LS Lab Assignment:
VM Migration∗

Credits to ARNO to help us solve many problems

VM Migration Basics

Question 1(a). Describe the differences between cold (or off-line) migration and live migration.

Answer:

Xen can migrate virtual machines between different servers, running Xen. There are two types of migration the offline migration and the live migration:

- **Cold Migration:**

The cold migration is a manual process that approximates the flow of live migration. “Migration begins by saving the domain. The administrator manually moves the save file and the domain’s underlying storage over to the new machine and restores the domain state. Because the underlying block device is moved over manually, there’s no need to have the same filesystem accessible from both machines, as would be necessary for live migration.” It matters in transporting the content of the Xen virtual disk. However, when performing the cold migration we must ensure that it copies the disk in such a way that is bit-forbit the same and has the same path on both physical machines.

- **Live Migration:**

The live migration is is the ability to move a domain from one physical machine to another transparently, that is, imperceptibly to the outside world. “As with cold migration, live migration transfers the domain’s configuration as part of its state; it doesn’t require the administrator to manually copy over a config file. Manual copying is, in fact, not required at all.” However, Live migration has some prerequisites. It relies on the domain’s storage being accessible from both machines and on the machines being on the same subnet. The last thing is because the copy phase occurs automatically over the network, the machines must run a network service.

Answer this question from the following sources:

Sources:

1- https://wiki.xenproject.org/wiki/Migration, 2-
http://wiki.prgmr.com/mediawiki/index.php/Chapter_9:_Xen_Migration

Question 1(b). What mechanism makes live migration almost instantaneous under light load?.

Answer:

Two techniques are involved to make this happen:

- **Pre-copy memory migration:**

1. Warm-up phase:

In pre-copy memory migration, the Hypervisor typically copies all the memory pages from source to destination while the VM is still running on the source. If some memory pages change "dirty" during this process, they will be re-copied until the rate of re-copied pages is not less than page dirtying rate.

1. Stop-and-copy phase:

After the warm-up phase, the VM will be stopped on the original host, the remaining dirty pages will be copied to the destination, and the VM will be resumed on the destination host.

- **Post-copy memory migration:**

Post-copy VM migration is initiated by suspending the VM at the source. With the VM suspended, a minimal subset of the execution state of the VM (CPU state, registers and, optionally, non-pageable memory) is transferred to the target. The VM is then resumed at the target. Concurrently, the source actively pushes the remaining memory pages of the VM to the target - an activity known as pre-paging. At the target, if the VM tries to access a page that has not yet been transferred, it generates a page-fault. These faults, known as network faults, are trapped at the target and redirected to the source, which responds with the faulted page.

Sources:

1- https://en.wikipedia.org/wiki/Live_migration

Question 2. What are the technical requirements to be able to coldly migrate VMs, and why?.

Answer:

- Both the source host and destination host must be running Xen and the xend daemon.
- The destination host must have enough disk space, memory capacity, and resources to run the domain after the migration.
- The source host and destination host machines must have the same architecture and virtualization extensions. For example, if the source host is running on x86-64 architecture with extensions, then you must ensure that the destination host does the same. This is stipulated so that you don't run into any mismatches in the instruction sets used by the kernel and the user libraries.

Source:

1-
https://www.packtpub.com/mapt/book/virtualization_and_cloud/9781847192486/9/ch09lvl1sec01/migration-requirements

Received feedback on this question **“You can migrate to different hypervisors.”**

Fix:

The file system wise “the underlying block device is moved over manually, there’s no need to have the same filesystem accessible from both machines, as would be necessary for live migration. All that matters is transporting the content of the Xen virtual disk.”

The configuration file wise “ensure that it copies the disk in such a way that is bit-forbit the same and has the same path on both physical machines. In particular, do not mount the domU filesystem on machine A and copy its files over to the new domU filesystem on machine B. This will cause the VM to crash upon restoration.”

