A common way to share files and/or directories with group members or others is to use the chmod command to change the permissions. However, chmod has limitations, so you may sometimes choose to use Access Control Lists (ACLs).

When you issue the command chmod g+rx *filename*, for example, all the members in your group (g) gain read (r) and search/execute (x) access to that file, as shown below:

```
% ls -l filename
-rw------- 1 zsmith s0101 9 Jun 10 12:11 filename

% chmod g+rx filename

% ls -l filename
-rw-r-x--- 1 zsmith s0101 9 Jun 10 12:11 filename
```

However, chmod does not allow you to select which members of your group or which specific individuals outside of your group can access your files/directories. ACLs provide a mechanism for greater control of file sharing. There are two ACL commands:

setfacl

Set file access control lists.

```
SYNOPSIS
     setfacl [-bkndRLPvh] [{-m|-x} acl_spec] [{-M|-X} acl_file] file ...

     setfacl --restore=file
```

A detailed usage explanation of setfacl and its options can be found via **man setfacl**. Among the options listed:

1. The -m or -M option lets you "modify" the ACL, where -m expects an ACL on the command line and -M expects an ACL from a file or from standard input
2. The -x or -X option removes the ACL entries
3. The -R or --recursive option applies operations to all files and directories recursively
4. The --test option allows you to test the effect of changing the ACL without actually changing it
5. The -b option removes all extended ACL entries except the base entries of the owner, group, and others

getfacl

Get file access control lists.

```
SYNOPSIS
     getfacl [-dRLPvh] file ...

     getfacl [-dRLPvh] -
```

A detailed usage explanation of getfacl and its options can be found via **man getfacl**.

Note: Before you grant another user or group access to certain files or directories, make sure that access to the parent directory (where the files or directories reside) is also allowed.

Example 1

To allow another user (*jbrown*) to have read/execute (rx) permission on a file (*filename*) and to view the ACL before and after an ACL change:

```
% ls -l filename
-rw------- 1 zsmith s0101 9 Jun 10 12:11 filename

% getfacl filename
# file: filename
# owner: zsmith
# group: s0101
user::rw-
group::---
other::---

% setfacl -m u:jbrown:rx filename

% getfacl filename
# file: filename
# owner: zsmith
# group: s0101
user::rw-
user:jbrown:r-x
group::---
```

mask::r-x
other::---

% ls -l *filename*
-rw-**r-x**---**+** 1 *zsmith* s0101 9 Jun 10 12:11 *filename*

Example 2

To remove all extended ACLs in Example 1 except the base entries of the owner, group, and others:

% setfacl -b *filename*

% ls -l *filename*
-rw------- 1 *zsmith* s0101 9 Jun 10 12:11 *filename*

% getfacl *filename*
# file: *filename*
# owner: *zsmith*
# group: s0101
user::rw-
group::---
other::---

Example 3

Continuing from Example 1, to test the granting of read/execute (rx) access to another group (group id 24176) without actually doing it:

% setfacl --test -m g:24176:rx *filename*
filename: u::rw-,u:*jbrown*:r-x,g::---,**g:g24176:r-x,m::r-x**,o::---,*

% getfacl *filename*
# file: *filename*
# owner: *zsmith*
# group: s0101
user::rw-
user:*jbrown*:r-x
group::---
mask::r-x
other::---

Example 4

To allow another user (*jbrown*) recursive access to a directory (dir.abc which contains a file *filename*):

% ls -ld dir.abc
drwx------ 2 *zsmith* s0101 17 Jun 10 13:19 dir.abc

% ls -l dir.abc
total 0
-rw------- 1 *zsmith* s0101 0 Jun 10 13:19 *filename*

% setfacl -R -m u:*jbrown*:rx dir.abc

% getfacl dir.abc
# file: dir.abc
# owner: *zsmith*
# group: s0101
user::rwx
user:*jbrown*:r-x
group::---
mask::r-x
other::---

% getfacl dir.abc/*filename*
# file: dir.abc/*filename*
# owner: *zsmith*
# group: s0101
user::rw-
user:*jbrown*:r-x
group::---
mask::r-x
other::---

