UNIVERSITY OF LONDON

IMPERIAL COLLEGE OF SCIENCE, TECHNOLOGY AND MEDICINE

EXAMINATIONS 1998

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 / I4.10

PARALLEL ALGORITHMS

Wednesday, April 29th 1998, 10.00 - 12.00

Answer THREE questions

For admin. only: paper contains 4 questions

1 A state generation program for state-transition systems (equivalent to a breadth first graph enumeration algorithm) has been implemented on a static wraparound square mesh of p processors where p is the square of an even number. Each state is assigned to a unique processor and careful load balancing has ensured that, after a start-up period and before the termination phase of the algorithm, all processors operate functionally in the same way, using a stack for its own states, as follows:

1) Take the state from the top of the stack;
2) Generate the set of next states (including any already generated);
3) Push those states to be processed locally onto the stack;
4) Store each other state generated into the output buffer allocated to that state's (remote) processor;
5) Repeat from 1).

The buffers are transmitted over the network when they become full with 8000 bytes of data, using a *blocking I-O cut-through* transfer; i.e. the processor waits for the corresponding acknowledgement to be received. Messages are divided into *flits* of size 100 bytes, which are sent between adjacent nodes serially. The header of any message consists of one flit and acknowledgement messages comprise only a header flit. State-messages (as stored in the buffers) are 200 bytes long.

a Take some time to understand the algorithm and list the main issues that you think influence its performance.

You may now ignore the start-up period and termination phase and assume that the stack is never empty in any processor. You may also ignore any queueing time for communication resources. Given that a total of A states are generated (including duplicates which you need not distinguish), that a processor takes c seconds to process one state locally and that I-O transfers have a combined start-up and receive overhead time of s seconds and per-hop latency for one flit of f seconds:

b calculate the number of messages generated per processor, assuming that new states generated belong to each of the p processors with equal probability;

c calculate the time spent by each processor computing new states;

d calculate the time spent waiting for communication to complete by each processor;

e calculate the speed-up and efficiency of the algorithm executing on this architecture;

f is the algorithm cost-optimal, and why?

The six parts carry, respectively, 20%, 15%, 10%, 25%, 15% *and* 15% *of the marks.*

2 a i) Explain the qualitative differences between the implementations of static and dynamic parallel algorithms on a multiprocessor computer. Give examples of algorithms of each type, suggesting appropriate interconnection networks.

ii) What problems arise in predicting the run time of a dynamic parallel algorithm on a multiprocessor with dynamic interconnection network?

b i) Define the term *d-hypercube* (i.e. a hypercube with d dimensions) interconnection network and show how to construct one from two (d–1)-hypercubes for $d>1$. Find the *diameter* of a d-hypercube.

ii) Give an algorithm to sort the items at the leaves of a binary tree of depth greater than d on a d-hypercube. (You may *assume* you have a parallel algorithm to *merge* the items in a hypercube when they are in order on each of two subcubes.)

3 a i) Describe an algorithm to compute the inner product (i.e. normal matrix multiplication) of two matrices using *block checkerboarding partitioning.*

ii) What is the parallel run time of your algorithm on p processors?

iii) Under what conditions is your algorithm *cost-optimal*?

b i) Describe an algorithm to implement *Gaussian elimination* on n linear equations in n unknown variables on p processors, where n is an integer multiple of p. Derive an expression for the parallel computation time, assuming arithmetic operations each take unit time to execute.

ii) Explain how efficiency can be improved using *pipelining* and estimate how big a reduction in computation time can be obtained How might a *cyclic* partitioning scheme improve performance?

Turn over .....

4 a A regular mesh of points labelled $\{(i,j) \mid 0 \le i,j \le n\}$ is defined on the quarter plane $\{(x,y) \mid x,y \ge 0\}$, where the point (i,j) represents the co-ordinate (x_i, y_j) and $x_{i+1}-x_i = y_{i+1}-y_i = h$ for $0 \le i \le n-1$. 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*, and are these estimates necessary?

b Consider the partial differential equation:

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0$$ with boundary condition $u(x,y) = x+y$ for (x,y) on the unit square with vertices at co-ordinates (0,0), (0,1), (1,1), (1,0).

i) Explain carefully how the method of *finite differencing* can be used to approximate the numerical solution to the differential equation.

ii) What opportunities for parallel computation are there?

iii) How would the solution change if the right hand side of the equation were replaced with the expression $(x-y)^2$?

iv) How would the solution change if the right hand side of the equation were replaced with the expression $\frac{\partial u}{\partial x} + \frac{\partial u}{\partial y}$?

c Briefly explain how the method of *finite elements* could be applied to solve the same equations. Where can parallelism be introduced?

The three parts carry, respectively, 30%, 50% *and* 20% *of the marks.*

End of paper