

# Table of Contents

|  |          |
|--|----------|
| <b><u>Lustre on Pleiades</u></b> .....                             | <b>1</b> |
| <u>Lustre Basics</u> .....   | 1        |
| <u>Pleiades Lustre Filesystems</u> .....                           | 4        |
| <u>Lustre Best Practices</u> .....                                 | 6        |
| <u>Lustre Filesystem Statistics in PBS Output File</u> .....       | 12       |
| <u>Using 'mtar' to Create or Extract Tar Files on Lustre</u> ..... | 13       |

# Lustre on Pleiades

## Lustre Basics

A Lustre filesystem is a high-performance, shared filesystem (managed with the Lustre software) for Linux clusters. It is highly scalable and can support many thousands of client nodes, petabytes of storage and hundreds of gigabytes per second of I/O throughput. On Pleiades, the Lustre filesystems are named `"/nobackupp."`

### Main Lustre components:

- Metadata Server (MDS)

1 or 2 per filesystem; service nodes that manage all metadata operations such as assigning and tracking the names and storage locations of directories and files on the OSTs.

- Metadata Target (MDT)

1 per filesystem; a storage device where the metadata (name, ownership, permissions and file type) are stored.

- Object Storage Server (OSS)

1 or multiple per filesystem; 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 up to ~15 OSTs. The aggregate bandwidth of a Lustre filesystem can approach the sum of bandwidths provided by the OSSes.

- Object Storage Target (OST)

multiple per filesystem; storage devices where the data in user files are stored. Under Linux 2.6 (current OS on Pleiades), each OST can be up to 8TB in size. Under SLES 11, each OST can be up to 16 GB in size. The capacity of a Lustre filesystem is the sum of the sizes of all OSTs.

- Lustre Clients

commonly in the thousands per filesystem; compute nodes that mount the Lustre filesystem, and access/use data in the filesystem.

### File Striping

A user file can be divided into multiple chunks and stored across a subset of the OSTs. The chunks are distributed among the OSTs in a round-robin fashion to ensure load balancing.

Benefits of striping:

- allows one to have a file size larger than the size of an OST
- allows one or more clients to read/write different parts of the same file at the same time and provide higher I/O bandwidth to the file since the bandwidth is aggregated over the multiple OSTs

Drawbacks of striping:

- higher risk of file damage due to hardware malfunction
- increased overhead due to network operations and server contention

There are default stripe configurations for each Lustre filesystem. However, users can set the following stripe parameters for their own directories or files to get optimum I/O performance:

#### 1. stripe\_size

the size of the chunk in bytes; specify with k, m, or g to use units of KB, MB, or GB, respectively; the size must be an even multiple of 65,536 bytes; default is 4MB for all Pleiades Lustre filesystems; one can specify 0 to use the default size.

#### 2. stripe\_count

the number of OSTs to stripe across; default is 1 for most of Pleiades Lustre filesystems (/nobackupp[10-60]); one can specify 0 to use the default count; one can specify -1 to use all OSTs in the filesystem.

#### 3. stripe\_offset

The index of the OST where the first stripe is to be placed; default is -1 which results in random selection; using a non-default value is NOT recommended.

Use the **lfs setstripe** command for setting the stripe parameters.

```
pfe20% lfs setstripe -s stripe_size -c stripe_count -o
stripe_offset dir|filename
```

For example, to create a directory called dir1 with a stripe\_size of 4MB and a stripe\_count of 8, do

```
pfe20% mkdir dir1
pfe20% lfs setstripe -s 4m -c 8 dir1
```

Also keep in mind that:

- When a file or directory is created, it will inherit the parent directory's stripe settings.

- The stripe settings of an *existing file* can not be changed. If you want to change the settings of a file, you can create a new file with the desired settings and copy the existing file to the newly created file.

