Introduction to the Cabeus Cluster (Overview of Grace Hopper)

April 23, 2025

NASA Advanced Supercomputing (NAS) Division

Authors:
Kyle G. Rollin, RedLine Performance Solutions
Thaddeus J. Norman, InuTeq, LLC

Cabeus Cluster, now with Grace Hopper

- Cabeus is a lunar crater in the south polar region of the moon. In Oct 2009 the NASA LCROSS (Lunar Crater Observation and Sensing Satellite) mission's rocket body struck its floor to examine the presence of water and other chemicals

  Fun videos to watch:

  https://www.youtube.com/watch?v=Wym1xL5qacw (educational with music)

  https://www.youtube.com/watch?v=3FHgrIuJUh8 (the flight+impact)

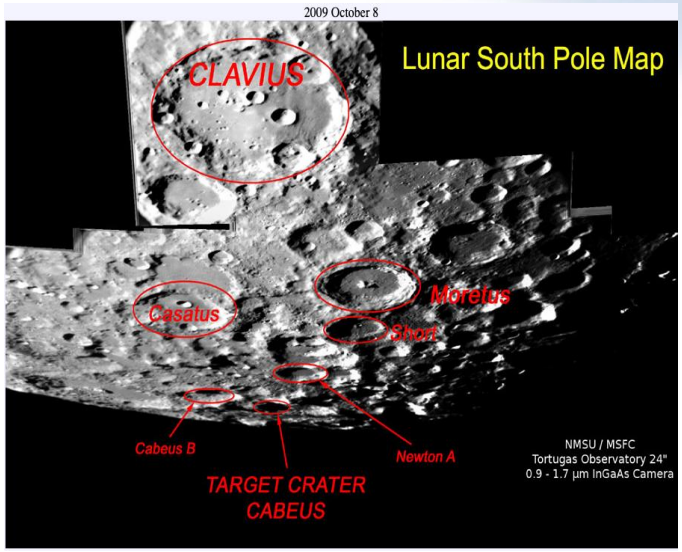

Image from https://apod.nasa.gov/apod/ap091008.html

- The Cabeus cluster, named after the crater, is a system for HPC and AI/ML applications that can benefit from the GPU technology
- Released for production work on Dec 22, 2023, Cabeus initially included 128 nodes, each node containing 1 AMD Milan CPU host + 4 NVIDIA A100 GPUs
- Older GPU nodes (sky_gpu, cas_gpu, rom_gpu) have been added to the Cabeus cluster.
- On March 3, 2025 the Grace Hopper nodes were released to the NAS userbase.

Cabeus Resources at a Glance

Systems	sky_gpu, cas_gpu, rom_gpu	mil_a100	gra_h100
Front-Ends (w internet access)	cfe01 - cfe02		cghfe01 – cghfe02
Front-End processor	Milan 7313P, 1 socket, 16 cores/socket		Grace-Hopper Superchip, 1 socket, 72 cores, 1 Hopper GPU
Compute Nodes	CPU + GPU (Details on next slide)		
Inter-Node IB Network	EDR (100 Gb/s)	HDR (200 Gb/s)	NDR (400 Gb/s)
Filesystems	$HOME, Lustre /nobackup, /nobackupnfs1, local /tmp (memory)		
PBS Server	pbspl4		pbs05a
Job Charging	applied		Charging started in Mar 2025
SBU allocation group	GPU (GPU Allocation)		

NAS GPU Compute Nodes

CPU Host + GPU Device	Grace + 1 Hopper GPU	Milan + 4 A100	Rome + 8 A100	Cascade + 4 V100	Skylake + 4 V100	Skylake + 8 V100
model type in PBS	gra_h100	mil_a100	rom_gpu	cas_gpu	sky_gpu	sky_gpu
# of nodes	350	128	2	38	17	2
hostname	cgh[1-3,6-15]n[01-26] cgh04n[01-12]	cb[01-5,8-9]n[01-12], cb[06-07]n[01-10], cb[10-12]n[01-08]	r101i5n[0-1]	r101i2n[0-17], r101i3n[0-15], r101i4n[0-3]	r101i0n[0-11,14-15], r101i1n[0-2]	r101i0n[12-13]
# of CPU socket/node	1 (Grace CPU)	1 (EPYC 7763)	2 (EPYC 7742)	2 (Platinum 8268)	2 (Gold 6154)	2 (Gold 6154)
# of CPU physical cores/node	72	64	128	48	36	36
CPU Host Memory/node	480 GB (LPDDR5X α)	512 GB (DDR4 β)	512 GB (DDR4)	384 GB (DDR4)	384 GB (DDR4)	384 GB (DDR4)
# of GPU cards/node	1 Hopper GPU	4 A100	8 A100	4 V100	4 V100	8 V100
GPU Device Memory per GPU card	96 GB (HBM3)	80 GB (HBM2e γ)	40 GB (HBM2e)	32 GB (HBM2)	32 GB (HBM2)	32 GB (HBM2)

