NATIONAL UNIVERSITY OF SINGAPORE
SCHOOL OF COMPUTING

FINAL ASSESSMENT PAPER - ANSWERS

AY2020/21 Semester 1

CS3210 — PARALLEL COMPUTING

November 2020 Time Allowed: 2 hours

INSTRUCTIONS TO STUDENTS

1. Submit by **Mon, 30 Nov, 11:00 am** on **Luminus Files->Student Submissions -> Final Exam Submission** a PDF file:
   - Name your PDF file using your student number: A1234567Z.pdf
   - The format of the answers has to **strictly** follow the format provided in the question.PDF file**. For example, if Question 7 (requirement and answer space) is on page 6 in the questions PDF file, it should be on page 6 in the answers PDF file submitted by you.**
   - In case you scan your answers, make sure that they can be read once included in the PDF. We will not grade answers that cannot be read.
   - When multiple submissions are made, we will grade only your most recent submission.
2. To solve the questions. You can either:
   - write your answers on the PDF directly,
   - print out, hand-write your answers, and scan into a PDF
   - hand-write your answers on clean paper, and scan into a PDF
   - write your answers in the DOCX, and save as PDF
   - have any other approach that would result in a PDF file
   - if you scan your answers, you may use Drive application on Android to add a new document, and choose Scan.
3. This assessment paper contains THREE (3) parts and comprises FOURTEEN (14) pages. The total number of marks in the paper is 70 and the paper counts towards 35% of your final grade.
4. Students are required to answer **ALL** questions within the space in this file.
5. Failing to submit your answers to Luminus folder by 11:00am on Mon, 30 Nov means you failed your exam (0 marks will be awarded). No late submission is allowed.
6. Address any questions you might have in Zoom chat or call 65168850.
7. Write your student number below. Do not write your name.

STUDENT NO: _____

Part A: General questions [26 marks].

1. [2 marks] True or false! Two threads created by different processes share memory space. Justify your answer.

Answer:

[2 marks] False – different processes have different memory space

[1 mark] if only explanation or justification is correct

2. [2 marks] True or false! For large number of nodes, a CCC (cube-connected cycles) interconnection network is easier to physically construct than a hypercube. Justify your answer.

Answer:

[2 marks] True – the degree of a node in CCC is always constant (3); hence the hardware load on a node does not increase with the increase in size of the network.

[1 mark] if only explanation or justification is correct

3. [2 marks] We used our lab machines for OpenMP and MPI programming. Can these systems be categorized as a cloud or a data center? Justify your answer.

Answer:

[2 marks] None – the system is just a collection of lab machines (computers) connected and accessible over a network. No special data center setup is in place.

[1 mark] Data center – if justified properly.

4. [2 marks] We used the SoC Computer Cluster machines for CUDA programming. Can these systems be categorized as a cloud or a data center? Justify your answer.

Answer:

[2 marks] Data center, with justification

5. We discussed the matrix multiplication problem in many contexts in this module. Assume you are required to implement an OpenMP program that does matrix multiplication for two large square matrices. The matrices are stored in two files and they do not fit in memory.
   Assume that computers (machines) with similar performance and configuration are used for all questions below.

   a. [4 marks] If you try to run a classic matrix multiplication implementation in OpenMP (like the one we used in Lab 2), you will experience some issues. Briefly mention and explain two issues that the program will face due to the large size of the matrices.

Answer:
[2 marks] With the invocation of many threads and each thread would need parts of matrix A and B, which will not fit in memory. This may result in excessive hard-disk thrashing as there will be multiple page-faults when the memory is unable to hold the huge data.
This is the main issue, should be mentioned by all answers.

Second issue can be related to page faults in some way, or cache misses for matrix B, etc.
[2 marks] Assuming that the processing elements store data in a row-major order, this process will result in a lot of cache misses when traversing matrix B as it may be traversing different cache lines for each element (or few elements). Since the matrix is relatively large, it increases the likelihood that the consecutive column element in matrix B is not on the cache line which results in poorer performance due to cache write policies.

   b. [4 marks] A solution proposed to speed up the matrix multiplication implementation is to use MPI instead of OpenMP, and distribute the run on multiple machines. How would you modify your OpenMP program into an MPI program? Give an overview of two essential changes in your implementation.
   Note: Syntax or API changes are expected, and they should not be mentioned as essential changes.

Answer:

In MPI, use a master-worker approach: the master reads and sends chunks of data to the workers.
[2 marks] **communication** instead of accessing shared memory
[2 marks] divide the matrices into **chunks that fit in memory** for each worker

c. [4 marks] Which of the two matrix multiplication implementations (OpenMP and MPI) do you expect to have better performance (i.e. smaller execution time)? Explain your choice using two reasons. State any assumptions made.
Note: Repeating your explanation from point a. will not receive marks. You need to compare the two implementations.

