File systems in the cluster

As you may have guessed, there are several file systems serving the cluster.

Each of these file systems has its own characteristics, making it the choice for a different workload.

Under Unix/Linux, file systems are simply *mounted* as “subdirectories” somewhere under the root of the file system (or “the root of the directory tree”), which is symbolized by the `/` (slash character). So, in case you are coming from a Windows environment, the first thing you'll notice is that there is no such thing as drive letters (`C:`, `D:`, etc.). Logically, everything appears to be in the same file system. You use a different file system by simply changing into its directory. File systems may be mounted almost anywhere in the file system, not just at the top. Use your favorite search engine to learn more about how unix organizes files in case you are not yet familiar. You'll need it.

For ease of use and consistency of access, we provide some automatically set variables called HOME, BIGWORK, SOFTWARE, PROJECT and TMPDIR that either point to your own directories in the file systems that are mounted under /home, /bigwork, /software and /project, or to the scratch space local to each compute node (/scratch). You can access the content of a variable by putting a '$' sign in front of its name, like in the commands `echo $HOME` or `cd $BIGWORK`. To make it clear that those are variables, we'll refer to them including the dollar sign.

Filesystem	Soft limit	Hard limit	Inode soft	Inode hard	Grace time	Backup	Connection	Access point
$HOME	10 GB	12 GB	-	-	4 weeks	Yes	Slow	/home/USERNAME
$BIGWORK	100 GB	1 TB	1.000.000	1.500.000	30 days	No	Fast	/bigwork/USERNAME
$PROJECT	10 TB	12 TB	10.000.000	15.000.000	30 days	No	Moderate	/project/GROUPNAME
$SOFTWARE	50 GB	100 GB	-	-	30 days	No	slow	/software/GROUPNAME

Table 1: The most important file systems with specifications

How to use the file systems

The following sections describe the characteristics and best utilization of each of the cluster's filesystems.

$BIGWORK - where you work

Your main working directory. It provides a comparatively high amount of storage space. Due to the fact that large amounts of data are quickly changing in this storage, it is impossible to provide a backup. All computations should be done here, except for jobs requiring $TMPDIR for a reason (see section $TMPDIR). $BIGWORK is connected via the InfiniBand network interface and thus inherently

much faster than $HOME. It is also being provided by a whole group of servers to increase performance. Bandwidth and scalability for multi-core jobs usually is very good, but performance for many small files is mediocre at best (for this use case, see $TMPDIR). Both the amount of data and the number of files that you can store are limited (see below, Quota and grace time).

$BIGWORK is mounted on all nodes in the cluster (all nodes see the same $BIGWORK file system).

$PROJECT - data sets and results of a group, but no access in compute jobs

Your (and your colleagues') project directory. Each project gets assigned a project directory to which all members of a project have read and write access at */project/<your-cluster-groupname>* (the environment-variable $PROJECT points to this location). In order to store individual user accounts' data in this directory in such a manner that everyone can keep track of what belongs to whom, we suggest each account create their own sub directory named $PROJECT/$USER and set suitable access rights (like `mkdir -m 0700 $PROJECT/$USER`). How you best organize your data within your group is left up to you, however. The group's quota for the project directory is usually set to 10 TB.

$PROJECT is ONLY mounted on the login and the transfer nodes, but NOT on the compute nodes, since the hardware serving it is quite modest. So you CAN NOT USE this file system IN YOU JOBS (use $BIGWORK instead).

Copying between $BIGWORK and $PROJECT is thus only possible on the transfer or the login nodes, where a fast connection between both file systems is available. It is intended to separately store files that are important for the work of an institute on the cluster, for preparation, curation and long time retention of both input data, results and setups. It is configured in such a way that all data are physically written in two copies. Due to the amount of data, however, no additional backup is provided.

We regard data in $PROJECT as belonging to the institute or the project, respectively. If you want to make sure that the project management at your institute can still access your data after you have left the LUH, put a copy of your project results here, and we can easily give access to the person in charge of the project.