V100: NVIDIA Volta A100: NVIDIA Ampere

α LPDDR5X: Low-Power Double Data Rate 5X
β DDR4: Double Data Rate 4
γ HBM: High Bandwidth Memory

SSH to CGHFE01 or CGHFE02

- From your local workstation (recommended approach)
  - Two-step login

    local_desktop% ssh sfe6.nas.nasa.gov (or use sfe7, sfe8)
    sfe6% ssh cghfe01 (or ssh cghfe02)

  - One-step login (need SSH Passthrough)

    https://www.nas.nasa.gov/hecc/support/kb/entry/232 ; NAS Help Desk: 1-800-331-8737

    Modify your local .ssh/config to include these two blocks:

```
Host cghfe01
    HostKeyAlias            cghfe01.nas.nasa.gov
    ProxyCommand            ssh -ax -oCompression=no sfe ssh-balance %h

Host cghfe02
    HostKeyAlias            cghfe02.nas.nasa.gov
    ProxyCommand            ssh -ax -oCompression=no sfe ssh-balance %h
```

    local_desktop% ssh cghfe01 (or ssh cghfe02)

- From a pfe or lfe

  pfe, lfe% ssh cghfe01 (or ssh cghfe02)

Note: SSH from cghfe01 and cghfe02 to pfes, lfes, pbspl1, pbspl4 is disabled

Available Software

- Grace Hopper is an ARM64 (AArch64) architecture (other NAS systems are x86_64).
- Available software that has been built/compiled with ARM64 can be seen with “module avail”:

```
---------------------------------------------------------- /nasa/arm/modulefiles/toss5 ----------------------------------------------------------

boost/1.87          hdf5/1.14.5_nv_serial  nvhpc-nompi/24.3    nvhpc/24.3            openmpi/4.1.7-cuda12.6-nv  scicon/cli_tools
cuda/12.6           hpcx/2.21              nvhpc-nompi/24.11   nvhpc/24.11           parmetis/4.0.3-ompi-32bit  scicon/mpiprof
go/1.23.3           metis/5.2.1-32bit      nvhpc-nompi/25.1    nvhpc/25.1            parmetis/4.0.3-ompi-64bit  szip/2.1.1
hdf5/1.12.2_nv_serial metis/5.2.1-64bit    nvhpc-openmpi3/24.3 openmpi/4.1.7-cuda12.6-gnu scicon/aws_cli_tools
```

- OpenMPI is the suggested MPI for Grace Hopper.
- HPC-X is an MPI package from NVIDIA that is OpenMPI + Additional optimizations, libraries, and tools.
- Currently available MPI libraries (as seen in “module avail”, above) are:
  - nvhpc/25.1 (NVIDIA HPC SDK version 25.1, HPC-X 2.21, Cuda 12.6)
  - nvhpc/24.11 (NVIDIA HPC SDK version 24.11, HPC-X 2.20, Cuda 12.6)
  - nvhpc/24.3 (NVIDIA HPC SDK version 24.3, HPC-X 2.17.1, Cuda 12.3)
  - hpcx/2.21 (HPC-X 2.21, no NVIDIA compilers, wrappers “wrap” GNU)
  - nvhpc-openmpi3/24.3 (OpenMPI version 3)
  - openmpi/4.1.7-* (Vanilla OpenMPIs built with NVIDIA or GNU compilers, respectively)

Note: NVIDIA states that 25.1 is the first NVHPC HPC SDK built for Grace Hopper, so recommend to try first.

