HIGH PERFORMANCE COMPUTING

HIGH PERFORMANCE COMPUTING

Proceedings of the International Symposium on
High Performance Computing
Montpellier, France, 22–24 March, 1989

edited by

J.–L. DELHAYE
Centre National Universitaire Sud de Calcul
Ministère de l'Education Nationale
Montpellier, France

E. GELENBE
Ecole des Hautes Etudes Informatiques
Université René Descartes
Paris, France

1989

NORTH-HOLLAND
AMSTERDAM • NEW YORK • OXFORD • TOKYO

ELSEVIER SCIENCE PUBLISHERS B.V.
Sara Burgerhartstraat 25,
P.O. Box 211, 1000 AE Amsterdam, The Netherlands

Distributors for the United States and Canada:

ELSEVIER SCIENCE PUBLISHING COMPANY, INC.
655 Avenue of the Americas
New York, N.Y. 10010, U.S.A.

ISBN: 0 444 88043 7

Printed in The Netherlands

PREFACE

The European Symposium on High Performance Computing was held in Montpellier, France, from March 22 to 24, 1989 at the new Congress Center in Le Corum. It was organized by CNUSC (Centre National Universitaire Sud de Calcul, Ministère de l'Education Nationale), in association with IBM and CERFACS, and was sponsored by the District de Montpellier, Département de l'Herault, Région Languedoc-Roussillon.

The development of large scale computing is now considered an essential factor in research productivity, in all scientific and technical domains. The purpose of this symposium was to offer an overview of the state-of-the-art through papers concerned with both methods and applications, and to promote the exchange of experience and information.

These proceedings contain most of the papers presented at the meeting. Some of the authors who made presentations about recent research in industry at the conference meeting did not submit written reports of their work to be included in this volume for obvious reasons.

The papers in these proceedings are grouped into seven subject areas: Numerical Algorithms and Tools, Parallel Architectures, Applications in Fluid Dynamics, Applications in Chemistry, Applications in Physics, New Application Areas, and Education and Training.

The first area is concerned with numerical algorithms and tools. New computer architectures, whether they are parallel or vector, require elaborate algorithmic work in order to enhance performance and numerical stability.

The second area is devoted to parallel architectures, either related to supercomputers (CRAY, IBM) or to computers with a high level of parallelism (Hypercubes).

The next three parts present experience in three areas which have been heavy supercomputing users for a long time: Fluid Dynamics, Chemistry, and Physics. A high proportion of these presentations is concerned with the use and effect of vectorization and parallelization on the IBM 3090/VF.

New application areas such as medicine and economics are discovering the contribution which can be made by High Performance Computing. The four presentations in Part VI are thus of great interest because they open new horizons to those subjects' specialists.

Finally, the last section contains several contributions made as part of an open discussion on the training of supercomputer users. Emphasis must be placed on rapidly educating and training research workers, engineers and scientific students.

I wish to express my gratitude to Michèle Batlle, Pierre Richard, Erol Gelenbe, Francesco Pavoncelli for their work on the Organizing Committee, and to John Connolly, Brian Davis, Ian Duff, Dieter Haupt, Jean-Claude Ippolito, Chris Jones, Domenico Laforenza, Steve Lavenberg, Larry Lee, and Piero Sguazzero for agreeing to be members of the Program Committee.

In addition, I would like to express my special thanks to a number of friends and colleagues who helped considerably in making this symposium a success. They include Isabelle Wurth the Symposium Secretary, Sylvie Flamand, Kathy Barbieri, Andree Balzac, Jean-Claude Ippolito, Philippe Falandry, Bernard Di Miceli, Philippe Gairard, and Herb Budd.

