CSCI-UA.0480-051: Parallel Computing
Final Exam (May 15th, 2023)
Total: 100 points

Important Notes- READ BEFORE SOLVING THE EXAM

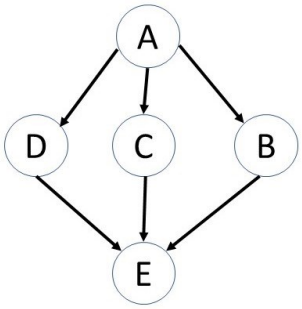
- **If you perceive any ambiguity in any of the questions, state your assumptions clearly and solve the problem based on your assumptions. We will grade both your solutions and your assumptions.**
- **This exam is take-home.**
- **The exam is posted on Brightspace, in the assignments section, at 2pm EST of May 15th.**
- **You have up to 24 hours from 2pm EST of May 15th to submit your answers on Brightspace (in the assignments section).**
- **You are allowed only one submission, unlike assignments and labs.**
- **Your answers must be very focused. You may be penalized for wrong answers and for putting irrelevant information in your answers.**
- **You must upload one pdf file.**
- **Your answer sheet must have a cover page (as indicated below) and one problem answer per page (e.g., problem 1 in separate page, problem 2 in another separate page, etc.).**
- **This exam has 3 problems totaling 100 points.**
- **The very first page of your answer is the cover page and must contain ONLY:**
  - **You Last Name**
  - **Your First Name**
  - **Your NetID**
  - **Copy and paste the honor code showed in the rectangle at the bottom of this page.**

Honor code (copy and paste the whole text shown below, in the rectangle, to the first page, cover page, of your exam)

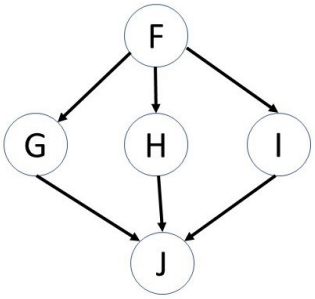
- You may use the textbook, slides, and any notes you have. But you may not use the internet.
- You may NOT use communication tools to collaborate with other humans.
- Do not try to search for answers on the internet. It will show in your answer, and you will earn an immediate grade of 0.
- Anyone found sharing answers or communicating with another student during the exam period will earn an immediate grade of 0.
- **"I understand the ground rules and agree to abide by them. I will not share answers or assist another student during this exam, nor will I seek assistance from another student or attempt to view their answers."**

Problem 1

Suppose we have the following two DAGs. Each DAG represents a process. That is, DAG 1 is a process and DAG 2 is another process. The two DAGs are totally independent from each other. The table shows the time taken by each task in each process.



DAG 1



DAG 2

TASK	TIME (ms)
A	15
B	5
C	5
D	15
E	15
F	5
G	15
H	15
I	15
J	5

a. [8 points] What will be the minimum time taken by each process if we execute it alone on one core, two cores, three cores, and four cores? That is, execute DAG 1 on one core, then on two cores, then on three cores, and then on four cores. Do the same for DAG 2. (Hint: You can put the result in the form of a table with three columns: #cores, time for DAG 1, and time for DAG 2. your table will have four rows for one, two, three, and four cores). Assume each core does not have hyperthreading technology.

#cores	DAG1	DAG2
1	55	55
2	45	40
3	45	25
4	45	25

b. [10 points] Based on the results you calculated in part a above, which DAG benefited more from using more cores? The two DAGs look similar, yet one DAG benefited from more cores. What is the main characteristic that this DAG has and makes it benefit from more cores?

[3] DAG 2 benefited more from more cores.
[7]The reason is that it has more load balancing among the parallel tasks {G, H, I}.
(If the answer just explained what the cores will execute and show a timeline but did not explicitly specify 'load balancing' then 4 points only instead of 7).

c. [10 points] Suppose DAG 1 is your media player and DAG 2 is your browser. You start these two processes exactly at the same time and you are using a machine with eight cores (with no hyperthreading technology). How long does each DAG take to finish? Justify your answer.

[3] DAG1 will take 45 and DAG2 will take 25.
[7] The reason is that with eight cores each DAG will have more than enough cores (each one does not need more than three cores) to get the same performance as if it is alone.