% ls -ld dir.abc
drwx**r-x**---**+** 2 *zsmith* s0101 17 Jun 10 13:19 dir.abc

% ls -l dir.abc

total 0
-rw-**r-x**---**+** 1 *zsmith* s0101 0 Jun 10 13:19 *filename*

Example 5

Continuing from Example 4, to recursively remove all permissions user *jbrown* for a directory:

% setfacl -R -x u:jbrown dir.abc

% getfacl dir.abc
# file: dir.abc
# owner: *zsmith*
# group: s0101
user::rwx
group::---
mask::---
other::---

% getfacl dir.abc/*filename*
# file: dir.abc/*filename*
# owner: *zsmith*
# group: s0101
user::rw-
group::---
mask::---
other::---

For more information on ACLs, read **man acl**.

NAS's in-house developed giveto script is built on the use of Access Control Lists (ACLs). It allows one user (the giver) to copy files and/or directories to a /nobackup directory of another user (the recipient).

giveto is installed under /usr/local/bin on all NAS compute systems and Lou.

In the example below, user zsmith gives a copy of his dir.abc directory on PFEs to user jbrown. The steps describe the giveto command used by each of them, and the results.

1. User *jbrown* uses the command giveto -i *zsmith* to automatically (a) create an INCOMING directory (if it does not already exist) under her /nobackup/*jbrown* and (b) grant user *zsmith* read/write/execute permission on this directory.

   pfe*21*:/u/*jbrown*% giveto -i *zsmith*
   nobackup[*1*] = /nobackup/*jbrown*

   pfe*21*:/u/*jbrown*% ls -ld /nobackup/*jbrown*/INCOMING
   drwx**rwx**---**+** 2 *jbrown* s0202 4096 Jun 14 12:18 /nobackup/*jbrown*/INCOMING

2. User *zsmith* uses the command giveto *jbrown* dir.abc to automatically (a) create a subdirectory called *zsmith*_0 under *jbrown*'s INCOMING directory, (b) copy dir.abc to /nobackup/*jbrown*/INCOMING/*zsmith*_0, (c) grant user jbrown read/write/execute permission on /nobackup/*jbrown*/INCOMING/*zsmith*_0, and (d) send an email to user *jbrown* regarding the copy.

   pfe*21*:/home1/*zsmith*> ls -ld dir.abc
   drwx------ 2 *zsmith* s0101 17 Jun 14 12:21 dir.abc/

   pfe*21*:/home1/*zsmith*> giveto *jbrown* dir.abc
   setfacl -m u:*jbrown*:rwx *zsmith*_0
   setfacl -m u:*jbrown*:rwx *zsmith*_0/giveto.log
   setfacl -m u:*jbrown*:rwx *zsmith*_0/dir.abc
   setfacl -m u:*jbrown*:rwx *zsmith*_0/dir.abc/foo2
   path = /nobackup/*jbrown*/INCOMING/*zsmith*_0
   total 12
   drwxrwx---+ 3 *zsmith* s0101 4096 Jun 14 12:29 .
   drwxrwx---+ 3 *jbrown* s0202 4096 Jun 14 12:29 ..
   drwxrwx---+ 2 *zsmith* s0101 4096 Jun 14 12:21 dir.abc
   -rw-rwx---+ 1 *zsmith* s0101 44 Jun 14 12:29 giveto.log

   Note: If the directory *zsmith*_0 already exists prior to this step, *zsmith*_1 would be used instead.

3. User *jbrown* receives an email from user *zsmith* with a subject line "giveto files". They see that the directory dir.abc has been copied successfully. Even though the directory /nobackup/*jbrown*/INCOMING/*zsmith*_0 is still owned by user *zsmith*, user *jbrown* now has permission to read/write/execute files and directories under /nobackup/*jbrown*/INCOMING/*zsmith*_0.