Jean-Loïc Delhaye
Chairman of the Organizing Committee
Montpellier, April 1989

CONTENTS

APPLICATIONS IN CHEMISTRY

APPLICATIONS IN PHYSICS

NUMERICAL ALGORITHMS AND TOOLS

High Performance Computing
J.-L. Delhaye and E. Gelenbe (Editors)
Elsevier Science Publishers B.V. (North-Holland), 1989

PARALLEL ALGORITHMS FOR SCIENTIFIC COMPUTING

David J. EVANS

Department of Computer Studies
University of Technology
Loughborough, Leicestershire, U.K.

Although mini-supercomputers cost an order of magnitude less than mainframes or supercomputers, their use in typical mainframe applications has been expanding rapidly beyond all expectations. Initially, their performance advantage was fast scientific computation with vector processing and embrionic parallel processing capabilities resulting in their suitability for simulation, scientific and technological applications.

More recently, however, with the increasing usage of parallel processing strategies and techniques, mini-supercomputers are emerging as serious contenders in an expanding arena of information processing applications.

Finally, the increasing trend of solving large linear systems derived from scientific and engineering problems is reconsidered and a parallel algorithm presented which obviates the need for 'linear solvers'.

1. INTRODUCTION

The past few years have seen the accelerated growth of a new breed of computers - *the mini-supercomputer*. These incredibly useful machines have transformed the computer scenario by their sheer capability and adaptability in many traditional application areas for they offer 25% of the power of a supercomputer for only 10% of the cost.

Clearly, then although the mini-supercomputer cannot compete on peak supercomputer performance they can provide a cost effective, high performance machine or work station in scientific and engineering application areas such as simulation and modeling, finite element/ difference and matrix analyses, numerical solution of differential equations and approximation of functions in CAD and graphics.

How and where did these machines evolve? Historically the initial methods of achieving mini-supercomputer performance was via the use of special purpose *attached processors* (array processors). This was achieved by the rapid development of ASIC (application specific integrated circuit) design tools at board-level and chip-level implementations. The 32 bit array processor with its high performance processing of streams of data by firmware was initially used widely in signal and image processing but more recently has entered the domain of SIMD computing whilst the 64 bit array processor with the ability to execute programs rather than sub-

routines and with vector processing capability has emerged as the forerunner of the mini-supercomputer.

However the earlier attached processor designs although adding extra processing ability suffered from severe host communication problems which demonstrated the need for executing more code within the attached processor. This need for large fast memories and wide bandwidth communication led to a *full function attached processor* containing all the features of a C.P.U. being incorporated.

The final stage was the elimination of the host system by introducing an operating system and the realisation of the *stand-alone mini-supercomputer*. In the future, we can expect to see the mini-supercomputer performance coming to the workstation market while off the shelf VSLI chip sets will be incorporated into desktop configurations.

Finally, mini-supercomputer performance can also be attained effectively by *multiprocessor implementations*. In fact, the recently introduced Sequent Computer Symmetry multi-processor system attains the 10-MFLOP threshold of supercomputer performance when equipped with an optional Weitek arithmetic co-processor.

Multiprocessor systems are optimized for multiple job throughput. Whilst *parallel processors*, in contrast, focus on the execution of single jobs. Both, however, employ a number of processors, and the prospective buyer should be more concerned with performance and cost than with what often amounts to architectural semantics.

2. CURRENT TRENDS IN PARALLEL ARCHITECTURES

Let us look briefly at some parallel architectures - a procedure which one might go through when deciding what sort of parallel architecture is best for a given class of problem.

To achieve parallelism by extending sequential architectures with such machines like the Cray 1S, or CDC Cyber or the latest line of VAX machine involves quite *expensive technology*. So I think we all agree that parallel processing is the correct approach for the future.

This calls into question what kind of parallel architecture to consider. You could put together a few very expensive processors. Some of the supercomputer companies, in fact, are recognising this trend and moving that way with the Cray XMP and the Japanese FUJITSU computers. A better approach is to put together many low cost processors - *micro-processors* in fact. I think the most convincing argument for this is that the microprocessor is the most cost effective package of a computer instruction.