Source:

1- http://wiki.prgmr.com/mediawiki/index.php/Chapter_9:_Xen_Migration

Question 3. What are the technical requirements to be able to live migrate VMs, and why?.

Answer:

- Both the source and destination hosts must have access to the root filesystem (and swap if specified) via the same path name. For example if the root filesystem is contained in a disk image with a path of /xen/xenguest.img then that image file must also be accessible at the same location on the target host. This is most commonly achieved by placing the image files on a file server such that it can be mounted via NFS.
- The target host must have sufficient memory to accommodate the migrated guest domain.
- Both systems must be running compatible processors.
- The source and destination machines must reside on the same subnet.
- The two systems need to be running compatible versions of Xen.
- Firewall settings (and SELinux if enabled) must be configured to permit communication between the source and destination hosts.
- Both systems must be configured to allow migration of virtual machines.

Sources:

1- https://support.citrix.com/article/CTX115813, 2-
http://www.virtuatopia.com/index.php/Migrating_Xen_domainU_Guests_Between_Host_Systems

Question 4. Form a group of two and discuss how you are going to migrate VMs to each other's hypervisor. Set up your systems so you can do both cold and live migrations. Describe your setup in your logs. Hint: Do not use LVM. Remember your eno2..

Answer:

I will work with Lennart

Cold Migration:

First, lets list the VMs:

```
kotaiba@bristol:~$ sudo !!
sudo xl list
Name                                        ID   Mem VCPUs  State    Time(s)
Domain-0                                     0  7013     4     r-----
399.9
Guest-01                                     6  1024     2     -b----
704.4
```

Save a file:

```
kotaiba@bristol:~$ sudo xl save 6 cold_bristol
Saving to cold_bristol new xl format (info 0x3/0x0/1506)
xc: info: Saving domain 6, type x86 PV
xc: Frames: 262144/262144  100%
xc: End of stream: 0/0    0%
```

List now:

```
kotaiba@bristol:~$ sudo xl list
Name                                        ID   Mem VCPUs  State    Time(s)
Domain-0                                     0  7013     4     r-----
408.3
```

Now, move the file to lennart using scp:

```
kotaiba@bristol:~$ scp cold_bristol
```

```
lennart@oxford.studlab.os3.nl:/home/lennart
The authenticity of host 'oxford.studlab.os3.nl (145.100.104.172)' can't be
established.
ECDSA key fingerprint is SHA256:8kkg/O/VA7YJqA8Ciz66x69/pamNMUdVWO/+kolcLZg.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'oxford.studlab.os3.nl,145.100.104.172' (ECDSA)
to the list of known hosts.
lennart@oxford.studlab.os3.nl's password:
cold_bristol                                                       100%
1026MB 102.6MB/s   00:10
```

Since Lennart also did the previous steps with me, now I have his file so I can easily restore it :

```
kotaiba@bristol:~$ sudo xl restore cold_oxford
Loading new save file cold_oxford (new xl fmt info 0x3/0x0/1501)
 Savefile contains xl domain config in JSON format
Parsing config from <saved>
xc: info: Found x86 PV domain from Xen 4.6
xc: info: Restoring domain
xc: info: Restore successful
xc: info: XenStore: mfn 0x194155, dom 0, evt 1
xc: info: Console: mfn 0x194154, dom 0, evt 2
```

List:

```
kotaiba@bristol:~$ sudo xl list
Name                                        ID   Mem VCPUs  State    Time(s)
Domain-0                                     0  7013     4     r-----
409.6
Guest-01                                     7  1024     2     -b----
0.0
```

now, when we try to login we use Lennart password:

```
kotaiba@bristol:~$ sudo xl consol Guest-01

Ubuntu 16.04 LTS Guest-01 hvc0

Guest-01 login: root
Password:
Last login: Thu Nov  2 12:29:57 UTC 2017 on hvc0
Welcome to Ubuntu 16.04 LTS (GNU/Linux 4.4.0-98-generic x86_64)

 * Documentation:  https://help.ubuntu.com/
root@Guest-01:~# exit
logout

Ubuntu 16.04 LTS Guest-01 hvc0

Guest-01 login:
```