   pfe*21*:/u/*jbrown*% ls -lrt /nobackup/*jbrown*/INCOMING
   total 4
   drwx**rwx**---**+** 3 *zsmith* s0101 4096 Jun 14 12:29 *zsmith*_0

   pfe*21*:/u/*jbrown*%ls -lrt /nobackup/*jbrown*/INCOMING/*zsmith*_0
   total 4
   drwx**rwx**---**+** 2 *zsmith* s0101 4096 Jun 14 12:21 dir.abc

   pfe*21*:/u/*jbrown*%ls -lrt /nobackup/*jbrown*/INCOMING/*zsmith*_0/dir.abc
   total 4
   -rw-**rwx**---**+** 1 *zsmith* s0101 8 Jun 14 12:21 *foo2*

   pfe*21*:/u/*jbrown*%getfacl /nobackup/*jbrown*/INCOMING/*zsmith*_0/dir.abc
   # file: /nobackup/*jbrown*/INCOMING/*zsmith*_0/dir.abc
   # owner: *zsmith*
   # group: s0101
   user::rwx
   user:*jbrown*:rwx
   group::---
   mask::rwx
   other::---

Read **man giveto** for more information.

The giveto script was created by NAS staff member Arthur Lazanoff.

Storing Data in the Home Filesystems for NAS Compute Resources

All data stored in your home filesystem of any NAS compute system are automatically backed up *daily* under script control.

If you lose important data from a home filesystem and need to have it restored, please send a request to the NAS Control Room at support@nas.nasa.gov and provide the following information:

- System name
- Your username
- The directory and/or file name(s) that you wish to restore
- The date the data was last touched

Note: Disk space quota limits are imposed on the home filesystems for NAS compute resources. If your data exceeds your disk space quota limits, reduce your usage by deleting unneeded files, copying important files to Lou, or copying them to storage space at your local site and then removing them from your home filesystem to NAS compute resources.

Storing Data in NAS /nobackup Filesystems

Data in any /nobackup filesystem of any NAS system are *not* backed up by NAS. You are responsible for performing your own backups.

Note: Disk space and inode quota limits are imposed on the Lustre filesystems (/nobackup*X*). If your data exceeds the soft quota limits, reduce your usage by removing some files and/or archiving any important data on Lou or on the storage space at your local site.

In addition, when the overall usage of a /nobackup filesystem is near its capacity, files can be deleted by system administrators. Normally, the few top users are sent emails and asked to clean up their disk space.

Using Shift to Copy Files to Lou

The NAS /nobackup filesystems (for example, /nobackup[*17-19, 27-29*]) are mounted on Lou. As a result, you can log into Lou and copy the files from your /nobackup filesystem to your home directory on Lou.

We recommend using the Shift tool (shiftc), which will automatically utilize the highest performing local file transport. You can also use shiftc if you need to initiate the transfer from a NAS compute front end, such as a PFE. If you need to encrypt the data, even within the NAS enclave, then use the shiftc --encrypt option.

A Lustre filesystem is a highly scalable, high-performance shared filesystem for Linux clusters that is managed with Lustre software. It can support many thousands of client nodes, petabytes of storage, and hundreds of gigabytes per second of I/O throughput. The NAS Lustre filesystems are named "/nobackupp*X*."

Each Lustre filesystem is actually a set of many small filesystems, which are referred to as object storage targets (OSTs). The Lustre software presents the OSTs as a single unified filesystem.

For more information about OSTs and other Lustre filesystem components, see the Main Lustre Components section below.

Useful Lustre Commands

The examples in this section use /nobackupp*17* as an example of a specific Lustre filesystem.

Listing Disk Usage and Quotas

To display disk usage and limits on your /nobackup directory:

pfe*21*% lfs quota -h -u *username* /nobackupp*17*

or

pfe*21*% lfs quota -h -u *username* /nobackup/*username*

To display usage on each OST, add the -v option:

pfe*21*% lfs quota -h -v -u *username* /nobackup/*username*

Listing Space Usage

To list space usage per OST and MDT in human-readable format for all Lustre filesystems or for a specific one:

pfe*21*% lfs df -h
pfe*21*% lfs df -h /nobackupp*17*

Listing Inode Usage

To list inode usage for all filesystems or for a specific one:

pfe*21*% lfs df -i
pfe*21*% lfs df -i /nobackupp*17*

Listing OSTs

To list all the OSTs for the filesystem:

pfe*21*% lfs osts /nobackupp*17*

Viewing Striping Information

To view the striping information for a specific file or directory:

pfe*21*% lfs getstripe *filename*
pfe*21*% lfs getstripe -d *directory_name*

Note: Omitting the -d flag will display striping for all files in the directory.