## Useful Commands for Lustre

- To list all the OSTs for the filesystem

```
pfe20% lfs osts
```

- To list space usage per OST and MDT in human readable format for all Lustre filesystems or for a specific one, for example, /nobackupp1:

```
pfe20% lfs df -h  
pfe20% lfs df -h /nobackupp1
```

- To list inode usage for all filesystems or a specific one, for example, /nobackupp1:

```
pfe20% df -i  
pfe20% df -i /nobackupp1
```

- To create a new (empty) file or set directory default with specified stripe parameters

```
pfe20% lfs setstripe -s stripe_size -c stripe_count -o  
stripe_offset dir|filename
```

- To list the striping information for a given file or directory

```
pfe20% lfs getstripe dir|filename
```

- To display disk usage and limits on your /nobackup directory (for example, /nobackupp1):

```
pfe20% lfs quota -u username /nobackupp1
```

or

```
pfe20% lfs quota -u username /nobackup/username
```

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

```
pfe20% lfs quota -v -u username /nobackup/username
```

See the **lfs man page** for more options and information.

# Pleiades Lustre Filesystems

**Summary:** The Lustre filesystems on Pleiades are called "nobackup." As the name suggests, these filesystems are for temporary use, and are not backed up. Lustre can handle many large files, but you cannot store those files on Pleiades; if you want to save them, move them to Lou.

---

Pleiades has several Lustre filesystems (`/nobackupp[1-6]`) that provide a total of about 6.795 petabytes of storage and serve thousands of cores. These filesystems are managed under Lustre software version 1.8.6.

Lustre filesystem configurations are summarized at the end of this article.

**WARNING:** As the names suggest, these filesystems are not backed up, so any files that are removed *cannot* be restored. Essential data should be stored on Lou[1-2] or on other, more permanent storage.

## Which /nobackup Should I Use?

Once you are granted an account on Pleiades, you will be assigned to use one of the Lustre filesystems. Find out which Lustre filesystem you have been assigned to by typing the following:

```
pfel% ls -l /nobackup/your_username
lrwxrwxrwx 1 root root 19 Feb 23 2010 /nobackup/username -> /nobackupp2/username
```

In the above example, the symlink from `/nobackup` to `/nobackupp2` shows that the user's assigned nobackup system is `/nobackupp2`.

## Default Quota and Policy on /nobackup

Disk space and inodes quotas are enforced on the `/nobackup` filesystems. The default soft and hard quota limits for inodes are 75,000 and 100,000, respectively. Those for the disk space are 500 gigabytes and 1 terabyte, respectively. To check your disk space and inodes usage and quota on your `/nobackup`, use the `lfs` command and type the following:

```
%lfs quota -u username /nobackup/username
Disk quotas for user username (uid nnnn):
  Filesystem  kbytes      quota   limit   grace   files   quota   limit   grace
/nobackup/username 1234  530000000 1100000000 -      567   75000 100000 -
```

The NAS quota policy states that if you exceed the soft quota, an email will be sent that lists your current usage and remaining grace period. It is expected that users will occasionally exceed their soft limit, as needed; however after 14 days, users who are still over their soft

limit will have their batch queue access to Pleiades disabled.

If you anticipate having a long-term need for higher quota limits, please send a justification via email to [support@nas.nasa.gov](mailto:support@nas.nasa.gov). This will be reviewed by the HECC Deputy Project Manager for approval.

For more information, see also, [Quota Policy on Disk Space and Files](#).

NOTE: If you reach the hard limit while your job is running, the job will die prematurely without providing useful messages in the PBS output/error files. A Lustre error with code **-122** in the system log file indicates that you are over your quota.

In addition, 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. The NAS Control Room staff normally will send out emails to those using the most space, asking them to clean up their files.