Answer:

MPI will get shorter execution time, if the tasks assigned to a worker work with data that fits in memory.
Any of the following points would get 2 marks:
[2 marks] Communication overhead would be smaller than page faults overhead.

[2 marks] Tasks in MPI are independent – no synchronization is needed.

[2 marks] In MPI the processor can exploit code (data) locality, while in OpenMP the computation is interrupted by the IO.

[2 marks] OpenMP performance is limited by the performance of the shared-memory machine used. In MPI, cores can be added to solve the problem faster.

6. In the **sequential** pseudo-code shown in Table 1, $\mathbf{V_x}$ represents one dimensional arrays of size **N**, $\mathbf{M_x}$ represents two dimensional arrays of size **N x N**.

Line#	Pseudo-code
1	`for (int i = 0; i<N; i++)`
2	`    for (int j = 0; j<N; j++)`
3	`        M3[j][i] = M1[i][j] * a + V1[j];`
11	`for (int i = 0; i<N; i++)`
12	`    for (int j = i; j<N; j++)`
13	`         M2[i][j] = M2[i-1][j] + M1[i][j];`
21	`for (int i = 0; i<N; i++)`
22	`    V2[i] = 0;`
23	`    for (int j = 0; j<N; j++)`
24	`        V2[i] = V2[i] + V1[j] * M3[i][j];`

Table 1: Pseudo-code for question 6

Loop level parallelization can be applied differently in the nested loops. For example, in a shared address space, loop (1-2-3) can be parallelized in two ways:

i. Parallelize only line 1, i.e. iterations with different i values are executed in parallel.
ii. Parallelize both lines 1 and 2, i.e. each pair of (i, j) value are executed in parallel.

a. [2 marks] Which approach i. or ii. would give better performance (i.e. smaller execution time) for the code in Table 1 and why? Give one reason to justify your choice.

Answer:

i. is better because the task granularity is more likely to be of the right size. In ii., the task size is too small.

b. [2 marks] For loop at lines 11-12-13 in Table 1, identify which parallelization approach(es) (i. and ii.) **cannot** be applied and briefly explain why.

Answer:

None of the approaches can be used because there is a data dependency at line 13 with the previous iteration of the loop.

c. [2 marks] For loop at lines 21-22-23-24 in Table 1, identify which parallelization approach(es) (i. and ii.) **cannot** be applied and briefly explain why.

Answer:
ii. cannot be used because all iterations at line 23 modify V2[i] at the same time (data race).

Part B: Performance [20 marks]

7. [4 marks] Assume that you have 10 cores that you can use to solve a problem in parallel. 95% of your code is parallelizable. Can you get a speedup of 5? If so, how many cores are needed?

Answer:

[4 marks]
S=5
f=0.05
5=1/ (0.05+0.95/x) => x=0.95/(1/5 -0.05)= 0.95/0.15= 6.33 -> 7 cores are enough

8. [2 marks] Is it possible to solve a problem of variable size (increasing size) to obtain a fixed speedup, provided that an unlimited number of processing units is available? Explain your answer.

Answer:

Yes, it is **possible** if we apply the case of Gustafson's Law. If the sequential fraction f decreases with the increase in problem size, a required speedup can be achieved.

9. A CPU "M1" has a clock rate of 3.3 GHz and the following cycles per instruction for instruction types A through D. A program P1 has a total number of instructions of 20,000 and the mix of instructions shown in Table 2:

Instruction Type	**A**	**B**	**C**	**D**
CPI for M1	5	3	2	1
Mix of instructions for P1 (%)	20	30	15	35

Table 2: Mix of instructions for question 9

a. [2 marks] Calculate the average CPI of P1 on CPU M1.

Answer:

CPI_{ave} = (5*20+3*30+2*15+1*35)/100 = 2.55

b. [2 marks] Calculate the user CPU time of P1 on M1.