File Striping

Files on the Lustre filesystems are striped automatically. This means they are transparently divided into chunks that are written or read simultaneously across a set of OSTs within the filesystem. The chunks are distributed among the OSTs using a method that ensures load balancing. Files larger than 100 gigabytes (GB) *must* be striped in order to avoid taking up too much space on any single OST, which might adversely affect the filesystem.

Striping benefits include:

- Allows one or more clients to read/write different parts of the same file at the same time, providing higher I/O bandwidth to the file because the bandwidth is aggregated over the multiple OSTs.
- Allows file sizes larger than the size of a single OST.

However, striping small files increases the number of objects in each file, resulting in increased overhead due to network operations —sometimes creating server contention. This is not likely to cause problems for files larger than 4 megabytes (MB), but in many

cases, striping can cause noticeably slower read/write performance for files smaller than 4 MB.

Progressive File Layout

Lustre's progressive file layout (PFL) feature enables stripe settings for different segments of a file to be configured in such a way that a small stripe count is used for the beginning segment, and larger stripe counts are used for latter segments as the file grows. All of the NAS Lustre filesystems have been configured with NAS-selected PFL settings.

To learn more about PFL, see Progressive File Layout with SSD and HDD Pools.

Main Lustre Components

Lustre filesystem components include:

Metadata Server (MDS)
: Service nodes that manage all metadata operations, such as assigning and tracking the names and storage locations of directories and files on the OSTs. There is 1 MDS per filesystem.

Object Storage Target (OST)
: Storage devices where users' file data is stored. The size of each OST varies from approximately 7 terabytes (TB) to approximately 22 TB, depending on the Lustre filesystem. The capacity of a Lustre filesystem is the sum of the sizes of all OSTs. There are multiple OSTs per filesystem.

Metadata Target (MDT)
: A storage device where the metadata (name, ownership, permissions and file type) are stored. There is 1 MDT per filesystem.

Object Storage Server (OSS)
: Service nodes that run the Lustre software stack, provide the actual I/O service and network request handling for the OSTs, and coordinate file locking with the MDS. Each OSS can serve multiple OSTs. The aggregate bandwidth of a Lustre filesystem can approach the sum of bandwidths provided by the OSSes. There can be 1 or multiple OSSes per filesystem.

Lustre Clients
: Compute nodes that mount the Lustre filesystem, and access/use data in the filesystem. There are typically thousands of Lustre clients per filesystem.

At NAS, Lustre (/nobackup) filesystems are shared among many users and many application processes, which can cause contention for various Lustre resources. This article explains how Lustre I/O works, and provides best practices for improving application performance.

How Does Lustre I/O Work?

When a client (a compute node from your job) needs to create or access a file, the client queries the metadata server (MDS) and the metadata target (MDT) for the layout and location of the file's stripes. Once the file is opened and the client obtains the striping information, the MDS is no longer involved in the file I/O process. The client interacts directly with the object storage servers (OSSes) and object storage targets (OSTs) to perform I/O operations such as locking, disk allocation, storage, and retrieval.

If multiple clients try to read and write the same part of a file at the same time, the Lustre distributed lock manager enforces coherency so that all clients see consistent results.

Jobs being run on NAS compute systems contend for shared resources in NAS's Lustre filesystem. Each server that is part of a Lustre filesystem can only handle a limited number of I/O requests (read, write, stat, open, close, etc.) per second. An excessive number of such requests, from one or more users and one or more jobs, can lead to contention for storage resources. Contention slows the performance of your applications and weakens the overall health of the Lustre filesystem. To reduce contention and improve performance, please apply the examples below to your compute jobs while working in our high-end computing environment.

Best Practices

Avoid Using ls -l

The ls -l command displays information such as ownership, permission, and size of all files and directories. The information on ownership and permission metadata is stored on the MDTs. However, the file size metadata is only available from the OSTs. So, the ls -l command issues Remote Procedure Calls (RPCs) to the MDS/MDT and OSSes/OSTs for every file/directory to be listed. RPC requests to the OSSes/OSTs are very costly and can take a long time to complete if there are many files and directories.

