CSCI-UA.0480-051: Parallel Computing
Midterm Exam (October 21st, 2020)

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 NYU classes, at the beginning of the Oct 21st lecture.**
- **You have up to 23 hours and 55 minutes to submit on NYU classes.**
- **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.**
- **Your answer sheet must have a cover page (as indicated below) and one problem answer per page. This exam has 6 problems.**
- **The very first page of your answer must contain:**
  - **You Last Name**
  - **Your First Name**
  - **Your NetID**
  - **Copy and paste the honor code showed in the rectangle at the bottom of this page.**

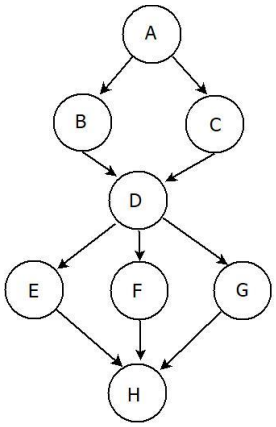
Honor code (copy and paste to the first page of your exam)

- 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. This includes but is not limited to G-Chat, Messenger, E-mail, etc.
- 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."**

The numbers between brackets in the solutions indicate the credit for each part of the answer.

Problem 1

Assume we have the following task flow graph where every node is a task and an arrow from a task to another means dependencies. For example, task B cannot start before task A is done.



The following table shows the time taken by each task in nano seconds.

Task	Time Taken
A	50
B	10
C	10
D	5
E	10
F	100
G	10
H	100

a. [4 points] What is the minimum number of cores needed to get the highest performance? What is the speedup given the number of cores you calculated?

We need two cores [2].
Speedup (2) = time sequential / time with 2 cores = 295/(50+10+5+100+100) = 295/265=1.11 [2]

b. [4 points] What is the span? What is the work?

[2] span = A→(B or C) → D → F → H = 50+10+5+100+100=265
[2] Work = total tasks = 295

c. [4 points] Given what you calculated in (b) above, what is the parallelism? And what does the parallelism number you calculated mean?

Parallelism = work/span = 295/265 = 1.11 [1]
It means that ceil (1.11) = 2 is the maximum number of cores after which we will see no performance gain [3].

d. [4 points] Is the number you calculated in (c) different than the number of you calculated in (a)? Justify.

No, they are the same. [1]
Because the parallelism gives the maximum number of cores to gain performance. That number is 2. And, in (a), we picked two cores as the minimum number of cores to gain performance. So, more than 2 will not give performance and less than 2 is the sequential. [3]

e. [4 points] Do you think we can have a loop in the task graph above (i.e. an arrow pointing to an earlier task)? If no, how can iterations be represented? If yes, how will you calculate the span in that case?

We can have loops. But we need to know the number of iterations beforehand to be able to analyze the parallelism. Even if nodes represent functions and those functions have loops, we need to know the number of iterations to have a prediction of the time taken by those functions. [4] [Note: Other logical answers can also be considered correct].

f. [4 points] Suppose we have two scenarios: In one of them we have x cores and in the other we have y cores. Choose x and y such that x cores gives you lower speedup but higher efficiency for the graph above. than y cores? Is it possible to do that? If yes, please given an example using the above graph (and calculate both the speedup and efficiency). If not, explain in 1-2 sentences why not.

Yes, it is possible. [1]
If x = 1 and y = 2. [1]
[1] In case x =1 → speedup = 1 → efficiency = 100% (lower speedup and higher efficiency)
[1] In case y = 2 → speedup = 1.11 → efficiency = 1.11/2 = 55.5% (higher speedup and lower efficiency)

Problem 2

Answer the following questions about technology and hardware design.

a. [4 points] What is the importance of Moore's law that is related to multicore processors?

Moore's law becomes related to performance because more transistors means more features which is translated to more performance [1]. However, more transistors when transistors became so small, means more non-manageable power consumption and dissipation [1]. This requires reducing frequency and voltage [1]. To sustain performance, we need to add more cores with low performance [1].

b. [4 points] Why it is not easy to just write a sequential program and let the compiler generate the parallel version of it? You answer must not exceed two sentences.

Because it is sometime impossible, at compile time, to know whether two parts of the software have dependencies or not. This is most apparent in case of pointers. [4]

c. [4 points] We say that in hardware pipelining, with five stages (fetch, decode, issue, execute, commit), the hardware can execute several instructions at the same time but each instruction is in different phase of its lifetime. That is, one instruction in its fetch phase, another in its decode phase, a third in the issue phase, etc. So, if we increase the number of stages from five, as we saw in class, to ten, does this guarantee we will have ten instructions being executed in parallel most of the time? Why? or Why not?

No, it does not guarantee that [1].
Because we may have conditional branches. And, even if we use branch prediction, the predictions can be wrong sometimes, reducing the number of instructions in the pipeline. [3]

d. [4 points] Since MIMD can implement all the other types (SISD, SIMD, and MISD). Then why did we build chips based on SIMD (like GPUs) and did not build all chips as MIMD?