After the Cold migration now I have Lennart Vm instead of mine, so I will delete it and put my Guest-01 again so we can get fresh start:

```
kotaiba@bristol:~$ sudo xl destroy 7
kotaiba@bristol:~$ sudo xl list
Name                                        ID   Mem VCPUs  State    Time(s)
Domain-0                                     0  5989     4     r-----     
441.0
kotaiba@bristol:~$ sudo xl restore cold_bristol
Loading new save file cold_bristol (new xl fmt info 0x3/0x0/1506)
 Savefile contains xl domain config in JSON format
Parsing config from <saved>
xc: info: Found x86 PV domain from Xen 4.6
xc: info: Restoring domain
xc: info: Restore successful
xc: info: XenStore: mfn 0x203170, dom 0, evt 1
xc: info: Console: mfn 0x20316f, dom 0, evt 2
```

Live Migration:

In order to make a live migration, we connected our both servers with UTp cable then we need to specify a shared storage between me and Lennart, although we found a really easy way to do that but Mick and Jab suggested to NFS (for future stuff):

Now, we will set up an NFS mount on Ubuntu 16.04:

Since, Lennart configure and act as NFS server to me. We can access the NSF together:

What I did to do that:

I installed first:

```
kotaiba@bristol:~$ sudo apt-get install nfs-kernel-server nfs-common
```

After that I created the same directory as Lennart did:

```
kotaiba@bristol:/$ sudo mkdir -p /nfs/home/lennart/oxford_nfs
```

and I mount it to his file at his serve IP:

```
kotaiba@bristol:/$ sudo mount 145.100.104.172:/home/lennart/oxford_nfs
/nfs/home/lennart/oxford_nfs
```

check:

```
kotaiba@bristol:/$ df -h
Filesystem                                Size  Used Avail Use% Mounted on
udev                                      3.8G     0  3.8G   0% /dev
tmpfs                                     769M  8.7M  760M   2% /run
/dev/sda1                                 213G  4.5G  197G   3% /
tmpfs                                     3.8G     0  3.8G   0% /dev/shm
tmpfs                                      5.0M     0  5.0M   0% /run/lock
tmpfs                                      3.8G     0  3.8G   0%
/sys/fs/cgroup
145.100.104.172:/home/lennart/oxford_nfs  213G  3.7G  198G   2%
/nfs/home/lennart/oxford_nfs

kotaiba@bristol:/nfs/home/lennart/oxford_nfs$ ll
total 8
drwxr-xr-x 2 nobody nogroup 4096 Nov  2 15:02 ./
drwxr-xr-x 3 root   root    4096 Nov  2 15:05 ../
-rw-r--r-- 1 root   root       0 Nov  2 15:02 hello_kotaiba
```

So, now we have shared NFS.

Now, let's proceed in the live migration:

Enable the live migration server, adding the following /etc/xend-config.sxp:

```
(xend-relocation-server yes)
(xend-relocation-address 192.168.0.2)
(xend-relocation-hosts-allow '^localhost$' '^oxford.studlab.os3.nl$')
```

STOP HERE AND LET'S START AGAIN

Our Ips:

- **Bristol eno2 IP: 192.168.0.2**
- **Oxford “Lennart” eno2 IP: 192.168.0.5**

After working on this and trying to fix stuff, unfortunately nothing works well. So we started again with another stuff:

Lennart created a new VM called PubMigr-01 and also new NFS shared (for the private subnet that we share) and I will use to mount from him and we both will use PubMigr-01 to test the live migration.

In order to fix the previous stuff:

First, I mount the NFS folder, I created this /home/xen/domains/PrivMigr-01 same at Lennart and I mount it to his IP:

```
kotaiba@bristol:/home/xen/domains/PrivMigr-01$ sudo mount
192.168.0.5:/home/xen/domains/PrivMigr-01 /home/xen/domains/PrivMigr-01
```

At this step, we have now shared folder on our private Subnets.