## Lustre File Systems Configurations

In the table below, /nobackupp[1-6] are abbreviated as nbp[1-6]. P=Petabytes; T=Terabytes

|                            | <b>Pleiades Lustre Configurations</b> |                      |                      |                      |                      |                      |
|----------------------------|---------------------------------------|----------------------|----------------------|----------------------|----------------------|----------------------|
| Filesystem                 | nbp1                                  | nbp2                 | nbp3                 | nbp4                 | nbp5                 | nbp6                 |
| # of MDSEs                 | 1                                     | 1                    | 1                    | 1                    | 1                    | 1                    |
| # of MDTs                  | 1                                     | 1                    | 1                    | 1                    | 1                    | 1                    |
| size of MDTs               | 0.9T                                  | 0.9T                 | 0.6T                 | 0.6T                 | 0.8T                 | 0.9T                 |
| # of usable inodes on MDTs | ~256x10 <sup>6</sup>                  | ~256x10 <sup>6</sup> | ~173x10 <sup>6</sup> | ~173x10 <sup>6</sup> | ~512x10 <sup>6</sup> | ~256x10 <sup>6</sup> |
| # of OSSes                 | 8                                     | 8                    | 8                    | 8                    | 8                    | 8                    |
| # of OSTs                  | 120                                   | 120                  | 60                   | 60                   | 120                  | 120                  |
| size/OST                   | 15T                                   | 15T                  | 7.1T                 | 7.1T                 | 15T                  | 7.1T                 |
| Total Space                | 1.7P                                  | 1.7P                 | 424T                 | 424T                 | 1.7P                 | 847T                 |
| Default Stripe Size        | 4M                                    | 4M                   | 4M                   | 4M                   | 4M                   | 4M                   |
| Default Stripe Count       | 1                                     | 1                    | 1                    | 1                    | 1                    | 1                    |

NOTE: After January 13, 2011, directories without an explicit stripe count and/or stripe size adopted the new stripe count of 1 and stripe size of 4MB. However, old files in that directory retain their old default values. New files that you create in these directories will adopt the new default values.

Each Pleiades Lustre filesystem is shared among many users. To get good I/O performance for your applications and avoid impeding the I/O operations of other users, read the related articles listed below.

# Lustre Best Practices

**Summary:** At NAS, Lustre filesystems (/nobackup) are shared among many users and many application processes, which causes 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 Pleiades contend for shared resources in NAS's Lustre filesystem. The Lustre server can only handle about 15,000 remote procedure calls (RPCs, inter-process communications that allow the client to cause a procedure to be executed on the server) per second. 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 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 home filesystem on Pleiades `/u/username`) or copy them from Lustre to `/tmp` on each node at the beginning of the job, and then access them from `/tmp`.

## Use a Stripe Count of 1 for Directories with Many Small Files

If you must keep small files on Lustre, be aware that `stat` operations are more efficient if each small file resides in one OST. Create a directory to keep small files, set the stripe count to 1 so that only one OST will be needed for each file. This is useful when you extract source and header files (which are usually very small files) from a tarfile. Use the Lustre utility `lfs` to create a specific striping pattern, or find the striping pattern of existing files.

```
pfel% mkdir dir_name
pfel% lfs setstripe -s 1m -c 1 dir_name
pfel% cd dir_name
pfel% tar -xf tarfile
```

If there are large files in the same directory tree, it may be better to allow them to stripe across more than one OST. You can create a new directory with a larger stripe count and copy the larger files to that directory. Note that moving files into that directory with the `mv` command will not change the strip count of the files. Files must be *created in* or *copied* to a directory to inherit the stripe count properties of a directory.

```
pfel% mkdir dir_count_4
pfel% lfs setstripe -s 1m -c 4 dir_count_4
pfel% cp file_count_1 dir_count_4
```

If you have a directory with many small files (less than 100 MB) and a few very large files (greater than 1 GB), then it may be better to create a new subdirectory with a larger stripe count. Store just the large files and create symbolic links to the large files using the `symlink` command `ln`.

```
pfel% mkdir bigstripe
pfel% lfs setstripe -c 16 -s 4m bigstripe
pfel% ln -s bigstripe/large_file large_file
```

## Use mtar for Creating or Extracting a tar file

A modified `gnu tar` command, `/usr/local/bin/mtar`, is Lustre stripe aware and will create tar files or extract files with appropriately sized stripe counts. Currently, the number of stripes is set to the number of gigabytes of the file.

