UNIVERSITY OF LONDON

IMPERIAL COLLEGE OF SCIENCE, TECHNOLOGY AND MEDICINE

EXAMINATIONS 1999

MEng Honours Degrees in Computing Part IV
MEng Honours Degree in Information Systems Engineering Part IV
MSci Honours Degree in Mathematics and Computer Science Part IV
MSc Degree 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
Diploma of Membership of Imperial College
Associateship of the City and Guilds of London Institute
Associateship of the Royal College of Science

PAPER 4.29 / I 4.10

PARALLEL ALGORITHMS

Thursday, April 29th 1999, 10.00 – 12.00

Answer THREE questions

For admin. only:
paper contains 4 questions

1 Items of size w words are searched for in an unordered database using a simple static parallel algorithm in which the database is partitioned into p equal sized parts and each of p processors searches exactly one part allocated to it.

a Compare and contrast the algorithm in the cases of (i) exhaustive search; (ii) search for the first matching item. How do you ensure the latter is faster in your parallel algorithm? What messages are generated by each processor in each case?

b How long, on average, does each processor spend searching in each case (i) and (ii), given a database of N items with n matching items uniformly spread across the p partitions? (Assume the time taken between successive comparisons is the constant k. in each processor.)

c In case (ii), assuming blocking I/O, show how to calculate the time spent waiting for communication by each processor. Hence calculate the speed-up and efficiency of the algorithm. (Neglect initialisation and assume that the average network latency is $L(p)$ seconds per word transmitted, including any contention for the network.)

d Assuming there is only one match in the entire database, under what conditions is the algorithm *cost-optimal*? What is its isoefficiency function when it is cost-optimal?

The four parts carry, respectively, 25%, 20%, 25% *and* 30% *of the marks.*

2 a Describe the *Parallel Random Access Machine* (PRAM) idealised model of computation.

b Consider n linear equations in n unknown variables $x_1, ..., x_n$, written in matrix form as $\boldsymbol{A}.\boldsymbol{x} = \boldsymbol{b}$ for matrix $\boldsymbol{A}$ and vector $\boldsymbol{b}$. Compare and contrast parallel algorithms to solve these equations on p processors, where p divides n, using

i) Gaussian elimination using block striped partitioning of $\boldsymbol{A}$;

ii) Jacobi iteration.

In each case, derive an expression for the parallel run time of your algorithm executing on a PRAM, assuming you have an algorithm of complexity $\theta(n^2/p)$ for a matrix-vector dot-product.

c Now suppose the matrix A is tri-diagonal, i.e. its only non-zero elements lie either on the diagonal or one position right or left of the diagonal. Devise a more efficient algorithm for Jacobi iteration, assuming n is an integer multiple of $3p$, indicating carefully where communication occurs.

The three parts carry, respectively, 25%, 50% *and* 25% *of the marks.*

Turn over .....

3 a Give a recursive definition of a *hypercube network with d dimensions.*

i) Show how to embed a ring-network of 2^d nodes into a d-dimensional hypercube such that nodes that are adjacent in the ring are directly connected in the hypercube.

ii) Show how to embed a $2^d \times 2^e$ wraparound mesh into a (d+e)-dimensional hypercube similarly.

b Describe an algorithm for the inner (dot) product of an $n\times n$ matrix $\boldsymbol{M}$ and n-component vector $\boldsymbol{v}$ on a network of p processors using a *block-checkerboarding* data partitioning scheme. Derive an expression for its asymptotic parallel execution time when executed on a hypercube network. You may neglect start-up and per-hop communication times. Under what conditions is your algorithm *cost-optimal*?

4 a Explain the method of finite elements for solving partial differential equations, paying particular attention to identifying potential sources of parallelism. Use one-dimensional equations to illustrate but indicate where the complexity of the method increases significantly at higher dimensions.

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

$$\frac{du}{dx} + ku = 0$$

over the interval $0 \le x < X$, where k and X are constants. Divide the interval into n equal sized finite elements and define the shape (or basis) functions $\phi_0, \phi_1, \ldots, \phi_{n-1}$ by:

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

for $0 \le i \le n-1$. That is ϕ_i is unity on the (i+1)th element and zero elsewhere.

i) Approximate your solution for $u(x)$ by a linear mixture of all the n shape functions, call it $v(x)$.

ii) Minimise the error by setting to zero, for each shape function, the inner product:

$$\int_0^X \phi_i(x)\left[\frac{dv}{dx} + k\,v\right]dx \qquad (0 \le i \le n-1)$$

iii) Deduce the approximation $v(x) = \alpha_0\left(1 - \frac{kX}{n}\right)^i$ for $iX \le nx < (i+1)X$, where α_0 is a constant.

End of paper