ECE 759
High Performance Computing for Engineering Applications
Assignment 1
Due Monday 2/3/2024 at 23:59 PM

Submit responses to tasks which don't specify a file name to Canvas in a file called `assignment1.pdf` (choose one of the formats). Submit all plots (if any) on Canvas. Do not zip your Canvas submission.

All *source code files* should be submitted in the `HW01` subdirectory on the `main` branch of your GitHub repo (which should be named `repo759`, as instructed in class). Please use the name `HW01` exactly as shown here (both in terms of capitalization & name). Other names like `hw1`, `hw01`, `HW1` will not be recognized by the repo scripts. The `HW01` subdirectory should have no subdirectories. For this assignment, your `HW01` folder should contain `task4.sh` and `task6.cpp`.

All submissions will be graded on *Euler*. Sometimes, code behaves differently on your computer and on Euler, so it is recommended that you test on *Euler* before you submit. Per CAE policies: do not run computational jobs directly on the head node; and do not launch interactive jobs on *Euler*. Use `sbatch` to submit your jobs and use the resources responsibly.

Use Piazza for resolving your questions and leaving comments on assignments. Read this document to help you use Piazza correctly, efficiently and wisely.

Finally, remember to invite the TAs and the instructor as collaborators to your GitHub repo. That will allow us to clone your repo and grade your work. You can find their GitHub account on the course Canvas page.

Please submit clean code. If you have a bit of time, learn how to use a formatter like clang-format.

Specify your GitHub link here: https://github.com/kaganaval6/repo759/tree/main/HW01

Note that your link should be of this format: `https://github.com/YourGitHubName/repo759/HW01`

---

1. (a) Read the files `timing.md` and `slurm_usage.md` from the `Assignments/general` directory of the ECE 759 Resource Repo. These documents will come into play in almost all your assignments throughout the semester.
   (b) Read the `hw_repos.md` file in the same directory. This is very important and must be done in order for you to turn in all the assignments for ECE 759.
   (c) At least skim `workflow.md`. This contains a quick guide for effectively working between your local computer and *Euler*. This was also discussed in class.

   NOTE: For this problem, when you are ready to state this in good faith, just say "I went through a) through c) and understand how to time code, how to submit my assignments with `git`, and what the recommended workflow is when it comes to working on my assignment".

   I went through a) through c) and understand how to write code, how to submit my assignments with git, and what the recommended workflow is when it comes to working on my assignment.

2. Write one line of `bash` code for each of the following sub-tasks (assume that all the files and directories mentioned exist). The purpose of this task is to get you familiar a bit with the Linux command line.
   a) Change the current directory to a subdirectory called `somedir` cd somedir
   b) Print out to the terminal the contents of a file called `sometext.txt`. The file exists in the current directory. cat sometext.txt

c) Print out to the terminal the last 5 lines of a plain text file called `sometext.txt`. The file exists in the current directory. tail -n 5 sometext.txt

d) Print out to the terminal the last 5 lines of *each* file that ends in the extension `.txt` and lives in the current directory tail -n 5 *.txt

e) Write a `for` loop which prints each integer from 0 to 6 (including 0 and 6).

for i in {0..6}; do echo $i; done

3. The purpose of this task is to get you familiar using *Euler*. On *Euler*, using the `module` command, answer the following questions.

a) Are there any modules loaded (`module list`) when you log in on *Euler*? No

b) What version (version number) of `gcc` is available to you without loading any modules? gcc (GCC) 14.1.1 20240522 (Red Hat 14.1.1-4)

c) List all `cuda` modules available on *Euler*.

d) List one other piece of software that has a module on *Euler* and write one sentence about what it does. (If you aren't familiar with any of the other software, google one up and write a sentence about it.)

c) nvidia/cuda/10.2.2, nvidia/cuda/11.0.3, nvidia/cuda/11.3.1, nvidia/cuda/11.6.0, nvidia/cuda/11.8.0, nvidia/cuda/12.0.0, nvidia/cuda/12.1.0, nvidia/cuda/12.2.0, nvidia/cuda/12.5.0 } Purely CUDA modules in Euler

Other modules like gromacs and nvidia/nvhpc-byo-cudall use cuda support or depends on cuda and aren't pure CUDA modules.

matlab/r2021a, matlab/r2021b, matlab/r2023a }

d) Matlab is a piece of software that has 3 modules on Euler. These are different versions of matlab which is a high level programming language and numerical computing environment used for math modeling, computation, data analysis and visualization, engineering simulations etc.

4. Write a bash script called `task4.sh` with a Slurm header which asks for

- 2 CPU cores
- A job name of `FirstSlurm`
- An output file called `FirstSlurm.out`
- An error file called `FirstSlurm.err`

and runs a single command to print the hostname of the machine (compute node) running the job. This job must be submitted by running `sbatch task4.sh` on *Euler*'s head node.

5. Research some useful Slurm tools (one sentence responses): answer on page 3 and page 4

a) In what directory does a Slurm job on *Euler* begin execution? You may run some jobs in different directories to check this.

b) Explain what `SLURM_JOB_ID` is in the environment of a running Slurm job.

c) How would you track the status of job(s) run by yourself? Assume that the job(s) have not been completed yet.

d) How would you cancel a job submitted by yourself? Assume that the job is still in the queue.

e) Explain what the following script header line specifies: `#SBATCH --gres=gpu:1`

f) (Optional) Explain what the following script header line specifies: `#SBATCH --array=0-9`

6. Write a C++ program called `task6.cpp` that:

   a) Takes a command line argument `N`. (If you are confused about command line arguments, it may be helpful for you to read this)

   b) Prints out each integer from `0` to `N` (including `0` and `N`) in ascending order with the `printf` function.

   c) Prints out each integer from `N` to `0` (including `N` and `0`) in descending order with `std::cout`.

   For each printing process, the integers should be separated by spaces on a single line ending in a newline.

   - Compile command: `g++ task6.cpp -Wall -O3 -std=c++17 -o task6`
   - Run command (**do not** run this on Euler's head node; do it through Slurm): `./task6 N`
   - Expected output (followed by a newline):
     `0 1 2 3 ··· N`
     `N ··· 3 2 1 0`
   - Example expected output for `N` = 6 (followed by a newline):
     `0 1 2 3 4 5 6`
     `6 5 4 3 2 1 0`

Q.5 (answers)

a) Jobs begin execution in the same directory from which sbatch was executed. SLURM_SUBMIT_DIR is an environment variable automatically set by slurm when you submit a job. It stores the directory from which job was submitted and you can compare directories using pwd and pwd.

b) SLURM_JOB_ID is an environment var. automatically set by Slurm when a job is submitted. It stores the unique job ID assigned to the running job.

c) To track status of jobs that haven't been completed yet, we use squeue command. By running basic command squeue -u $USER I can see a table with Job ID, state, time elapsed and assigned nodes.

(filter the list to only jobs run by me)

d) we can use scancel, to cancel a job that is still in queue or running.

To cancel a specific job we use command

scancel <job_#>

To cancel all my jobs, I can use command

scancel -u $USER

e) The script header line #SBATCH --gres=gpu:1 allocates a single GPU core.

f) #SBATCH --array=0-9 directive in a slurm script specifies a job array the runs 10 jobs with task IDs ranging from 0 to 9