PHILOSOPHICAL TRANSACTIONS OF THE ROYAL SOCIETY A MATHEMATICAL, PHYSICAL & ENGINEERING SCIENCES

General Discussion

R. E. Overill and M. V. Wilkes

Phil. Trans. R. Soc. Lond. A 1988 **326**, 499
doi: 10.1098/rsta.1988.0100

Phil. Trans. R. Soc. Lond. A **326**, 499 (1988)

Printed in Great Britain

General discussion

R. E. Overill (*Department of Computing, King's College London*). I believe that this meeting has demonstrated the need for a change in attitude by both computer scientists and conventional scientists. Professor Hoare has suggested that computer scientists have a useful contribution to make in the area of effective use of multiprocessor architectures; this will require them to enter into much closer contact with colleagues involved in science and engineering applications. Professor Parkinson has pointed out how reluctant many such computer users are to relinquish their ill-structured, undocumented, sequential programs for better computational tools. However, the impressive results presented by Professor Wallace, which would have been uncontemplatable with older approaches, should provide a powerful stimulus for any hidebound computational scientist to explore the exciting possibilities opened up by effective use of the new generation of multiprocessor architectures.

M. V. Wilkes, F.R.S. (*Olivetti Research Ltd, Cambridge*). In the past, advances in computer hardware engineering have created opportunities for research in system programming. For example, the coming of computers capable of supporting time-sharing led to a great wave of activity in the design of operating systems. Personal computers, with their bit-mapped displays, led to a similar wave of activity in computer graphics. In both cases, before the potentiality of the new hardware could be exploited by the ultimate users, software development in the area of system programing was necessary.

Many people thought that the same thing would happen with parallel computing. It has not done so, either for multiprocessors with a common memory, or for computers with private memories which communicate with each other by passing messages. There is, I think, a reason for this. To get speed by parallelism one must first identify opportunities for spreading the work over a number of processors and then take steps to ensure that the load is well balanced so that the total processor idle time is minimized.

Unfortunately there are severe theoretical limitations to the amount of load balancing that can be achieved by a compiler. Run-time load balancing has possibilities, but little practical progress has been made. Load balancing can, however, be achieved by an application programmer who thoroughly understands his problem and also understands the timing constraints of the particular parallel computer on which it is to be run. Thus the advent of multiprocessor systems has created opportunities for the application programmer, but has short circuited the systems programmer who, as I have said above, made an essential contribution on earlier occasions.

The development of efficient parallel programs calls for much effort. This will limit parallel programing to certain application areas and we have yet to see how extensive those areas are. To some extent one might hope that experience with parallel programming could be transferred from one application area to another, or from one particular design of parallel computer to another. However, I would not expect that any principles of general interest will emerge. This need not deter those whose aim is to secure results in their own scientific or engineering disciplines.