NATIONAL UNIVERSITY OF SINGAPORE

SCHOOL OF COMPUTING

CS3210 – Parallel Computing

(Semester 1: AY2019/20)

Final Assessment Paper - Answers

Time Allowed: 2 hours

INSTRUCTIONS TO STUDENTS

1. Write your Student Number only. Do not write your name.
2. This assessment paper contains **FOUR** questions and comprises **FOURTEEN** printed pages.
3. Students are required to answer **ALL** questions within the space in this booklet.
4. This is a CLOSED BOOK assessment. An A4 handwritten reference sheet and calculators are allowed.
5. Feel free to use pencil for answering the questions.
6. Write your Student Number below.

STUDENT NO: _____

This portion is for examiner's use only

Question	Marks	Remarks
Q1	/20	
Q2	/15	
Q3	/30	
Q4	/15	
Total	/80	

Q1. [20 marks] General parallelism questions

a. List and explain two advantages and two disadvantages in using threads in OpenMP and CUDA. Mention your assumptions. (8 marks)

Answer:

Threads in CUDA are light weight, faster to create/destroy, have restricted resources to use (registers, shared memory)

But threads in CUDA are grouped in warps and it is advisable for threads in the same warp to have the same control flow.

Students had to mention 4 points about CUDA and OpenMP. 2 general points about threads were acceptable, but 2 points had to be specific to CUDA and OpenMP.

b. Heterogeneous platforms achieve good performance while running programs in an energy efficient way. Explain how this is achieved by giving two examples of how energy consumption can be improved when using heterogeneous platforms. (4 marks)

Answer:

Core performance can be adjusted based on the tasks that have to be completed. For example, turn off or decrease the performance of the big cores when the phone is in the pocket and performing short, sequential, background tasks.

c. Explain the differences between data centers and cloud computing. Give at least two differences. (4 marks)

Answer:

Cloud computing is a data center with layer of software over it. The cloud can offer services at different levels (platform, software, infrastructure), while the data centers offer the infrastructure service only. Cloud services are pay-per-use model, while data centers are a huge investment in the beginning.

d. In designing parallel programs, task size (granularity) is important. Explain by giving two examples how task granularity impacts the program's performance (4 marks).

Answer:

A small granularity can introduce a lot of parallelism overhead (creation and merging of threads/process, use of shared resources or communication overhead). A large granularity might make exploiting parallelism difficult (for example, some tasks are too long and take longer than the rest; or there are not enough tasks to run on all cores)

Q2. [20 marks] Parallelism and performance

a. Intel Xeon Phi processor uses a mesh network with "YX" message routing among its cores. In YX routing, messages move vertically in the mesh until they reach the destination row, and then move horizontally through the network until they reach the destination node.
Consider the mesh shown in Figure 1. Each link in the network is capable of transmitting one byte of data per clock.
Two messages are sent on the mesh. Both messages are sent at the same time. Message 1 is sent from node 0 to node 14. Message 2 is sent from node 11 to node 13. Both messages contain two packets of information and each packet is 4 bytes in size.
Your friend looks at message workload and says: "It looks like we're going to need a more complicated routing scheme to avoid contention."
Do you agree or disagree? Explain your answer (3 marks).

Answer:
Contention does not exist. It will take 3×4=12 cycles before the first packet from Message 1 arrives at node 11. However, after 8 cycles the Message 2 has already completely been transmitted over this link, so the link is free.

