UNIVERSITY OF LONDON
IMPERIAL COLLEGE OF SCIENCE, TECHNOLOGY AND MEDICINE

EXAMINATIONS 2004

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

Wednesday 28 April 2004, 14:30
Duration: 120 minutes

Answer THREE questions

Paper contains 4 questions
Calculators required

Section A (Use a separate answer book for this Section)

1 a Consider a parallel algorithm running on $p \geq 1$ processors and assume that its *best serial run time*, which is equivalent to the *problem size*, is the time elapsed in the parallel algorithm's execution on a single processor, i.e. for $p = 1$. Define the following terms relating to the algorithm's quantitative behaviour:

i) Parallel run-time, T_p

ii) Speed-up, S_p

iii) Efficiency, E_p

iv) Cost, C_p

What is meant by saying that an algorithm is cost-optimal and scalable? Define the *Isoefficiency function* of a cost-optimal algorithm and explain how it describes scalability.

b i) Describe the DNS algorithm for multiplying two $n \times n$ matrices using $p = n^3$ processors.

ii) Can the algorithm be cost-optimal when executing on a PRAM? Justify your answer, carefully explaining any PRAM variant you use.

c Consider now the implementation of the DNS algorithm on a hypercube of $p = q^3$ processors, where q divides n.

i) Using a block-checkerboard partitioning of the matrices A and B over plane 0 initially, how does your algorithm change?

ii) Assuming communications along each dimension are in a sub-hypercube of q processors, what is the maximum number of links traversed in each communication step?

iii) Assuming store-and-forward communications and ignoring per-hop times and start-up times, show that the parallel run-time is $T_p = n^3/p + t_w n^2/p^{2/3} \log_2 p$ where t_w is the time to transmit one matrix element over one link.

iv) Hence show that the algorithm is cost-optimal provided $n^3 \geq \Theta(p(\log_2 p)^3)$.

The three parts carry, respectively, 40%, 25%, and 35% of the marks.

2 a Compare and contrast the *Jacobi* and *Gauss-Seidel* methods for iteratively solving a set of linear equations, in terms of parallel run-time on p processors.

b Outline the *method of finite elements* for solving partial differential equations, paying particular attention to identifying potential sources of parallelism. Illustrate using one-dimensional equations (single dependent variable) but indicate where the scope for parallel computation increases significantly at higher dimensions.

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

$$\frac{\mathrm{d}u}{\mathrm{d}x} + ku = 0$$

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

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

for $0 \leq i \leq n-1$. That is, ϕ_i is 1 on the $(i+1)^{\text{st}}$ element and is 0 everywhere else.

i) Approximate your solution for $u(x)$ by a linear mixture $v(x)$ of the shape functions, with coefficients to be estimated.

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

$$\int_0^X \phi_i(x) \left[\frac{\mathrm{d}v}{\mathrm{d}x} + kv\right] \mathrm{d}x \quad (0 \leq i \leq n-1)$$

iii) Deduce the approximation

$$v(x) = \alpha_0 \left(1 - kX/n\right)^i$$

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

The three parts carry, respectively, 35%, 25%, and 40% of the marks.

Section B (Use a separate answer book for this Section)

3 Consider the following algorithm that performs a collective data operation on a square 2D wraparound mesh with p nodes. Here *node* is the rank (number) of the processor (from 0 to $p-1$) and *input_vector* is an m-word input vector.

```
procedure collective_mesh(node, input_vector, p)
    left = node − (node mod √p) + (node − 1 + √p) mod √p
    right = node − (node mod √p) + (node + 1) mod √p
    result = input_vector
    msg = result                                   /* point 1 */
    for i = 1 to √p − 1 do
        send msg to right
        receive msg from left
        result = result ∪ msg
    end for
    up = (node − √p + p) mod p                     /* point 2 */
    down = (node + √p) mod p
    msg = result
    for i = 1 to √p − 1 do
        send msg to down
        receive msg from up
        result = result ∪ msg
    end for
end procedure                                      /* point 3 */
```

a Given a 9 processor mesh with input vector $\{i\}$ on node i (for $i = 0, 1, 2, \ldots, 8$), draw three diagrams to show the contents of the result vector on each mesh node at each of the points 1, 2 and 3 in the algorithm.

b To which MPI primitive does this operation correspond?

c Derive a *general* expression for the parallel run time of this algorithm. You may assume start-up time is t_s, hop time is negligible, per-word transfer time is t_w and the time taken to union two vectors with k (one word) elements is kt_a.

d Define **procedure** collective_hypercube(*node*, *input_vector*, *d*) which implements the same collective data operation on a hypercube with $p = 2^d$ nodes.

e Derive a general expression for the parallel run time of the hypercube algorithm you devised in part (d). Hence explain which architecture performs the collective data operation more efficiently for large messages.

The five parts carry, respectively, 25%, 10%, 20%, 20%, and 25% of the marks.

4 a State whether each of the following statements is True or False. In each case, give a full justification for your answer.

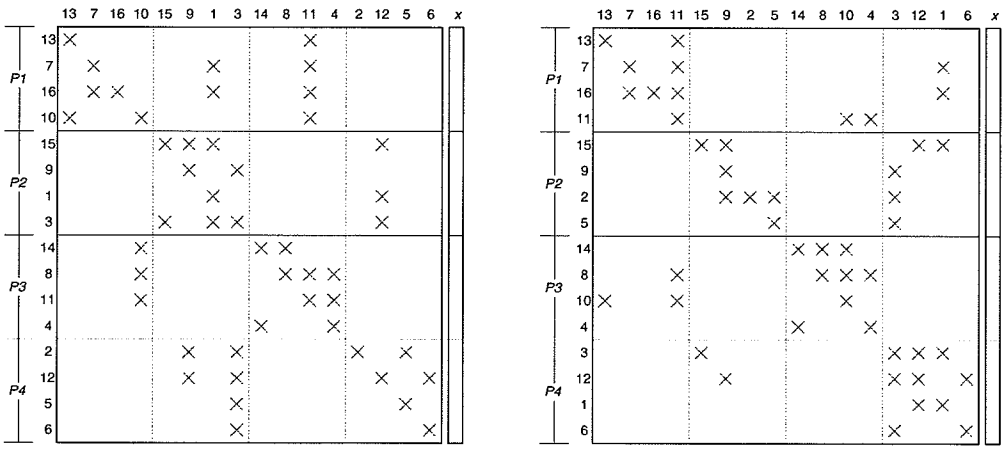
i) It is cheaper to build a 64-processor hypercube than to build a 64-processor 3D wraparound mesh. You may assume that processing elements and links cost the same for both architectures and that links are bidirectional.

ii) The worst-case time taken to send a message between two processors on a 64-processor 3D wraparound mesh will be higher than that on a 64-processor hypercube.

iii) If the parallelisation overhead for an algorithm with problem size W running on p processors is $O_p = \sqrt[3]{W}p^2$, then the problem size must increase by a factor of 8 to maintain the same efficiency when the number of processors is doubled.

iv) In a dynamic Asynchronous Round Robin (ARR) load-balancing scheme, the worst-case minimum number of requests for work guaranteed to include at least one request to each of p processors is $V(p) = p^2 - 3p + 3$.

b Consider the following two balanced row-striped partitionings (derived from the same input matrix) to be used in parallel sparse-matrix vector multiplication across 4 processors :



i) Which partitioning (left or right) was produced using hypergraph partitioning, and which using graph partitioning? Justify your answer.

ii) For the graph partitioning, what is the edge cut? For the hypergraph partitioning, what is the hyperedge cut?

The two parts carry, respectively, 60%, and 40% of the marks.