T_{ex} = N* CPI * (1/clock_rate) = 20000 * 2.55 * (1/(3.3 *10^9)) = 0.0155ms

10. Answer points a.-c. based on the description shown below.
You are required to analyze the performance of four implementation used for summing up an array. The pseudo-code and execution time for four versions are shown below in the next five tables:

Line#	Pseudo-code **sum_sequential**
1	`long long sum_ sequential (long long *a, int n) {`
2	`    int i;`
3	`    long long sum = 0;`
4	`    for (i = 0; i < n; i ++) {`
5	`        sum += a[i];`
6	`    }`
7	`    return sum;`
8	`}`
9	

Table 3: Psuedo-code for sum_sequential

Line#	Pseudo-code **sum_omp_critical**
11	`long long sum_omp_critical (long long *a, int n) {`
12	`    int i;`
13	`    long long sum = 0;`
14	`    #pragma omp parallel for shared (a, sum) private (i) \`
15	`       schedule (static, 1)    //each thread gets 1 iteration`
16	`    for (i = 0; i < n; i ++) {`
17	`        #pragma omp critical`
18	`        sum += a[i];`
19	`    }`
20	`    return sum;`
	`}`

Table 4: Psuedo-code for sum_omp_critical

Line#	Pseudo-code **sum_omp_atomic**
21	`long long sum_omp_atomic (long long *a, int n) {`
22	`    int i;`
23	`    long long sum = 0;`
24	`    #pragma omp parallel for shared (a, sum) private (i) \`
25	`                schedule (static, 1)`
26	`    for (i = 0; i < n; i ++) {`
27	`        __atomic_fetch_add (&sum, a[i], __ATOMIC_RELAXED);`
28	`    }`
29	`    return sum;`
30	`}`

Table 5: Psuedo-code for sum_omp_atomic

*GCC `__atomic_*` intrinsics compile to x86 atomic instructions (vs. locks for #pragma)

*`__ATOMIC_RELAXED` implies no inter-thread ordering constraints (relaxed consistency).

Line#	Pseudo-code **sum_omp_reduction**
31	`long long sum_omp_reduction (long long *a, int n) {`
32	`    int i;`
33	`    long long sum = 0;`
34	`    #pragma omp parallel for shared (a) private (i) \`
35	`          schedule (static, 1) reduction (+:sum)`
36	`    for (i = 0; i < n; i ++) {`
37	`        sum += a[i];`
38	`    }`
39	`    return sum;`
40	`}`

Table 6: Psuedo-code for sum_omp_reduction

Function	**Execution time (ms)**
sum_sequential	3.7
sum_omp_critical	48.2
sum_omp_atomic	26.7
sum_omp_reduction	1.3

Table 7: Wall-clock time for the functions in tables 3-6 (n = 2^{20}, default (-O0) compiler optimizations)

a. [3 marks] Compute the speedup for each of the implementations shown in tables 3-6 based on the measurements shown in table 7.

Answer:

S_critical = 3.7/48.2 = 0.076
S_atomic = 3.7/26.7 = 0.138
S_reduction = 3.7 / 1.3 = 2.84

b. [2 marks] Explain the possible reason for the differences in performance among the three OpenMP implementations shown in tables 3-7.

Answer:
[1 mark] Omp_critical vs. omp_atomic: atomic operations are faster than mutex operations (locks)
[1 mark] Omp_reduction vs (omp_critical and omp_atomic): independent additions are done in parallel and don't need a critical section in the omp_reduction (possibly using a binary tree), while the additions are done sequentially in the other implementations due to the atomic and lock operations.

c. [4 marks] Mention two other performance metrics (except for execution time) that you would use to further understand the performance. Explain how to use them in your analysis of the implementations shown in tables 3-6.

Answer:
Any of the following would get 2 marks:
[2 marks] – investigate the time spent waiting; the only waiting happens for lock contention
[2 marks] – check IPC – if low, it is likely that the atomic instructions are contended
[2 marks] – check cache misses or page faults
[2 marks] – check the call stack using perf

d. [1 mark] All four codes were run on the same machine. How would you explain the differences in execution time between `sum_sequential` and `sum_omp_reduction`, as shown in Table 7?

Answer:
In sum_omp_reduction the additions are done in parallel, while in sum_sequential the additions are done sequentially.

Part C: Atmospheric model [24 marks]

Answer questions 11 – 16 based on the description of the **atmospheric model.**

An **atmospheric model** simulates three atmospheric processes (wind, clouds, and precipitation) that influence weather or climate. This model may be used to study the evolution of tornadoes, to predict tomorrow's weather, or to study the impact on climate of an increase in the concentrations of atmospheric carbon dioxide. Like many numerical models of physical processes, an atmospheric model solves a set of partial differential equations, in this case describing the fluid dynamics of the atmosphere. The behavior of these equations on a continuous space is approximated by their behavior on a finite set of regularly spaced points in that space.

Typically, these points are located on a rectangular latitude-longitude grid of size N*2N (2D array of size N*2N) and each grid point contains an array of values, representing quantities such as pressure, temperature, wind velocity, and humidity. To compute the values for each grid point, the values of eight surrounding points are needed as shown in Figure 1. The problem is computational-intensive (the bottleneck is computation). You may consider that the atmospheric processes (wind, clouds, and precipitation) are independent of each other and they have the same amount of computation.