- Use ls by itself if you just want to see if a file exists
- Use ls -l *filename* if you want the long listing of a specific file

Avoid Having a Large Number of Files in a Single Directory

Opening a file keeps a lock on the parent directory. When many files in the same directory are to be opened, it creates contention. A better practice is to split a large number of files (in the thousands or more) into multiple subdirectories to minimize contention.

Avoid Accessing Small Files on Lustre Filesystems

Accessing small files on the Lustre filesystem is not efficient. When possible, keep them on an NFS-mounted filesystem (such as your NAS compute resources home filesystem /u/*username*) or copy them from Lustre to /tmp on each node at the beginning of the job, and then access them from /tmp.

Keep Copies of Your Source Code on the NAS Compute Resources Home Filesystem and/or Lou

Be aware that files under /nobackup are *not* backed up. Make sure that you save copies of your source codes, makefiles, and any other important files on your NAS compute resources home filesystem. If your NAS compute resources home directory quota isn't large enough to keep all of these files, you can request a larger quota and/or create tarred copies of these files on Lou.

Avoid Accessing Executables on Lustre Filesystems

There have been a few incidents on NAS compute systems where users' jobs encountered problems while accessing their executables on the /nobackup filesystem. The main issue is that the Lustre clients can become unmounted temporarily when there is a very high load on the Lustre filesystem. This can cause a bus error when a job tries to bring the next set of instructions from the inaccessible executable into memory.

Executables run slower when run from the Lustre filesystem. It is best to run executables from your NAS compute resources home filesystem. On rare occasions, running executables from the Lustre filesystem can cause executables to be corrupted. Avoid copying new executables over existing ones of the same name within the Lustre filesystem. The copy causes a window of time (about 20 minutes) where the executable will not function. Instead, the executable should be accessed from your home filesystem during runtime.

Limit the Number of Processes Performing Parallel I/O

Given that the numbers of OSSes and OSTs of various NAS Lustre filesystems are about a hundred or fewer, there will be contention if a large number of processes of an application are involved in parallel I/O. Instead of allowing all processes to do the I/O, choose

just a few processes to do the work. For writes, these few processes should collect the data from other processes before the writes. For reads, these few processes should read the data and then broadcast the data to others.

Understand the Effect of Stripe Counts/Sizes for MPI Collective Writes

For programs that call MPI collective write functions, such as MPI_File_write_all, MPI_File_write_at_all, and MPI_File_write_ordered, it is important to understand the effect of stripe counts on performance.

Background

MPI I/O supports the concept of collective buffering. For some filesystems, when multiple MPI processes are writing to the same file in a coordinated manner, it is much more efficient for the different processes to send their writes to a subset of processes in order to do a smaller number of bigger writes. By default, with collective buffering, the write size is set to be the same as the stripe size of the file.

With Lustre filesystems, there are two main factors in the HPE (SGI) MPT algorithm that chooses the number of MPI processes to do the writes: the stripe count and number of nodes. When the number of nodes is greater than the stripe count, the number of collective buffering processes is the same as the stripe count. Otherwise, the number of collective buffering processes is the largest integer less than the number of nodes that evenly divides the stripe count. In addition, MPT chooses the first rank from the first *n* nodes to come up with *n* collective buffering processes.

Note: Intel MPI behaves similarly to HPE (SGI) MPT on Lustre filesystems.

Enabling Collective Buffering Automatically

You can let each MPI implementation enable collective buffering for you, without any code changes.

HPE (SGI) MPT automatically enables collective buffering for the collective write calls using the algorithm described above. This method requires no changes in the user code or in the mpiexec command line. For example, if the stripe count is 1, only rank 0 does the collective writes, which can result in poor performance. Therefore, experimenting with different stripe counts on the whole directory and/or individual files is strongly recommended.

