EE 451: Parallel and Distributed Computation

PA6b — Spring 2021

Due date: Monday 12th April 2021 11:59 PM

1. **Examples**

   Copy example files to your home directory.

   1. Login to HPC
   2. Copy

      ```
      cp -r /project/xuehaiqi_652/6a .
      ```

   The `hello.cu` contains the CUDA implementation of HelloWorld.

   1. Login to HPC
   2. Setup MPI toolchain:

      ```
      module purge
      module load gcc/8.3.0 cuda/10.1.243
      ```

   3. Compile

      ```
      nvcc -O3 hello.cu
      ```

   4. Run

      ```
      srun -n1 --gres=gpu:1  -t1 ./a.out
      ```

      The option `-t` specifies the limit of run time. Setting it as a small number will get your program scheduled earlier. For more information on `srun` options, you can use `man srun` to find out.

   5. Profile (optional)

      ```
      srun -n1 --gres=gpu:p100:1 --partition=debug nvprof ./a.out
      ```

   6. Allocate a machine

      ```
      salloc -n1 --gres=gpu:1 --mem=16G -t10
      // After the allocation, you will log on the machine and have
         10 minutes to perform multiple operations
      ./a.out
      // edit, compile, and run again without waiting for a new
         allocation
      ./a.out
      ./a.out
      ```

2. (15 points) Refer to the kernel `setRowReadColPad` in the file `checkSmemSquare.cu`. Make a new kernel named `setColRedRowPad` that pads by one column. Then, implement the operation of writing by columns and reading from rows. Run your code and observe the output. Optional: check memory transactions with `nvrpof`.

3. (15 points) Refer to the kernel `setRowReadCol` in the file `checkSmemRectangle.cu`. Make a new kernel named `setColRedRow` that writes by columns and reads by rows. Run your code and observe the output. Optional: check memory transactions with `nvrpof`.

4. (15 points) Design a CUDA program to perform matrix multiplication $C = A \times B$. The size of each matrix is $1K \times 1K$. Each element is 1 byte. The matrices are initially stored in the global memory of GPU. The GPU has one streaming multiprocessor (SM) with $1K$ CUDA cores. Each CUDA core runs at 1GHz and can perform one floating point operation in each clock cycle. The peak bandwidth between the GPU and the global memory is 100 GB/s and the cache of GPU has been disabled.

   - Design a simple CUDA program using straightforward vector inner product approach.
     - Write the pseudo code for your CUDA program including both kernel function and host function.
     - Derive a lower bound on the execution time of your kernel function? Explain.
   - Assuming the GPU has an on-chip buffer (shared memory) which can store $3 \times 32 \times 32$ elements and has a peak access bandwidth of 1 TB/s.
     - Write the pseudo code for an optimized CUDA program using block matrix multiplication approach including both kernel function and host function.
     - Derive a lower bound on the computation time, the local data access time in SM, and the global data access time of your optimized kernel function. Explain.

5. (10 points) If a kernel is launched with only one Warp (32 threads) in each thread block, the *__syncthreads*() instruction can be left out. Do you think this is correct? Explain.

Youwei Zhuo: youweizh@usc.edu

6. (30 points) Odd-even transposition sort algorithm (`https://www.geeksforgeeks.org/odd-even-transposition-sort-brick-sort-using-pthreads/`) is a parallel sorting algorithm. Design a CUDA program to sort a 2K-element integer array using odd-even sort transposition algorithm with 1K threads. The array is initially stored in the host memory. Write the pseudo code of your host function and kernel function. State any assumptions you may make.

   Submission Instructions: Submit your code, screenshots, and a performance report as described above.