There are many advantages of using microprocessor technology in a parallel architecture. First, there is a tremendous amount of market forces associated with using a current chip design for the processor. Secondly, another advantage is to have a wide range of high quality compilers available and software tools such as the Kuck & Associates parallel preprocessor, etc. Finally, the importance of the communications software must not be forgotten when it comes to building a system around standard microprocessors and with an operating system like UNIX available.

The choice now is whether to have a tightly coupled or a *shared memory* architecture, or a loosely coupled or *distributed memory* architecture? Both types of systems are now the subject of extensive competition in the marketplace.

More recently it has become evident that the shared memory architectures with a moderate number of processors have a programming model that is much more closely aligned with the traditional uni-processor approach. So ultimately the machine is easier to use and program. On the other hand, loosely coupled architectures, i.e. hypercube, or mesh-type topologies do have the advantage of using a few hundred or even a few thousand processors. However because of the communication problems that are involved, trade-off's are made to decrease the number of interconnects between the processors. In addition, the mapping of the algorithms on to such architectures has to be seriously reconsidered which can often involve extensive reprogramming overheads.

Thus midrange multi-processors, sometimes called multi-microcomputers achieve super-minicomputer and even mini-supercomputer performance by combining 16 to 64 processors with memory and I/O subsystems, usually in tightly coupled configurations. Like low-end multi-processors, they use proprietary system buses, standard microprocessor I/O buses and variations on the UNIX theme.

Parallel processing systems are specialized multi-processor systems. They fall into two categories: common backplane implementations that achieve parallel operation through the use of computer technology, and architectures like hypercubes and systolic arrays in which processors communicate with a limited number of cohorts. Such systems are the focus of much activity in terms of both research and publicity.

Few technology issues remain unresolved in the multi-processor field. Loosely and tightly coupled systems of minicomputer and microprocessor-based APU's are now available from several suppliers. Dynamic load balancing is now common, and the capability to parallelize tasks is rapidly emerging.

Differences between implementation strategies usually revolve around the specific tasks for which the systems are best suited. In general, low-end multi-processors are used in the workstation, network server and multi-user applications, while multi-microcomputers and large multi-processors compete with super-minicomputers.

The forecast for the computer industry as a whole for the remainder of the decade predicts a growth of roughly 10% per year. If the forecast is right, growth in the multi-processor segment would presumably have to come at the expense of (and be resisted by) the established firms. New market entrants such as mini-supercomputer and RISC suppliers will provide fierce competition.

Clearly, a pitched battle will rage (with a high casulty rate) between super-minicomputer, mini-supercomputer and multi-microcomputer/parallel-processor suppliers for what is now the high end of the super-minicomputer market. The flexibility of multi-processor systems (variable granularity, incremental growth capability and fault-tolerance, etc.) suit them well for specialised markets and midrange multitasking applications. Finally

parallel computers can be expected to replace an entire segment of mid-range machines, including those currently categorized as mini, supermini and mainframe. Single processor-based equipment will hold down the low end, with massive supercomputers on the other end of the price and performance range, (Fig.1).

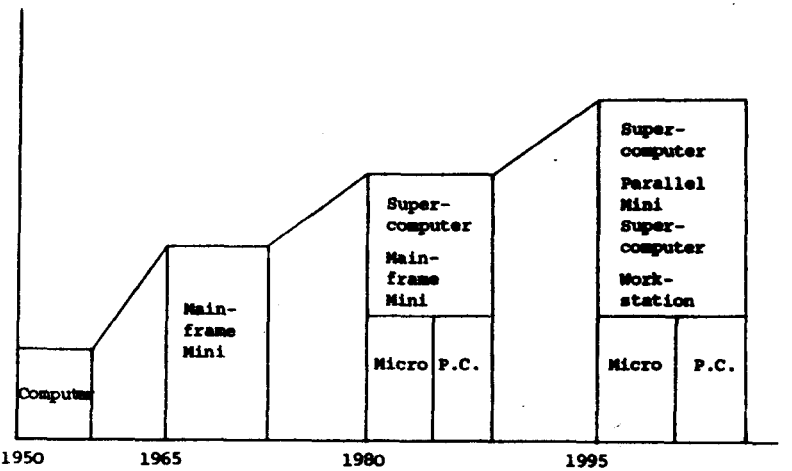


FIGURE 1: Changes in computer markets due to the impact of parallel architectures.