Some student considered that the packets are sent byte by byte, not by packet. 2 marks were awarded to such solutions.

b. Answer points i. – iii. based on the following task dependence graph in Figure 1:

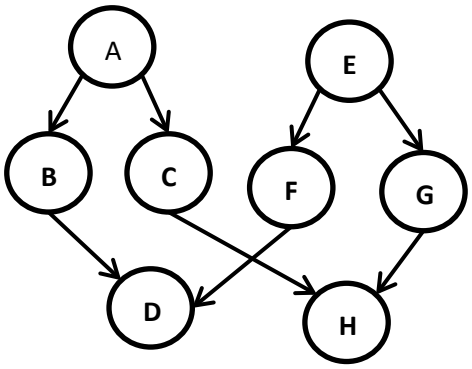


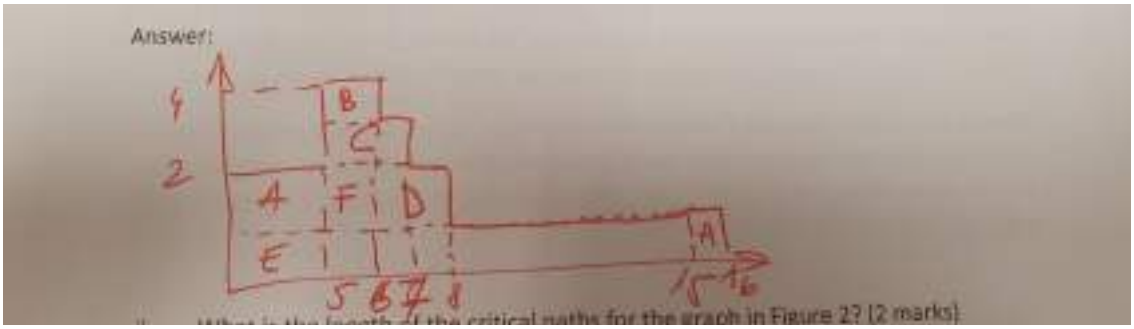
Figure 1: Task dependency graph

Assume tasks are of different sizes and they take the following units of time to execute:

Task	A	B	C	D	E	F	G	H
Time units	5	1	2	2	5	1	10	1

i. Plot the degree of concurrency (or degree of task parallelism) over time for the graph in Figure 2. (2 marks)

Answer:



ii. What is the length of the critical paths for the graph in Figure 2? (2 marks)

Answer:

16: E->G->H

iii. What is the maximum achievable speedup given infinite resources (Figure 2)? (2 marks)

Answer:

27/16 = 1.6875

c. A program spends 20% of its execution time within inherently sequential code. What is the maximum speedup achievable by a parallel version of the program? Briefly explain what law applies best for this scenario (4 marks).

Answer:

F =0.2
S = 1/(0.2 + 0.8/infinite) = 5

Amdhal's Law

d. An application running on 10 processors spends 3% of its time in serial code. What is the maximum speedup for this application? Briefly explain what law applies best for this scenario (2 marks).

Answer:
S = (10*T+0.03*T) / (T +0.03 * T) = p/(1+f) + (p+f)/(1+f) = 10.03 / 1.03 = 9.737

Gustafson Law.

It was accepted to consider T as the parallel time.

Problem description for Q3: TOP10

Consider the problem of finding the maximum, MAX (A), of N integer numbers stored in an array A. Additionally, the program needs to find the numbers that are at most 10% less than the maximum. Essentially the program needs to return MAX(A) and all a_i values from A such that: a_i >= 0.9 * MAX(A). Assume N is very large. We call this problem TOP10.

In a simplistic sequential approach, TOP10 problem would be solved by first finding the maximum value MAX(A) in A, followed by another iteration through the elements of A to find the numbers that are at most 10% less than MAX(A). For brevity of your pseudo-code, you can assume you have access to a **sequential** sort function (but it is not mandatory to use it).

Assume you need to solve the TOP10 problem in parallel, using a shared memory machine and a distributed memory machine. You may have two different solutions (algorithms) for each of the machines. You should aim to optimize your algorithms considering that N is very large.

Q3. [28 marks] Parallel Programming Models

a. You are first solving TOP10 problem for the shared memory machine and assume A can fit in memory. Explain how would you design your parallel algorithm? You may use OpenMP-like pseudo-code to show your algorithm.

   Assume that the the number of elements greater than 0.9*MAX(A) is much smaller than N. In your solution, focus on identifying the problem's parallelism and on explaining the steps of your solution. Do not focus on optimizing your solution (4 marks).