After that I change my /etc/xen/xend-config.sxp file, and set the following configuration:

```
(xend-relocation-server yes)
```

```
(xend-relocation-port 8002)
(xend-relocation-address '')
(xend-relocation-hosts-allow '')
```

Now, we assumed that everything is works, but when we tried to do the migration we got a problem with the ssh root user.

so I changed my ssh configuration file in order to accept root login:

```
#changed this:
#PermitRootLogin prohibit-password
PermitRootLogin yes
```

then, restart:

```
kotaiba@bristol:/etc/ssh$ sudo service ssh restart
kotaiba@bristol:/etc/ssh$ sudo service sshd restart
```

Now, everything is works, lennart migrate this vm to me:

Before migration:

```
kotaiba@bristol:/home/xen/domains/PrivMigr-01$ sudo xl list
Name                                        ID   Mem VCPUs  State   Time(s)
Domain-0                                     0  5989     4     r-----
1259.6
Guest-01                                     9  1024     2     -b----
311.1
```

After migration (loot at his wiki it said (successful migration, I entered my password there)

```
kotaiba@bristol:~$ sudo xl list
Name                                        ID   Mem VCPUs  State   Time(s)
Domain-0                                     0  5989     4     r-----
1315.4
Guest-01                                     9  1024     2     -b----
316.4
PrivMigr-01                                 10  1024     2     -b----
0.1
```

Let's test if its works:

```
kotaiba@bristol:~$ sudo xl consol PrivMigr-01
root@PrivMigr-01:~#
root@PrivMigr-01:~#
root@PrivMigr-01:~#
```

It works :D.

What I did, is that I migrated the same vm “PrivMigr-01” to him again from my server:

```
kotaiba@bristol:~$ sudo xl migrate --live 10 192.168.0.5
root@192.168.0.5's password:
migration target: Ready to receive domain.
Saving to migration stream new xl format (info 0x3/0x0/1548)
Loading new save file <incoming migration stream> (new xl fmt info
0x3/0x0/1548)
 Savefile contains xl domain config in JSON format
Parsing config from <saved>
xc: info: Saving domain 10, type x86 PV
xc: info: Found x86 PV domain from Xen 4.6
xc: info: Restoring domain
xc: info: Restore successful
xc: info: XenStore: mfn 0x1f3e3b, dom 0, evt 1
xc: info: Console: mfn 0x1f3e3a, dom 0, evt 2
migration target: Transfer complete, requesting permission to start domain.
migration sender: Target has acknowledged transfer.
migration sender: Giving target permission to start.
migration target: Got permission, starting domain.
migration target: Domain started successsfully.
migration sender: Target reports successful startup.
Migration successful.
```

List again:

```
kotaiba@bristol:~$ sudo xl list
Name                                        ID   Mem VCPUs  State    Time(s)
Domain-0                                     0  5989     4     r-----
1342.1
Guest-01                                     9  1024     2     -b----
320.2
```

Sources:

1- http://wiki.prgmr.com/mediawiki/index.php/Chapter_9:_Xen_Migration, 2- https://www.digitalocean.com/community/tutorials/how-to-set-up-an-nfs-mount-on-ubuntu-16-04

Question 5. Together think of a definition of the downtime of a VM, and how to best measure that downtime. Write down your definition and measurement method(s) for both.

Answer:

- **(a) Cold migration:** and **(b) Live migration:**

The downtime of a VM, is the amount of time that the specific VM needs to get back to it's normal operation after the migration process. In other word if I want to migrate my VM, how much time it takes to back to back to its normal operation and functions. For these two types of migration downtime we used the network layer measurement (which is is ping the VM and see the time that it will be unreachable when we apply the migration) “Operational” —> “Not operational” —> “Operational again”, we missured the “not operational time in seconds”. However, the downtime for cold and live migration differs thats why we adapted the ping interval depends on each one.

