UNIVERSITY OF LONDON
IMPERIAL COLLEGE OF SCIENCE, TECHNOLOGY AND MEDICINE

EXAMINATIONS 2000

MEng Honours Degree in Information Systems Engineering Part IV
MEng Honours Degrees in Computing Part IV
MSc in Advanced Computing
for Internal Students of the Imperial College of Science, Technology and Medicine

This paper is also taken for the relevant examinations for the Associateship of the City and Guilds of London Institute

PAPER C429=I4.10

PARALLEL ALGORITHMS

Thursday 18 May 2000, 10:00
Duration: 120 minutes

Answer THREE questions

Paper contains 4 questions

1 a Define the *efficiency*, E_p, of an algorithm executing on a parallel computer comprising p processors. What is its cost, C_p, and what is meant by saying that an algorithm is *cost-optimal*?

b i) What is meant by the term *scalability*?

ii) The *overhead*, $O_p(W) = C_p - W$ is the amount of additional computation not required in the best serial algorithm, for a problem with (best) serial run time W. Show that $W = \frac{E_p}{1-E_p} O_p(W)$.

iii) Suppose that $I(p, E)$ is a solution for W in the equation of part ii). In what sense does $I(p, E)$ measure the scalability of the system?

c A certain algorithm has complexity $\Theta(n^2)$, i.e. has $W = kn^2$ for some constant k. It is executed on p processors, where p is a power of 2, such that the only additional run-time is due to communication and the parallel computation time is W/p. The interconnection network is a hypercube.

i) Show that an all-to-all broadcast can be achieved by every node simultaneously collecting all the data accumulated at each of its directly connected neighbours in turn; i.e. there is one step per dimension of the hypercube and the amount of data transmitted at each step doubles. Hence show that the asymptotic per-word communication time is $\Theta(p)$.

ii) The communication in the algorithm comprises four all-to-all broadcasts of messages of length $\sqrt{W/p}$ amongst the p processors. Show that the algorithm is scalable with isoefficiency function of order p^2. (You may neglect transmission start-up and per-hop times.)

2 a Describe an algorithm to compute the matrix-product of two $n \times n$ matrices using *block checkerboarding* data partitioning and answer the following questions in relation to your algorithm.

i) What is the parallel *computation* time of your algorithm on p processors? (Assume all arithmetic operations take unit time. An asymptotic result is acceptable.)

ii) What is its *storage* requirement?

iii) How much data is transmitted in each communication?

b Suggest a way in which the storage requirement of your algorithm for part a could be reduced by a factor of approximately $\sqrt{p}$. (You may assume p is a perfect square.)

c Suggest a way in which the computation time of a matrix-product algorithm could be minimised by using n^3 processors, for example using the DNS algorithm. Show that the parallel run time cannot be better than $\Theta(\log_2 n)$, even on a PRAM.

3 a Outline the method of *finite differencing* for solving partial differential equations and contrast it with the *finite elements* method. Identify potential sources of parallelism in each case.

b Consider the first-order differential equation for the function $u(x)$

$$u\frac{\mathrm{d}u}{\mathrm{d}x} - k = 0$$

over the interval $0 \le x < 1$, where k is a constant. The interval is partitioned into n equal sized finite elements and we define the *shape functions* $\phi_0, \phi_1, \ldots, \phi_{n-1}$ by:

$$\phi_i(x) = \begin{cases} 1 & \text{for } i/n \le x < (i+1)/n \\ 0 & \text{otherwise} \end{cases}$$

for $0 \le i \le n-1$. (I.e, ϕ_i is 1 on the $(i+1)^{\text{th}}$ element and 0 elsewhere.)

i) Write down an approximation, $v(x)$, for the solution for $u(x)$ as a linear mixture of all the n shape functions.

ii) Determine a recurrence formula for the coefficients of this approximation by setting to zero the inner products of each shape function and the residual error obtained by substituting v for u in the left hand side of the differential equation.

Note that the inner product of functions $f(x)$ and $g(x)$ is $\int_0^1 f(x)g(x)\mathrm{d}x$.

iii) Deduce the approximation $v(x) = \sqrt{\frac{2ki}{n} + \alpha_0^2}$

for $i \le nx < (i+1)$, where α_0 is a constant.

4 a Compare and contrast *static* and *dynamic* parallel algorithms and their implementations on a multiprocessor computer. Illustrate your answer with examples and indicate appropriate interconnection networks for each.

b A state graph generation algorithm for the analysis of a communication protocol in a large network has been written such that all the processors in a parallel computer hold the data for the same number of states in their local memories *and* these states are randomly distributed across the memories.

i) Outline a simple parallel algorithm that explores the whole graph to detect illegal states, i.e. errors in the protocol. It should begin in the processor holding the root state, the other processors being idle. Show each processor's computation and communication.

ii) Can the communication overhead be reduced, and if so, how?

iii) Suppose now that an exploration is performed on a shared memory multiprocessor. Idle processors request work through an additional controller, which allocates half the largest work unit of the currently active processors. For example, if three processors have 4, 8 and 4 units before the allocation, all four will have 4 units afterwards. Assuming there are 2^n states, 2^m processors ($n >> m$) and that the graph is a binary tree, estimate a lower bound for the number of times the graph is split. When might it be split more often?

End of paper