Answer:

Any approach that divides the data into chunks to find the local maximum is accepted. No need to optimize.

Split A into P tasks, each task processing N/P elements (so data parallelism). Let's call A_K the elements assigned to task K. A simple approach is for each task K to take one pass, determine the maximum value, MAX(A_K) and compute 0.9*MAX(A_K). In a second pass we identify all elements >= 0.9*MAX(A_K), which we denote as $a_{jk.}$

We can then use an odd-even approach to collect the global MAX(A) and ai into the memory of task K

b. What type of parallelism (data or task) does TOP10 problem entail for your solution from point a.? Explain your choice (2 marks).

Answer:

Data or task is acceptable, depending on the implementation.

c. You are now solving TOP10 problem for the distributed memory machine. Assume the machine has P cores (each one with its memory) and A cannot fit in memory. However, each core can hold N/P in its local memory. P << N, and P is large.

   Which data distribution pattern would you use to solve TOP10 problem? Present the data distribution for array A (4 marks).

Answer:

Blockwise data dist. Each P gets N/P elements of A

d. Use the distributed machine from point c.
Assume you are using MPI to implement a parallel program to solve TOP10 problem. Initially A is located at process rank 0 (on disk, as A cannot fit in the memory of a single node), and result should be located at process rank 0 at the end of the execution.
   - Explain how the data and task distribution would be.
   - Explain what each participant process would do.
   - Explain how the processes communicate at each step.
   - Draft your solution in MPI pseudo-code. (5 marks)

   You might find MPI_Reduce/MPI_AllReduce useful here.

```
int MPI_Reduce(const void *sendbuf, void *recvbuf, int count,
MPI_Datatype datatype, MPI_Op op, int root, MPI_Comm comm)
```

Description: Combines the elements provided in the input buffer of each process in the group (`sendbuf`), using the operation `op`, and returns the combined value in the output buffer (`recvbuf`) of the process with rank `root`.

```
int MPI_Allreduce(const void *sendbuf, void *recvbuf, int count,
MPI_Datatype datatype, MPI_Op op, MPI_Comm comm)
```

Description: Same as MPI_Reduce except that the result appears in the receive buffer of all the group members.

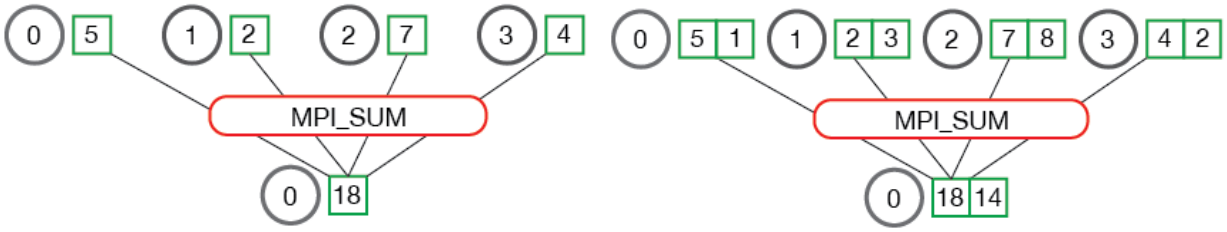


Figure 2: MPI_Reduce usage example

Answer:

Each P gets N/P elements of the array (scatter) and finds the maximum for those elements.

Use MPI_AllReduce to find the maximum.