## Keep Copies of Your Source Code on the Pleiades Home Filesystem and/or Lou

Be aware that files under `/nobackup[p1-p6]` are *not* backed up. Make sure that you have copies of your source codes, makefiles, and any other important files saved on your Pleiades home filesystem or on Lou, the NAS storage system.

## Avoid Accessing Executables on Lustre Filesystems

There have been a few incidents on Pleiades 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 home filesystem on Pleiades. 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.

## Increase the `stripe_count` for Parallel Writes to the Same File

When multiple processes are writing blocks of data to the same file in parallel, the I/O performance for large files will improve when the `stripe_count` is set to a larger value. The stripe count sets the number of OSTs the file will be written to. By default, the stripe count is set to 1. While this default setting provides for efficient access of metadata--for example to support the `ls -l` command--large files should use stripe counts of greater than 1. This will increase the aggregate I/O bandwidth by using multiple OSTs in parallel instead of just one. A rule of thumb is to use a stripe count approximately equal to the number of gigabytes in the file.

Another good practice is to make the stripe count be an integral factor of the number of processes performing the write in parallel, so that you achieve load balance among the

OSTs. For example, set the stripe count to 16 instead of 15 when you have 64 processes performing the writes.

## Limit the Number of Processes Performing Parallel I/O

Given that the numbers of OSSes and OSTs on Pleiades 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.

## 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 be 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 also: [Lustre Basics](#)

## **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.

# Lustre Filesystem Statistics in PBS Output File

For a PBS job that reads or writes to a Lustre file system, a Lustre filesystem statistics block will appear in the PBS output file, just above the job's PBS Summary block. Information provided in the statistics can be helpful in determining the I/O pattern of the job and assist in identifying possible improvements to your jobs.

The statistics block lists the job's number of Lustre operations and the volume of Lustre I/O used for each file system. The I/O volume is listed in total, and is broken out by I/O operation size.

The following Metadata Operations statistics are listed:

- Open/close of files on the Lustre file system
- Stat/statfs are query operations invoked by commands such as `ls -l`
- Read/write is the total volume of I/O in gigabytes

The following is an example of this listing:

```
=====
                        LUSTRE Filesystem Statistics
-----
nbp10 Metadata Operations
      open      close      stat      statfs      read(GB)      write(GB)
      1057      1058      1394         0             2             14
Read   4KB   8KB  16KB  32KB  64KB  128KB  256KB  512KB  1024KB
      9     3     1     0     1     0     3     2     319
Write  4KB   8KB  16KB  32KB  64KB  128KB  256KB  512KB  1024KB
      138   13     1    11    36     9    21    37   12479
-----
Job Resource Usage Summary for 11111.pbsp11.nas.nasa.gov

CPU Time Used           : 00:03:56
Real Memory Used        : 2464kb
Walltime Used           : 00:04:26
Exit Status              : 0
```

The read and write operations are further broken down into buckets based on I/O block size. In the example above, the first bucket reveals that nine data reads occurred in blocks between 0 and 4 KB in size, three data reads occurred with block sizes between 4 KB and 8 KB, and so on. The I/O block size data may be affected by library and system operations and, therefore, could differ from expected values. That is, small reads or writes by the program might be aggregated into larger operations, and large reads or writes might be broken into smaller pieces. If there are high counts in the smaller buckets, you should investigate the I/O pattern of the program for efficiency improvements.

For tips for improving Lustre I/O, see [Lustre Best Practices](#) for multiple tips to improve the Lustre I/O performance of your jobs.

