NCHC Grace 2025 Tutorial Hands-on

Level: Beginner / Intermediate

Goals

The goal of this tutorial session is to provide an overview of NVIDIA Grace Superchip and allow attendee to experience first hand the product thanks to a system provided by the HPC Advisory Council.

For this tutorial, there are two options:
1. Bring Your Own Code (BYOC)
2. Guided examples

Here the list of examples provided in the repository:
* `00-arm-kernels`
* `00-gemm_example`
* `00-hello_world`
* `00-stream`
* `01-cp2k`
* `01-hpcg`
* `01-lulesh`

How to interact with the system

Connecting to the system

After receiving the public and private SSH keys (for example, `rdmaworkshop30`), run the following command to login to the system:

```
ssh -i '~/rdmaworkshop30' -t rdmaworkshop30@155.248.177.18  \
       ssh rdmaworkshop30@gw.hpcadvisorycouncil.com
```

For Windows, do place bith prvate and publich SSH keys in the same folder, open the Windows terninal and type:

```
ssh -i rdmaworkshop30 -t rdmaworkshop30@155.248.177.18  \
       ssh rdmaworkshop30@gw.hpcadvisorycouncil.com
```

Transfer data in/out to/from the system

Below an example how to transfer local `DATA` folder in the `$HOME` directory of the remote system using `rsync` (example using `rdmaworkshop30` user):

```
rsync -av -e \
  "ssh -i '~/rdmaworkshop30' -t rdmaworkshop30@155.248.177.18 ssh -o StrictHos
  ./DATA rdmaworkshop30@gw.hpcadvisorycouncil.com:/global/home/users/rdmaworks
```

Then, another an example how to transfer `DATA` from the `$HOME` directory of the remote system into the current directory on your local machine using `rsync` (example using user `rdmaworkshop30`):

```
rsync -av -e \
   "ssh -i '~/rdmaworkshop30' -t rdmaworkshop30@155.248.177.18 ssh -o StrictHo
   rdmaworkshop30@gw.hpcadvisorycouncil.com:/global/home/users/rdmaworkshop30/
```

Node allocation and job submission

Once on the system, you will be on a x86 node. Thea is composed by 3 different partitions: `gg` (Grace CPU Superchip nodes), `gh` (GH200 nodes) and `ggcompile` . Compilation and execution need to happen in one of these 3 partitions.

We provide few commands to simplify interaction with the SLURM resource manager:
* `interactive` : start an interactive session on the dedicated Grace CPU Superchip used for compilation purposes.
* `submit-gg` : submit a job to a single Grace CPU Superchip node, max 15 minutes walltime.
* `submit-gh` : submit a job to a single GH200 node, max 15 minutes walltime.

NOTE - When submitting any job via `srun` or `sbatch` always specify the reservation for this tutorial: `--reservation=gg-nchc` for Grace-Grace nodes and `--reservation=gh-nchc` for Grace-Hopper nodes.

NOTE - We kindly ask to avoid block nodes for long period of time so everybody has a chance to submit and work on the examples.

SW environment provided

Appropriate software environment is loaded by default. Below a table summarizing modulefile names (some self-explanatory) and what they provide

Module	Description	Install Location
`gcc/13.3.0-gcc-11.4.1`	GNU 13.3.0 (suggested default)	-
`acfl/24.10.1-gcc-13.3.0`	Arm HPC Compiler 24.10 (and associated Arm Performance Library	`${ACLF_HOME}`
`nvhpc/25.1-gcc-13.3.0`	NVIDIA HPC SDK 25.1 with CUDA-aware MPI (provided by HPCX)	`${NVHPC_HOME}`
`nvhpc/24.11-gcc-13.3.0`	NVIDIA HPC SDK 24.11 with CUDA-aware MPI (provided by HPCX)	`${NVHPC_HOME}`
`ucx/1.18.0-gcc-13.3.0`	UCX 1.18.0 built for GNU (cma and xpmem shm transport + IB)	`${UCX_HOME}`
`openmpi/5.0.6-gcc-13.3.0`	OpenMPI 5.0.6 built for GNU (uses UCX)	`${OMPI_HOME}`
	Arm	

`armpl-gcc/24.10-gcc-13.3.0`	Performance Library 24.10 for GNIU	`${ARMPL_HOME}`
`openblas/0.3.29-gcc-13.3.0`	OpenBLAS 0.3.29 built for GNU 13.3.0 (multi-thtead support)	`${OPENBLAS_HOME}`
`openblas/0.3.29-nvhpc-25.1`	OpenBLAS 0.3.29 built for NVHPC 25.1 (multi-thtead support)	`${OPENBLAS_HOME}`
`openblas/0.3.29-nvhpc-24.11`	OpenBLAS 0.3.29 built for NVHPC 21.11 (multi-thtead support)	`${OPENBLAS_HOME}`
`fftw/3.3.10-gcc-13.3.0`	FFTW 3.3.10 built for GNU 13.3.0 (no MPI)	`${FFTW_HOME}`
`fftw/3.3.10-nvhpc-25.1`	FFTW 3.3.10 built for NVHPC 24.11 (no MPI)	`${FFTW_HOME}`
`fftw/3.3.10-nvhpc-24.11`	FFTW 3.3.10 built for NVHPC 24.11 (no MPI)	`${FFTW_HOME}`
`nvpl/24.7-gcc-13.3.0`	NVIDIA Performance Library 24.7 (provides	`${NVPL_HOME}`

