UNIVERSITY OF LONDON
IMPERIAL COLLEGE OF SCIENCE, TECHNOLOGY AND MEDICINE

EXAMINATIONS 2001

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

Friday 4 May 2001, 10:00
Duration: 120 minutes

Answer THREE questions

Paper contains 4 questions
Calculators not required

1 a Explain the data partitioning techniques for matrix operations *striping (by row or column)* and *checkerboarding,* distinguishing between the *block* and *cyclic* variants. In what circumstances would *cyclic* partitioning provide a more efficient implementation?

b In a certain simulation program, there are a set of n event types, $E_1,\ldots E_n$. The simulated time of the j^{th} occurrence of event E_i is T_{ji} and the time elapsed between the j^{th} time E_i is *enabled* and T_{ji} is an *input* d_{ji} to the simulator. In this simulator, the following relationship holds:

$$T_{ji} = d_{ji} + \max_k (T_{j-1,k}+s_{ki})$$

where s_{ki} is the time elapsed between $T_{j-1,k}$ and the next enablement of E_i.

i) Write this relationship in matrix form as an *inner-product* (cf. 'dot-product' in arithmetic) defined in terms of appropriate operators. ***Hint***: The j^{th} vector $\mathbf{T}_j$ has components T_{ij} which can be written in the form $(m_{i1}\otimes T_{j1}) \oplus (m_{i2}\otimes T_{j2}) \oplus \ldots \oplus (m_{in}\otimes T_{jn})$ for certain numbers m_{ji} and operators $\otimes$ and $\oplus$ which you must define $(1\le i,j\le n)$.

ii) Describe two parallel algorithms to execute your inner product on n processors using first *row-striping* and then *column-striping* data partitioning schemes.

iii) Outline how to modify the *row-striping* algorithm to implement *block-striping* when there are $p < n$ processors. (Assume p divides n.)

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

2 a A discrete, regular mesh $\{(i,j) \mid 0\le i,j\le n\}$ is defined on the quadrant $\{(x,y) \mid x,y \ge 0\}$, where the point (i,j) represents the co-ordinate (x_i, y_j) and $x_{i+1}-x_i = y_{j+1}-y_j = h$ for $0\le i,j\le n-1$.

i) Use *central differencing* to estimate the partial derivatives $\partial u/\partial x$, $\partial u/\partial y$, $\partial^2 u/\partial x^2$ and $\partial^2 u/\partial y^2$ of some function $u(x,y)$ at internal points. How can these derivatives be estimated at *boundary points*?

ii) Estimate the second derivative $u_{xy} \equiv \partial u_y/\partial x$ at (x_i, y_j) by applying your result for part i) to the function u_y, itself approximated by part i).

b Describe how the method of *finite differencing* can be used to transform a differential equation into a set of linear equations. Explain the role of the boundary conditions or initial conditions in the solution of these equations.

c Consider the partial differential equation:

$$u_{xy} = x+y$$

defined for $x,y\ge 0$ with initial conditions $u(x,0) = 1/(1+x)$ and $u(0,y) = 1/(1+y)$. Derive a set of linear equations that approximately solve this equation in the upper right quadrant, using a square mesh of size h. Indicate an iterative algorithm for obtaining the solution. Where can parallel computation be used?

The three parts carry, respectively, 35%, 30% *and* 35% *of the marks.*

3 In a vector reduction, each of p processors supplies a vector of n values, and n element-by-element reductions are performed (using a user-supplied commutative, associative operator) to produce a vector of n results on every processor. So the result of performing a vector reduction over the four 3-vectors {4,1,2}, {1,5,3}, {2,3,4} and {0,2,6} using the max operator is {4,5,6}, {4,5,6}, {4,5,6} and {4,5,6}. A simple exchange algorithm for performing this vector reduction operation on a hypercube with bidirectional links is:

```
procedure hypercube_vector_reduce(node, input_vector, d)
begin
    result_vector = input_vector
    for i = 0 to d - 1    /* d main steps */
        dest = node XOR 2^i   /* determine exchange partner */
        send result_vector to dest  /* exchange vectors */
        receive message_vector from dest
        result_vector = OP(result_vector, message_vector)
    end for
end
```

This program is executed by each processor in the hypercube. Here `d` is the dimension of the hypercube ($p = 2^d$), `node` is the processor's label/number and `input_vector` is the n-element input vector on the processor. `XOR` is the bitwise exclusive-or operator and `OP` is a user-supplied function that applies the reduction operator over two n-vectors on an element-by-element basis.

a Suppose the vectors {4,1,2}, {1,5,3}, {2,3,4} and {0,2,6} are held on nodes 0 (binary 00), 1 (binary 01), 2 (binary 10) and 3 (binary 11) of a 4 processor hypercube respectively. Draw a diagram showing the communication that takes place and the contents of `result_vector` on each hypercube node for each step of the vector reduction algorithm using the max operator.

b Derive a general expression for the parallel run time of the algorithm on a hypercube with p processors. You may assume each vector element uses one word of storage, message start-up time is t_s, hop time is negligible, per-word transfer time is t_w and the time to perform a reduction operation on two words is t_a. What is the parallel cost?

c Derive an expression for the sequential run time (i.e. if p vectors were to be reduced all on a single processor). Are there conditions under which the parallel algorithm will be cost-optimal?

d Predict the speedup and efficiency of the algorithm when reducing vectors of length 100 on a 32 processor hypercube that has t_s = 100ns, t_w = 50ns and t_a = 450ns. What is the maximum speedup that can be achieved for any vector length on this particular hypercube?

e Which MPI primitives could be used to implement the communication that takes place during each main step of the algorithm in an efficient way?

The five parts carry, respectively, 15%, 25%, 20%, 25% and 15% of the marks.

4 a State whether each of the following statements is True or False. In each case, give a *brief* (one sentence) justification for your answer.

i) An EREW PRAM can compute the maximum of n elements in O(1) time using n^2 processors.

ii) It is possible to observe an efficiency above 1 in practice.

iii) Node-to-node latency in a 2D wraparound mesh with cut-through routing and negligible hop time will appear to be constant, i.e. independent of the co-ordinates of the sending and receiving nodes.

iv) An upper bound on the communication time to send a message of length m words between two nodes in a p processor 3D wraparound mesh with store-and-forward routing is given by $t_s + 3\lfloor \sqrt[3]{p}/2 \rfloor t_h + mt_w$ where t_s is message start up time, t_h is per-hop time and t_w is per-word transfer time.

b The following is a sequential algorithm for performing a breadth first search of a directed graph:

```
E = {s0}
Q.insert( s0 )
while (Q not empty) do begin
    s = Q.remove()
    for each s'∈ succ(s) do begin
        if s'∉ E do begin
            Q.insert(s')
            E = E ∪ {s'}
        end
    end
end
```

Here s_0 is the root node, Q is a FIFO queue of unexplored nodes, and E is the set of explored nodes. Function *succ*(s) returns the set of successor nodes of node s.

i) Describe how you would extend this algorithm to implement a message passing scheme for performing a breadth-first search of a directed graph in parallel. Include a discussion of what data structures you would need, how nodes would be assigned to processors, when messages would be sent and how you would achieve a good load balance.

ii) What problem related to the number of messages would you have to overcome if you attempted to implement your algorithm in practice? How could you solve the problem?

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

End of paper