HIP Lecture Series

Introduction to HIP Exercises

For the HIP Lecture Series, the examples can be retrieved from this repository.

```
git clone https://github.com/olcf/hip-training-series
```

Below are instructions for doing the exercises on OLCF Frontier. For NERSC Perlmutter, there are two different approaches we'll explore for the hands-on exercises. The first is the conventional, simpler approach of separate repositories and build systems for each platform. For this approach, please see Lecture 1 Exercises for Perlmutter.

For the single repository, portable build system approach, see the instructions for Nvidia below. NERSC has installed the AMD ROCm software stack to enable developers on Perlmutter to begin developing more portable applications with a single repository. This can greatly reduce code porting and maintenance efforts.

This markdown document is located at `'Lecture1/01 Exercises for HIP Introduction.md'` contains the instructions to run the examples. You can view it in github for better readability or download the pdf file `'Lecture1/01 Exercises for HIP Introduction.pdf'` which has been generated from the markdown document.

For the first interactive example, get an slurm interactive session

```
salloc -N 1 -p batch --gpus=1 -t 10:00 -A <project>
```

Use your project id in the project field. If you do not remember it, run the command without the -A option and it should report your valid projects.

```
module load PrgEnv-amd
module load amd
module load cmake
export CXX=${ROCM_PATH}/llvm/bin/clang++
```

Basic examples

```
cd hip-training-series/Lecture1/HIP/vectorAdd
```

Examine files here – README, Makefile, CMakeLists.txt and vectoradd.hip. Notice that the Makefile requires ROCM_PATH to be set. Check with module show rocm or echo $ROCM_PATH. Also, the Makefile builds and runs the code. We'll do the steps separately. Check also the HIPFLAGS in the Makefile. There is also a CMakeLists.txt file to use for a cmake build.

For the portable Makefile system

```
make vectoradd
srun ./vectoradd
```

This example also runs with the cmake system

```
mkdir build && cd build
cmake ..
make
srun ./vectoradd
```

Now clean up from these exercises before the next part.

```
cd ..
make clean
rm -rf build
module unload PrgEnv-amd
module unload amd
```

```
module unload cmake
```

We can use a SLURM submission script, let's call it `hip_batch.sh`. There is a sample script for some systems in the example directory.

```
#!/bin/bash
#SBATCH -p batch
#SBATCH -N 1
#SBATCH --gpus=1
#SBATCH -t 10:00
#SBATCH -A <your project id>

module load PrgEnv-amd
module load amd
module load cmake
cd $HOME/hip-training-series/Lecture1/HIP/vectorAdd

make vectoradd
srun ./vectoradd
```

Submit the script `sbatch hip_batch.sh`

Check for output in `slurm-<job-id>.out` or error in `slurm-<job-id>.err`

To use the cmake option in the batch file, change the build commands in the batch file to

```
mkdir build && cd build
cmake ..
make
srun ./vectoradd
```

Compile and run with Cray compiler

```
module load PrgEnv-cray
module load amd-mixed
module load cmake
CXX=CC CRAY_CPU_TARGET=x86-64 make vectoradd
srun ./vectoradd
```

And with the cmake build system.

```
module load PrgEnv-cray
module load amd-mixed
module load cmake
mkdir build && cd build
CXX=CC CRAY_CPU_TARGET=x86-64 cmake ..
make
srun ./vectoradd
```

Before moving onto another example, first clean up from the previous work.

```
cd ..
make clean
rm -rf build
module unload PrgEnv-cray
module unload amd-mixed
module unload cmake
```

Now let's try the hip-stream example. This example is from the original McCalpin code as ported to CUDA by Nvidia. This version has been ported to use HIP.

```
module load PrgEnv-amd
module load amd
module load cmake
cd $HOME/HPCTrainingExamples/HIP/hip-stream
make
srun ./stream
```

Note that it builds with the hipcc compiler. You should get a report of the Copy, Scale, Add, and Triad cases.

Test the code on an Nvidia system – Add `HIPCC=nvcc` before the make command or `-DCMAKE_GPU_RUNTIME=CUDA` to the cmake command. (See README file)

For the hands-on exercise on the NERSC Perlmutter system, there is a reservation under the account ntrain8. Get an allocation with

```
salloc -N 1 -C gpu -A ntrain8 --reservation=hip_aug14 -q shared -c 32 -G 1 -t 1:00:00
```

Outside of reservation window, you can do:

```
salloc -N 1 -C gpu -A <your_project> -q shared -c 32 -G 1 -t 1:00:00
```

Load the environment for Nvidia. Note that there is an order required. To load the HIP module, the GNU environment must already be loaded. Then load the HIP environment. Once the HIP module is loaded, you can load the Nvidia programming environment.

```
module load PrgEnv-gnu/8.3.3
module load hip/5.4.3
module load PrgEnv-nvidia/8.3.3
module load cmake
```

Build the example

```
cd ~/hip-training-series/Lecture1/HIP/vectorAdd
HIPCC=nvcc make vectoradd
srun ./vectoradd
```

Cleanup

```
make clean
```

For the cmake build

```
mkdir build && cd build
cmake -DCMAKE_GPU_RUNTIME=CUDA ..
make
srun ./vectoradd
```

Cleanup

```
cd ..
rm -rf build
```

On your own:

1. Check out the saxpy example in `hip-training-series/Lecture1/HIP`
2. Write your own kernel and run it

More advanced HIP makefile

The jacobi example has a more complex build that incorporates MPI. The original Makefile has not been modified, but a CMakeLists.txt has been added to demonstrate a portable cmake build. From an interactive session, try the following steps

```
cd $HOME/hip-training-series/Lecture1/HIP/jacobi

module load PrgEnv-amd
module load amd
module load cray-mpich
module load cmake

mkdir build && cd build
cmake ..
make
srun -n 1 ./Jacobi_hip
```