File System Evaluation

Due date: May 17th 2024, at noon

1 Introduction

The goal of this exercise series is to evaluate several file systems using the *bonnie++* benchmark and to report on the differences in performance observed with respect to the type of file system.

The submission format is quite free but should reflect the instructions in bold from section 2.5.

1.1 Physical Machine

The bare metal machine used for this exercise series is located in the faculty's cluster room and has the following properties:

CPU:	Broadcom BCM2711, Quad core Cortex-A72 (ARM v8) 64-bit SoC @ 1.8 GHz
Storage:	240 GB (over one SSD)
RAM:	2x 4 GB LPDDR4-3200 SRAM
OS:	Linux Ubuntu LTS 22.04

The machine belongs to a cluster of 24 identical machines deployed by Canonical's Metal as a Service (MAAS). The machines are connected to the network by two Cisco Catalyst switches using Ethernet.

1.2 Remote Connection

In order to connect remotely via the SSH protocol to your machine, you need to have access to a program that can communicate using this protocol. The programs mentioned in table 1 are recommended for the different types of operating systems.

Linux	OpenSSH	Debian-based: `apt-get install openssh` RPM-based: `yum install openssh` Red Hat-based: `dnf install openssh` openSUSE: `zypper install openssh`
macOS	(native SSH client)	-
Windows	OpenSSH	Official tutorial

Table 1: Operating systems and recommended SSH clients.

NOTE: For this lab you are supposed to have provided a public key in the *ed25519* type that was put on the machine by the system administrator. For Windows, the Windows PowerShell allows for key pair generation through `> ssh-keygen`, which is also the command for most Unix-derived systems. Make sure the public (`.pub`) and private (no extension) pair are located inside a `.ssh/` folder inside your home directory.

Once the SSH client is installed, you may follow the operating system specific instructions on how to connect to the machine running on the cluster.

1.2.1 Unix

Remote connect with the following: $ ssh chasseral-2.maas.
If asked, confirm the fingerprints of the router and the machine by entering yes and confirming with <ENTER>.
In order to retain this configuration after a reboot or power-off of your machine, create a file ~/.ssh/config with the following content:

```
Host iiun-cluster
     Hostname       clusterinfo.unineuchatel.ch
     User           fri-2024
     IdentitiesOnly yes
     IdentityFile   INSERT_PRIVATE_KEY_PATH_HERE
Host chasseral-*.maas
     User                  ubuntu
     StrictHostKeyChecking no
     ForwardAgent          yes
     ProxyJump             iiun-cluster
     IdentitiesOnly        yes
     IdentityFile    INSERT_PRIVATE_KEY_PATH_HERE
```

Listing 1: Configuration of ~/.ssh/config.

and replace INSERT_PRIVATE_KEY_PATH_HERE with your private key path, which should resemble ~/.ssh/id_ed25519.

1.2.2 Windows

Steps are identical to that of Unix but should be reproduced from the Windows PowerShell once the OpenSSH routine has been set up following the official tutorial. Make sure to respect Linux-style path writing for the INSERT_PRIVATE_KEY_PATH_HERE (that is, ~/.ssh/id_rsa in lieu of C:\Users\<yourName>\.ssh\id_rsa).

Make sure the config file inside C:\Users\<yourName>\.ssh has no extension.

2 Setting up

To execute a shell command as *root* user, prefix the command with the $ sudo <cmd> program. Commands entailing a # hashtag symbol are meant to be run with root privileges on your dedicated remote machine. You may execute $ sudo su to enter this execution mode. Know the risks.

2.1 Disk Partitioning and Installation of File System

Print the list of all available block devices on the machine on the solid-state drive (SSD):

```
ubuntu@chasseral-2:~\$ lsblk
NAME        MAJ:MIN RM   SIZE RO TYPE MOUNTPOINTS
loop0         7:0    0  59.1M  1 loop /snap/core20/1856
loop1         7:1    0 109.6M  1 loop /snap/lxd/24326
loop2         7:2    0  43.2M  1 loop /snap/snapd/18600
loop3         7:3    0  46.4M  1 loop /snap/snapd/18940
sda           8:0    0 223.6G  0 disk
|-sda1        8:1    0   512M  0 part /boot/efi
|-sda2        8:2    0  93.1G  0 part /
`-sda3        8:3    0 129.9G  0 part
mmcblk1     179:0    0  29.7G  0 disk
`-mmcblk1p1 179:1    0  29.7G  0 part
```

Listing 2: Disk partitioning on chasseral nodes.

Ignore attention to the `loop` type blocks: those are virtual devices generated by the OS. As can be seen from the output of `lsblk`, the machine has a single SSD `sda` of approximately 240 GB. There is a first partition `sda1`, a second partition `sda2` and a third partition `sda3` with increasing size. Those are respectively a boot partition, a root partition (indicated by the backslash) with the operating system having a size of 93 GB, and an empty partition of 130 GB. The `mmcblk1` is a plugged-in SD card. **DO NOT TOUCH THIS!** Breaking it would render your machine inoperable on the cluster and force manual intervention from administrators. Since the SDD disposes of three partitions, the installation of a file system is possible. The commands for installing the different file systems only differ in some command line parameters. In order to be consistent, each partition will be given as label the name of file system installed. The following example illustrates the installation of a XFS transactional/journaling file system, the creation of a mount point and finally mounting the file system under the mount point.

```
1 # mkfs.xfs -f -L XFS /dev/sda3
2 # mount -t xfs -o defaults /dev/sda3 /mnt
```

Listing 3: Formatting and mounting of an XFS file-system.