# Using 'mtar' to Create or Extract Tar Files on Lustre

NAS's in-house developed **mtar** program is a modification of GNU **tar** version 1.25. It is exactly equivalent to **tar** except that, if it detects a Lustre filesystem, then it restripes files as they are "tarred" and/or "untarred" for better performance. Specifically:

- The stripe count of files extracted on a Lustre filesystem will be dynamically selected based on the original file size, so, small files will be extracted with small stripe counts and large files will be extracted with large stripe counts
- Tar files created on a Lustre file system will have a stripe count based on the sum of the sizes of all component files

**TIP:** We recommend using **mtar** in place of **tar** when creating or extracting from a tar file on Lustre.

Currently, the number of stripes set by **mtar** is essentially the number of gigabytes of that file (for disk storage, 1 GB = 10<sup>9</sup> bytes), limited by the number of object storage targets in that Lustre filesystem.

Tar files created with **gzip** (**-z**), **bzip2** (**-j**), and arbitrary compression (**--use-compress-program**) will preserve the striping of the uncompressed tar file.

## Using mtar

**mtar** is available in `/usr/local/bin` on the Pleiades front-ends (pfe[20-27], bridge[1-4]). Usage of **mtar** is exactly the same as **tar** and you don't have to know anything extra, as it all happens automatically.

The following example demonstrates its usage and the comparison between **mtar** and **tar**. Note that some output has been removed for clarity.

```
%ls -l *_file
-rw----- 1 zsmith s0101 16800000112 Aug  3 14:58 17g_file
-rw----- 1 zsmith s0101  1200000008 Aug  3 14:51 2g_file
-rw----- 1 zsmith s0101          1215 Aug  3 15:04 2k_file

%lfs getstripe *_file
17g_file
lmm_stripe_count:  1
2g_file
lmm_stripe_count:  1
2k_file
lmm_stripe_count:  1
```

Notice that the default stripe count is 1 on all Pleiades Lustre filesystems.

### Comparison of tar and mtar

| <b>tar</b>  | <b>mtar</b>   |
|---|---|
| <pre>%tar cvf tar.tar 17g_file 2g_file 2k_file %lfs getstripe tar.tar tar.tar lmm_stripe_count: 1</pre>                                 | <pre>%mtar cvf mtar.tar 17g_file 2g_file 2k_file %lfs getstripe mtar.tar mtar.tar lmm_stripe_count: 19</pre>                              |
| <pre>%tar xvf tar.tar %lfs getstripe *_file 17g_file lmm_stripe_count: 1 2g_file lmm_stripe_count: 1 2k_file lmm_stripe_count: 1</pre>  | <pre>%mtar xvf mtar.tar %lfs getstripe *_file 17g_file lmm_stripe_count: 17 2g_file lmm_stripe_count: 2 2k_file lmm_stripe_count: 1</pre> |
| <pre>%tar xvf mtar.tar %lfs getstripe *_file 17g_file lmm_stripe_count: 1 2g_file lmm_stripe_count: 1 2k_file lmm_stripe_count: 1</pre> | <pre>%mtar xvf tar.tar %lfs getstripe *_file 17g_file lmm_stripe_count: 17 2g_file lmm_stripe_count: 2 2k_file lmm_stripe_count: 1</pre>  |
| <pre>%tar zcvf tar.tgz tar.tar %lfs getstripe tar.tgz tar.tgz lmm_stripe_count: 1</pre>   | <pre>%mtar zcvf mtar.tgz mtar.tar %lfs getstripe mtar.tgz mtar.tgz lmm_stripe_count: 19</pre>   |

Notice that the **tar**-created archive has a default stripe count, while the **mtar**-created archive has a stripe count based on the sizes of component files. In addition, **tar**-extracted files all have a default stripe count, while **mtar**-extracted files have a variable stripe count depending on size. Also notice that using **mtar** with compression preserves striping of the uncompressed tar file.

The **mtar** script was created by NAS staff member Paul Kolano.