CUDA: How-To

CS 3220 / CS 5220 Lecture 4-F

Jason Hibbeler
University of Vermont

Spring 2024

Topics

- Vermont Advanced Computing Core (VACC)
- Compiling CUDA programs
- Running CUDA programs

Vermont Advanced Computing Core

VACC: a research facility offering computing services to UVM faculty, staff and students

- website

For CS 3220 / CS 5220, you may use the VACC to run CUDA programs on an Nvidia GPU

You can also use your own system, if you download and install the CUDA environment

Obtaining Access

You all now have access to the systems in the VACC

Logging In

First ssh to vacc-user1.uvm.edu

- if you're coming from your laptop, and your username on your laptop is different than your UVM netid, then do this:

```
$ ssh your_netid@vacc-user1.uvm.edu
```

Make a directory `$HOME/CUDA` to hold your CUDA programs

Setting Your Environment

My VACC environment was not set to have the Nvidia compiler available by default

- add the following lines to your `$HOME/.bashrc`

```
PATH="$PATH:/usr/local/cuda-11/bin"
export PATH
```

This will let you use the command `nvcc` to compile GPU programs

Creating a Program

The convention for CUDA programs is to give the files a `.cu` extension

You'll have to use a text editor such as vim or nano

- it's just bare Linux—there's no IDE here

Compiling

The command to compile a CUDA program is nvcc

```
$ nvcc myprogram.cu
```

Running a Program

This is a little more complex

- the systems in the VACC are used by many people
- it's not practical to give a user an interactive session
- instead, you will submit jobs to the job scheduler (called slurm) that manages run requests

Basic flow:

1. create your program
2. submit it as a batch job
3. look at the results

Batch Submission

Get the script `run-batch-job.slurm` from gitlab

Modify this line:

```
#SBATCH --mail-user=jhibbele@uvm.edu
```

Change it so that it has your UVM email address

Batch Submission

Also, you can modify this line:

```
#SBATCH --job-name=my-program
```

This determines the name of the output status file that is created when the job is launched

Batch Submission

And you can optionally change the line that actually launches your job:

```
./CUDA/a.out
```

It should match the name and location of your compiled CUDA program (which will by default be `a.out`)

Batch Submission

You can also tell the job scheduler to send you an email message when various events occur:

```
# Request email to be sent at begin and end, and if fails;
# options are NONE, BEGIN, END, FAIL, ALL
#SBATCH --mail-type=FAIL
```

This will send you an email message if the job fails

Batch Submission

After you get your program running correctly, submit it one time with this option

```
# Request email to be sent at begin and end, and if fails;
# options are NONE, BEGIN, END, FAIL, ALL
#SBATCH --mail-type=BEGIN
#SBATCH --mail-type=END
```

This will send you an email message when the jobs starts and when the job ends

And I will ask you to forward to me these emails when you submit your CUDA assignment

Batch Submission

Then run your job this way:

```
$ sbatch run-batch-job.slurm
```

Viewing Output

The output from your program will go into the file specified in your script

```
#SBATCH --output=%x_%j.out
```

The `%x` will be the job name, and the `%j` will be the job ID assigned by the scheduler

- for example, `my-program_161731.out`

Checking for Errors

CUDA kernels run asynchronously

- there's no return code from the call

Instead, check for an error this way:

```
addKernel<<<blocks, threadsPerBlock>>>(d_x, d_y, d_z, pitch, N);
cudaDeviceSynchronize();

cudaError_t err = cudaGetLastError();
const char *msg = cudaGetErrorName(err);
printf("error = |%s|\n", msg);
```

If there's no error, you'll get `cudaSuccess`

Computing CPU Elapsed Time

Here's how to compute elapsed time on the CPU:

```
#include <sys/time.h>

struct timeval t1, t2;
float elapsedTime;

gettimeofday(&t1, NULL);
// do some work
gettimeofday(&t2, NULL);

elapsedTime = (t2.tv_sec - t1.tv_sec) * 1000.0;      // sec to ms
elapsedTime += (t2.tv_usec - t1.tv_usec) / 1000.0;   // us to ms
printf("%f ms\n", elapsedTime); // elapsed time in milliseconds
```

Example Program

Try compiling and running this program:

`vecaddi-gridstride.cu`

from the `Examples/CUDA` directory in gitlab

Copying a File from VACC to Laptop

Suppose you have a file `myprogram.cu` in your CUDA directory, and you want to copy it to your laptop

On your laptop, do this:

```
$ scp netid@vacc-user1.uvm.edu:CUDA/myprogram.cu .
```

Note the dot at the end.

This will put the file in your current directory on your laptop (after it prompts you for your UVM netid password)

Copying a File from Laptop to VACC

Suppose you have a file `otherprogram.cu` on your laptop, and you want to copy it to your directory in the VACC

On your laptop, cd to the directory containing the file, and do this:

```
$ scp otherprogram.cu netid@vacc-user1.uvm.edu:.
```

Again, note the dot at the end.

This will put the file in your home directory on vacc-user1 (after it prompts you for your UVM netid password)