Using the different disks

When you execute the command:

df

```
]$ df -h
Filesystem                  Size  Used Avail Use% Mounted on
devtmpfs                    7.8G     0  7.8G   0% /dev
tmpfs                       7.8G     0  7.8G   0% /dev/shm
tmpfs                       7.8G  1.4M  7.8G   1% /run
tmpfs                       7.8G     0  7.8G   0% /sys/fs/cgroup
/dev/sdb1                    40G   14G   27G  34% /
/dev/sda5                   1.8T  964G  844G  54% /state/partition2
/dev/sdb5                   1.8T  1.1T  676G  63% /state/partition1
/dev/sdb2                    20G  3.3G   17G  17% /var
tmpfs                       1.6G     0  1.6G   0% /run/user/0
10.2.2.10:/mnt/labdata/ocs  9.6T  7.8T  1.4T  86% /home/ocs
tmpfs                       1.6G     0  1.6G   0% /run/user/1132
10.2.2.15:/home             1.8T  736G 1004G  43% /sharelab/bioinf
```

The first column is the "device": how this disk is seen by the operative system.
The last column is the "data": where, in the directory tree, the data that this disk contains can be accessed (in which directory the device is mounted)

You will find basically 3 types of disks

1.- Remote disks:

Those disks that its device name starts with a computer name or a IPadress followed by ":"/somethinghere are remote disks usually protected by RAID, you will be able to access them in the same place on any computer in our system. Example:

```
10.2.2.15:/home             1.8T  736G 1004G  43% /sharelab/bioinf
10.2.2.10:/mnt/labdata/ocs  9.6T  7.8T  1.4T  86% /home/ocs
```

Remote disks mount on demand. This way, when you do a "df" or "ls /sharelab" or "ls /share" it is highly probable that you do not see all the available discs (including the ones that you may be looking for). They will appear when you access them (e.g.: ls /sharelab/bioinf). After a while of not being used they unmount automatically.

They are usually main data directories, but not necessarily.
Usually "`/home/$USER`" is a remote disk and it is your main data directory, but not necessarily, if "/home/$USER" is not your main data directory avoid creating there any file or directory.
Highly intense IO (Input-Output) operations (AKA: lots of reading and writing per second) should be avoided in these disks. For these purposes local disks are better.

2- Local disks:

All our computers contain directories called:
`/state/partitionN`
Where "N" is a number.
These are local disks; they could be HDDs, NVMe or RAID virtual disks.
Take into account that:
NVMes are faster but they are smaller than HDDs.
RAID disks are groups of disks that are presented as a single one, this is done to offer certain protection against disk failure.

You can see if they are hdds (`/dev/sdxx`), nvme (`/dev/nvmexxxx`) or a raid virtual disk (`/dev/mdxx`) with the command "`df` ".
If these disks are not raid virtual disks, they are not protected against failures, so you should never use them for storage of something "valuable".
They are "local" so its contents is different in each computer.
Whatever data you leave in a NVMe must be eliminated as soon as the run that generated it finishes. NVMes cannot be used as storage units.
Unless the `/state/partitionXX` has been assigned as our main storage unit, we usually only run jobs in `/state/partitionXX` if we know they do a lot of IO (Input-Output) operations (AKA: reading and writing lots of data from/to the disks) or they generate a huge amount of data, so if it is not the case, do not worry about it. We usually do it for running MD simulations that they are run as a batch of concatenated jobs.
We prepare and keep all the files in the our main storage disk and in case we decide to run the job in `/state/partitionN`, we only move what we need to the `/state/partition1` right before starting the job. If the input data is small, we usually don't move it. We usually do something like this:

```
cd /state/partition1/myuser
mkdir mynewjobdir
cd mynewjobdir
/home/myuser/myprogramdir/myprogram.sh /home/myuser/mydatadir/myinputdata
```

the program generates the data in the current directory (/state/partition1/myuser/mynewjobdir)
when it is done, we look at the data or maybe we run some analysis scripts and later or either we move the whole dir:

```
mv /state/partition1/myuser/mynewjobdir /home/myuser/mydatadir/
```

or we just move the analysis results:

```
mv /state/partition1/myuser/mynewjobdir/newjob_analysis /home/myuser/mydatadir/
```

For doing that, if it is a very repetitive process, we use scripts like `state_partition1_example_job.sh` (you can find it in the same place as this guide and has been copied at the end of this document for convenience)
This script is a bit complex with some error control, most of the times we do not use the error control (lines from 10 to 32). You are welcome to use it if you want.
This script has not been written by hand; in these cases, we usually create a larger script which oversees creating these ones.
The use of scripts automatizes the process, so we don't have to worry for certain things like making mistakes creating directories an all of that, but it takes an extra effort, sometimes it is worth it and sometimes it is not, so we don't necessarily do it.
What we actually do every time we have to run something in a remote computer that will take more than some minutes to finish is to use "terminal multiplexers": screen or tmux. If you do not know them, I strongly advise that you take a look to this page:
http://linuxcommand.org/lc3_adv_termmux.php
Using them will prevent losing your job if there is a network failure between your computer and the one you are using for running your job, and at the same time, you can get many "terminals" with only one ssh connection and many other features if you know how to use them. (IMPORTANT! execute tmux or screen in the running computer, eg: gin, orujo or stro)

3.- The others

They are virtual devices used by the Operative system to handle some operations, please ignore them.

state_partition1_example_job.sh:

```
#!/bin/bash

RESULTDIR=/home/txino/GTA/GTAscript/V4/RUN/BLOUT
JOBSDIR=/home/txino/GTA/GTAscript/V4/RUN/BLOUT/JOBS
TJID=0224104558_089
WJID=Sma00K279a..9..Smasm454-2lDM5j
JOBNAME=Sma00K279a
WORKDIR=/state/partition1/$USER/$WJID/$TJID/$JOBNAME

function endscript
{
    if ls ERR* 1>/dev/null 2>&1;then
        touch ERROR
        touch $JOBSDIR/$JOBNAME.ERROR
    else
        echo "NO ERROR FILE"
        touch DONE
        touch $JOBSDIR/$JOBNAME.DONE
    fi

    rm -f $JOBSDIR/$JOBNAME.RUNNING

    cd $RESULTDIR
    rsync --remove-source-files -av $WORKDIR $RESULTDIR/ && rm -rf $WORKDIR

    rmdir /state/partition1/$USER/$WJID/$TJID
    rmdir /state/partition1/$USER/$WJID
    rmdir /state/partition1/$USER

    date +%Y%m%d_%H%M%S
    exit
}

if hostname -A >/dev/null 2>&1;then
    hostname -A
else
    hostname
fi

date +%Y%m%d_%H%M%S
mkdir -p $WORKDIR
cd $WORKDIR || { touch ERR_cdworkdir; endscript; }
pwd
touch $JOBSDIR/$JOBNAME.RUNNING
####  END HEADER #######

#### JOB CODE ##########


/home/txino/GTA/GTAscript/V4/BLTOJSON/bltojson.pl
/home/txino/GTA/GTAscript/V4/Sma00K279a..9..Smasm454-2lDM5j-rum_newblat-fst.fsgi.json Sma00K279a

rc=$?; if [[ $rc != 0 ]]
then
    echo "ERROR: /home/txino/GTA/GTAscript/V4/BLTOJSON/bltojson.pl
"/home/txino/GTA/GTAscript/V4/Sma00K279a..9..Smasm454-2lDM5j-rum_newblat-fst.fsgi.json Sma00K279a"
exec error \$rc=$rc"
    touch ERR_BLTJ
fi


#### END JOB CODE ######

####  START FOOTER #######

endscript
```