Please note: Backing up your data regularly from $BIGWORK to $PROJECT storage and/or to your institute's servers is considered essential, since $BIGWORK is designed as scratch file system. $PROJECT should also NOT be considered being a real backup as well, since it is also disk based. Disks may fail, and computing centers may also suddenly burn to the ground when you least need it.

$HOME - configuration files and setups. "Don't do big work in your home", though (pun intended)

Your home directory is the directory in which you are normally placed directly after a login from the command line. You should only place small and particularly important files here that would be especially time-consuming to create anew, like templates for job setups, important scripts, configuration files for software etc. DO NOT PUT DATA HERE that you use in compute jobs and DO NOT USE THIS DIRECTORY TO WRITE RESULTS. $HOME is provided by only one server which intentionally is connected via relatively slow gigabit ethernet to the remainder of the system (to protect the server from becoming overloaded to quickly). In comparison to $BIGWORK, this file system is very slow and particularly unsuitable to handle data intensive tasks. If you succeed

overloading $HOME, all users will notice.

If you overstep your quota (see following section) in this directory, – which will happen quickly if you do not head the abov advice and let your compute jobs run and put data here – you will make yourself unable to log in graphically with tools like X2Go. You will then need to delete data before you can continue to work. In case you run into that situation, do not make matters worse by simply deleting “anything”, because there are important files here that e.g. give you access to parts of the system. If you just indiscriminately delete files here, you'll definitely embarass yourself and need to ask for administrative assistance next, which takes *much* longer than if you had instead taken some time to check which files were created by a job of yours.

So you should only place important and small files in your $HOME that would be particularly difficult and laborious to re-create (like configs etc.), but meticulously avoid accesssing data here from within compute jobs. If you disregard that rule, your jobs will quite probably not only take much longer to complete, but at the same time also impede other users’ work, due to the technical properties described above. You should point all directories that possibly have been set automatically by the software you install/use - quite often e.g. temporary directories – to $BIGWORK. Do NOT use symbolic links between $BIGWORK and $HOME in a compute job to creatively cheat around the circumstances, since these links need to get looked up as well. Use the environment variable $BIGWORK for convenient access in a computation, and avoid putting anything in your $HOME that can go somewhere else. Do NOT place pip, (ana)conda, CRAN/R or similar installations here. Use your $SOFTWARE directory and, if possible, share with your colleagues. Also have a look at the exercise.

Due to the limits in the amount of data and implicitly only slowly changing data, we have a daily backup of data in $HOME.

Memory Aid: “Never WORK at HOME”. It simply is not built for that, and others will feel the impact of what you are doing. Do not abuse $HOME as a backup directory for your compute results. That's a really bad idea.

$HOME is mounted on all nodes in the cluster (all nodes see the same $HOME file system).

$TMPDIR - local to each node, different disk types

Within jobs, the variable $TMPDIR points to local storage available directly on each node. Whenever local storage is needed, such as when you are handling thousands of files that would make you hit your $BIGWORK inode limit, $TMPDIR should be used. The second reason may be that your jobs due to heavy parallelization are now I/O-bound and thus need a node that is specifically equipped with local fast SSDs. Only a few nodes are equipped with these SSDs yet, and they usually belong to an institute that bought them within our FCH service, so that is a limited option for those who do not have access to a specific FCH partition.