	BLAS, LAPACK, FFTW and ScaLAPACK)	
`cuda/12.5.1-gcc-13.3.0`	CUDA 12.5.1 (GNU 13.3.0 as host compiler)	`${CUDA_HOME}`
`ucx/1.18.0-gcc-13.3.0-cuda-12.5.1`	UCX 1.18.0 built for GNU (cma and xpmem shm transport + IB) + GPU DIRECT	`${UCX_HOME}`
`openmpi/5.0.6-gcc-13.3.0-cuda-12.5.1`	CUDA-aware OpenMPI 5.0.6 built for GNU (uses UCX for GPU)	`${OMPI_HOME}`
`papi/7.1.0-gcc-13.3.0`	PAPI 7.1.0	`${PAPI_HOME}`
`python/3.13.1-gcc-13.3.0`	Python 3.13.1	`${PYTHON_HOME}`

NOTE - `modulefiles` tool is configured to auto-load dependencies. If some key software is missing please ask, we may be able to build it *on-the-fly* using spack.

NOTE - Please always use `gcc/13.3.0-gcc-11.4.1` instead of default GCC provided by the OS.

NOTE - Please use `${SCRATCH_FAST}` for I/O purposes.

System walkthrough (live demo)

Live demonstration of how Grace CPU Superchip and GH200 nodes are presented to the user

Useful commands to explore the system:

`ml load papi`

`lscpu`

`numactl`

`papi_avail`

`perf list`

On the GH200 nodes is also possible to look at the GPU:

`ml load cuda`

`nvidia-smi -q -d TEMPERATURE`

`nvidia-smi -q -d POWER`

Compile and run SVE FMLA mini-benchmark

REFERENCE: https://nvidia.github.io/grace-cpu-benchmarking-guide/foundations/FMA/index.html

Load the appropriate environment (only once) and start an interactive session on the compilation node:

```
interactive
```

Loading appropriate modules:

```
module load gcc/13.3.0-gcc-11.4.1
```

Compile:

```
cd ~/nchc25-tutorial/00-arm-kernels
make -j
```

Run SVE example:

```
perf stat -e ase_spec,sve_inst_spec ./arithmetic/fp64_sve_pred_fmla.x
```

Run NEON example:

`perf stat -e ase_spec,sve_inst_spec ./arithmetic/fp64_neon_fmla.x`

Compile and run simple DGEMM Fortran code calling

Load the appropriate environment (only once) and start an interactive sesison on the compilation node:

```
interactive
```

Loading appropriate modules:

```
module load gcc/13.3.0-gcc-11.4.1
module load nvpl/24.7-gcc-13.3.0
module load openblas/0.3.29-gcc-13.3.0
```

Compile:

```
cd ~/nchc25-tutorial/00-gemm_example
make
```

Run:

```
time env OMP_NUM_THREADS=8 ./dgemm-NVPL.x
```

Run NVPL variant with binding and perf:

```
export OMP_NUM_THREADS=8
numactl --physcpubind=1 --membind=1 \
    perf stat -e ase_spec,sve_inst_spec env ./dgemm-NVPL.x
```

Run OpenBLAS variant with binding and perf:

```
export OMP_NUM_THREADS=8
numactl --physcpubind=1 --membind=1 \
perf stat -e ase_spec,sve_inst_spec env ./dgemm-.OPENBLASx
```

Compile and run simple "Hello World" MPI example

Load the appropriate environment (only once) and start an interactive sesison on the compilation node:

```
interactive
```

Loading appropriate modules:

```
module load gcc/13.3.0-gcc-11.4.1
module load openmpi/5.0.6-gcc-13.3.0
```

Compile with OpenMPI:

```
cd ~/nchc25-tutorial/00-hello_world
mpicc -mcpu=native hello_world.c -o mpi_hello_world.x
```

Run locally on the node:

```
mpirun -np 16 --map-by ppr:16:node:PE=4 --report-bindings ./mpi_hello_world.x
```

Prepare a submission script for Grace CPU Superchip (called `submit-cpu.slurm`:

```
#!/bin/bash
module purge
module load gcc/13.3.0-gcc-11.4.1
module load openmpi/5.0.6-gcc-13.3.0
cd $SLURM_SUBMIT_DIR

mpirun -np 144 \
    --map-by ppr:144:node:PE=1 \
    --report-bindings \
    ~/nchc25-tutorial/00-hello_world/mpi_hello_world.x
```

Submit and look for a file named `slurm-XXXXXX.out` (`XXXXXX` is thre Slurm JOBID):

```
submit-gg submit-cpu.slurm
```