To help you prepare for running jobs on Aitken's Milan nodes, this short user guide includes information on the general configuration of Milan nodes, compiling your code, and running PBS jobs.

Aitken Milan Front-End Systems

Two new Aitken front ends, afe01 and afe02, serve the Aitken Milan nodes. Each AFE includes one or two AMD EPYC 7763 processors with a total of 128 physical cores (sum of two sockets), and 512 gigabytes (GB) of memory.

Note: The Aitken Milan compute nodes are not accessible from the PFEs.

You can access the AFEs using one of the following methods.

Initiating SSH from Your Local Desktop (Recommended)

- Two-step login:

  ```
  local_desktop% ssh sfe6.nas.nasa.gov  (or replace sfe6 with sfe7 or sfe8)
  sfe6% ssh afe01    (or ssh afe02)
  ```

- One-step login (using SSH passthrough):

  Before you begin: Follow the instructions in Setting Up SSH Passthrough, and make sure that the configuration file in the .ssh directory of your local desktop contains the following lines:

  ```
  Host afe01
     HostKeyAlias          afe01.nas.nasa.gov
     ProxyCommand          ssh -ax -oCompression=no sfe ssh-balance %h

  Host afe02
     HostKeyAlias          afe02.nas.nasa.gov
     ProxyCommand          ssh -ax -oCompression=no sfe ssh-balance %h
  ```

  Once you have SSH Passthrough set up, you can log into the AFE in one step:

  ```
  local_desktop% ssh afe01   (or ssh afe02)
  ```

Initiating SSH from a PFE

Note: This method is not recommended but is made available for convenience.

```
pfe% ssh afe01 (or ssh afe02)
```

Aitken Milan Compute Nodes

Aitken includes 512 Milan compute nodes, which are partitioned into four physical racks. Each node contains two 64-core AMD EPYC 7763 Milan sockets (2.45 GHz) and 512 GB of memory.

These nodes are inter-connected with four-lane High Data Rate (4X HDR) devices and four-lane High Data Rate (4X HDR) switches. They are currently housed in the NAS main building and have access to the NAS compute resources home and Lustre filesystems.

Note: For the long term, the Milan nodes will be relocated to be near the existing Aitken resources when a new compute module becomes available. At that time, these nodes will be connected to the Rome nodes with additional HDR cables on the 8th dimension of the Aitken cluster hybercube.

Operating System and Software Modules

The operating system on all Milan compute nodes is the Red Hat Enterprise Linux-based Tri-Lab Operating System Stack Version 4 (TOSS 4). Batch jobs use :aoe=toss4 by default, so you do not need to add :aoe=toss4 in your PBS resource request. The default $MODULEPATH includes:

- /usr/share/Modules/modulefiles
- /nasa/modulefiles/toss4
- /nasa/modulefiles/spack/
- /nasa/modulefiles/pkgsrc/toss4/

Compilers For Milan Nodes

There are several compilers you can use to build your application:

Intel Compilers

Some Intel compiler modules are available in the /nasa/modulefiles/toss4 directory, however, several are in the /nasa/modulefiles/testing directory. To see what versions are available in these two directories, run:

```
module use -a /nasa/modulefiles/testing
module avail comp-intel
```

To build your application using the Intel compilers, use `-march=core-avx2`.

Existing executables that are built for the Aitken Rome nodes should run on the Milan nodes.

Note: Although AMD recommends using `-axCORE-AVX2`, we do not recommend using it, as the resulting executable will run but will not actually take advantage of AVX2 instructions. Executables built with `-xCORE-AVX2`, `-xSSE4.2`, and so on will not run on Milan processors.

GNU Compilers

The default version of GNU Compiler Collection (GCC) under TOSS 4 (/usr/bin/gcc) is GCC 8.5.0. However, support for the Rome architecture is only available with GCC 11.xx and later. GCC version 13.2 has been installed and is accessible via:

```
module use -a /nasa/modulefiles/testing
module load gcc/13.2
```

The recommended compiler flag for Milan with gcc is: `-march=znver3`

AMD Optimizing C/C++ Compilers (AOCC)

To see available versions before loading an AOCC compiler, use the `module avail comp-aocc` command. For example:

```
module use -a /nasa/modulefiles/testing
module avail comp-aocc
---------- /nasa/modulefiles/testing ----------
comp-aocc/4.0.0  comp-aocc/4.1.0  cmp-aocc/4.2.0

---------- /nasa/modulefiles/toss4 ------------
comp-aocc/3.2.0
```

The AOCC C, C++ and Fortran compiler drivers are clang, clang++ and flang, respectively.