Intel MPI also does collective buffering, similar to HPE (SGI) MPT, when the I_MPI_EXTRA_FILESYSTEM and I_MPI_EXTRA_FILESYSTEM_LIST variables are set appropriately, as follows:

```
mpiexec.hydra  -env I_MPI_EXTRA_FILESYSTEM on \
      -env  I_MPI_EXTRA_FILESYSTEM_LIST lustre \
      -np xx a.out
```

Enabling Collective Buffering via Code Changes

In this method, you provide "hints" in the source code to inform MPI what to do with specific files. For example:

```
  call MPI_Info_create(info, status)
  call MPI_Info_set(info, "romio_cb_write", "enable", STATUS)
  call MPI_Info_set(info, "striping_unit", "1048576", STATUS)
  call MPI_Info_set(info, "striping_factor", "16", STATUS)
  ...
  call MPI_File_open(MPI_COMM_WORLD, file_name, MPI_MODE_WRONLY, info, unit, status)
```

Note: The hints are only advisory and may not be honored. For example, SGI MPT 2.12r26 honors these hints, but MPT 2.14r19 does not. Intel MPI 5.0x honors these hints when the I_MPI_EXTRA_FILESYSTEM and I_MPI_EXTRA_FILESYSTEM_LIST variables are set appropriately, as follows:

```
mpiexec.hydra  -env I_MPI_EXTRA_FILESYSTEM on \
      -env  I_MPI_EXTRA_FILESYSTEM_LIST lustre \
      -np xx a.out
```

Stripe Align I/O Requests to Minimize Contention

Stripe aligning means that the processes access files at offsets that correspond to stripe boundaries. This helps to minimize the number of OSTs a process must communicate for each I/O request. It also helps to decrease the probability that multiple processes accessing the same file communicate with the same OST at the same time.

One way to stripe-align a file is to make the stripe size the same as the amount of data in the write operations of the program.

Avoid Repetitive "stat" Operations

Some users have implemented logic in their scripts to test for the existence of certain files. Such tests generate "stat" requests to the Lustre server. When the testing becomes excessive, it creates a significant load on the filesystem. A workaround is to slow down the testing process by adding sleep in the logic. For example, the following user script tests the existence of the files WAIT and STOP to decide what to do next.

```
touch WAIT
 rm STOP
```

```
while ( 0 <= 1  )
 if(-e WAIT) then
   mpiexec ...
   rm WAIT
 endif
 if(-e STOP) then
   exit
 endif
end
```

When neither the WAIT nor STOP file exists, the loop ends up testing for their existence as quickly as possible (on the order of 5,000 times per second). Adding sleep inside the loop slows down the testing.

```
touch WAIT
 rm STOP

 while ( 0 <= 1  )
  if(-e WAIT) then
    mpiexec ...
    rm WAIT
  endif
  if(-e STOP) then
    exit
  endif
  sleep 15
 end
```

Avoid Having Multiple Processes Open the Same File(s) at the Same Time

On Lustre filesystems, if multiple processes try to open the same file(s), some processes will not able to find the file(s) and your job will fail.

The source code can be modified to call the sleep function between I/O operations. This will reduce the occurrence of multiple, simultaneous access attempts to the same file from different processes.

```
100  open(unit,file='filename',IOSTAT=ierr)
     if (ierr.ne.0) then
      ...
     call sleep(1)
     go to 100
     endif
```

When opening a read-only file in Fortran, use ACTION='read' instead of the default ACTION='readwrite'. The former will reduce contention by not locking the file.

```
open(unit,file='filename',ACTION='READ',IOSTAT=ierr)
```

Avoid Repetitive Open/Close Operations

Opening files and closing files incur overhead and repetitive open/close should be avoided.

If you intend to open the files for read only, make sure to use ACTION='READ' in the open statement. If possible, read the files once each and save the results, instead of reading the files repeatedly.

If you intend to write to a file many times during a run, open the file once at the beginning of the run. When all writes are done, close the file at the end of the run.

See Lustre Basics for more information.

Use the Soft Link to Refer to Your Lustre Directory

Your /nobackup directory is created on a specific Lustre filesystem, such as /nobackupp17 or /nobackupp18, but you can use a soft link to refer to the directory no matter which filesystem it is on:

/nobackup/*your_username*

By using the soft link, you can easily access your directory without needing to know the name of the underlying filesystem. Also, you will not need to change your scripts or re-create any symbolic links if a system administrator needs to migrate your data from one Lustre filesystem to another.