3. FUNDAMENTAL ISSUES IN PARALLEL ALGORITHM DESIGN FOR LINEAR SYSTEMS

Research into parallelism indicates that often we cannot directly extend the techniques used in serial algorithms to parallel computers because efficient serial algorithms do not necessarily lead to efficient parallel algorithms. Our example illustrates the *'divide and conquer'* strategy. This (very old) idea is probably the first that would occur to anyone designing an algorithm for parallel machines. One first partitions the problem into several subproblems and then the parallel machine solves the subproblems simultaneously. Unless the subproblems are independent, the data must be communicated between the subproblems. For the success of a divide and conquer technique, it is necessary that the gains in solving the subproblems in parallel are not outweighed by the extra work required to obtain the solution to the whole problem from that of the subproblems. An essential ingredient of the strategy is that the subproblems are rearranged in 'explicit' form so that a parallel algorithm can be achieved.

The solution of banded and tridiagonal linear systems occurs frequently in the solution of partial differential equations in Computational Mathematics. Here we introduce a new parallel algorithm which can be readily extended to cope with multi-dimensional boundary value problems.

Consider the following second order parabolic equation,

$$\frac{\partial U}{\partial t} = \frac{\partial^2 U}{\partial x^2}, \quad 0 \leq x \leq 1, \quad 0 < t \leq T \qquad (3.1)$$

subject to the initial-boundary conditions,

$$U(x,0) = f(x),\ 0<x<1\ ,$$

$$U(0,t) = g(t),\ 0<t\leq T\ , \tag{3.1a}$$

and $$U(1,t) = h(t)\ .$$

A uniformly-spaced network whose mesh points are $x_i=i\Delta x$, $t_j=j\Delta t$ for $i=0,1,\ldots,m,m+1$ and $j=0,1,\ldots,n,n+1$ is used with $\Delta x=1/(m+1)$, $\Delta t=T/(n+1)$ and $\lambda=\Delta t/(\Delta x)^2$, the mesh ratio.

A weighted approximation to the differential equation (3.1) at the point $(x_i,t_{j+\frac{1}{2}})$ is given by,

$$-\lambda\theta u_{i-1,j+1}+(1+2\lambda\theta)u_{i,j+1}-\lambda\theta u_{i+1,j+1} = \lambda(1-\theta)u_{i-1,j}+(1-2\lambda(1-\theta))u_{ij}+$$

$$\lambda(1-\theta)u_{i+1,j}\ ,\ i=1,2,\ldots,m. \tag{3.2}$$

This approximation can be displayed in a more compact matrix form as,

$$\begin{bmatrix} a & b & & & & \\ c & a & b & & & \\ & c & a & b & & \\ & & \ddots & \ddots & \ddots & \\ & & & c & a & b \\ & & & & c & a \end{bmatrix}_{(m\times m)} \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_{m-1} \\ u_m \end{bmatrix}_{j+1} = \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_{m-1} \\ f_m \end{bmatrix}_j\ , \tag{3.3}$$

i.e., $$Au = f\ , \tag{3.4}$$

where, $c = -\lambda\theta$, $a = (1+2\lambda\theta)$, $b = -\lambda\theta$;

$$f_1 = \lambda(1-\theta)(u_{0j}+u_{2j})+\lambda\theta u_{0,j+1}+(1-2\lambda(1-\theta))u_{1,j};$$

$$f_i = \lambda(1-\theta)(u_{i-1,j}+u_{i+1,j})+(1-2\lambda(1-\theta))u_{ij},\ i=2,3,\ldots,m-2,m-1;$$

$$f_m = \lambda(1-\theta)(u_{m-1,j}+u_{m+1,j})+(1-2\lambda(1-\theta))u_{mj}+\lambda\theta u_{m+1,j+1}$$

and

$$u = (u_{1,j+1},u_{2,j+1},\ldots,u_{m,j+1})^T \text{ and } f = (f_1,f_2,\ldots,f_m)^T. \tag{3.4a}$$