The recommended compiler flag for Milan with AOCC is: `-march=znver3`

Note: The default optimization level is `-O1` in AOCC 3.2, and `-O2` in AOCC 4.1 and newer versions.

For more compiler options, see AMD's article Compiler Options Quick Reference Guide for AMD EPYC 7xx3 Series Processor.

MPI Libraries For Milan Nodes

HPE MPT

Use the modulefile `mpi-hpe/mpt`, which is available in the /nasa/modulefiles/toss4 directory:

```
module load mpi-hpe/mpt
```

OpenMPI

There are a few OpenMPI modulefiles available in the /nasa/modulefiles/testing directory.

```
module use -a /nasa/modulefiles/testing
module avail openmpi
---- /nasa/modulefiles/testing ----
openmpi/4.1.5-toss4-intel_2022.1  openmpi/4.1.6-toss4-gnu  openmpi/4.1.6-toss4-aocc
```

Note: There are several newer versions that are not listed in the output above.

The compiler used to build each of these modulefiles is indicated in their filenames. For example, `openmpi/4.1.6-toss4-gnu` was built with gcc 8.5.0, and `openmpi/4.1.6-toss4-aocc` was built with aocc 4.2.0.

After loading one of these modulefiles, run the command `ompi_info` to learn more about each installation.

Running OpenMP or MPI/OpenMPI Hybrid Codes

Each Milan node has 128 physical cores. When simultaneous multi-threading (SMT) is enabled in BIOS, there are 256 logical cores for use by OpenMP or MPI/OpenMPI hybrid codes.

To check whether SMT is enabled, run the following command on a Milan compute node:

```
cat /proc/cpuinfo | grep processor | wc -l
```

If the output shows 256, then SMT is enabled.

Note: During Milan's early production, some nodes were configured with the SMT on and some with the SMT off. Currently, all nodes have been set with SMT on.

We recommend using the mbind.x tool for process and thread binding, as shown in the following example with four MPI processes and 32 OpenMP threads per process:

```
mpiexec -n 4 /u/scicon/tools/bin/mbind.x -t 32 a.out
```

If you use HPE MPT, you can also use the omplace tool for pinning MPI/OpenMP hybrid code.

Running PBS Jobs on Aitken Milan Nodes

Aitken's Milan nodes are accessible only through PBS jobs submitted from afe01 and afe02 and served by the PBS server pbspl1.

To request Aitken Milan nodes, use :model=mil_ait in your PBS script, as shown in the example below.

Sample PBS Script For Aitken Milan Nodes

```
#PBS -l select=2:ncpus=128:mpiprocs=128:model=mil_ait
#PBS -l walltime=8:00:00
#PBS -q normal
module use -a /nasa/modulefiles/testing
module load mpi-hpe/mpt
module load comp-intel
cd $PBS_O_WORKDIR
mpiexec -np 256 ./a.out
```

For more information, see AMD Milan Processors.

To help you prepare for running jobs on Aitken's Rome nodes, this short user guide includes information on the general configuration of Rome nodes, compiling your code, and running PBS jobs.

Overview of Aitken Rome Nodes

Aitken includes 2,008 Rome nodes, which are partitioned into sixteen physical racks. Each node contains two 64-core AMD EPYC 7742 Rome sockets (2.25 GHz) and 512 GB of memory. The nodes are connected to the Aitken InfiniBand network (ib0 and ib1) via four-lane High Data Rate (4X HDR) devices and four-lane High Data Rate (4X HDR) switches for internode communication. The ib1 fabric is used mainly for I/O and is connected to the Lustre filesystems. In addition, Aitken Rome nodes share the same home filesystems, PFEs, and PBS server as Electra, and the Aitken Cascade Lake and Milan nodes. You can access Aitken Rome nodes only through PBS jobs submitted from the PFEs.