- Compilers: nvhpc contain nvc, nvc++, nvfortran, and nvcc (C/C++ CUDA). System GNU is 11.5.0
- Other software available: Boost, Go, HDF5, METIS/ParMETIS
- MPI Profiling: scicon/mpiprof
  - https://www.nas.nasa.gov/hecc/support/kb/using-mpiprof-for-performance-analysis_525.html

PBS Queues for Grace Hopper

- The following commands show the queue definitions and status for Grace Hopper

Overall queue summary for Grace Hopper (qstat –q or –Q for more details):

```
Queue            Memory CPU Time Walltime Node   Run   Que   Lm  State
---------------- ------ -------- -------- ---- ----- ----- ----  -----
gpu_normal         --       --    24:00:00   84     4     6   --   E R
gpu_long           --       --    120:00:0    8     0     1   --   E R
gpu_devel          --       --    02:00:00    2     0     1   --   E R
gpu_wide           --       --    12:00:00   --     0     1   --   E R
```

Per-job settings for specific queues (qstat –Qf <queue name>):

gpu_normal

resources_max.nodect = 84
resources_max.walltime = 24:00:00
resources_min.mem = 256mb
resources_min.ncpus = 1
resources_default.mem = 16gb
resources_default.ncpus = 1
resources_default.ngpus = 1
resources_default.place = scatter:excl

gpu_devel

resources_max.nodect = 2
resources_max.walltime = 02:00:00
resources_min.mem = 256mb
resources_min.ncpus = 1
resources_default.mem = 32gb
resources_default.ncpus = 1
resources_default.ngpus = 1
resources_default.place = scatter:excl

gpu_long

resources_max.nodect = 8
resources_max.walltime = 120:00:00
resources_min.mem = 256mb
resources_min.ncpus = 1
resources_default.mem = 16gb
resources_default.ncpus = 1
resources_default.ngpus = 1
resources_default.place = scatter:excl

Note: gpu_wide not shown here but has no limit on number of nodes and max walltime of 12 hours.

Queue ACL Example:

acl_groups = a1438,a1441,a1448,a1452,a1457,a1486,a1495,a1502,a1503,a1512,a1521,a1532,
a1550,a1607,a1608,a1613,a1614,a1837,a1849,a1961,a2108,a2109,a2253,a2286,a2290,
a2292,a2299,a2301,c1454,css,cstaff,e0721,e0847,e0916,e1305,e2101,e2286,e2287,
e2296,e2300,e2302,g1099,g28100,hsp,n1853,n1855,n2001,n2008,n2203,s1007,s1061,
s1082,s1126,s1153,s1488,s1720,s1873,s1991,s1994,s2157,s2380,s2423,s2424,s2458,
s2557,s2703,s2739,s2748,s2764,s2805,s2810,s2829,s2853,s2857,s2896,s2908,s2916,
s2917,s2965,s2994,s3006,s3104,s3136,s3168,s3171,s5692,s7614,s7985,sciben,scicon,
tools,vistech

- Your GID should be in the ACL for gpu_normal, gpu_long, gpu_devel, and gpu_wide as long as it has GPU SBUs

Sample PBS Script and Interactive Jobs


```
#PBS –l select=8:ngpus=1:ncpus=72:mem=450GB:model=gra_h100   (this specifies 8 nodes with 1 GPU each,
                                                              72 CPU cores and 450 GB host memory)
#PBS –lplace=scatter:excl        (optional, since scatter:excl is default queue placement)

#PBS -l walltime=24:00:00
#PBS –q gpu_normal               (Available queue options are gpu_normal, gpu_devel, gpu_long)
#PBS –W group_list=a1234         (Specify the GID to use if it is not your default GID in /etc/passwd)
#PBS –j oe                       (to combine PBS output and error to PBS output file)
#PBS –koed                       (specify writing PBS output/error directly to their final destination)

qstat –f $PBS_JOBID              (specify this if you want to check how resources are allocated to this job)

cd $PBS_O_WORKDIR
module purge
module load …

mpiexec –np 576 –machinefile $PBS_NODEFILE –x LD_LIBRARY_PATH –x PATH <executable>
```

- --machinefile $PBS_NODEFILE is needed because NVIDIA provided MPI libraries are not PBS Aware
- -x LD_LIBRARY_PATH and –x PATH are sometimes needed to propagate those env variables to remote nodes.

