The Endeavour supercomputer comprises two nodes, Endeavour3 and Endeavour4. Each of the two systems includes 896 cores and six terabytes (TB) of global shared memory. See Endeavour Configuration Details for more hardware information.

Endeavour 3 & 4 are accessible through the PFEs and are served by PBS server pbspl4. They have access to the NAS compute resources home filesystem and NAS Lustre nobackup filesystems.

Update to Endeavour Support Model: Due to the age of Endeavour's hardware, we are lowering the priority of those nodes in our monthly patching work. This means that Endeavour may not always be returned to service at the same time as newer systems, and the patch windows may need to be extended to account for additional time needed to bring the nodes online.

Operating System

Endeavour 3 & 4 are running the Red Hat Enterprise Linux-based Tri-Lab Operating System Stack (TOSS4).

Connecting to Endeavour

To access the systems, submit PBS jobs from the PFEs with the server name pbspl4 included in the qsub command line or PBS script. You cannot log directly into Endeavour3 and 4 from the PFEs unless you have a PBS job running there.

Accessing Your Data

Use your NAS home filesystems (/u/*username*) and your Lustre filesystem (/nobackup) for Endeavour.

For more information on using Lustre filesystems, see Lustre Basics and Lustre Best Practices.

Compiling and Running Your Code

There are no default modules for compilers, Message Passing Interface (MPI) components, or math libraries. To use these modules, you must first load them.

OpenMP Applications

Load an Intel compiler module and use the Intel compiler -qopenmp option to build the executable. Since the Cascade Lake processors used in Endeavour3 and 4 support AVX512, you can experiment with different compiler versions and flags for better performance. For example:

```
% module load comp-intel/2020.0.166
% ifort -O2 -qopenmp program.f
```

or

```
% module load comp-intel/2020.4.304
% ifort -Ofast -ipo -axCORE-AVX512,CORE-AVX2,AVX -xSSE4.2 -qopenmp program.f
```

Set a specific number of OpenMP threads in your PBS job, and use a thread-pinning method to ensure that the threads do not get in the way of each other on the same core or migrate from one core to another. For example:

```
#PBS ...

module load comp-intel/2020.4.304
setenv OMP_NUM_THREADS 28

/u/scicon/tools/bin/mbind.x -cs -t28 -v ./a.out
```

or

```
setenv KMP_AFFINITY compact # or: setenv KMP_AFFINITY scatter
./a.out > output
```

or

```
setenv KMP_AFFINITY disabled
dplace -x2 ./a.out > output
```

See Process/Thread Pinning Overview for more information about different pinning methods.

Note: The default OMP_STACKSIZE is set to 200 megabytes (MB). If your application experiences segmentation fault (segfault), experiment with increasing the $OMP_STACKSIZE value to see if it helps.

Applications that Require Scientific and Math Libraries

The Intel Math Kernel Library (MKL) is included in Intel compiler modules (versions 11.1 and later). SeeMKL to learn how to link to

MKL libraries. In many cases, the following commands may be sufficient:

```
% module load comp-intel/2020.4.304
% ifort -O2 program.f -mkl
```

Note: By itself, the -mkl option implies -mkl=parallel, which will link to the threaded MKL library. If your application can benefit from using multiple OpenMP threads when the MKL routines are called, you should also set the environment variable OMP_NUM_THREADS, as shown in the OpenMP example in the previous section.

If you want to use a non-threaded MKL library, change the -mkl option to -mkl=sequential.

MPI Applications

HPE's latest Message Passing Toolkit (MPT) library is recommended. Load the library as shown in the following example:

```
% module load comp-intel/2020.4.304
% module load mpi-hpe/mpt
% ifort -O2 program.f -lmpi
```

During runtime, load the Intel compiler and the MPT modules and use mpiexec to launch the executable:

```
#PBS ...
module load comp-intel/2020.4.304
module load mpi-hpe/mpt

mpiexec -np xx ./a.out > output
```

Running PBS Jobs

Each of the Endeavour nodes has 32 sockets, and each socket includes 28 cores and 192 gigabytes (GB) of physical memory. In each socket, PBS has access to 28 cores and 185 GB of memory, which is known as a minimum allocatable unit (MAU). In addition, one of the sockets is set aside for system operations, leaving 31 sockets available for user jobs.

PBS does not allow jobs to share resources in the same socket. However, the amount of resources in a socket that a job will receive is restricted to the amount requested. This restriction is enforced by the Linux kernel cgroups feature. For example, if you request one core and 100 GB of memory, one socket will be assigned exclusively for your job, but 27 cores and 85 GB of memory of the socket will not be accessible by your job.