Cold migration = ping interval 0.20, Live migration ping = 0.000001.

Question 6. Perform cold migrations with your partner, and measure the downtime. Do not take just a single measurement! Compute mean and median of your chosen metric.

Answer:

Please, look at question 4 for details, but here we will only do the measurement.

Please notice that we document the experiment on Lennart machine so I will put the code from the file (we have both same thing)

From my machine, I restored the VM and move it to our shared folder so Lennart can restore it:

```
root@bristol:~# xl list
Name                                        ID   Mem VCPUs  State    Time(s)
Domain-0                                     0  5989     4     r-----
185.4
Guest-01                                     1  1024     2     -b----
3.3
PrivMigr-01                                  4  1024     2     -b----
41.8
root@bristol:~# xl save 4 cold_test
Saving to cold_test new xl format (info 0x3/0x0/1548)
xc: info: Saving domain 4, type x86 PV
xc: Frames: 262144/262144  100%
xc: End of stream: 0/0    0%
root@bristol:~#  mv cold_test /home/xen/domains/PrivMigr-01/
```

Now, We ping the vm with time interval 0.20 second, then we will do the migration process, after that we will count the number of “Unreachable packet” then time it with 0.25 so we can get the exact time.

```
root@oxford:/home/xen/domains/PrivMigr-01# xl restore cold_test
```

```
Loading new save file cold_test (new xl fmt info 0x3/0x0/1548)
 Savefile contains xl domain config in JSON format
Parsing config from <saved>
xc: info: Found x86 PV domain from Xen 4.6
xc: info: Restoring domain
xc: info: Restore successful
xc: info: XenStore: mfn 0x21b52b, dom 0, evt 1
xc: info: Console: mfn 0x21b52a, dom 0, evt 2

root@oxford:/home/xen/domains/PrivMigr-01# xl list
Name                                        ID   Mem VCPUs  State    Time(s)
Domain-0                                     0  3940     4     r-----
2664.8
PrivMigr-01                                 84  1024     2     -b----
0.0
```

We did that 3 times, pinged with interval of 0.20 and append the result to a text file. Then we count the unreachable and times it with 0.20 so we got the following down time seconds: 42 seconds, the second time 36.40, and the third time it was 35.60. Calculating the median = 36.40 seconds and the mean is 38 seconds.

Question 7. Perform live migrations with your partner, and measure the downtime. Compute mean and median of your chosen metric.

Answer:

As we did above, but we will do it now with the live migration with a ping interval 0.000001:

```
root@oxford:/etc# xl list
Name                                        ID   Mem VCPUs  State    Time(s)
Domain-0                                     0  3940     4     r-----
1817.2
PrivMigr-01                                 83  1024     2     -b----
13.9

root@oxford:/etc# xl migrate --live 83 192.168.0.11
root@192.168.0.11's password:
Saving to migration stream new xl format (info 0x3/0x0/1548)
migration target: Ready to receive domain.
Loading new save file <incoming migration stream> (new xl fmt info
0x3/0x0/1548)
 Savefile contains xl domain config in JSON format
Parsing config from <saved>
xc: info: Saving domain 83, type x86 PV
xc: info: Found x86 PV domain from Xen 4.6
xc: info: Restoring domain
xc: info: Restore successful
```

```
xc: info: XenStore: mfn 0x1ab63f, dom 0, evt 1
xc: info: Console: mfn 0x1ab63e, dom 0, evt 2
migration target: Transfer complete, requesting permission to start domain.
migration sender: Target has acknowledged transfer.
migration sender: Giving target permission to start.
migration target: Got permission, starting domain.
migration target: Domain started successsfully.
migration sender: Target reports successful startup.
Migration successful.
</code.

Again, we run ping command with 0.000001 time interval and calculate
"unreachable" and times it with "number of unreachable packets * 0.000001 =
downtime",

<code>
root@oxford:~# cat test2.txt | grep Unreachable
```

However, there was no one single microsecond downtime here. Since we use –live option the transfer is only done when both machines are ready. Which results in our experiment to downtime of 0 seconds using the live migration.