Or to run interactively, submit a job from cghfe01 or cghfe02 using the qsub command:

```
cghfe01:~> qsub –I –q gpu_normal –lselect= 8:ngpus=1:ncpus=72:mem=450GB:model=gra_h100,walltime=24:00:00
```

SBU Charging and Accounting

- When you request a Grace Hopper node via PBS, you get charged for the whole node.
- The Grace Hopper SBU rate (per hour) is fixed at: 18.34.
- A successful job submission requires
  - An SBU allocation specifically for GPUs for the Group-ID (GID, e.g., a1234) you want to use

    New GPU projects should create a request for GPU allocations via https://request.hec.nasa.gov
  - Positive remaining value of the allocation

    pfe, cfe, pbspl1 or pbspl4% acct_ytd [a1234]

Project	Host/Group	Fiscal Year	Used	% Used	Limit	Linear Remain	YTD Usage	Project Exp Date
a1234	gpu	2024	5407.459	47.63	11352.000	5944.541	168.80	09/30/25

  - The GID is added to the Access Control List (ACL) of the PBS queue to be submitted to; done by NAS

    cghfe01 or cghfe02% qstat –fQ gpu_normal | grep acl_groups

    ```
    acl_groups = a1234, a1235, ….
    ```

- Use command acct_query to check SBUs charged to each job (-o) for using gra_h100

  pfe, cfe, pbspl1 or pbspl4% acct_query –u username –p gid –olow –c cabeus_GH –b 03/01/25

  Note: SBU accounting data usually is available within a few hours after a job is completed

  Note: the acct_ytd and acct_query commands do not work on cghfe01 or cghfe02.

HECC Data Science

- Data Science Conda Environments:
  - Miniconda for Nodes with x86_64-Based CPUs
    - TensorFlow: tf2_16
    - PyTorch: pyt2_6
  - Miniconda for Nodes with aarch64-based CPUs
    - TensorFlow: tf2_18_gh
    - PyTorch: pyt2_6_gh
- Deploying Environments on Nvidia GH200 Nodes:
  - Accessing Environments
    - module use -a /swbuild/analytix/tools/modulefiles
    - module load miniconda3/gh2
    - source activate <*environment_name*>

Common Python Packages

- Pandas – Data analytic platform
- Dask – Distributed data analytic platform
- Scikit-learn – Machine learning platform
- Matplotlib – Plotting and graphics platform
- Pillow – A fork of Python Image Library (PIL)
- Jupyter Lab – A collaborative and integrated development environment
- To request or suggest a package:
  - Contact: support@nas.nasa.gov

Creating a Custom Conda Environment

- Accessing NAS conda environments:
  - module use -a /swbuild/analytix/tools/modulefiles
  - module load miniconda3/gh2
  - source activate *environment_name*
- Adding packages for local use:
  - pip install --user *package_name*
- Creating a new conda environment:
  - Acquire a token from Data Science Team
  - yaml files: /swbuild/analytix/tools/miniconda3_ 241216 _yml/
    - Change name on first line!
  - export CONDA_PKGS_DIRS=/nobackup/$USER/.conda/pkgs
  - export CONDA_ENVS_PATH=/nobackup/$USER/.conda/envs
  - conda env create -f <filename>

Profiling Bootcamp Conda Environments

- Load Analytix modules files
  - module use -a /swbuild/analytix/tools/modulefiles
  - module load miniconda3/gh2
- HPC Bootcamp
  - source activate hpc_profiler
  - module load nvhpc/24.3
  - cp –R /nobackupp27/analytix/bootcamps/HPC_Profiler/_profiler <your directory>
  - cp –R /nobackupp27/analytix/bootcamps/nvbandwidth <your directory>
- AI Bootcamp
  - source activate ai_profiler
  - cp –R /nobackupp27/analytix/bootcamps/AI-Profiler/workspace <your directory>

Jupyter Lab & Tensorboard

- To Launch Jupyter Lab
  - On a GH200 Node:
    - jupyter lab --sock $HOME/sockets/jupyter.sock --no-browser
  - On your local system:
    - ssh -l <username> -o "StrictHostKeyChecking ask" -L 8080:<path to socket> \
      -o ProxyJump=sfe,cghfe01 <your assigned node>
  - Open Firefox and go to:
    - https://127.0.0.1:8080 or https://localhost:8080
- To Launch Tensorboard
  - In Jupyter Lab:
    - !tensorboard --logdir=../log
  - On your local system:
    - ssh -l <username> -o "StrictHostKeyChecking ask" -L 6006:localhost:6006 \
      -o ProxyJump=sfe,cghfe01 <your assigned node>
  - In Jupyter Lab:
    - click http:///localhost:6006

Bootcamp Jupyter Notebook Locations

- HPC Bootcamp
  - ~/HPC_Profiler/_profiler/jupyter_notebook
  - ~/nvbandwidth/memory_coherent/jupyter_notebook
- AI Bootcamp
  - ~/AI-Profiler/workspace/jupyter_notebook

LET'S GET STARTED!