AMD GPU servers

Overview

Austin's own Advanced Micro Devices (AMD) has most generously donated a number of GPU-enabled servers to UT.

While it is still true that AMD GPUs do not support as many 3rd party applications as NVIDIA, they do support many popular Machine Learning (ML) applications such as TensorFlow, PyTorch, and AlphaFold, and Molecular Dynamics (MD) applications such as GROMACS, all of which are installed and ready for use.

Our recently announced ***AMD GPU pod is available for both research and instructional use*, for any UT-Austin affiliated PIs**. To request an allocation, ask your PI to contact us at rctf-support@utexas.edu, and provide the UT EIDs of those who should be granted access.

Two BRCF research pods also have AMD GPU servers available: the Hopefog and Livestong PODs. Their use is restricted to the groups who own those pods. See Livestrong and Hopefog pod AMD servers for specific information.

GPU-enabled software

AlphaFold

The AlphaFold protein structure solving software is available on all AMD GPU servers. The **/stor/scratch/AlphaFold** directory has the large required database, under the **data.3** sub-directory. There is also an AMD example script **/stor/scratch/AlphaFold/alphafold_example_amd.sh** and an **alphafold_example_nvidia.sh** script if the POD also has NVIDIA GPUs, (e.g. the Hopefog pod). Interestingly, our timing tests indicate that AlphaFold performance is quite similar on all the AMD and NVIDIA GPU servers.

Pytorch and TensorFlow examples

Two Python scripts are located in **/stor/scratch/GPU_info** that can be used to ensure you have access to the server's GPUs from **TensorFlow** or **PyTorch** . Run them from the command line using time to compare the run times.

- **Tensor Flow**
  - **time (python3 /stor/scratch/GPU_info/tensorflow_example.py)**
    - should take ~30s or less with GPU, > 1 minute with CPUs only
    - this is a simple test, and on CPU-only servers multiple cores are used but only 1 GPU, one reason why the times are not more different
- **PyTorch**
  - **time (python3 /stor/scratch/GPU_info/pytorch_example.py)**
    - ***Note: this test script is not yet working on AMD servers***

If GPUs are available and accessible, the output generated will indicate they are being used.

TensorFlow

The AMD-GPU-specific version of TensorFlow, **Tensorflow-rocm** 2.9.1 is installed on all AMD GPU servers. This version works with ROCm 5.1.3+. If you need to install your own version with **pip**, specify this version:

```
pip install tensorflow-rocm==2.9.1
```

You may also need to adjust your **LD_LIBRARY_PATH** as follows:

```
export LD_LIBRARY_PATH="/opt/rocm-5.1.3/hip/lib:$LD_LIBRARY_PATH"
```

Resources

ROCm environment

ROCm is AMD's equivalent to the CUDA framework. ROCm is open source, while CUDA is proprietary.

We have multiple versions of the ROCm framework installed in the **/opt** directory, designated by a version number extension (e.g. **/opt/rocm-5.1.3**, **/opt/rocm-5.2.3**). The default version is the one pointed to by the **/opt/rocm** symbolic link, which is generally the latest version.

To specify a specific ROCm version, set the **ROCM_HOME** environment variable; for example:

```
export ROCM_HOME=/opt/rocm-5.1.3
```

You may also need to adjust your **LD_LIBRARY_PATH** as follows:

```
export LD_LIBRARY_PATH="/opt/rocm-5.1.3/hip/lib:$LD_LIBRARY_PATH"
```

Command-line diagnostics

- GPU usage: **rocm-smi**
- CPU and GPU details: **rocminfo**
- What ROCm modules are installed: **dpkg -l | grep rocm**
- GPU GPU/CPU communication bandwidth test
    - between GPU2 and CPU: **rocm-bandwidth-test -b2,0**
    - between GPU3 and GPU4: **rocm-bandwidth-test -b3,4**

Sharing resources

Since there's no batch system on BRCF POD compute servers, it is important for users to monitor their resource usage and that of other users in order to share resources appropriately.

- Use **top** to monitor running tasks (or **top -i** to exclude idle processes)
    - commands while top is running include:
    - **M** - sort task list by memory usage
    - **P** - sort task list by processor usage
    - **N** - sort task list by process ID (PID)
    - **T** - sort task list by run time
    - **1** - show usage of each individual hyperthread
        - they're called "CPUs" but are really hyperthreads
        - this list can be long; non-interactive **mpstat** may be preferred
- Use **mpstat** to monitor overall CPU usage
    - **mpstat -P ALL** to see usage for all hyperthreads
    - **mpstat -P 0** to see specific hyperthread usage
- Use **free -g** to monitor overall RAM memory and swap space usage (in GB)
- Use **rocm-smi** to see GPU usage

AMD GPU and ROCm resources

ROCm GPU-enabling framework

Best starting places:

- **ROCm Video series**
    - https://community.amd.com/t5/instinct-accelerators-blog/rocm-open-software-ecosystem-for-accelerated-compute/ba-p/418720
    - Especially the **Introduction to AMD GPU Hardware**: Link
        - Provides hardware background and terminology used throughout other guides
    - Also
        - **GPU Programming Concepts**
            - Part 1 - HIP framework (like NVIDIA CUDA): Link
            - Part 2 - Device management, synchronization, MPI programming: Link
            - Part 3 - Device code, shared memory & thread synchronization: Link
        - **GPU Programming Software** (compilers, libraries & tools): Link
- **AMD ROCm resources Learning Center**: https://developer.amd.com/resources/rocm-resources/rocm-learning-center/
    - Especially:
        - Introduction to ROCm (Video, PDF)
        - Introduction to HIP (Video, PDF)
        - Introduction to Deep Learning on ROCm (Video, PDF)

Training Guides

1. Introduction_to_AMD_7002_processor.pdf
2. Radeon_Instinct_HPC_Training_2020.pdf

3. Radeon_Instinct_ML_Training_2020.pdf