CSCI-UA.0480-009

Parallel Computing

Exam 2

Fall 2019 (60 minutes)

Last Name: **First Name:** **NetID:**

- This exam contains **8 questions** with a total of **50 points** in **5 pages.**
- If you must make assumptions to continue solving a problem, state your assumptions clearly.

1. [6 points] What are the three characteristics of a GPU friendly application?
   - Computationally intensive
   - Independent computations
   - Similar computations

2. [2 points] If four threads, in OpenMP, on a four-core processors, execute the instruction x++ where x is a shared variable initialized to 1, what are the possible values that x could have after the execution of the threads (assume no synchronization or precautions were taken)?

2, 3, 4, 5

3. [4 points] We have learned about the concept of warps in GPUs. Warps are transparent to the programmer, yet it is a very useful concept to know about. As a programmer, state two reasons that make the concept of warps beneficial to your GPU code.

   - Number of threads/block must be as close as possible to multiple of 32 to increase utilization.
   - Mut be careful with if-else in kernels to reduce branch divergence.

4. [12 points] For each variable in the following code: identify the scope of the variable, justify your choice, and for each variable identify potential race condition, if any, and justify. You can assume that some variable declarations have been made before this piece of code.

```
#pragma omp parallel for
for (i = 0; i < 1000; i++) {
        int x = 0;
        c++;
        for (j = i; j < 1000; j++)
                x += compute(c, a[j]);
        b[i] = x; }
```

Variable	Private/ shared	Why?	race cond? (Y/N)	Why?
a[]	shared	Declared outside the parallel construct.	N	No modification
b[]	shared	Declared outside the parallel construct.	Y	Shared and modified
c	shared	Declared outside the parallel construct.	Y	Shared and modified
i	private	Loop index is private by default in OpenMP	N	Private
j	shared	Declared outside the parallel construct.	Y	Shared and modified
x	private	Declared inside the parallel construct	N	Private

5. [4 points] Suppose a GPU has the following configurations:
1024 threads/block, 1024 threads/SM, 4 blocks/SM, and 32 threads/warp.
What is/are the best configuration(s) for threads per block to implement matrix addition: 4x4, 8x8, 16x16, 32x32, or 64x64? Justify your answer.

Best configuration is the one that will maximize number of threads in the SM (1024 in this problem). We can do that with two configurations:
16x16 blocks and use 4 blocks in the SM → 16x16x4 = 1024
32x32 blocks and use 1 block in the SM → 32x32 = 1024

6. [6 points] When there is a critical section in OpenMP, we need to deal with it. State three different ways we can deal with that in OpenMP.

- #pragma omp atomic

- #pragma omp critical

- locks

7. Assume we have the following piece of code that runs on a four-core processors. Each element of the array A[] is incremented by 1 and if it crosses 10, a counter is incremented. A[0] is also updated as shown. Also, assume variables have been defined before that code.

```
1.  #pragma omp for
2.  for (i = 1; i < 100; i++)
3.  {
4.        A[i]++;
5.        if (A[i] > 10)
6.              count++;
7.        #pragma omp atomic
8.        {
9.           A[0] = count*2;
10.          if( A[0] > 100) A[0] = 0;
11.       }
12. }
```

a. [4 points] There are two mistakes that makes the above code incorrect. What are they? What is the fix?

Line(s) containing the bug	Bug description	The fix
6	Critical section not dealt with	Either use private(count) close or use #pragma omp atomic
7	The atomic clause cannot be used in multi-line body.	#pragma omp critical

b. [3 points] With the code as written above, how many threads are generated? Is there a way to increase that? If so, what is it? If not, why not?

Only one thread exists because there is no *parallel* clause.
We can increase that using #pragma omp parallel for

8. [9 points] Circle the correct answer among the choices given. If you circle more than one answer, you will lose the grade of the corresponding question.

1. In OpenMP the code that is not covered by #pragma is executed by all threads.

a. True b. False c. Depends on the number of cores

2. Suppose each thread in OpenMP will do different tasks. If we know the number of tasks beforehand (i.e. while writing the code), then we use:

a. tasks b. sections c. critical sections

3. If the work in a for-loop body increases with the loop index. Then the best schedule to use in OpenMP is:

a. static b. dynamic c. They are both the same

4. Assume we dynamically allocate an array of int. Each thread, in OpenMP, may potentially update the same element in the array. The best way to deal with that is:

a. critical section without names b. critical sections with names c. locks

5. What type of parallelism is exploited by GPU based on Flynn's taxonomy?

a. SISD b. SIMD c. MISD d. MIMD

6. If a grid is 2D and has 2x2 blocks. Each block is 3D and has 2x3x2 threads. How many total threads we have?

a. 36 b. 48 c. 24 d.12

7. We can have two blocks executed on the same SM at the same time.

a. True b. False c. Depends on the OS d. Depends on the compiler

8. Threads in two different blocks in GPU cannot synchronize.

a. True b. False c. Depends on the OS d. Depends on the compiler

9. A variable in shared memory of an SM can be accessed by all threads in the grid.

a. True b. False c. Depends on the OS d. Depends on the compiler