Note: Aitken Milan nodes use a different set of front ends. See Preparing to Run on Aitken Milan Nodes for more information.

Operating System and Software Modules

The operating system on all Rome compute nodes is the Red Hat Enterprise Linux-based Tri-Lab Operating System Stack Version 4 (TOSS 4). Batch jobs use `:aoe=toss4` by default, so you do not need to add `:aoe=toss4` in your PBS resource request. The default $MODULEPATH includes:

- /usr/share/Modules/modulefiles
- /nasa/modulefiles/toss4
- /nasa/modulefiles/spack/
- /nasa/modulefiles/pkgsrc/toss4/

Compilers for Rome Nodes

Note: Some executables built under the TOSS 3 environment will work under TOSS 4; however, some may not. If you need to rebuild your executable under TOSS 4, and you experience issues such as missing certain software packages or libraries, send an email to support@nas.nasa.gov

There are several compilers you can use to build your application:

- Intel Compilers:

  Some Intel compiler modules are available in the `/nasa/modulefiles/toss4` directory, however, several are in the `/nasa/modulefiles/testing` directory. To see what versions are available in these two directories, run:

  ```
  module use -a /nasa/modulefiles/testing
  module avail comp-intel
  ```

  Existing executables that are built with support for AVX512 (for running on Skylake and Cascade Lake processors) will not run on the Rome nodes. Others that are built with support for AVX2 may run on the Rome nodes; however, you should always check the result of your runs using existing executables. To rebuild your application using Intel compilers, use `-march=core-avx2`.

  Note: Although AMD recommends using -axCORE-AVX2, we do *not* recommend using it, as the resulting executable will run but will not actually take advantage of AVX2 instructions. Executables built with -xCORE-AVX2, -xSSE4.2, etc., will not run on Rome processors.
- GNU Compilers:

  The default version of GNU Compiler Collection (GCC) under TOSS 4 (/usr/bin/gcc) is GCC 8.5.0. However, support for the Rome architecture is only available with GCC 9.x and later. Among the several GCC versions available in the /nasa/pkgsrc/toss4 directory, load the 9.3 or 10.3 version for use with the Rome nodes. For example:

  ```
  module avail gcc
  -- /nasa/modulefiles/pkgsrc/toss4/ --
  gcc/6.5  gcc/7.5  gcc/8.4  gcc/9.3  gcc/10.3  gcc/13.2
  module load gcc/10.3
  ```

  The recommended compiler flag to use with GCC is `-march=znver2`.
- AMD Optimizing C/C++ and Fortran Compilers (AOCC)

  To see available versions before loading an AOCC compiler, use the `module avail comp-aocc` command. For example:

  ```
  module avail comp-aocc
  -- /nasa/modulefiles/toss4 --
  comp-aocc/3.2.0

  -- /nasa/modulefiles/testing --
  comp-aocc/4.0.0  comp-aocc/4.1.0  comp-aocc/4.2.0  comp-aocc/5.0.0

  module load comp-aocc/3.2.0
  ```

  The recommended compiler flag to use with AOCC is `-march=znver2`.

- PGI Compilers:

  Some PGI compiler modules are available in the /nasa/modulefiles/testing directory. There is no support specifically for Rome processors. If you want to compile your code using a PGI compiler, the recommended compiler flag is -tp=zen

For more compiler options, see Compiler Options Quick Reference Guide for AMD EPYC 7xx2 Series Processors.

MPI Library for Rome Nodes

HPE MPT is the only recommended MPI library for use on the AMD Rome nodes under TOSS 4. Use thempi-hpe/mpt.2.30 modulefile in the /nasa/modulefiles/toss4 directory. Earlier versions such as mpi-hpe/mpt.2.25 do not work properly for the Rome nodes.

```
module load mpi-hpe/mpt.2.30
```

Note: Although there is more than one version of MPT library installed on NAS systems, mpi-hpe/mpt.2.30 is currently the only reliable version for running on the Rome nodes. You can access this version by loading mpi-hpe/mpt, which points to the recommended version (currently mpi-hpe/mpt.2.30) and environmental settings:

```
% module load mpi-hpe/mpt
```

Running OpenMP or MPI/OpenMPI Hybrid Codes

Each Rome node has 128 physical cores. When Simultaneous Multi-Threading (SMT) is enabled in BIOS, there are 256 logical cores for use by OpenMP or MPI/OpenMPI hybrid codes.