This atmospheric model is implemented and solved on a distributed memory machine with P processing units. Each processing unit has its own memory unit (distributed memory machine). Assume P<<N.

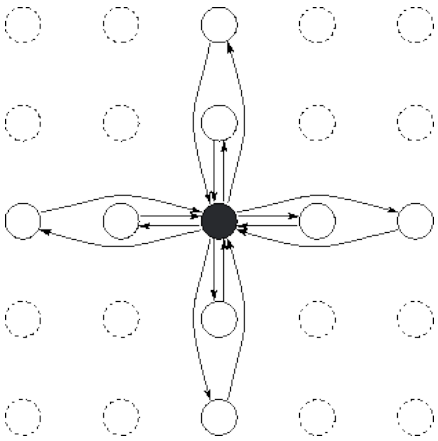

Figure 1: Information needed for computing the values for a grid point in each atmospheric process.

11. [4 marks] Explain how to distribute the computation of the **atmospheric model** (data and tasks) on processing units. Your goal is to achieve good execution time with good utilization of the resources.

Answer:

To achieve good utilization, need to equally load the CPUs.
Need to explain what you distribute: data or tasks. Both are possible, but they would need to be explained in different ways.
-1 mark: if the student says task distribution and goes ahead and explains data distribution.
Q11-12 are marked together.

12. [2 marks] What type of data distribution are you using in question 11 to solve the **atmospheric model**? Use a diagram or explanation to illustrate your approach.

Answer:
2D distribution - Checkerboard.
1D distribution - needs to be shown and explained (blockwise)

-2 marks is explanation is not clear or it does not make sense when considered together with Q11.

13. [2 marks] If you could design a distributed memory machine to speed up the **atmospheric model**, what type of interconnect network would you use to connect your processing units? Justify your choice.

Answer:

If 2D distribution: need a 2D mesh (or similar)
If 1D distribution: need a 1D network – linear, ring…
-1 mark for mismatch in answers with Q11-12

14. [6 marks] Sketch your implementation in MPI pseudocode for the **atmospheric model**. Initially data is located at process rank 0 (on disk, as data cannot fit in the memory of a single node), and results should be located at process rank 0 at the end of the execution. In your pseudo-code, show at least the following:
    a. How the data and/or task distribution is done.
    b. What each participant process would do.
    c. How the processes communicate at each step.

Answer:

Best would be to use a master/worker for data parallelism.
[1 mark] Master sends chunks and receives chunks.

Worker:

- [1 mark] Receives chunk
- [1 mark] Computes in a loop the atmospheric processes for a given number of steps (or time)
    - [2 marks] Needs to communicate at each step with the neighboring workers to get updated values for the borders (sendrecv would be best, no need to worry about synchronization)
- [1 mark] Sends chunk back to master after a given number of steps (or time)

15. [4 marks] Assume now that the distributed system that you are using has P processing units, but each 4 processing units share a memory unit (distributed shared-memory system). How would you change your approach to implement the **atmospheric model**? Explain.
    Note1: If you think that no change is needed, explain how your initial approach deals with the issues that come from using shared memory.
    Note2: You can use other programming paradigms you have studied in the module.

Answer:

Two issues should be mentioned:

- [2 marks] Memory for the 4P units – need to take care what each worker gets such that there are no page faults
- [2 marks] The 4 processing units should be utilized, even if memory is not enough:
    - Use OpenMP to speed up computation for a chunk sent on a 4P unit.

[1 mark] – for no change, and explaining that everything would be fine.

16. The accuracy of the results predicted by the **atmospheric model** can be increased using three methods (that can be combined):
    i. Reducing the grid size by increasing N.
    ii. Compute a localized model focusing on only one region on Earth.
    iii. Run multiple simulations by choosing different perturbations to focus on. Choose the most accurate prediction by comparing the results and choosing the one that has more similarity points with others.

    a. [3 marks] The atmospheric model is used for weather prediction in Australia (which has approx. 1.5% of the total surface of the Earth) and the computation must use the most recent available data and be completed 24 hours before the day starts.
    Explain the approach you would implement to improve the accuracy.

Answer:

[3 marks] Use ii. + iii. – stop iii. when you meet the time deadline.
i. cannot be used because we have a time deadline.
[2 marks] for only one approach used

    b. [3 marks] The atmospheric model is used for climate change prediction and the computation must be accurate.
    Explain the approach you would implement to improve the accuracy.

Answer:

[3 marks] Use i. + iii. – stop iii. when you meet an accuracy level.
i. cannot be used because we do climate modeling for the Earth.
[2 marks] for only one approach used

--END of PAPER--

BLANK PAGE