CSCI-GA.3033-004

Graphics Processing Units (GPUs): Architecture and Programming

Fall 2017 – (90 minutes)

Last Name: First Name: NetID:

- If you must make assumptions to continue solving a problem, state your assumptions clearly.
- You answer on the question sheet. You can use extra white papers if you want.

(8 questions …. Total of 30 points)

1. [3 points]We have seen many issues that can affect the performance of a kernel. State three issues that can affect the performance of a kernel, in no more than one sentence each.

- Branch divergence
- Non-coalesced memory access
- Too much resources per block, reducing number of blocks executed in parallel per SM.

2. [3 points] State three useful usages of streams.

- Execute multiple kernels at the same time
- Overlap communication and computation
- Ensure ordered execution for commands in the same stream

3. [4 points]Assume we have the following kernel for matrix multiplication. We launch it with matrices of size 1000x1000 and each block is 16x16 threads (i.e 2D blocks of 16x16). How many warps will have control divergence? Justify

```
__global__ void MatrixMulKernel(float* M, float* N, float* P, int Width)
{
   int Row = blockIdx.y*blockDim.y+threadIdx.y;
   int Col = blockIdx.x*blockDim.x+threadIdx.x;
   if ((Row < Width) && (Col < Width)) {
      float Pvalue = 0;
      for (int k = 0; k < Width; ++k)
         {Pvalue += M[Row*Width+k] * [k*Width+Col];}

      P[Row*Width+Col] = Pvalue;
   }
}
```

A: 500.

Explanation:There will be 63 blocks in the horizontal direction. 8 threads in the x dimension in each row will be in the invalid range. Every two rows form a warp. Therefore, there are 1000/2 =500 warps that will straddle the valid and invalid ranges in the horizontal direction. As for the warps in the bottom blocks, there are 8 warps in the valid range and 8 warps in the invalid range. Threads in these warps are either totally in the valid range or invalid range.

4. [3 points] For a vector addition, assume that the vector length is 4000, each thread calculates one output element, and the thread block size is 512 threads. The programmer configures the kernel launch to have a minimal number of blocks to cover all output elements. How many threads will be in the grid? To get full credit, show all steps.

A. 4096
ceil(4000/512)*512 = 8 * 512 = 4096. Another way to look at it is the minimal multiple of 512 to cover 4000 is 512*8 = 4096.

5. [2 points] In class, we have seen tiling as a useful technique in matrix multiplication. State two scenarios where tiling is useful at:

- To allow more the usage of shared memory to reduce global memory access.
- To allow dealing with very large matrices that cannot fit in global memory.

6. [2 points] We know that CUDA does not allow synchronization among threads in different blocks. Suppose CUDA allows this. State one potential problem that may arise.

- Not all blocks of the grid are executing at the same time. Some are waiting to be scheduled. This can be a cause of deadlock.

7. [2 points] We have seen many type of memories in the GPU. One of them is the local memory. Given that registers are used on per thread basis, what is the point of having a local memory?

For local variables that do not fit in a register, like arrays, structures, etc.

8. For the following sequential code (Assume matrix B is initialized to 0):

```
#define N 64
float A[N][N];
float B[N][N];
for (k=0; k<N; k++) {
        for (j=0; j<N; j++) {
                for (i=0; i<N; i++) {
                        B[i][j] += A[j][k];
                }
        }
}
```

Assume the GPU we will use has 8 SMs, and can have at most 8 blocks per SM, 512 threads per block, and 1024 threads per SM. Each SM has 64KB of shared memory. The GPU also has 6GB of global memory and 64KB of constant memory.

a. [2points] If you write a CUDA version of that code, what is the best location to store the matrix A? Why?

Constant memory because it is read only and its size is small.

b. [3 points] What will be the geometry of the CUDA kernel (grid and block sizes and dimensions) to get the most parallelism? Justify your choice

One possible solution (there are other correct solutions):
To use all SMs, and hence maximize parallelism, we want 8 SMs. The matrix B has 64*64= 4096 elements. If each thread is responsible for 1 element (to increase parallelism) then each block has 4096/8 = 512 threads, which is multiple of 32 (warp size). One possible arrangement is then to have 2x4 grid of 32x16 threads.

c. [2 points] Where will you store matrix B? Justify.

Initially B will be partitioned among the 8 shared memories of each SM. Then each block will write its portion to global memory at the end.

d. [4 points] Assume there no shared memory in the SM. Write the kernel that will be executed by each thread depending on your answer of question b above. We assume matrices A and B are already at the device global memory.

We show the code neglecting coping the data to the shared memory

```
int row;
int col;

row = blockIdx.y*blockDim.y + threadIdx.y;
col =  blockIdx.x*blockDim.x + threadIdx.x;

B[row][col] = 0;
for(int i = col; i < N; i++)
        B[row][col] += A[col][i];
```