To check whether SMT is enabled, run the following command on a Rome compute node:

```
cat /proc/cpuinfo | grep processor | wc -l
```

If it shows 256, then SMT is enabled.

We recommend using the mbind.x tool for process and thread binding, as shown in the following example with four MPI processes and 32 OpenMP threads per process:

```
mpiexec -n 4 /u/scicon/tools/bin/mbind.x -t 32 a.out
```

If you use HPE MPT, you can also use the omplace tool for pinning MPI/OpenMP hybrid code.

Running PBS Jobs on Aitken Rome Nodes

To request Aitken Rome nodes, use :model=rom_ait in your PBS script, as shown in the example below.

The devel, normal, long, debug, and low queues are available for use with Rome nodes.

Note: The Standard Billing Unit (SBU) rate for Rome processors is 4.06.

Sample PBS Script For Aitken Rome Nodes

```
#PBS -l select=2:ncpus=128:mpiprocs=128:model=rom_ait
#PBS -l walltime=8:00:00
module load mpi-hpe/mpt
module load comp-intel/2020.4.304

cd $PBS_O_WORKDIR

mpiexec -np 256 ./a.out
```

For more information, see AMD Rome Processors.

To help you prepare for running jobs on Aitken's Cascade Lake compute nodes, this short user guide includes information on the general configuration of Cascade Lake nodes, compiling your code, running PBS jobs, and checking allocation usage.

Overview of Aitken Cascade Lake Nodes

Aitken includes 1,152 Cascade Lake nodes, which are partitioned into eight physical racks. Each node contains two 20-core Xeon Gold 6248 sockets (2.5 GHz) and 192 GB of memory. The nodes are connected to the Aitken InfiniBand network (ib0 and ib1) via four-lane Enhanced Data Rate (4X EDR) devices and four-lane High Data Rate (4X HDR) switches for internode communication. The ib1 fabric is used mainly for I/O and is connected to the NAS Lustre filesystems. In addition, Aitken Cascade Lake nodes use the NAS compute resources home filesystems, front-end systems (PFEs), and PBS server (pbspl1) as Electra and the Aitken Rome nodes. You can access Aitken Cascade Lake nodes only through PBS jobs submitted from the PFEs.

Note: Aitken Milan nodes use a different set of front ends. See Preparing to Run on Aitken Milan Nodes for more information.

Compiling Your Code For Cascade Lake Nodes

The Cascade Lake processors include the Advanced Vector Extensions 512 (AVX-512). Intel AVX-512 optimizations are included in Intel compiler version 16.0 and later versions. We recommend that you test the latest Intel compiler, through the command `module load comp-intel/`, which may include better optimizations for AVX-512.

For Cascade Lake-specific optimizations, use the compiler option `-xCORE-AVX512`.

If you want a single executable that will run on any of the Aitken and Electra processor types, with suitable optimization to be determined at runtime, you can compile your application using the option:
`-O3 -axCORE-AVX512,CORE-AVX2 -xAVX`.

Note: The use of either of these options, -xCORE-AVX512 or -axCORE-AVX512, could either improve or degrade performance of your code. Be sure to check performance with and without these flags before using them for production runs.

Running PBS Jobs on Aitken Cascade Lake Nodes

To request Aitken Cascade Lake nodes, use `:model=cas_ait` in your PBS script, as shown in the example below.

Note: Because MPT 2.15 and earlier versions do not support the ConnectX-5 host channel adapters (HCAs), the environment variables MPI_IB_XRC and MPI_XPMEM_ENABLED have been disabled for jobs running on Cascade Lake. If your MPI applications perform significant MPI collective operations and rely on having these two variables enabled to get good performance, use MPT 2.17 or newer versions. You can use the NAS-recommended MPT version with the command:
`module load mpi-hpe/mpt`

Sample PBS Script For Aitken Cascade Lake Nodes

```
#PBS -l select=10:ncpus=40:mpiprocs=40:model=cas_ait
#PBS -l walltime=8:00:00
#PBS -q normal

module load mpi-hpe/mpt
module load comp-intel

cd $PBS_O_WORKDIR

mpiexec -np 400 ./a.out
```

For more information, see Cascade Lake Processors.

