Module Code: XJCO322101

Module Title: Parallel Computation

School of Computing

Semester 2 2021/22

Examination Information

- There are **6** pages to this exam.
- There are **2 hours** to complete the exam.
- You are given an additional **15 minutes** to mitigate for technical issues.
- Answer **all 2** questions.
- The number in brackets **[ ]** indicates the marks available for each question or part question.
- You are reminded of the need for clear presentation in your answers.
- The total number of marks for this examination paper is **50**.

Question 1

(a) Answer either 'True' or 'False' for each of the following questions, giving a reason for each of your answers.

(i) Parallel code running on $p > 1$ processing units always runs faster than the equivalent serial algorithm.

[1 mark]

(ii) Parallel code running on p processing units cannot run more than p times faster than the optimal serial algorithm.

[1 mark]

(iii) Thread safety is not relevant to distributed memory systems.

[1 mark]

(iv) GPUs are well suited to data parallel problems.

[1 mark]

(v) It is possible to synchronise between all threads on a GPU.

[1 mark]

(b) The parallel execution time t_{p} for a parallel algorithm running on p processing units can be much less than the serial execution time t_{s} for an equivalent algorithm. The speed–up S is defined as the ratio $S = t_{\mathrm{s}}/t_{\mathrm{p}}$. Given a serial fraction of code f, Amdahl's law predicts a maximum speed–up, $S \leq \frac{1}{f+\frac{1-f}{p}}$, whereas the Gustafson–Barsis law predicts $S \leq p + f(1-p)$.

(i) For $f = 0.1$ and $p = 16$, calculate the maximum speed-up as predicted by Amdahl's law, and as predicted for the Gustafson-Barsis law. You can either use a calculator, or rearrange the equations into a simple rational. Which law predicts a higher $S^{\max}$, and why?

[2 marks]

(ii) What does Amdahl's law predict for $f = 0$? Similarly, what does the Gustafson–Barsis law predict for $f = 0$? Explain your answers.

[2 marks]

(iii) Suppose you are writing parallel code that sorts a large database of size n. Given that it is beyond your control to change the value of n, which of the above two laws, Amdahl or Gustafson–Barsis, is most relevant to this situation? Explain you answer.

[2 marks]

(c) The following serial code demonstrates a simple function in C that identifies where the largest value lies in a floating point array of size `N`, returning the corresponding array index.

```
1     int argmax( float *array, int N )
2     {
3         int i, maxIndex=0;
4         float maxValue = array[0];
5
6         for( i=1; i<N; i++ )
7               if( array[i] > maxValue ) {
8                 maxIndex = i;
9                 maxValue = array[i];
10              }
11
12        return maxIndex;
13    }
```

When called from a multi-threaded context, *i.e.* when multiple threads call `argmax` simultaneously for the same `array`, it is found that most times this function works as expected, but occasionally gives an answer that does not agree with the corresponding serial code.

(i) Identify one reason why the function `argmax` may fail when called simultaneously by multiple threads, using the line numbers as given to explain you answer.

[2 marks]

(ii) It is suggested that enclosing some lines in a `#pragma omp critical` region may solve the problem. Which lines would you include in the critical region? Again, use the line numbers as given.

[2 marks]

(iii) However, profiling runs show that the parallel code with the critical region is very slow. Why is this?

[2 marks]

(d) The atomic compare-and-exchange operator compares the value of a memory location to a given value, changing it to a second given value only if the result of this comparison was true. For instance, for integer variables in OpenCL, the atomic compare-and-exchange is provided as `atomic_cmpxchg(int *p,int cmp,int val)`, and sets `*p=val` only if `*p==cmp`. It always returns the original value of `*p`.

(i) Give two advantages, and two disadvantages, of atomic operations compared to the use of critical regions or locks.

[4 marks]

(ii) Consider the following snippet of code that uses `atomic_cmpxchg`. The pointer declared as `int *p` points to an integer variable that can be accessed by all threads.

```
1     while(true) {
2        int value = *p;
3        if( atomic_exchg(*p,value,value*2)==value ) break;
4     }
```

What function does this code perform? Explain your answer making explicit reference to line numbers where appropriate, highlighting where the actions of other threads may cause problems.

[4 marks]

[Question 1 Total: 25 marks]

Question 2

Many parallel algorithms require, at some stage, variables distributed across multiple processing units to be reduced to a single value by a binary operation. This reduced value must then be made accessible to all processing units. For instance, in a series of vector operations, it may happen that the result of the scalar product of two vectors must then be made available to all processing units for the next stage in the calculations.