Because we have many applications that follow a SIMD executions [2]. And SIMD allows to have way more execution units than MIMD because we get rid of many front ends (fetch, decode units) [2].

e. [4 points] We saw four different categories of architecture in Flynn's taxonomy. What is the classification of single core with speculative execution and superscalar capability? Justify your choice.

It is MIMD [2] because it can executes several instructions at the same time and each instruction uses different data [2].

f. [4 points] As the number of cores increases, cache coherence protocols are implemented as directory-based and not snoopy, why is that (in 1-2 sentences)?

Because snoopy requires the usage of bus interconnections [2]. Bus is not a scalable topology [2].

Problem 3

Answer the following questions regarding parallel programming techniques.

a. [5 points] State five reasons, each one not exceeding two sentences, about why it is almost impossible to eliminate load imbalance in parallel programs.

Even if all threads are assigned the same amount of work:

- One thread can have cache misses while the others not.
- One thread can have page faults while the others not.
- A core where a thread is assigned can become hot and must reduce its own frequency and voltage, hence comes slower.
- More than one thread is assigned to the same core, due to the larger number of threads related to cores, and hence will execute sequentially.
- The code of thread itself may have if-else structures which makes it hard to ensure that all threads will follow the same path.
- Coherence delays some threads when more than one threads are trying to update the same cache block.
- Accessing the memory or cache does not have the same latency (NUMA and NUCA).

[Only five reasons are needed.]

b. [4 points] Why is it good to start by writing a sequential code of the problem to be solved then parallelize that code? State one reason in 1-2 sentences.

To have something to compare with when we are done with the parallel code. Otherwise, how will you know the speedup of your code?

c. [10 points, 1 per entry] For each one of the following problems, choose whether you will parallelize it using task-level, or divide and conquer. Then add 1-2 sentences justifying your choice.

Problem	**Task level or divide and conquer**	**Justification**
matrix multiplication	Task level	Elements of the result matrix are totally independent.
factorial n	Divide and conquer	You can do factorial for smaller numbers (divide and conquer). But, you have to combine them to get the final answer.
matrix transpose	Task level	If you give corresponding elements in the matrix (in triangular fashion) to different threads/processes, they can be done in parallel.
matrix addition	Task level	Elements of the result matrix are totally independent.
find all even numbers in a list	Task level	You can divide elements of the list among threads/processes and results can be generated in parallel.

Problem 4

[4 points] Suppose we have the following two pieces of code.

….
x = TRUE;
….
	…
	if(!x)
	  x = TRUE;
	….

A program wants to decide whether to use the code on the left or the code on the right in a multithreaded program. It is obvious that both are doing the same thing. Which one do you think has the potential of being faster? and why? (Not giving a justification will not earn you any credit).

Even though the code on the right will be translated to more assembly instructions that the code on the left, it is the one on the right that can potentially be faster. [1]
The main reason is that the code on the left will potentially trigger the coherence protocol more often than the one on the right, that has the if-condition. [3]

Problem 5

[9 points 3 points for each a, b, and c]
Suppose we have a quad-core processor. Each core has its own L1 cache. There is a shared L2 cache. The block size for all caches is 32 bytes in size. We have an array of 256 integers. We want to increment each element of that list by one. We want to do that by writing a multithreaded program. The elements of this array will be divided among the four cores. State which elements of that array will be assigned to which core to reduce the possibility of false sharing and ensure load balancing, in each one of the following scenarios. Also, for each one, give one sentence justification:

a. Each L1 cache is 64KB of size and is 2-way set associative.
b. Each L1 cache is 32KB of size and is 4-way set associative.
c. Each L1 cache is 16KB and is 8-way set associate.

Simply ensure that no two threads on different cores will be updating elements of the same cache block [1].
Since each cache block is 32 bytes = 8 integers, then any two threads must be working on elements that are 8 elements away in the array from each other [1].
This can be easily done by giving each thread 256/4 = 64 elements to work on [1].

It is the same answer for a, b, and c. Because the answer has nothing to do with the cache size of associativity.

Problem 6

Answer the following questions regarding MPI.

a. [4 points] If we run MPI on a shared memory machine, do we risk suffering from coherence? Justify in 1-2 sentences.

No [1],
because each process has its own virtual address space. The OS will ensure that no two virtual address spaces from different threads map to the same physical address [3].

b. [4 points] In 1-2 sentences explain why MPI gives each process a rank?

To be able to use this rank to make each process do a different work (or different part of the problem to be solved) based on its rank. Otherwise, they will redundantly do the same work.

c. [4 points] We saw the API MPI_Comm_size() that gives the total number of processes. But, we already specify the number of processes when we write execution command. Then, why do we need such an API? State two reasons.

Because the MPI programmer does not know, at the time of code writing, the number of processes that the user will type.

d. [4 points] What is the reason for having something called communicator in MPI?

To be able to broadcast or do collective operations to a sub-set of processes, if we have a communicator that encompasses this sub-set.

e. [4 points] Most collective calls in MPI have either a field for source process or a field for destination process, but never both. Why is that? State your reason in 1-2 sentences.

Because collective call involves either broadcasting data from a source to others (needs source), gathering processed data from other processes to a destination (needs destination), or collectively requires results in all processes, like MPI_Allreduce() (does not need neither source not destination).