When you request PBS resources for your job, you must specify one or more Aitken or Electra node model (processor) types, so it is helpful to keep this information in mind:

- Usage charges on each HECC node type are based on a common Standard Billing Unit (SBU).
- The total physical memory of each node type varies from 128 GB to 512 GB.

The following table provides the SBU rate and memory for each Aitken and Electra node type:

Processor Type	SBU Rate/Node	Memory/Node
Milan	4.38 (128 cores/node)	512 GB
Rome	4.06 (128 cores/node)	512 GB
Cascade Lake	1.64 (40 cores/node)	192 GB
Skylake	1.59 (40 cores/node)	192 GB
Broadwell	1.00 (28 cores/node)	128 GB

Note: The amount of memory available to a PBS job is less than the node's total physical memory, because the system kernel can use up to 4 GB of memory in each node.

Checking Available Nodes Before Requesting Resources

If your application can run on any of the Aitken or Electra processor types, you might be able to reduce wait time by requesting a type that has more free nodes (nodes currently unoccupied by other running jobs).

Before you submit your job, you can find out the current numbers of used and free nodes for each processor type by running the node_stats.sh script, which is accessible by loading the scicon/cli_tools modulefile. For example:

```
% module load scicon/cli_tools
% node_stats.sh
```

You can also identify which processor models are in use for each job currently running on HECC systems. Run the following command, and check the Model field in the output:

```
% qstat -a -W o=+model
```

Resource Request Examples

As shown in these examples, use the model=[bro_ele,sky_ele,cas_ait,rom_ait, mil_ait] attribute to request the node model type(s) for your job.

The default operating system for all model types is TOSS (Tri-Lab Operating System Stack). For all model types, the default is aoe=toss4:aoe=toss4 to your PBS resource request.)

Example 1: Basic Resource Request

Each of the following sample command lines requests a single node model type for a 128-process job:

```
#PBS -l select=5:ncpus=28:model=bro_ele
#to run all 28 cores on each of the 5 Broadwell nodes
#(12 cores in the 5th node will go unused)

#PBS -l select=4:ncpus=40:model=sky_ele
#to run all 40 cores on each of the 4 Skylake nodes
#(32 cores in the 4th node will go unused)

#PBS -l select=4:ncpus=40:model=cas_ait
#to run all 40 cores on each of the 4 Cascade Lake nodes
#(32 cores in the 4th node will go unused)

#PBS -l select=1:ncpus=128:model=rom_ait
#to run all 128 cores on 1 Rome node

#PBS -l select=1:ncpus=128:model=mil_ait
#to run all 128 cores on 1 Milan node
```

You can specify both the queue type -q normal, debug, long, low, or devel and the processor type. For example:

```
#PBS -q normal
```

```
#PBS -l select=128:ncpus=128:model=mil_ait
```

Example 2: Requesting Different ncpus in a Multi-Node Job

For a multi-node PBS job, the number of CPUs `ncpus` used in each node can be different. This feature is useful for jobs that need more memory for some processes, but less for other processes. You can request resources in "chunks" for a job with varying `ncpus` per node.

This example shows a request of two resource chunks. In the first chunk, one CPU in one Broadwell node is requested, and in the second chunk, 28 CPUs in each of three other Broadwell nodes are requested:

```
#PBS -l select=1:ncpus=1:model=bro_ele+3:ncpus=8:model=bro_ele
```

Example 3: Requesting Multiple Node Model Types

A PBS job can run across more than one node model type if the nodes are all from Electra or all from Aitken, but not across the systems. This can be useful in two scenarios:

- When you cannot find enough free nodes within one model for your job
- When some of your processes need more memory while others need much less

You can accomplish this by specifying the resources in chunks, with one chunk requesting one node model type and another chunk requesting a different type.

