CSCI-UA.0480-003

Parallel Computing

Final Exam

Spring 2014 - May 15th (90 minutes)

NAME: **ID:**

- This exam contains 8 questions with a total of 20 points in **four pages.**
- The exam is open book/notes.
- If you have to 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.

1. If three threads, in OpenMP, execute the instruction x++ where x is a shared variable initialized to 0, what are the possible values that x could have after the execution of the threads (assume no synchronization or precautions were taken)? Clearly explain how each value(s) results.

x=1 x=2 x=3 This is because x++ involves reading, updating, and writing. If interleaved, you may get wrong results.

2. When is loop-unrolling **not** beneficial? Assume we are talking about CUDA.

- **If it will add more branch divergence**
- **If it makes a thread require more registers and hence may reduce number of blocks assigned to an SM.**

3. Suppose we have a system with 1 CPU and 4 GPUs. The CPU can reach a performance of 0.5 GFLOPS, while each GPU can reach 1GFLOPS. If you have an application which is 50% parallelizable (i.e. 50% of the application is assigned to CPU and the other 50% to the GPUs), what is the average peak performance?

Scenario 1: CPU executes first then GPUs
max (0.5, 4*1) = 4 GFLOPS

Scenario 2: Both CPU and GPUs work in parallel:
0.5 + 4 = 4.5

4. Parallelize the following code using openMP pragmas. Assume that the target machine has a cache block size of 128 bytes, and that the size of an int is 4 bytes. A, B, and C are arrays and matrices of integers.
Be sure to explicitly specify the "schedule" options that should be used, even if you want to use the default options. Rewrite the code. You can assume that the variable P represents the number of processors to be used. Also assume that N is large (in the tens of thousands or more).

```
C[0] = 1;
for (i=1;i<N;i++){
        C[i] = C[i-1];
        for (j=0;j<N;j++){
                C[i] *= A[i][j] + B[i][j];
        }
}
```

Answer:

```
C[0] = 1;
for (i=1;i<N;i++){
        C[i] = C[i-1];
        #pragma omp parallel for schedule(static,N/P) reduction(*:product)
        for (j=0;j<N;j++){
                product *= A[i][j] + B[i][j];
        }
        C[i] *= product;
}
```

5. In bulleted list state the source(s) performance loss that we may face in: MPI, OpenMP, and CUDA (a bulleted list for each).
MPI:

- Communiation
- Synchronization
- Process creation overhead

OpenMP:

- Synchronization
- Coherence
- Communication (from memory to cores)
- Memory access

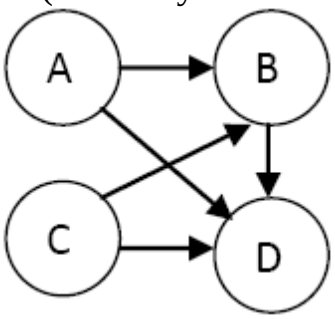
CUDA:

- Branch divergence
- Global memory access
- Data transfer

6. Explain whether we can have race condition in MPI, OpenMP, and CUDA, and justify in each case.

Only in CUDA and OpenMP because threads in both cases share memory.

7. Assume that all instructions of an application can be partitioned into 4 groups (A, B, C, D), with the following dependency. Each group contains 25% of the instructions. How to schedule them on your system (CPU + 4 GPU coprocessors) to achieve the best possible performance? Assume an instruction group can be assigned to either a CPU or GPU (Hint: Pay attention to communication cost!)



A on a GPU, C, B, and D in another GPU
or
A, B, and D on CPU (or GPU) and C on another GPU (or CPU)

8. Beside overlapping data-transfer and computation, state two other scenarios where streams are useful.

- **For correctness because a kernel launch and cudamemcpy are non-blocking. So, without putting them on the same stream, the memory copy may be executed before its kernel finishes its computations.**

- **Execute two kernels in parallel, if the hardware supports this.**