EE 451: Parallel and Distributed Computation

PA7 — Spring 2021

Due date: Friday 16th April 2021 11:59 PM

1. **Examples**

   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 -arch=sm_20 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 `test_shfl_up` in the file `simpleShfl.cu`. Invoke it with a negative delta as follows:

   ```
   test_shfl_up<<<1, BDIMX>>>(d_outData, d_inData, -2);
   ```

   Check the results.

Youwei Zhuo: youweizh@usc.edu

3. (15 points) Refer to the kernel `test_shfl_wrap` in the file `simpleShfl.cu`.Make a new kernel that can generate the following result:

   Initial: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

   Result: 2 4 6 8 10 12 14 16 18 20 22 24 26 28 30 0

4. (15 points) Refer to the kernel `test_shfl_xor` in the file `simpleShfl.cu`.Make a new kernel that can generate the following result:

   Initial: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

   Result: 1 1 5 5 9 9 13 13 17 17 21 21 25 25 29 29

5. (15 points) Refer to the kernel `test_shfl_xor_array` in the file `simpleShfl.cu`. Make a new kernel that just performs one operation as follows:

   ```
   value[3] = __shfl_xor(value[0], mask, BDIMX);
   ```

   Check the results.

6. (20 points) Refer to the kernel `test_shfl_wrap` in the file `simpleShfl.cu`. Make a new kernel that can shift double-precision variables in a wrap-around wrap approach.

7. (20 points) Refer to the kernel `warpReduce` in the file `reduceIntegerShfl.cu`. Write an equivalent function that uses the `__shfl_down` instruction instead.

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

Youwei Zhuo: youweizh@usc.edu