This example shows a request for one Skylake node (which provides ~185 GB of memory per node) and three Broadwell nodes (which provide ~120 GB of memory per node):

```
#PBS -lplace=scatter:excl:group=model

#PBS -lselect=1:ncpus=24:mpiprocs=24:model=sky_ele+3:ncpus=28:mpiprocs=28:model=bro_ele
```

Running Jobs on Athena

To run a PBS job on the Athena compute nodes, submit the job from one of the Athena front-end systems (ATHFEs) to one of the queues managed by the PBS server pbs06a: devel, normal, long, and wide.

Note: You cannot submit jobs to run on Athena from a PFE.

Running Jobs on Aitken and Electra

To run a PBS job on the Aitken or Electra compute nodes, submit the job from one of the PFEs to one of the queues managed by the PBS server pbspl1.

Aitken Milan Nodes: You can submit batch jobs to run on Aitken's Milan nodes (mil_ait) either from a PFE or from one of the two Aitken Milan front-end systems (afe01, afe02). Interactive PBS jobs for Aitken Milan nodes must be submitted from the AFEs because currently there is no direct network connection to the PFEs.

Running Jobs on Endeavour

To run a PBS job on the shared memory systems Endeavour3/4, submit the job from a PFE to one of the queues managed by the PBS server pbspl4.

Running Jobs on Cabeus

To run a PBS job on:

- Milan A100 (mil_a100) submit the job from a Cabeus front end (cfe01 or cfe02) to one of the queues managed by the PBS server pbspl4.
- Grace Hopper nodes (gh200), submit your job from a Cabeus Grace Hopper front end (cghfe01 or cghfe02) to one of the queues served by the PBS server pbs05a.

PBS Queues

To view a complete list of all queues, including their defaults or limits on the number of cores, wall time, and priority, run theqstat command from an appropriate front-end system as follows:

```
qstat -Q
```

To view more detailed information about each queue, such as access control, defaults, limits on various resources, and status:

```
qstat -fQ queue_name
```

Brief summaries of the various queues are provided in the following sections.

Queues for Athena Users

You can run jobs on Athena using any of the following queues:

```
Queue   NCPUs/      Time/
name     max/def    max/def    pr
======= =====/=== ======/====== ===
devel    256/ 1    02:00/--     100
normal    --/ 1    08:00/--       0
long      --/ 1   120:00/--       0
wide     128/64    24:00/--      20
```

Queues for Aitken and Electra Users

You can run jobs on Aitken and Electra using any of the following queues:

```
Queue   NCPUs/      Time/
name     max/def    max/def    pr
======= =====/=== ======/====== ===
low       --/  8  04:00/ 00:30 -10
normal    --/  8  08:00/ 01:00   0
long      --/  8 120:00/ 01:00   0
debug     --/  8  02:00/ 00:30  15
devel     --/  1  02:00/    -- 101
```

The normal, long and low queues are for production work. The debug and devel queues have higher priority and are for debugging and development work.

For the debug queue, each user is allowed a maximum of 2 running jobs using a total of up to 128 nodes. For the devel queue, each user is allowed only 1 job (either queued or running).

Queues for Cabeus GPU Users

For information about queues for the Cabeus GPU nodes, see Requesting GPU Resources.

See also:

- Preparing to Run on Cabeus Grace Hopper Nodes
- Preparing to Run on Cabeus A100 GPU Nodes

Queues for Endeavour Users

Jobs submitted from PFEs to the e_normal, e_long, e_debug, and e_low queues are routed to run on the Endeavour shared-memory system, which has its own PBS server.

See Endeavour PBS Server and Queue Structure for more information.

Queue for Lou Users

The ldan queue provides access to one of the Lou data analysis nodes (LDANs) for post-processing archive data on Lou.

The ldan queue is managed by the same PBS server as all of the Electra and Aitken queues and is accessible by submitting jobs from an LFE or PFE.

Each job on the ldan queue is limited to 1 node and 72 hours. Each user can have a maximum of 2 running jobs on this queue.

See Lou Data Analysis Nodes for more information.

Queues with Restricted Access

The vlong Queue

Access to the vlong queue requires authorization from the NAS User Services manager.

