

THE AUSTRALIAN NATIONAL UNIVERSITY
Second Semester Examination – October 2001

COMP4300
Parallel Systems

Study Period: 15 minutes

Time Allowed: 2 hours

Permitted Materials: Calculator

Exam questions total 40 marks.

Answer ALL parts of ALL questions.

Point form is recommended for ‘descriptive’ questions.

*Clarity and conciseness in answers will be highly valued;
marks may be lost for supplying irrelevant information in
answers.*

Question 1 [2 marks]

What, Where and Why! Using only a few sentences, and in layman's terms, explain what parallel systems are, where they might be used, and why they are used.

Question 2 [1 mark]

What is meant by *superlinear speedup*? Give one reason why a parallel program may show *superlinear speedup*.

Question 3 [2 marks]

MPI-2 permits both *active* and *passive* remote memory access (RMA). What is RMA, and how do the two protocols differ?

Question 4 [1 mark]

What are SIMD parallel computers? Give an example of an operation that could be performed (efficiently) on a SIMD parallel computer. (You may express your example either as a mathematical operation or using a few lines of code).

Question 5 [1 mark]

What are MPI communicators and when are multiple different communicators useful?

Question 6 [1 mark]

In message passing what is meant by *store+forward* and *cut-through* routing?

Question 7 [2 marks]

What do you understand by coarse and fine grain parallelism? Give two factors that limit the scalability of parallel programs.

Question 8 [1 mark]

Outline one class of problems for which a pipeline parallelization approach may be applicable.

Question 9 [1 mark]

What is deadlock? Illustrate your answer with a potential deadlock situation.

Question 10 [2 marks]

Using send and receive message passing calls, show how you would construct a barrier. How does the communication cost of your barrier scale with the number of processors calling the barrier?

Question 11 [7 marks]

You are attempting to parallelize your code on a distributed memory parallel computer using MPI. You are taking an incremental approach, slowly introducing parallel segments into your code. So far you have successfully implemented the basic `MPI_Init` and `MPI_Finalize` calls and sorted out input processing etc such that the code now runs N independent calculations on N processors, with each processor giving the same (correct) result. In other words you have fully replicated all memory requirements and computational operations on all processors.

You have identified the time dominant part of your code to be the multiplication of two matrices (**A** and **B**), adding the result to a third matrix (**C**). This is illustrated by the following equation:

$$c_{i,j} = c_{i,j} + \sum_{k=0}^{l-1} a_{i,k} b_{k,j} \quad (0 \leq i < n, 0 \leq j < m)$$

with the relevant part of the source code given below:

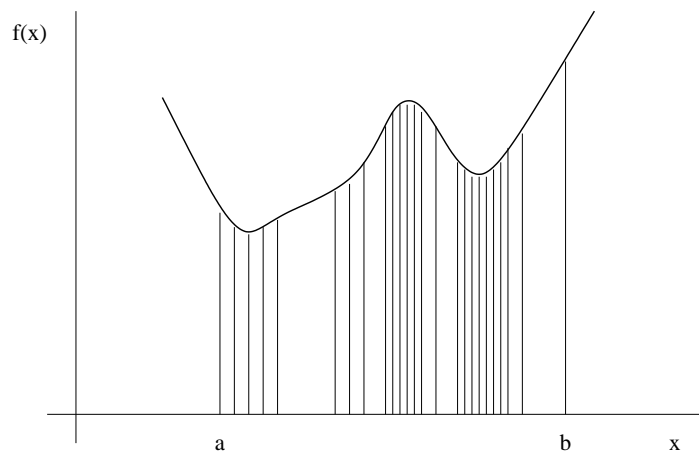
```
/* Point-1 */
for (i=0; i< n; i++){
    for (j = 0; j < m; j++){
        for (k = 0; k < l; k++){
            c[i][j] = c[i][j] + a[i][k] * b[k][j];
        }
    }
}
/* Point-2 */
```

- (a) Indicate how you would parallelize the above piece of code using message passing. Your parallelization scheme should ensure that all processors have identical copies of the matrices **A**, **B** and **C** after `Point-2` (so that each processor can continue execution and obtain the same, correct, result). Note that at `Point-1` all processors currently have identical copies of all matrices. You are free to reorder the three loops providing that the overall result is correct!
- (b) Analyse the computation and communication costs of your parallel algorithm.
- (c) Discuss briefly how you would implement a fully distributed version of the above matrix product. Assume **A**, **B** and **C** are square with $l = m = n$, and that each matrix is distributed over n^2 processors, such that there is just one element of each matrix on each processor.

Question 12 [6 marks]

- (a) Explain the difference between a process and a thread. What advantage do threads offer for parallel programming?
- (b) What are the roles of “mutex” and “condition” variables in the pthreads library?
- (c) Give one design feature of the Cray MultiThreaded Architecture (MTA) that makes it especially well suited to multithreaded programs.
- (d) Many shared memory machines are NUMA based. Explain what is meant by NUMA and how the Global Array programming model has been designed with NUMA in mind.

Question 13 [6 marks]



In the above diagram the area between $x = a$ and $x = b$ can be evaluated numerically by dividing the region into a number of trapezoids, calculating the area of each trapezoid separately, and then summing together the results. Such “numerical quadrature” schemes are well suited to parallelization via a domain decomposition approach.

- (a) Explain what is meant by domain decomposition.
- (b) In the above diagram an adaptive quadrature has been used. This produces more trapezoids in regions where the function ($f(x)$) changes quickly. Why is this likely to give rise to a load balancing problem?
- (c) What is the difference between static and dynamic load balancing? Outline one scheme for dynamic load balancing.
- (d) If a parallel task can spawn new parallel tasks what complication does this cause for dynamic load balancing schemes? Briefly outline how this problem may be addressed?

Question 14 [7 marks]

A ring, square 2-D torus, binary tree, and hypercube are 4 topologies that have been used to build networks for parallel computers.

- (a) Illustrate each of the above network topologies for some moderate number of processors.
- (b) The diameter is one criteria that is commonly used to compare different network topologies. What do you understand the diameter of a network to be? Give and define two other criteria that are often used to assess networks.
- (c) Compare the 4 topologies given above on the basis of the 3 criteria given in part (b) of this question.
- (d) The Compaq SC has a fat tree network. How does a fat tree differ from a binary tree?

