Project 3 - Matrix Transpose

- This is due **24:00, Friday, December 23.** Three teams will do their presentations at class on Monday, December 26 (*with bonus*).
- Send an email to the TA (lc3018121@sjtu.edu.cn) with:
  - **Subject**: in the form of `[CS433] Project 3: Team 1`
  - Your **source code** (*see instructions - Part 2* & 3) & **documentation** (*see Instructions - Part 4*).
  - If you are willing to do a presentation for us, please tell me.
  - Feedback on the project itself, if any.
- Please feel free to ask questions via email or QQ.

Instructions

Summary: In this project, you will:

1. Set up your CUDA development tools or just learn to run CUDA programs on Pi cluster.
2. Finish several versions of matrix transpose (CPU, naive GPU, shared memory, no bank conflict, loop unrolling).
3. Analyze performance of them.

Part 1: Setting up your development environment.

1. If your laptop is equipped with an NVIDIA graphics card with CUDA capability (Check your GPU on this compatibility table.), you can install CUDA following the instructions here.
2. You can also run CUDA programs directly on Pi cluster. See Using cuda on Pi.
   - You can find your username and password in the QQ group file (*"PI集群账号.pdf"*).
   - SSH login node: 202.120.58.229 or 202.120.58.230 or 202.120.58.231
   - SSH Port: 22
   - Send the IP address of your dormitory to the TA if you have not yet.
3. You can get the GPU information by running the `deviceQuery` sample attached on the course website.
   - There are several different types of GPUs on Pi, such as k20, k40, k80. You can select where to run your program by changing the `<node_type>` in `deviceQuery.slurm` . The types of node are gpu(k20), k40 or k80.
   - Use instruction below to compile it:

     ```
     $ nvcc -I./inc/ deviceQuery.cpp -o deviceQuery
     ```

Part2: Finish different versions of matrix transpose.

See Matrix transpose performance for details.

You should write a `README.md` with:

1. How to run your program.

Part3: Evaluate performance

You should figure out by your self how to timing the kernel execution of CUDA.

- You can repeat the kernel several times if the execution time is too short to test.
- The relevant performance metric is the **effective bandwidth**, calculated in **GB/s** as twice the size of the matrix – once for reading the matrix and once for writing – divided by the time of execution.

Part4: Write-up

Finish your documentation with:

1. How you implement the different versions. Any problems you met?
2. How you time the kernel execution of CUDA.
3. What the hardware information of your GPU device is.
4. What the results are and what's your opinion on them.

And you're done!