Preserve Corrupted Files for Investigation

When you notice a corrupted file in your /nobackup directory, it is important to preserve the file to allow NAS staff to investigate the cause of corruption. To prevent the file from being accidentally overwritten or deleted by your scripts, we recommend that you rename the corrupted file using:

pfe% mv *filename filename*.corrupted

Note: Do not use cp to create a new copy of the corrupted file.

Report the problem to NAS staff by sending an email to support@nas.nasa.gov. Include how, when, where the corrupted file was generated, and anything else that may help with the investigation.

See the following articles for more information:

- Lustre Basics - Setting Stripe Parameters
- Shift Command Options

Reporting Problems

If you report performance problems with a Lustre filesystem, please be sure to include the time, hostname, PBS job number, name of the filesystem, and the path of the directory or file that you are trying to access.Your report will help us correlate issues with recorded performance data to determine the cause of efficiency problems.

Filesystems on NAS systems have the following types of quotas:

- Limits on the total disk space occupied by your files
- Limits on the number of files (represented by inodes) you can store, regardless of size. For quota purposes, directories count as files

Hard and Soft Quota Limits

NAS quotas have hard limits and soft limits. Hard limits should never be exceeded. Soft limits can be exceeded temporarily, for a grace period of 14 days. If your data remains over the soft limit for more than 14 days, the soft limit is enforced as a hard limit. On some filesystems, this means that write attempts will fail. On other filesystems, it will mean that your PBS jobs will no longer start. To reduce your data to below the quota limits, you can delete unneeded files or copy important files elsewhere, such as the Lou mass storage system, and then remove them locally.

Filesystem quotas are shown in the following table:

	NAS Compute Resources	**Lou**
$HOME	NFS	XFS
Space: soft	8 GB	none
Space: hard	10 GB	none
Inode: soft	none	250,000
Inode: hard	none	300,000
/nobackup	Lustre /nobackupp*X*	N/A
Space: soft	1 TB	N/A
Space: hard	2 TB	N/A
Inode: soft	500,000	N/A
Inode: hard	600,000	N/A

To learn how to check your disk space, inode usage, and quotas, see the following articles:

- Home Filesystems for NAS Compute Resources
- NAS Lustre Filesystems
- The Lou Mass Storage System

Email Warnings and Consequences

It is expected that your data will exceed your soft limits as needed. When this occurs, you will begin to receive daily emails to inform you of your current disk space and how much of your grace period remains. However, if your data is still over the soft limit or if you reach the hard limit, restrictions are put in place.

Once you exceed your filesystem limits, PBS will return the following error and prevent you from submitting new jobs:

```
qsub: Filesystem quota exceeded! Please reduce your usage or request an increase before submitting.
```

The following table shows the quota enforcement policy for each type of filesystem.

Filesystem	**Grace Period**	**Enforcement**
/nobackupp*X*	2 weeks	Batch jobs won't run
/nobackupnfs2	2 weeks	Writes are disabled
NAS Compute Systems home directory	2 weeks	Writes are disabled
Lou home directory	2 weeks	Writes are disabled

More details are available in the following sections.

Lustre Filesystems (/nobackupp*X*)

If your grace period has expired, then your batch queue access is restricted and no new jobs can be run until your data on the Lustre filesystem is reduced to an amount below the soft limit. Any jobs that are running will continue to run. If you reach the hard limit, your batch access is immediately restricted, but any running jobs will continue.

NAS Compute Systems Home Directory and /nobackupnfs2

If your grace period has expired, then any writes to that filesystem will fail, and any jobs that attempt to write to the filesystem will

fail. You will be unable to write to the filesystem until your data is reduced to an amount below the soft limit. Similarly, if you reach the hard limit, any writes will immediately fail and you will be unable to write to the filesystem until your data is reduced to an amount below the hard limit.

Lou Home Directory

If your grace period has expired, then any writes that attempt to create new files will fail. You will be unable to create any new files until the number of files is reduced to below the soft limit. If you reach the hard limit, any writes that attempt to create new files will fail until the number of files is reduced to below the hard limit.

The maximum size of a tar file moved to Lou should not exceed 2 TB. If you need to archive larger files, please contact the NAS Control Room at support@nas.nasa.gov for assistance.