Performance

Question 8. What are the most important differences between NFS and SMB? Explain in approximately 200 words.

Answer:

NFS: Network File System. SMB: Server Message Block.

NFS and SMB are two protocols that provide the functionality to create a shared storage between machines.

- The main difference in these two protocols is that SMB(SAMBA implementation of SMB) is originally developed for Windows, however it also works on Linux. NFS is originally developed for Unix systems and it functions on linux (However, now Windows Server supports it natively).
- SMB uses Windows-style access control lists, whereas NFS uses Unix-style file permissions (User ID owner, Group ID owner, and read/write/execute permissions).
- SMB is an insecure protocol. All data transferred is not encrypted as it gets sent over the network. Also NFS is insecure just like SMB for the same exact reason (unencrypted transfer). However, NFS version 4 (aka NFSv4) can be configured to use encryption.
- Since I answered this question after I did the experiments bellow, I notice that NFS is faster than SMB on Linux servers, because we did exact same expirments for both and the result of the

experiment is that NFS is faster for copying large files and a huge number of small files.

I got this image from the source[1] that simulate what we did in question 10, and compare NFS and SMB Performance.

NFS vs SMB – Benchmark for Write Operations

Source:

1- https://ferhatakgun.com/network-share-performance-differences-between-nfs-smb/, 2- https://www.reddit.com/r/homelab/comments/2fawvq/eli5_what_is_the_difference_between_smb_cifs_and/

Question 9. Together with your partner, design an experiment to compare the performance of NFS and SMB as VM shared storage. Distinguish between raw I/O performance and the performance under a realistic workload. E.g. what if the VM was running an Apache Web server? Discuss the design with a lab teacher.

Answer:

Me and Lennart want to do two different tests for our VM shared storage:

Test 1- In order to test the raw I/O performance, we will create a very large file and copy it into the shared storage.

Test 2- in order to simulate the realistic workload we will create a lot of small files and copy it into the shared storage, because as the example you mentioned in the question the Apache web server usually handle a lot of small files so in our test we will simulate that.

Question 10. Configure both NFS and SMB on your systems. Perform the experiment and show the results in your log. Try to explain any remarkable differences. Hint: root is a nobody when it comes to NFS.

Answer:

Again, I'm working with Lennart on the experiment so the the steps that we did on his machine I will put it in my wiki

For our experiments, we will create one file share for NFS and one file share for SMB:

```
/home/nfs_exp
/home/smb_exp
```

I will be the SMB server and the NFS client.

Install samba and configure as the following:

```
sudo apt-get install samba

Then set password:

sudo smbpasswd -a <user_name>

Create a directory:

root@bristol:~# mkdir /home/kotaiba/
```

Now, we edit /etc/samba/smb.conf file add the follwoing:

```
[smb_exp]
path = /home/smb_exp
valid users = kotaiba lennart
read only = no
```

Restart it:

```
sudo service smbd restart
```

Now, Lennart will mount it, and also can connect to it using:

```
 smbclient //192.168.0.11/smb_exp -U kotaiba
```

Test, it:

```
root@bristol:/home/smb_exp# ll
total 12
drwxrwxrwx 2 root    root    4096 Nov  3 18:06 ./
drwxr-xr-x 5 root    root    4096 Nov  3 17:50 ../
-rwxrwxrwx 1 root    root       5 Nov  3 18:06 hello*
-rwxrwxrwx 1 root    root       0 Nov  3 17:56 hello-lennart-can-we-go-to-
the-weekend-?*
-rwxrwxrwx 1 kotaiba kotaiba    0 Nov  3 18:04 henk*
```

as we see we both are able to create users and do everything.