In general, do not simply assume $TMPDIR to be faster than $BIGWORK. $TMPDIR can be used for temporary files for applications that *imperatively* require a dedicated temporary directory. There will probably be a performance improvement if you have many very small files and use $TMPDIR. You may get a big performance boost if you can use a node that is equipped with a local SSD scratch disk, see the section hardware specifications of cluster compute nodes. Lustre (BIGWORK) is not good at handling large numbers of small files, which is the reason that the number of files is strictly limited on BIGWORK. The only option for these jobs is to pack datasets containing millions of files into archive files, e.g. using the `tar` command and then “locally” extract them on the fast SSD scratch disk before the job starts. An example how to do this would be the pair `tar -cf <archive_file.tar>`

`<directory_to_archive>` and `cd $TMPDIR ; tar -xf $BIGWORK/archive_file.tar`. In case of easily compressible data (for example text files, but usually NOT jpg files), you may try using mild compression to possibly increase bandwidth from disk and optimize the size of your archive, but if you are using millions of small jpegs, it's better not to do that — those are already relatively well compressed, and you'll usually not gain much in terms of file size, but it may cost you additional time to decompress. You may also want to have a look at how striping works on Lustre. See below for an explanation.

$TMPDIR **is strictly local to each node**. This means that **if you write data to the local $TMPDIR on one node, it will NOT be visible on any other node in the cluster**. If your job occupies two or more nodes, the tasks on each node will see a different local directory. You should have an idea of the size of $TMPDIR first, which you can easily check *before* submitting your jobs using the command `scontrol show nodes <nodename_here>`, or (of course) within your job using `df -h $TMPDIR`. The parameter “TmpDisk” shows you the configured size of the scratch/$TMPDIR partition in MiB.

Some FCH partitions (nodes we run for various institutes) have expressly been equipped with large scratch SSDs by their owners due to their particular work load (many small files that can not reasonably be stored in Lustre).

Please note: As soon as a job finishes, all data stored under $TMPDIR will be deleted automatically.

$SOFTWARE - install software here

This directory enables a project to install their own software in a way so the whole group can use it. You should also use this directory (or a personal subdirectory you create here) for installations done via pip, conda, R/CRAN and the like.

$SOFTWARE points to a cluster-wide shared directory that is placed in `/software/<your-cluster-groupname>`. If you want to narrow down the list of usernames in your project that should have access (e.g. read) to your software installation directory, we recommend creating a subdirectory that only your account can access using the command:

```
mkdir -m 0700 $SOFTWARE/mysoft
```

Thereafter, you may grant additional usernames the access you desire using an access control list (ACL):

```
setfacl --mask --modify user:<username>:rx,default:user:<username>:rx
$SOFTWARE/mysoft
```

Check the ACL you set using

```
getfacl $SOFTWARE/mysoft
```

See `man acl`, `man setfacl` and `man getfacl` for more information about ACLs. Subdirectories created in $SOFTWARE can only be removed by the user who created them (the “sticky bit” is set for this directory) to prevent colleagues with no read/execute access to subdirectories from accidentally destroying your work.

In case you are the project manager of a project with us and your institute has special requirements

(like, only a few persons should be able to install/maintain software in this directory), talk to us and we'll change ownership to the account of the project manager's account so you can completely decide for yourself whom to grant which permissions. In this case, you should take extra care that file permissions are set properly to make sure only members of your group can write into or have access to the directory so the executables you'll find here stay your own. Whomever you grant access may change files here.

The group's quota for the software directory is set to 50 GB. There's also no backup at this time. It would be an exceptionally bad idea to abuse $SOFTWARE to store job results. Not only because it's a shared and fairly limited ressource in terms of storage, but also since this directory is only being served via Ethernet by just one server and thus again not suitable for massively parallel workloads (similar to HOME, but not quite as desastrous when overloaded). But it's a way to give you some storage for installations that you might possibly share in your work group, and it somewhat mitigates the problem that $HOME must be so strictly limited. It also keeps the many small files that some software installations create away from Lustre ($BIGWORK) to keep the load created by many metadata operations as small as possible there.

Quota and grace time

On the shared storage systems (i.e. $HOME, $BIGWORK and $PROJECT), only a fraction of the whole disk space is made available to you or your account, respectively. This amount is designated as *quota*. The purpose is to ensure everybody gets a share of the ressources, and also to limit the effects of accidents that otherwise could easily fill up the whole system. In case your current quota on $BIGWORK or $PROJECT would severely impede your work, let us know what you would need, stating account name, amount needed and the time frame (i.e. for how many months you would need an increased quota).