The first command (Listing 3, line 1) changes when installing a different file system, hence the type parameter `-t` of the `mount` program has to be adjusted as well. Generally the recipe of the last command remains as follows.

```
# mount -t <selected FS (lowercase)> -o defaults /dev/sda3 /mnt
```

Listing 4: Receipt of for mouting a FS.

In Table 2 you find a list of file systems, their type and installation command.

File System	**Type**	**Command**
Btrfs	copy-on-write	`# mkfs.btrfs -f -L BTRFS /dev/sda3`
ext4	transactional	`# mkfs.ext4 -F -L EXT4 /dev/sda3`
NTFS	transactional	`# mkfs.ntfs -f -F -L NTFS /dev/sda3`
VFAT	FAT	`# mkfs.vfat -n VFAT /dev/sda3`
XFS	journaling	`# mkfs.xfs -f -L XFS /dev/sda3`

Table 2: File systems and installation commands.

NOTE: the VFAT file system has to be mounted with the following mount options:

```
# mount -t vfat -o noauto,user,uid=1000,gid=1000 /dev/sda3 /mnt
```

Listing 5: Mouting of VFAT FS.

Once the execution of a benchmark measurement for a file system is complete, the file system can be unmounted and its partition be deleted by using the `fdisk` program, as shown in Listing 6:

```
# umount /mnt
```

```
# fdisk /dev/sda
```

Listing 6: Unmounting a FS.

You would use `d` to delete a partition, then 3 to delete the third partition. Type and enter `p` to print the partition table and `w` to write the table to the disk and exit `fdisk`.

`fdisk` will also be used in order to create a partition on `sda3`. For doing so, use the following sequence of commands in the shell:

```
# fdisk /dev/sda
```

Listing 7: Unmounting a FS.

Enter `n` to add a new partition. Press `<ENTER>` three times to accept default partition number, accept default first sector and accept default last sector and use remaining space on virtual disk. Type `p` to print the partition table then `w` to write the table to the disk and exit `fdisk`.

Often, `fdisk` finds the signature of the old file system. If this happens, simply confirm to erase the old signature.

2.2 Transferring Files Between Systems

In order to copy files from the machine running on the cluster onto your local laptop, the following can be used:

```
student@localhost:~$ scp chasseral-2.maas:/home/ubuntu/<DIR>/<FILE> .
```

Listing 8: SCP from a remote machine into your local machine.

To upload files to the machine in the cluster, swap the second last with the last argument in Listing 10, while clearly indicating the path and filename that has to be uploaded.

2.3 Required libraries

To compile the benchmarking took, you will need `gcc`. You will furthermore need some utilities for the benchmarking computations and to expand the archive:

```
$ sudo apt update
$ sudo apt install build-essential
$ gcc --version # check if it worked
$ sudo apt install -y linux-tools-$(uname -r)
$ sudo apt install zip
```

Listing 9: Installation of building dependencies.

2.4 Benchmark

In the *student*'s home directory you must place the *bonnie++* archive. You may also place the `bonnie.sh` script:

```
student@localhost:~$ scp ./bonnie++-1.98.tgz ./bonnie.sh chasseral-2.maas:~/
```

Listing 10: Copy of bonnie files to cluster node.

The archive contains the *bonnie++* file system benchmark that will be used to evaluate the performance of the different file systems. It is recommended to quickly get acquainted with it through its short Wikipedia page.

In order to install *bonnie++* the archive's content has to be extracted and compiled as instructed by the following commands on the cluster machine:

```
$ tar -xf bonnie-1.98.tar # extract the source code from the archive
$ cd bonnie++-1.98 # change into the bonnie++ directory
```

```
$ ./configure --prefix=`pwd`/debian/bonnie++/usr --mandir=`pwd`/debian/bonnie++/usr/share/man
$ make MORECFLAGS=-std=c++11 # compile the executable of the bonnie++ benchmark
```

Listing 11: Compilation of bonnie.

bonnie++ offers a variety of command line options which can be consulted by browsing its manual with $ man ./bonnie++.8 within the bonnie++-1.98 directory. The typical *bonnie++* command used throughout this exercise series will look as follows

```
# ./bonnie++ -d /mnt -s <int list> -n <int list> -r <int> -u 0 -x 5 -z <int>
```

Listing 12: Bonnie example of command-line.

where

- /mnt is the mount point of the partition on which the file system to be evaluated is installed
- -s <int list> specifies the file size in *MB* created during the benchmark
- -n <int list> specifies the number of files times 1024 that are created for the file operation test
- -r <int> specifies the amount of RAM in *MB* used (should be at least half the size of the -s option)
- -u 0 execute the program as user and group *root*
- -x <int> specifies the number of times the benchmark is repeated with the same parameters (can be omitted to do a single run)
- -z <int> specifies the random seed

