GPU Programming

Bikash Gogoi
CS14B039

September 24, 2017

1. **Find all possible outputs for the following code.**

```
#include <cuda.h>
#include <stdio.h>

__global__ void alloutputs(int *counter) {
  atomicAdd(counter, 1);
  printf("%d\n", *counter);
}

int main() {
  int *counter, hcounter = 0;
  cudaMalloc(&counter, sizeof(int));
  cudaMemcpy(counter, &hcounter, sizeof(int), cudaMemcpyHostToDevice);
  alloutputs<<<1, 34>>>(counter);
  cudaDeviceSynchronize();
  return 0;
}
```

Solution:

If warp containing threads 0-31 completes first:

32 (32 times) and 34 (2 times) {0 of 2 incremented counter}
33 (32 times) and 34 (2 times) {1 of 2 incremented counter}
34 (32 times) and 34 (2 times) {2 of 2 incremented counter}

If warp containing threads 32-33 completes first:

34 (32 times) and 2 (2 times) {0 of 32 incremented counter}
34 (32 times) and 3 (2 times) {1 of 32 incremented counter}
.. ..
.. ..
34 (32 times) and 34 (2 times) {32 of 32 incremented counter}

2. **Write a kernel in pseudo-code (or CUDA) which takes an integer parameter whose value ranges from 1..32. Accordingly, the kernel achieves amount of coalescing. Thus, a value of 32 indicates fully coalesced access, while that of 1 indicates a fully- uncoalesced access.**

Solution:

```
__global__ void func(int *n, int *arr)
{
  int id = threadIdx.x;
  int a = arr[(33-n)*id];
}
```

3. **Write a kernel in pseudo-code (or CUDA) which takes an integer square matrix NxN size as a parameter and find the saddle point in it. A saddle point is an element which is maximum in its column and the minimum in its row. If there are multiple saddle points, print all. It is possible that there is no saddle point in a matrix. Assume that the number of threads with which the kernel is launched is equal to N.**

Solution:

```
__global__ void func(int *arr, int N)
{
  int i = threadIdx.x;
  int max = 0;
  for (int j=1; j<N; ++j) {
    if (arr[i*N+max] < arr[i*N+j]) {
      max = j;
    }
  }

  int isSaddle = 1;
  for (int k=0; k<N; ++k) {
    if (arr[k*N+max] < arr[i*N+max]) {
      isSaddle = 0;
    }
  }

  if (isSaddle) {
    printf("(%d, %d)\n", i, max);
  }
}
```