When job accounting is enabled, your job will be charged by the number of MAUs assigned exclusively for your job.

The maximum amount of resources a job can request is:

- **Endeavour3:** 868 cores with 5,750 GB are the maximum resources available for PBS jobs.
- **Endeavour4:** 868 cores with 5,750 GB are the maximum resources available for PBS jobs.

Endeavour PBS Queues

The PBS server pbspl4 manages jobs for Endeavour3 and 4 (:model=cas_end) and the NAS A100 GPU nodes (:model=[mil_a100]). Among the queues listed by the command qstat -Q @pbspl4, you can run jobs on the two systems using the e_normal, e_long, e_vlong, and e_debug queues. You can also find detailed settings of each queue, such as the maximum walltime and the default or minimum ncpus and memory, using the following command line (the e_normal queue is used in the example):

```
pfe% qstat -fQ e_normal@pbspl4
```

PBS Commands

Run PBS commands (such as qsub, qstat, and qdel, etc.) from the PFEs and specify the PBS server, pbspl4. For example:

```
pfe21% qsub -q queue_name@pbspl4 job_script
pfe21% qstat -nu username @pbspl4
pfe21% qstat job_id.pbspl4
pfe21% qdel job_id.pbspl4
```

If you do not normally run jobs on Aitken and Electra (which also use the PFEs to submit jobs), and your primary workload is on Endeavour3 and Endeavour4, you may want to consider setting the PBS_DEFAULT environment variable to pbspl4. This allows you to simplify the PBS commands, as follows:

```
pfe21% setenv PBS_DEFAULT pbspl4
pfe21% qsub job_script
pfe21% qstat -nu username
pfe21% qstat job_id
pfe21% qdel job_id
```

Job Submission Examples

The following examples demonstrate how PBS allocates resources for your job.

Note: If you do not specify a resource attribute (such as ncpus or mem), a default value for that attribute will be assigned to your job.

- To request using all the resources in two MAUs:

  pfe*21*% qsub -I -lselect=2:ncpus=28:mem=185G:model=cas_end -q *queue_name*@pbspl4

- To request 56 cores:

  pfe*21*% qsub -I -lselect=1:ncpus=56:model=cas_end -q *queue_name*@pbspl4

  Your job will be given access to 56 cores across in two sockets and only the default amount of memory (32G).

- To request 1,850 GB of memory:

  pfe*21*% qsub -I -lselect=1:mem=1850G:model=cas_end -q *queue_name*@pbspl4

  Your job will be given access to 1,850 GB of memory spread in ten sockets and only the default number of cores (one).

- If you do not specify which Endeavour node to run your job, PBS will assign one for you. The following example specifies Endeavour3 and requests 28 cores and 185 GB of memory:

  pfe*21*% qsub -I -lselect=host=endeavour3:ncpus=28:mem=185G:model=cas_end -q *queue_name*@pbspl4

Job Accounting

The Standard Billing Unit (SBU) rate for using one MAU per hour is 1.31.

Your group's HECC SBU allocation is charged for usage on any HECC resources, including Electra, Aitken, and Endeavour. The output from the acct_ytd command does not have separate entries for the different systems. To find Endeavour3 and Endeavour4 usage for your group, use the acct_query command instead.

For example, to view the statistics of all of your jobs that ran on Endeavour3 on March 10, 2021, type:

pfe*21*% acct_query -c endeavour3 -d 03/10/21 -u *your_username* -olow

See Job Accounting Utilities to learn more about using the acct_ytd and acct_query tools.

Endeavour jobs should be submitted from the PFEs with the specification of the PBS server pbspl4 using one of the two methods described below. Without specifying pbspl4, your Endeavour job will be rejected by the default PBS server, pbspl1, for jobs submitted from the PFEs.

Specify pbspl4 in the Command Line

You can explicitly add the server name pbspl4 to your command line. For example:

pfe*25*% qsub -q *queue_name*@pbspl4 *job_script*

2468.pbspl4.nas.nasa.gov

In the example above, a queue is specified. If you submit the job without specifying a queue, PBS will assign the job to an appropriate Endeavour queue. For example:

pfe*25*% qsub -q @pbspl4 *job_script* 2468.pbspl4.nas.nasa.gov

Additional examples of PBS command lines that specify the pbspl4 server:

pfe*25*% qstat -nu *username* @pbspl4

pfe*25*% qstat 2468.pbspl4@pbspl4

pfe*25*% qalter -lwalltime=2:00:00 2468.pbspl4@pbspl4

pfe*25*% qdel 2468.pbspl4@pbspl4

Change the Default Server to pbspl4

If your primary workload is on Endeavour, and you do not normally run jobs other NAS distributed memory cluster systems, set the environment variable PBS_DEFAULT to pbspl4. This allows you to omit pbspl4 from your command lines. For example:

pfe*25*% setenv PBS_DEFAULT pbspl4

pfe*25*% qsub -q *queue_name job_script*

2468.pbspl4.nas.nasa.gov

To submit a job without specifying a queue:

pfe*25*% qsub *job_script*

2468.pbspl4.nas.nasa.gov

Additional examples of PBS command lines you can use after you set the PBS_DEFAULT environment variable to pbspl4:

pfe*25*% qstat -nu *username*

pfe*25*% qstat 2468

pfe*25*% qalter -lwalltime=2:00:00 2468

pfe*25*% qdel 2468

Queue Structure

Each Endeavour node contains 896 cores with six terabytes (TB) of memory.

To use these resources, submit PBS jobs to one of the Endeavour queues. Endeavour queue names begin with an "e_" prefix.

View the general queues for Endeavour by using the qstat command:

pfe*25*% qstat -Q @pbspl4 | grep e_

```
Queue    Ncpus/      Time/         State counts
name      max/def    max/def   jm T/Q/H/W/R/E/B pr
======== =====/=== ======/===== == ============= ==
e_normal    --/  8  08:00/01:00 -- 0/0/0/0/0/0/0  0
e_long      --/  8  72:00/36:00 -- 0/0/0/0/3/0/0  0
e_vlong     --/  8 600:00/24:00 -- 0/0/0/0/0/0/0  0
e_debug    128/  8  02:00/00:30 -- 0/0/0/0/0/0/0 15
```

Among the Endeavour queues, the higher-priority e_debug queue is the only one that has a limit on the maximum number of cores (128 cores) per job. If you want to use the e_debug queue, you must specify -q e_debug either in the qsub command line or inside the

PBS script.

To view more information about each Endeavour queue:

pfe*25*% qstat -fQ *queue_name*@pbspl4

The PBS batch scheduler supports a feature called shrink-to-fit (STF). This feature allows you to specify a range of acceptable wall times for a job, so that PBS can run the job sooner than it might otherwise. STF is particularly helpful when scheduling jobs before an upcoming dedicated time.

For example, suppose your typical job requires 5 days of wall time. If there are fewer than 5 days before the start of dedicated time, the job won't run until after dedicated time. However, if you know that your job can do enough useful work running for 3 days or longer, you can submit it in the following way:

```
qsub -l min_walltime=72:00:00,max_walltime=120:00:00 job_script
```

When PBS attempts to run your job, it will initially look for a time slot of 5 days. When no such time slot is found between now and the dedicated time, it will look for shorter and shorter time slots, down to the minimum wall time of 3 days.

If you have an existing job that is still queued, you can use the `qalter` command to add these `min_walltime` and `max_walltime` attributes:

```
qalter -l min_walltime=hh:mm:ss,max_walltime=hh:mm:ss job_id
```

You can also use the `qalter` command to change the wall time:

```
qalter -l walltime=hh:mm:ss job_id
```

Note: For jobs on Endeavour, be sure to include both the job sequence number and the PBS server name, pbspl4, in the *job_id* (for example, 2468.pbspl4).

If you have any questions or problems, please contact the NAS Control Room at (800) 331-8737, (650) 604-4444, or by email at support@nas.nasa.gov.

During job execution, it can be useful to find out the amount of time remaining for your PBS job. This allows you to decide if you want to gracefully dump restart files and exit before PBS kills the job.

If you have an MPI code, you can call MPI_WTIME and see if the elapsed wall time has exceeded some threshold in order to decide if the code should go into the shutdown phase.

For example:

```
include "mpif.h"

real (kind=8) :: begin_time, end_time

begin_time=MPI_WTIME()
do work
end_time = MPI_WTIME()

if (end_time - begin_time > XXXXX) then
   go to shutdown
endif
```

In addition, the following library has been made available on NAS compute systems for the same purpose:

```
/u/scicon/tools/lib/pbs_time_left.a
```

To use this library in your Fortran code, you need to:

1. Modify your Fortran code to define an external subroutine and an integer*8 variable

   ```
   external pbs_time_left
   integer*8 seconds_left
   ```

2. Call the subroutine in the relevant code segment where you want the check to be performed

   ```
   call pbs_time_left(seconds_left)
   print*,"Seconds remaining in PBS job:",seconds_left
   ```

   Note: The return value from `pbs_time_left` is only accurate to within a minute or two.

3. Compile your modified code and link with the above library using, for example:

   ```
   LDFLAGS=/u/scicon/tools/lib/pbs_time_left.a
   ```