A maximum of 1024 nodes and 16 days are allowed on this queue.

To avoid losing all useful work in the event of a system or application failure, ensure that your application includes checkpoint and restart capability; this is especially important when using this queue.

The wide Queue

The wide queue is a high priority queue used for jobs that meet either one of the following conditions:

- The number of requested NCPUs (number of nodes * number of cores per node) exceeds 25% of the user's mission shares
- The number of nodes requested of a processor model exceeds 25% of the total number of nodes of that model. See the Electra and Aitken configuration pages for the total number of nodes for each model.

Users do not have direct access to the wide queue. Instead, a job that meets the above criteria is moved into the wide queue by NAS staff. Only 1 job in the wide queue is allowed among Electra and Aitken. When more than one job meets the criteria, preference is given to the job that has been waiting for longer time or whose owner does not have other jobs running.

Athena: You can submit jobs to the Athena wide queue (with minimum of 64 nodes and maximum of 128 nodes for up to 24 hours) without requesting approval from NAS staff.

Various Special Queues

Special queues are higher priority queues for time critical projects to meet important deadlines. Some examples of special queues are armd_spl, astro_ops, heomd_spl, kepler, smd1, smd2, smd_ops, and so on. Request for creation and access of a special queue is carefully reviewed by the mission directorate or by the program POC who authorizes the project's SBU allocation. The request is rarely granted unless the project's principal investigator demonstrates that access to a special queue is absolutely necessary.

Advanced Reservation Queues

An advanced reservation queue allows you to reserve resources for a certain period ahead of time. One circumstance in which the NAS User Services manager may authorize the use of a reservation queue is when a user plans to run an extra-large job, such as one exceeding 15,000 cores, which would never start running without a reservation.

A reservation queue is created by NAS staff for the user and is normally named with the letter R followed by a number.

To find the status of existing PBS advanced reservations, issue the pbs_rstat command on a PFE.

Note: Queues such as alphatst, ded_time, diags, dpr, idle, resize_test, and testing are used for staff testing purposes, and are not accessible by general users.

NAS provides a special queue on Electra and Aitken, called the devel queue, that is designated for development work. Using this queue for development work provides faster turnaround than using other, non-designated queues.

Note: The devel queue should not be used for production work. Misuse of the devel queue may result in loss of access to the queue. See devel Queue Policies for more information.

Flexible Node Types

From the devel queue, you can access nodes of all Aitken and Electra processor types.

Requesting Resources

Because the devel queue does not have a default processor type, you must specify the processor type by including the :model=*xxx* attribute in your PBS resource request (where *xxx* is bro_ele, sky_ele, cas_ait or rom_ait. Note that mil_ait is not allowed to be used in devel queue.) For example, to request a Rome node:

```
#PBS -q devel
#PBS -l select=1:ncpus=128:model=rom_ait
```

Submitting Jobs

To submit your job to the devel queue, run the qsub command:

```
pfe21% qsub -q devel job_script
1234.pbspl1.nas.nasa.gov
```

To check the status of a job submitted to the devel queue, run the qstat command:

```
pfe21% qstat -u your_username devel
```

Determining Available Resources

In order to obtain the fastest possible start time, run the node_stats.sh script to view the number and types of nodes currently available in the devel queue before you submit your job. For example:

```
pfe21% node_stats.sh
```

devel Queue Policy

When using the devel queue, be aware of the following policy:

- Each user is allowed only one job on the devel queue, either running or queued. You cannot stack up a list of queued jobs or create self-submitting jobs.
- The maximum wall-clock time of a job is always 2 hours.
- A job may request no more than 512 nodes, either of a single processor/node type or a mixture of different processor/node types.
- The PBS scheduler predominately considers which users have used the most devel queue resources to decide which jobs to run next.
- devel queue usage is applied to your mission share.

The devel queue is intended for developing and testing code and cases to prepare them for production runs in other queues, such as the normal and long queues. It should not be used for performing parameter studies or for breaking a long production run into multiple consecutive runs that each fit within the 2-hour limit.

Disregarding these devel queue policies and guidelines may result in loss of access to the queue.