EE 451: Parallel and Distributed Computation

PA5b — Spring 2021

Due date: Monday 29th March 2021 11:59 PM

1. **Examples**

   Copy example files to your home directory.

   1. Login to HPC
   2. Copy

      ```
      cp -r /project/xuehaiqi_652/cuda .
      ```

   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 --mem=16G -t1 ./a.out
      ```

      The option `-t` specifies the limit of run time. Setting it as a small number will get your program scheduled earlier. The option `--mem` specifies the minimum memory requirement. Setting it as a small number will get your program scheduled earlier. However, you need it when you run `sumArraysOnGPU` and `sumMatrixOnGPU`. 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. (40 points) In `reduceInteger.cu`, refer to the kernel `reduceUnrolling8` and replace the following code segment:

```
// unrolling 8
int a1 = g_idata[idx];
int a2 = g_idata[idx + blockDim.x];
int a3 = g_idata[idx + 2 * blockDim.x];
int a4 = g_idata[idx + 3 * blockDim.x];
int b1 = g_idata[idx + 4 * blockDim.x];
int b2 = g_idata[idx + 5 * blockDim.x];
int b3 = g_idata[idx + 6 * blockDim.x];
int b4 = g_idata[idx + 7 * blockDim.x];
```

with the functionally equivalent code below:

```
int *ptr = g_idata + idx;
int tmp = 0;
// Increment tmp 8 times with values strided by blockDim.x
for  *int i = 0; i < 8; i++) {
    tmp += *ptr;
    ptr += blockDim.x;
}
g_idata[idx] = tmp;
```

Compare the performance.

3. (30 points) Refer to the kernel `reduceInterleaved` and the kernel `reduceCompleteUnrollWraps8` and implement a version of each for `floats`. Compare their performance and explain any differences (profiling with `nvprof` is optional). Are there any differences compared to operating on integer data types?

4. (30 points) Refer to the file `nestedHelloWorld.cu` and implement a new kernel using the methods illustrated in Figure 3-30 (Change `igrid` to 2; Only one thread calls the kernel recursively). To compile it, add the following options.

```
nvcc -O3 -arch=sm_35 -rdc=true hello.cu
```

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

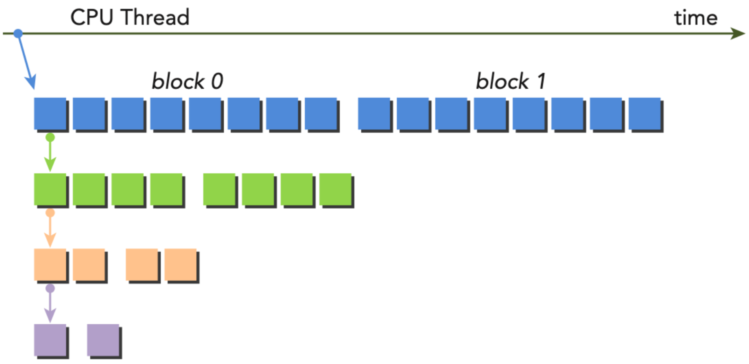

FIGURE 3-30