CSCI-UA.0480-003

Parallel Computing

Midterm Exam

Spring 2020 (70 minutes)

[50 points]

- All questions in this exam require very short answers [1-2 lines]. You will be penalized for wrong answers as well as unneeded sentences. So, be very specific in your answer.
- Before you start solving the exam, copy and paste the honor code in "midterm exam" description in the assignment section on NYU classes. At the bottom of the honor code put your last name/ first name and your NetID. Your submission will not be accepted without it.

1. [2 points] Why speculative execution helps making the best use superscalar capability?

There are many branches in any program. Branches need to be executed to know which instruction to fetch and this delays the fetch and hence affects the superscalar capability. Branch prediction solves this issue and is called the speculative execution.

2. [2 points] Suppose we have a four-core machine. Each core has two-way hyperthreading technology. You want to run four processes each of which uses three threads. Does the system need to use multitasking? Justify your answer.

Yes, we need. Because we have eight logical cores but need to execute 12 threads.

3. [4 points] State two reasons as to why in shared-memory machines coherence protocol is not enough to take care of critical sections and we still need to use techniques such as locks.

- **Critical section may consist of several instructions and can span several cache blocks but coherence deals only with one block access at a time.**
- **Two threads may execute on the same core and hence use the same cache. In which case, coherence won't help much.**

4. [2 points] Can two processes write to the same physical memory address? Justify.

No, they cannot because each process has its own virtual address space and the OS will ensure, though page table entries, that not two processes write to the same physical address.

5. [2 points] Can two threads, corresponding to the same process, write to the same physical memory address? Justify.

Yes, they can because they share the same virtual address space. So, if they write to the same virtual address it is writing to the same physical address.

6. [4 points] For each one of the following performance measures, indicate whether that measure is higher is better (H) or lower is better (L)

- Speedup: H
- CPI: L
- MIPS: H
- Throughput: H
- Efficiency: H
- Response time: L
- Cycle time: L
- Frequency: H

7. Assume the following three processes in MPI and the operations they do at each time step of the same program. Also assume that the destination is always process 1 and that the reduction operation is MPI_MAX.

Time	Process 0	Process 1	Process 2
0	x = 7; y = 3;	x = 3; y = 9;	x = 7; y = 5;
1	MPI_Reduce(&x, &y, …)	MPI_Reduce(&y, &x, …)	MPI_Reduce(&x, &y, …)
2	MPI_Allreduce(&y, &x, …)	MPI_Allreduce(&x, &y, …)	MPI_Allreduce(&x, &y, …)

a. [6 points] Fill in the following table with values of x and y in each process:

After Step 1:

	Process 0	Process 1	Process 2
x	7	9	7
y	3	9	5

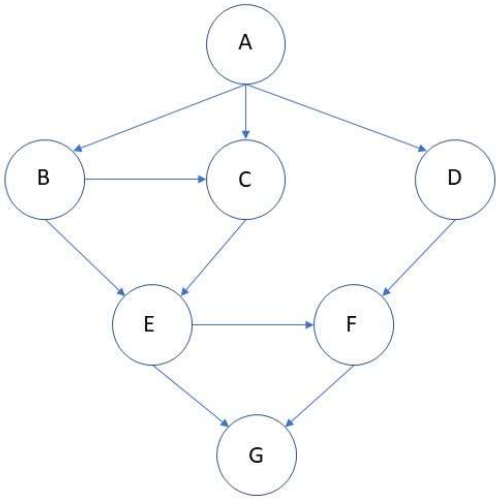
After Step 2:

	Process 0	Process 1	Process 2
x	9	9	7
y	3	9	9

b. [4 points] Can the above code produce wrong result if a process moves to step 2 while the other processes are still in step 1? Justify your answer.

No, the code cannot produce wrong results because collective calls are blocking. Hence, no process can proceed to the next instruction till all processes are done with that collective call.

8. Suppose we have the following DAG that represents a program.



Also suppose each task takes the following amount of times:

A	5
B	5
C	2
D	9
E	1
F	5
G	5

a. [1 point] What is the span of that DAG? **A D F G → 5+9+5+5 = 24**

b. [1 point] What is the work of that DAG? **The total of all nodes = 32**

c. [3 points] What is the minimum number of cores that we need to get the best performance? Show how you reached that number.

Two cores → ceil(work/span) = ceil(32/24) = 2

d. [2 points] Given the number of cores that you calculated above, what is the total execution time? Show how you reached this result.

5 + 9 + 5 + 5 = 24

e. [4 points] If you can remove one dependence (i.e. one arrow) only, which one will you remove to get the best performance? And how many cores do you need to get this best performance? Show all the steps that lead to your answer in order to get full credit. **No arrow can be legally removed and give best performance because D is much longer than others.**

9. Suppose that MPI COMM WORLD consists of three processes: 0,1, and 2. Also suppose the following code is executed (my_rank contains the rank of the executing process):

```
int x, y, z;

switch(my_rank) {

        case 0:
                x=5; y=7; z=11;
                MPI_Bcast(&x, 1, MPI_INT, 0, MPI_COMM_WORLD);
                MPI_Bcast(&z, 1, MPI_INT, 1, MPI_COMM_WORLD);
                break;
        case 1:
                x=2; y=4; z=6;
                MPI_Bcast(&y, 1, MPI_INT, 0, MPI_COMM_WORLD);
                MPI_Send(&y, 1, MPI_INT, 2, 43, MPI_COMM_WORLD);
                MPI_Bcast(&y, 1, MPI_INT, 1, MPI_COMM_WORLD);
                break;
        case 2:
                x=1; y=3; z=5;
                MPI_Bcast(&z, 1, MPI_INT, 0, MPI_COMM_WORLD);
                MPI_Recv(&x, 1, MPI_INT, 1, 43, MPI_COMM_WORLD, &status);
                MPI_Bcast(&y, 1, MPI_INT, 1, MPI_COMM_WORLD);
                break;
        }
```

a. [9 points] What will be the values of x, y, and z for each of the 3 processes after executing the above code?

	Po	**P1**	**P2**
X	5	2	5
Y	7	5	5
Z	5	6	5

b. [4 points] If we execute the above code with *mpiexec -n 4* what will happen? Explain your answer.

We will have a deadlock because process 3 will not call the collective API.