There is a *soft quota* and a *hard quota*. For each of these two quota types, two independent parameters are set, namely *block quota* (amount of gigabytes you can store) and *inode quota* (number of files you can create). So there are four values you need to consider.

The *hard quota* is the absolute upper bound which you can not exceed until an administrator raises the limit. The *soft quota*, on the other hand, may be exceeded for some time, the so-called *grace* time.

Exceeding any of the two parameters (*block* or *inodes*) of your *soft quota* starts the respective *grace timer*.

During the *grace time*, you are allowed to exceed your *soft quota* up to the limits of your respective *hard quota*. After the grace timer has run out, you will not be able to store any new data, unless you reduce disk space usage below the *soft quota*. *Since we get asked from time to time: no, the system will NOT delete random files of yours after the grace period has expired. What is on disk stays on disk. You will just not be able to write new data as long as you are over the soft limit when the grace period is over.*

As soon as your disk consumption falls below the *soft quota* again, the *grace time* counter for that file system and that parameter is reset. On $HOME and $BIGWORK, quotas are valid for each individual user, whereas on $PROJECT, quotas are accounted for the whole group (the project you are a member of, usually a four-letter code followed by a five-digit number; use the `id` command to check).

For $BIGWORK, $PROJECT and $SOFTWARE, the relevant (user/group) *block* and *inode* quota grace

times are 30 days. On $HOME, grace time is 10 days.

The *quota* mechanism protects users and system against possible errors of others, limits the maximum disk space available to an individual user, and keeps the system performance as high as possible by avoiding unneccessary clutter. In general, we ask you to delete files which are no longer needed to keep the file systems fast. Low disk space consumption is especially helpful on $BIGWORK in order to optimise system performance. You can query your disk space usage and *quota* with the command `checkquota` – see also exercise. In case the standard system quota values are too low for you, talk to us and tell us which account needs how many GB for how long (and why) - we try to accommodate requests within reason. The quota values are set such that usually 90 % of our users should not feel much of limits, except for the usual need to periodically clean up and backup results.

Please note: If your quota is exhausted on $HOME, you will not be able to login graphically using X2Go any more. Connecting using *ssh* (without -X) will still be possible.

Lustre file system and stripe count

Please note: All statements made in this section also apply to $PROJECT storage.

On the technical level, $BIGWORK is comprised of multiple components which make up the storage system. Generally speaking, using $BIGWORK without changing any default values should usually work well. However, it may be useful under certain circumstances to change the so called *stripe count*, in particular if you need to handle files larger than at least 1 GB, or if you access different parts of the same file in highly parallel applications running on several nodes. Balancing I/O, investing some time in understanding Lustre and testing different setups may result in a higher performance and a better-balanced use of the overall system, which in turn is beneficial for all users.

Data on Lustre-based file systems such as $BIGWORK is saved on OSTs, *Object Storage Targets*. The *location* of the data (which OST(s)?) is registered with a directory server (“MDS”) and stored on a MDT, the *Meta Data Target*. Each OST and the MDT as well looks like one large hard drive (a “block device”) to the system. In reality, each Target consists of several hard drives bundled together in the background by a storage controller to ensure e.g. some fault tolerance and higher performance, but this is not visible to the computers using the file system. The first takeaway message here is that Lustre stores files on several things that look like separate hard drives, which may be taken into consideration when you need more performance, but together the whole thing just looks like another file system in a subdirectory ($BIGWORK in this case).