(a) For shared memory systems, where the processing units are threads, all threads can read the memory location containing the reduced value. However, they should not begin subsequent calculations until the reduction calculation has been completed.

(i) For a GPU, suppose the reduction had been completed by threads within a single work group. Why is it beneficial to use local, rather than global, memory for intermediate calculations in this situation?

[1 mark]

(ii) Still for a GPU, how would you ensure the result of the reduction performed by a single work group, in local memory, has been completed, and can be read by all threads for the subsequent calculations? Explain your answer.

[4 marks]

(b) For distributed memory systems, where processing units are processes, the issue becomes communicating the result of the reduction to all processes. Suppose that, after the reduction, the reduced value is known only to one process, *e.g.* rank 0 for MPI.

(i) What form of collective communication should be used to send the reduced value to all processes? You do not need to give the actual MPI function name, but may do so if you like.

[1 mark]

(ii) Someone suggests using point-to-point instead of collective communication, and you rightly point out that this will likely be slower than using collective communication. Justify this claim by estimating how the communication time t_{comm} varies with the total number of processes `numProcs` for both methods. You should assume that the collective communication uses a binary tree.

[4 marks]

(iii) Given barriers are not used in the binary tree, how might the necessary synchronisation be achieved?

[1 mark]

(iv) In fact, MPI already provides a function `MPI_Allreduce` that both reduces, and distributes the final answer to all processes. One possible implementation is essentially a combination of binary trees. An example is given in Fig. 1 for `numProcs=4`. Redraw Fig. 1 for the case `numProcs=2`, for which there will be 2 levels rather than 3, and therefore 4 nodes in total.

[2 marks]

(v) How many communications are there in total?

[1 mark]

(vi) Returning to Fig. 1, note that in the final row of communications, some processes send two partial sums whereas others send none. How would you alter this final exchange of partial sums to make the communication better balanced, *i.e.* so processes send at most one partial sum? Use the given rank numbers in your answer.

[2 marks]

(c) Notice that Fig. 1 is a task graph. Assume that each task (node) corresponds to the same amount of time, including those on the top and bottom rows.

(i) What is the work and span of the task graph given in Fig. 1? What is the maximum performance as predicted by the work-span model?

[3 marks]

(ii) Suppose there are $p = 2^m$ processes. What is the work, span, and prediction of the work-span model now, for arbitrary m?

[3 marks]

(iii) It has been assumed that each task takes the same time to execute. Suppose each task now takes a different, but known, time to execute. Describe in general terms how you would modify the definition of work and span, and the prediction of the work–span model, for this situation. You do not need to derive expressions or perform actual calculations, but should explain your answer.

[3 marks]

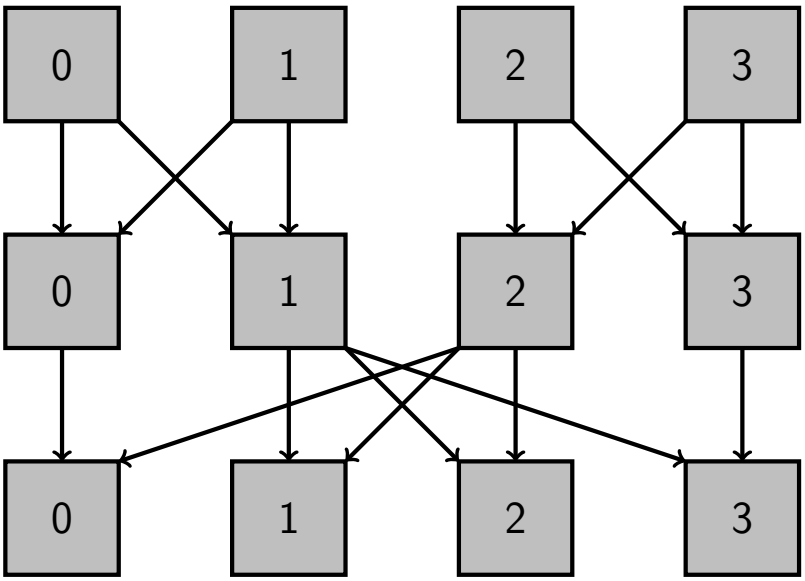

Figure 1: Figure for questions Q2(b) and Q2(c). Each square represents a binary operation and the numbers inside the squares correspond to the rank.

[Question 2 Total: 25 marks]

[Grand Total: 50 marks]

End