Note: The Shift tool, which is a convenient way to transfer files to Lou, can created a set of smaller independent tar files from a single large directory. For more information, see Shift File Transfer Overview.

Changing Your Quotas

If an account needs larger quota limits, send an email justification tosupport@nas.nasa.gov. Your request will be reviewed by management for approval.

All NAS systems share several filesystems intended to provide working space for compute jobs. These filesystems, called "nobackup" (/nobackupp*X*), provide more than 90 petabytes (PB) of disk space and serve many thousands of cores. The /nobackup filesystems are Lustre-based filesystems managed under Lustre software version 2.*x*.

Using the Lustre /nobackup Filesystems

As the name suggests, these "nobackup" filesystems are for temporary use, and are not backed up. Lustre can handle many large files, but you cannot use the Lustre filesystems for long-term storage. If you want to save your files, move them to Lou.

Each Lustre filesystem is shared among many users. To learn how to achieve good I/O performance for your applications and avoid impeding the I/O operations of other users, read the related articles listed at the bottom of the page.

WARNING: Any files that are removed from Lustre /nobackup filesystems *cannot* be restored. You should store essential data on the Lou mass storage system (lfe[*5-8*]) or on other, more permanent storage.

Which Lustre Filesystem is Assigned to Me?

Once you are granted a NAS account, one of the Lustre filesystems will be assigned to you. To find out which one, run the ls command as follows:

```
% ls -l /nobackup/your_username
```

In the output, look for the filesystem name (nobackupp*X*). For example, the following output shows that the user's assigned filesystem is /nobackupp17:

```
lrwxrwxrwx 1 root root 19 Sep 23  2021 /nobackup/your_username -> /nobackupp17/your_username
```

Default Quota and Policy on /nobackup Filesystems

Disk space and inode quotas are enforced on the /nobackup filesystems. The default soft and hard quota limits for inodes are 500,000 and 600,000, respectively. Quotas for the disk space are 1 terabyte and 2 terabytes, respectively. To check your disk space, inode usage, and quota on your /nobackup filesystem, run the lfs command as follows:

```
% lfs quota -h -u username /nobackup/username
Disk quotas for user username (uid nnnn):
     Filesystem    used   quota   limit   grace   files   quota   limit   grace
/nobackup/username
                     4k   1.024T  2.048T       -       1  500000  600000       -
```

It is expected that your data will occasionally exceed the soft limit. However, after a 14-day grace period, your access to the batch queues will be restricted and you will not be able to run new jobs until your data is reduced below the soft limit. If you reach the hard limit, your batch access will immediately be restricted, but any running jobs will continue.

Once you exceed your filesystem limits, PBS will return the following error and prevent you from submitting new jobs:

```
qsub: Filesystem quota exceeded! Please reduce your usage or request an increase before submitting.
```

If you anticipate a long-term need for higher quota limits, please send a justification via email to support@nas.nasa.gov. Your request will be reviewed by HECC management for approval.

For more information, see Quota Policy on Disk Space and Files.

Note: When a Lustre filesystem is full, the jobs writing to it will hang. A Lustre error with code -28 in the system log file indicates that the filesystem is full. NAS support staff will typically send emails to those using the most space, with a request to clean up their files.

Lustre Filesystem Configuration: Progressive File Layout

Lustre's Progressive File Layout (PFL) feature enables filesystems to dynamically change the stripe count for files based on their size. This means that the files are dynamically striped, therefore, you should ***not*** set custom stripe size or stripe counts.

The Lustre filesystems are called /nobackupp[*17-19, 27-29*]. They are equipped with a small number of solid state drives (SSDs) in addition to the hard disk drives (HDDs).

The HDD configurations are shown in the following table with the abbreviation nbp*X*. In the table, P = petabytes; T = terabytes.

Lustre PFL HDD Configurations

Filesystem	nbp17,18	nbp19	nbp27,28	nbp29
# of OSTs	32	16	32	16
size/OST	565 T	565 T	565 T	565 T
Total Space	18.6 P	9.3 P	18.6 P	9.3 P

Different PFL configurations are set among the SSD and HDD pools. To learn more, see Progressive File Layout with SSD and HDD Pools.