2.5 Instructions

Choose at least 2 file systems and **run** the benchmark for different file sizes. Try to obtain as many results as possible within two weeks. After installing a file system on the second partition /dev/sda3 and mounting the partition, you can run the benchmark. To not waste any of your time, we have simplified your workflow as follows:

```
$ cd # change into your home directory
$ screen # start the terminal multiplexer
$ # press SPACE
$ ./bonnie.sh # run the script. Use $ sudo chmod u+x bonnie.sh in case of issues
$ # press CTRL + A and then D to detach from the terminal multiplexer
$ exit # terminate the SSH connection
```

It is recommended to take a few minutes to go through the shell script. You will see it launches many experiments sequentially.

screen is a screen manager, that will keep running the bonnie++ benchmark while you are disconnected from the cluster machine. In the meantime you can do some other activity before reconnecting to the machine, once the benchmark is done. To do so, run this sequence of commands after reconnecting with SSH:

```
$ screen -r # reattach to the previous terminal session
$ # do whatever you need to do, e.g., compress the results to a ZIP  archive
$ zip -r mybonnierun.zip ./out # create an archive of the script's output
$ # copy the results to your local machine
$ rm -Rf ./out # remove (delete) the script's output
$ exit # terminate the terminal multiplexer
$ exit # terminate the SSH connection
```

In order to quit the screen manager `screen` you have to execute `exit` once. If you ever feel overwhelmed by the screens, just kill them using `pkill screen`. The output of the benchmark can also be redirected to a file and copied later on onto the local machine. From the remote home directory, you could for instance run a single experiment:

```
# ./bonnie++-1.98/bonnie++ -r 1024 -s 20480 -n 256 -b -d /mnt -u 0 >> results
# cat results
Version  1.98       ------Sequential Output------ --Sequential Input- --Random-
                    -Per Chr- --Block-- -Rewrite- -Per Chr- --Block-- --Seeks--
Name:Size etc        /sec %CP  /sec %CP  /sec %CP  /sec %CP  /sec %CP  /sec %CP
chasseral-2 20G   524k  98 63.0m   8 15.9m   3  805k  99 20.7m   2 117.3   4
Latency             38280us     178ms     250ms   20653us     175ms     463ms
Version  1.98       ------Sequential Create------ --------Random Create--------
chasseral-2     -Create-- --Read--- -Delete-- -Create-- --Read--- -Delete--
              files  /sec %CP  /sec %CP  /sec %CP  /sec %CP  /sec %CP  /sec %CP
                256     1   0     1  99     1   0     1   0     1  99     1   0
Latency               374ms     195us     345ms     418ms      53us     350ms
1.98,1.98,chasseral-2,1,1652906166,20G,,8192,5,524,98,64492,8,16250,3,805,99,21229,2,
117.3,4,256,,,,,58,0,367784,99,54,0,59,0,394830,99,92,0,38280us,178ms,250ms,20653us,175ms,
463ms,374ms,195us,345ms,418ms,53us,350ms
```

As you can see, the last line of this file is pretty-printed in the lines that precede it. Using this last line (you could just copy-paste it), the next command allows even prettier display:

```
$ echo "1.98,1.98 [...] 418ms,53us,350ms" | ./bonnie++-1.98/bon_csv2html >> results.html
```

which gives a nice rendering of the experiment:

Version 1.98		Sequential Output						Sequential Input				Random Seeks			Sequential Create						Random Create						
	Size	Per Char		Block		Rewrite		Per Char		Block				Num Files	Create		Read		Delete		Create		Read		Delete		
		M/sec	% CPU	M/sec	% CPU	M/sec	% CPU	M/sec	% CPU	M/sec	% CPU	/sec	% CPU		/sec	% CPU	/sec	% CPU	/sec	% CPU	/sec	% CPU	/sec	% CPU	/sec	% CPU	
yahoocluster-10	20G	524	98	63	8	16	3	805	99	21	2	117.3	4	256	58	0	367784	99	54	0	59	0	394830	99	92		0
	Latency	38280us		178ms		250ms		20653us		175ms		463ms		Latency	374ms		195us		345ms		418ms		53us	350ms			

Generate scatter plots from the results in ~/out (throughput on x-axis, latency on y-axis) from the set of five points (the -x parameter is set to 5 in the bonnie.sh script) created by the benchmark per file system. Notice the missing values and **find** the explanation for it in the man pages. **Explain** in a couple of sentences the performance differences of the various file system types and **describe** your observations. The report should cover the *file I/O tests* as well as the *file creation tests*.

NOTE: Never shutdown the remote machine. It will stay on for two weeks for you to work on.

2.6 Benchmark++

Instead of using the mount point /mnt of the 3rd partition, use the directory /dev/shm. **Run** the benchmark directly on the /dev/shm directory. **Compare** your results with the previous results and **explain** your observations. Try to figure out why the results for this specific directory are different.