By default, files are written to a single OST each, regardless of their size. This corresponds to a *stripe count* of one. The *stripe count* determines how many OSTs will be used to store data. Figuratively speaking, it determines in how many stripes a file is being split. Splitting data over multiple OSTs can increase access speeds, since the read and write speeds of several OSTs and thus a higher number of hard drives are used in parallel. At the same time, one should only distribute large files in this way, because access times can also increase if you have too many small requests to the file system. Depending on your personal use case, you may need to experiment to find the best setting. When doing this, consider that the current workload of the whole system may influence your results.

Please note: If you are working with files larger than 1 GB, and for which access times e.g. from within parallel computations could significantly contribute to the total duration of a compute job, please consider setting *stripe count* manually according to section.

Stripe count is set as an integer value representing the number of OSTs to use, with -1 indicating all available OSTs. It is advised to create a directory below $BIGWORK and set a *stipe count* of -1 for it. This directory can then be used e.g. to store all files that are larger than 100 MB. For files significantly smaller than 100 MB, the default stripe count of one is both better and sufficient.

Please note:In order to alter the of existing files, these need to be copied, see section. Simply moving files with `mv` is not sufficient in this case.

Exercises

In this section we consider several examples on working with the cluster storage systems.

Using file systems

```
# where are you? lost? print working directory!
pwd

# change directory to your bigwork/project/home directory
cd $BIGWORK
cd $PROJECT
cd $HOME

# display your home, bigwork & project quota
checkquota

# make personal directory in your group's project storage
# set permissions (-m) so only your account can access
# the files in it (0700)
mkdir -m 0700 $PROJECT/$USER

# copy the directory mydir from bigwork to project
cp -r $BIGWORK/mydir $PROJECT/$USER
```

Setting stripe count

```
# get overall bigwork usage, note different fill levels
lfs df -h

# get current stripe settings for your bigwork
lfs getstripe $BIGWORK

# change directory to your bigwork
cd $BIGWORK

# create a directory for large files (anything over 100 MB)
mkdir LargeFiles
```

```
# get current stripe settings for that directory
lfs getstripe LargeFiles

# set stripe count to -1 (all available OSTs)
lfs setstripe -c -1 LargeFiles

# check current stripe settings for LargeFiles directory
lfs getstripe LargeFiles

# create a directory for small files
mkdir SmallFiles

# check stripe information for SmallFiles directory
lfs getstripe SmallFiles
```

Use newly created LargeFiles directory to store large files

Altering stripe count

Sometimes you might not know beforehand, how large files created by your simulations will turn out. In this case you can set *stripe size* after a file has been created in two ways. Let us create a 100 MB file first.

```
# enter the directory for small files
cd SmallFiles

# create a 100 MB file
dd if=/dev/zero of=100mb.file bs=10M count=10

# check filesize by listing directory contents
ls -lh

# check stripe information on 100mb.file
lfs getstripe 100mb.file

# move the file into the large files directory
mv 100mb.file ../LargeFiles/

# check if stripe information of 100mb.file changed
lfs getstripe ../LargeFiles/100mb.file

# remove the file
rm ../LargeFiles/100mb.file
```

In order to change stripe, the file has to be copied (`cp`). Simply moving (`mv`) the file will not affect stripe count.

First method:

```
# from within the small files directory
```

```
cd $BIGWORK/SmallFiles

# create a 100 MB file
dd if=/dev/zero of=100mb.file bs=10M count=10

# copy file into the LargeFiles directory
cp 100mb.file ../LargeFiles/

# check stripe in the new location
lfs getstripe ../LargeFiles/100mb.file
```

Second method:

```
# create empty file with appropriate stripe count
lfs setstripe -c -1 empty.file

# check stripe information of empty file
lfs getstripe empty.file

# copy file "in place"
cp 100mb.file empty.file

# check that empty.file now has a size of 100 MB
ls -lh

# remove the original 100mb.file and work with empty.file
rm 100mb.file
```

From:
https://docs.cluster.uni-hannover.de/ - **Cluster Docs**

Permanent link:
https://docs.cluster.uni-hannover.de/doku.php/guide/storage_systems

Last update: **2026/10/15 18:59**