CSCI-UA.0480-003

Parallel Computing

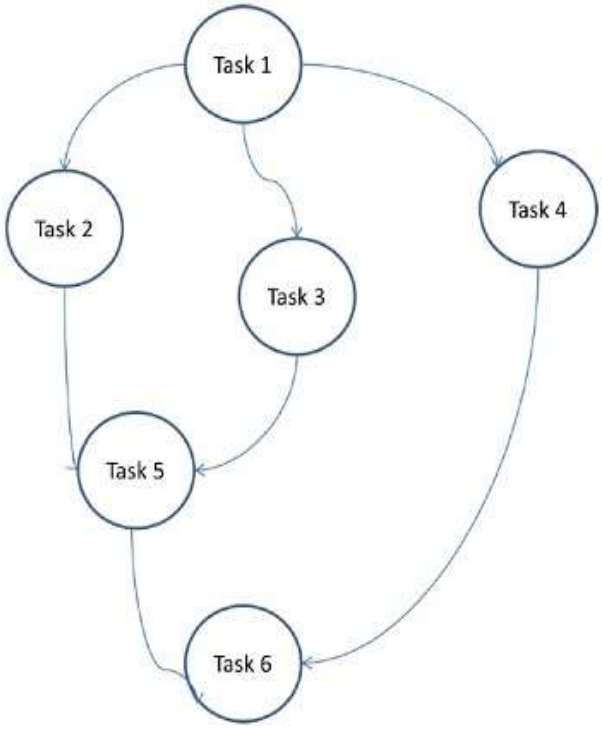
Midterm Exam

Spring 2019 (60 minutes)

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

- This exam contains **6 questions** with a total of **30 points** in **4 pages.**
- If you must make assumptions to continue solving a problem, state your assumptions clearly.

1. Assume we have the following task graph. A task can be thought of as function/procedure. An arrow from a task to another means that the first task generates data needed by the second one. Do not make any assumptions regarding cache misses, coherence, etc. The table indicates the time taken by each task.



Task#	Duration
1	5
2	10
3	10
4	100
5	20
6	5

a. [2 points] What is the smallest number of cores needed to get the best performance? Explain.

Two cores only are needed.
Because task 4 is too long so, if we use more cores to speedup the execution of tasks 2, 3, and 5, we still need to wait for task 4 to finish.

b. [2 points] Based on the number of cores you picked in part (a) above, what is the total run time?

It will be task 1 + task 4 + task 6 = 5 + 100 + 5 = 110

c. [2 points] If we use 6 cores, what will be the efficiency?

Efficiency = Speedup/#cores
Speedup = time single core / time 6 cores = 150/110
→ Efficiency = (150/110)/6 = 15/66 = ~22.7%

d. [2 points] What is the span for the above graph? what is the work?

Span = task 1 → task 4 → task 6 = 110
Work = total amount of time spent on all tasks = 150

e. [1 point] What is the parallelism of that graph?

Parallelism = work/span = 150/110 = ~1.4

f. [1 point] What does the number you calculated in (e) mean?

It means more than 2 cores, that is ceil(150/110), will not yield any more speedup or performance.

g. [2 points] Suppose that task 2 takes 100 units of time instead of 10, what will be your answer to question (a) above? Justify

We can use three cores: will finish tasks 2, 3, 4, and 5 in 100 +20 = 120 units of time. The we add tasks 1 and 6 reaching 130.
If we use two cores, tasks 2, 3, 4, and 5 finish in 100 + 10 + 20 = 130 units of time, making the total reach 140. The difference is small though.

2. [3 points] We have studied Amdahl's law in class. Suppose that the fraction of the sequential part of your program is 50% and we have four cores. What is the best action to get better performance: trying to reduce the sequential part to 20%? Or keeping it at 50% but use five cores instead of four? Justify your answer using Amdahl's law.

Amdahl's law → Speedup = 1/[F + (1-F)/p] where F is the percentage of sequential part and p is the number of cores.

Speedup (F = 20% and 4 cores) = 1/(0.2 + (0.8/4)) = 5/2
Speedup (F = 0.5 and 5 cores) = 1/(0.5 + (0.5/5)) = 5/3

The first scenario is better because it yields higher speedup.

3. [4 points] Suppose you have an MPI program and a sequential program. State two reasons that can make the MPI running with p processes slower than the sequential code (p > 1). [Note: You will be penalized for a wrong statement.]

- **Not enough parallelism to overcome the overhead of creating processes.**
- **Too much communication overhead**
- **Severe load balancing with large number of synchronization points**

4. [3 points] Suppose we want to parallelize factorial n (that is, n!), using MPI, with p processes. Assume that n is divisible by p and n > p. State the steps needed to do it such that the result is at process 0? Initially, only process 0 has the value of n. Do not write code but you can mention the name of the APIs you will use. Put your answer in a numbered list (step 1, step 2, ….).

1. **Process 0 uses <u>MPI+Bcast</u> to send n to all processes.**
2. **Each process calculates its range such that: process 0 takes 1→n/p, process 1 takes (n/p)+1 →2n/p, …., till process p-1 takes (p-1)(n/p)+1 → n This can be done with an MPI as: rank*(n/p)+1 → (rank+1)(n/p)**
3. **Each process multiples its own numbers range and obtains partial results**
4. **All processes use <u>MPI_Reduce</u> with operation <u>MPI_PROD</u> and destination 0.**

5. [2 points] As we increase the number of processes in MPI, does coherence protocol overhead becomes more severe? Justify.

No, no coherence is needed at all in MPI because the processes are not sharing any address space. For example, if two processes try to update variable x at the same time, they are accessing two different variables.

6. [6 points] 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 0 and that the reduction operation is MPI_SUM.

Time	Process 0	Process 1	Process 2
0	x = 2; y = 3;	x = 3; y = 4;	x = 4; y = 5;
1	MPI_Reduce(&x, &y, …)	MPI_Reduce(&y, &x, …)	MPI_Reduce(&x, &y, …)
2	MPI_Reduce(&y, &x, …)	MPI_Reduce(&x, &y, …)	MPI_Reduce(&x, &y, …)

Fill in the following table with values of x and y in each process
After Step 1:

	Process 0	Process 1	Process 2
x	2	3	4
y	10	4	5

After Step 2:

	Process 0	Process 1	Process 2
x	17	3	4
y	10	4	5