,no_subtree_check)

sudo chown 1000:1000 /srv/nfs/share # So that user “nasa” can write
sudo chmod 755       /srv/nfs/share
sudo exportfs -arv

sudo systemctl enable --now nfs-server
```

On Client VM

```
sudo apt install nfs-common

sudo mkdir -p /mnt

sudo mount -t nfs <SERVER_IP>:/srv/nfs/share /mnt
```

Submission

- Please submit commands screenshots in a single PDF on NTU COOL.
  - On Server
    - `cat /etc/exports`
    - `sudo exportfs -v`
  - On Client
    - `mount`
    - `df -h`