Problem 2

Suppose we have the following MPI+OpenMP piece of code. The whole program is not shown. The lines with dots (i.e. lines 2, 7, 10, and 12) contain some other code not needed for this problem. Some of the variables are used but not declared (e.g. num, procID, n, and arrays A and B). Assume they have been declared in a code not shown here. Assume this program runs on an eight cores processor. Each core is a four-way hyperthreading. Each core has its own private level 1 cache. Each level 1 cache is 32KB. Each core also has its own level 2 cache of size 1MB each. There is a shared level 3 cache of size 8MB. The program is executed using the following command: (progname is the name of the program's executable).

mpiexec -n 4 *./progname*

```
1.  int main(int argc, char **argv){
2.  .....
3.      float finalresult, sigma;
4.      MPI_Init(&argc, &argv);
5.      MPI_Comm_size(MPI_COMM_WORLD,  &num);
6.      MPI_Comm_rank(MPI_COMM_WORLD, &procID);
7.  .....
8.      sigma = procID / num;
9.      finalresult = findNum(A, B,  n, sigma);
10. .....
11.     MPI_Finalize();
12. .....
13. }
14.
15.   float  findNum(float * A, float  * B, int n, float sigma)
16.   {
17.     float result = 0, total = 0;
18.     int i;
19.     #pragma omp parallel for reduction(+:result) numthreads(4)
20.     for (i=0; i<n; i++)
21.     {
22.       float factor;
23.       factor = A[i] * B[i];
24.       result += factor * sigma *i;
25.     }
26.      MPI_Allreduce(&result, &total,1,MPI_FLOAT,MPI_SUM,MPI_COMM_WORLD);
27.     return total;
28.  }
```

a. [20 points] Fill in the right column of the following table with a short answer to the questions on the left. Please do not write any justification unless the question asks for it explicitly.

How many processes were created in the whole system?	4
How many threads are generated in total?	16
What is the maximum number of those threads (that you mentioned above) that can execute in parallel?	16
How many copies of the variable 'sigma' were created in the whole system?	4
How many copies of the variable 'factor' were created in the whole system?	16
How many threads will execute the code in line 12 (the dots ….)?	4
How many threads will execute the code in line 24?	16
How many threads will execute the code in line 26?	4
Is there a race condition for line 23?	No
Justify your answer to the above question in one line	'Factor' is private. Tt is declared inside the parallel region.

b. [8 points] Is there a situation where threads, in the above code, accessing the arrays A[] and B[] can cause the coherence protocol to start? If yes, what is the situation? If not, why not?

[3] No
[5] A[] and B[] are not written and only read from. Coherence is not activated by reading.

c. [8 points] Given the code above, how many virtual address spaces did the OS create? Justify your answer in 1-2 lines only.

[4] Four virtual addresses
[4] The OS creates a virtual address space per process.

d. [8 points] Is there a possibility that a process reaches line 27 before the other processes? If yes, will this cause a problem, and what is that problem? If not, why not?

[3] No
[5] MPI_Allreduce is a collective call and hence is blocking.

e. [8 points] If one of the processes created for the above code crashed for some reason, do we risk having a deadlock? Justify.

[3] Yes
[5] If the process crashes before executing MPI_Allreduce, then a deadlock will occur.

Problem 3

For each question below, choose all correct answers. That is, a question may have one or more correct answers.

Note on grading: (-1) for each wrong choice (i.e. if a wrong answer is chosen or right answer not chosen).

a. [4 points] Suppose a process wants to send data to a subset of processes. That process has the following options:
1. Split the communicator to smaller ones and use collective communication.
2. Make a series of send and receive to each one of the destination processes.
3. Use broadcast call.
4. Split each process into multiple threads and let threads communicate through shared memory.

b. [4 points] A warp in an NVIDIA GPU:
1. is transparent to the programmer.
2. consists of a maximum of 32 threads.
3. has each four threads share the same fetch and decode hardware.
4. suffer from thread divergence, also called branch divergence, possibility.

c. [4 points] A block in CUDA can be split among two SMs.
1. This statement is always true.
2. This statement is true if the block has more threads than the number of SPs in the SM.
3. This statement is always false.
4. This statement is false only if the number of threads in the block is less than the number of SPs in the SM.

d. [4 points] The following characteristics are needed for a code to be GPU friendly.
1. computation intensive
2. independent computations
3. similar computations
4. large problem size

e. [4 points] Choose all the correct statements from the following one:
1. If one program has a higher speedup than another program, for the same number of cores, it means that the program with the higher speedup also has higher efficiency than the other one.
2. MPI can run on distributed memory machines and shared memory machines.
3. OpenMP can run on distributed memory machines and shared memory machines.
4. Power consumption/dissipation is the main reason we moved from single core to multicore.