Now, For the NFS part I will mount it (Since Lennart already did everything):

```
kotaiba@bristol:/home$ sudo mount 192.168.0.10:/home/nfs_exp /home/nfs_exp/
```

Now, let's do the Tests:

Test 1- I/O performance (Large file)

First, Let's create a 1 Gb file:

```
root@oxford:/home# dd if=/dev/zero of=1GB.out bs=1MB count=1000
1000+0 records in
1000+0 records out
1000000000 bytes (1.0 GB, 954 MiB) copied, 2.34724 s, 426 MB/s
```

- **NFS test**

Copy the files into the /home/nfs_exp (NFS share file):

```
root@oxford:/home# time cp 1GB.out ./nfs_exp/

real    0m2.395s
user    0m0.024s
sys 0m1.224s
```

Now, From my machine I will copy it to /dev/null:

```
root@bristol:/home/nfs_exp# time cp 1GB.out /dev/null

real    0m3.979s
user    0m0.000s
sys 0m0.724s
```

- **SMB**

Same procedure for the SMB, First Copy into the /home/smb_exp (SMB share)

```
root@oxford:/home# time cp 1GB.out ./smb_exp/

real    0m8.607s
user    0m0.016s
sys 0m1.224s
```

Now, From my machine I will copy it to /dev/null:

```
root@bristol:/home/smb_exp# time cp 1GB.out /dev/null

real    0m8.508s
user    0m0.016s
sys 0m0.584s
```

We noticed two things from the experiments above, in case of 1GB file the NFS shared is ~5 seconds faster then the SMB shared. In addition to that, In case of NFS shared copying to the storage need less time than copying from the storage.

Share Type	Copy To	Copy From
NFS	0m2.395s	0m3.979s
SMB	0m8.607s	0m8.508s

Test 2- Realistic Workload (Apache Small files)

First, we will create 1000 files, each file 1000 bytes:

```
root@oxford:/home/smallfiles# (for i in {1..1000}; do dd if=/dev/zero
of=file$i bs=1KB count=1; done)
```

- **NFS**

Copy these files into /home/nfs_exp/smallfiles (NFS shares):

```
root@oxford:/home/smallfiles2# time (for i in {1..1000}; do cp file$i
/home/nfs_exp/smallfiles; done)

real    0m6.488s
user    0m0.716s
sys 0m5.460s
```

Now, Let's copy these files from the NFS share to /dev/null:

```
root@bristol:/home/nfs_exp/smallfiles# time (for i in {1..1000}; do cp
file$i /dev/null; done)

real    0m52.106s
user    0m0.740s
sys 0m50.936s
```

- **SMB**

Again, Copy these files into /home/smb_exp/smallfiles (SMB shares):

```
root@oxford:/home/smallfiles2# time (for i in {1..1000}; do cp file$i
/home/smb_exp/smallfiles; done)

real    0m8.250s
user    0m0.996s
sys 0m5.208s
```

Now, Copy these files from the SMB share to /dev/null:

```
root@bristol:/home/smb_exp/smallfiles# time (for i in {1..1000}; do cp
file$i /dev/null; done)

real    0m56.219s
user    0m0.704s
sys 0m53.176s
```

As we notice here, the NFS file shares perform better than the SMB file share:

Share Type	**Copy To**	**Copy From**
NFS	0m6.488s	0m52.106s
SMB	0m8.250s	0m56.219s

Source:

1-https://arstechnica.com/civis/viewtopic.php?f=20&t=248900, 2-
https://help.ubuntu.com/community/How%20to%20Create%20a%20Network%20Share%20Via%20Samba%20Via%20CLI%20%28Command-line%20interface/Linux%20Terminal%29%20-%20Uncomplicated%2C%20Simple%20and%20Brief%20Way%21, 3-
http://www.heatware.net/linux-unix/create-large-many-number-files-thousands-millions, 4-
https://www.linuxquestions.org/questions/programming-9/creating-large-size-file-for-testing-379950/