Find the TOP10 elements for each chunk of data and gather result at root.

e. Use the distributed machine from point c. Assume that there are no collective operations for reduction implemented in MPI (no MPI_Reduce/MPI_AllReduce). Sketch tasks dependency graph for solving TOP10 problem in MPI. Note the following:
   - Number of cores P is large, and the master-slave approach might not scale.
   - Attempt to optimize your approach for solving TOP10 problem.
   - Briefly explain what each task entails.
   - Does the type of parallelism change compared to point b.? (4 marks)

Answer:

Use a binary tree to find the maximum of the maximum for different chunks of data. Find the local maximum and TOP10 elements at the same time (not different stages).

f. Use the distributed machine from point c. Assume that there are no collective operations for reduction implemented in MPI (no MPI_Reduce/MPI_AllReduce). Optimize your implementation for solving TOP10 problem in MPI.
   - Sketch how the communication is implemented. Sketch the new pseudo-code.
   - Discuss the communication efficiency for your implementation.
   - Explain why your solution is optimized.
   - You may use MPI_Gather / MPI_Scatter.
   - Mention any other assumptions you have made. (7 marks)

Answer: N/A

g. Use the distributed machine from point c. What type of interconnection network would you use to connect your P processors to make your implementation in MPI from point f. faster? Justify your choice with an example of how your chosen interconnect would improve the performance (4 marks).

Answer:

Any connection that can embed a tree: Hypercube or tree.

Q4. [15 marks] Memory model.

a. Explain two performance issues you might observe when using shared and distributed memory architectures, respectively? (4 marks)

Answer:

Memory consistency, cache coherence or communication cost.

b. Is there a problem related to cache coherence in distributed memory systems? Explain you answer (4 marks).

Answer:

There are two valid answers, accepted only if justified properly (no points only for saying Yes/No).

No problem: No shared memory. Updating the cache is done based on explicit messages.

There is a problem: In distributed systems, nothing about cache or memory is shared, but the file system might be shared. If so, the same problems arise by caching files in main memory. The network substitutes for the system bus. (students might not get this issue because it was not mentioned din class).

c. Assume the following program segments are executed on three processors of a multiprocessor machine. Initially before execution, all variables are equal to 0.

Table 1

P1	P2	P3
(1) A = 1	(2) U = A	(4) V = B
	(3) B = 1	(5) W = A

i. There are totally 8 possible final states of U, V, W as listed in Table 2 (below). Indicate whether it is valid or invalid under a **sequential consistency model** after executing the statements from Table 1 (above). (4 marks)

Table 2

Case	U final state	V final state	W final state	Valid (Yes/No)
1	0	0	0	Y
2	0	0	1	Y
3	0	1	0	Y
4	0	1	1	Y
5	1	0	0	Y
6	1	0	1	Y
7	1	1	0	N
8	1	1	1	Y

ii. Choose one of the final states that you think is invalid under sequential consistency from Table 2. and show that it is invalid. Refer to the statements in Table 1 using their identifier.

Answer:
1,1,0 is not possible.
If V is 1, P2.1. and P2.2. must happen before P3.1. But if we want U=1, P1.1 must happen before P2.1. Hence W will be 1 because it is executed after P3.1.

iii. Show what final states are valid under relaxed consistency models after executing the statements from Table 1.

- TSO (1mark):

Case	U final state	V final state	W final state	Valid (Yes/No)
1	0	0	0	Y
2	0	0	1	Y
3	0	1	0	Y
4	0	1	1	Y
5	1	0	0	Y
6	1	0	1	Y
7	1	1	0	N
8	1	1	1	Y

reordering of statements is not possible in TSO.

- PC (1 mark):

Case	U final state	V final state	W final state	Valid (Yes/No)
1	0	0	0	Y
2	0	0	1	Y
3	0	1	0	Y
4	0	1	1	Y
5	1	0	0	Y
6	1	0	1	Y
7	1	1	0	Y
8	1	1	1	Y

Case 7 is possible now.

- PSO (1 mark):

Case	U final state	V final state	W final state	Valid (Yes/No)
1	0	0	0	Y
2	0	0	1	Y
3	0	1	0	Y
4	0	1	1	Y
5	1	0	0	Y
6	1	0	1	Y
7	1	1	0	N
8	1	1	1	Y

Reordering of statements is impossible in PSO. Reordering (4) and (5) would not be allowed because that entails reordering R->R as each statement is a store and a load.

END OF PAPER