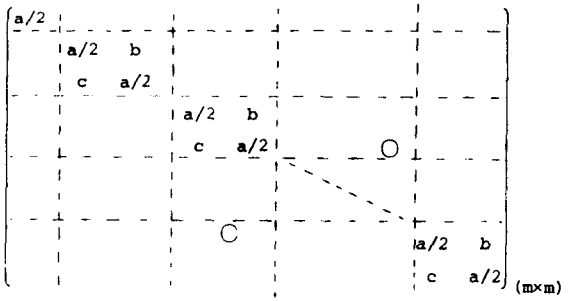
We note that f is a column vector of order m consisting of the boundary values as well as known u values at time level j while u are the values at time level (j+1) which we seek. We also recall that (3.4) corresponds to the fully implicit, the Crank Nicolson, the Douglas and the classical explicit methods when θ takes the values 1, $\frac{1}{2}$, $\frac{1}{2}$, $-1/12\lambda$ and 0 with accuracies of the order $O([\Delta x]^2+\Delta t)$, $O([\Delta x]^2+[\Delta t]^2)$, $O([\Delta x]^4+[\Delta t]^2)$ and $O([\Delta x]^2+\Delta t)$ respectively.

Let us assume we have an *even number of intervals* (corresponding to an odd number of internal points, i.e. m odd) on the real line $0\leq x\leq 1$. A similar analysis holds for m even. We can then perform the following

splitting of the coefficient matrix A:

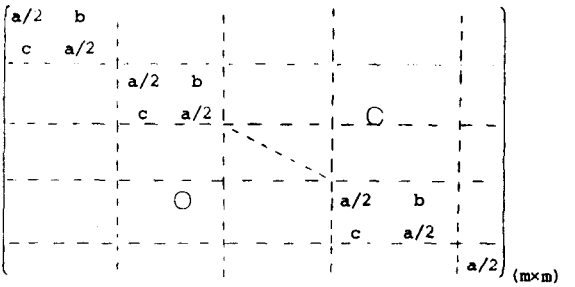
$$A = G_1 + G_2 \ , \tag{3.5}$$

where,

$G_1 =$ 

, (3.6)

and

$G_2 =$ 

. (3.7)

It is assumed that the following conditions are satisfied:

(i) G_1+rI and G_2+rI are non-singular for any $r>0$, $\theta \geq \frac{1}{2}$.

(ii) For any vectors f_1 and f_2 and for any $r>0$, the systems,

$$\begin{aligned} (G_1+rI) &= f_1 \\ \text{and} \quad (G_2+rI) &= f_2 \ , \end{aligned} \tag{3.8}$$

are more easily solved in explicit form since they consist of only (2×2) subsystems.

Thus, with these conditions, system (3.4) becomes,

$$(G_1+G_2)u = f \ . \tag{3.9}$$

The Alternating Group Explicit (AGE) iteration consists of writing (3.9) as a pair of equations

$$\begin{aligned} (G_1+rI)u &= (rI-G_2)u+f \ , \\ \text{and} \quad (G_2+rI)u &= (rI-G_1)u+f \ . \end{aligned} \tag{3.10}$$

The AGE method using *the Peaceman and Rachford variant* for the *stationary case* (r=constant) is given by,

$$\begin{aligned} (G_1+rI)u^{(p+\frac{1}{2})} &= (rI-G_2)u^{(p)}+f \ , \\ \text{and} \quad (G_2+rI)u^{(p+1)} &= (rI-G_1)u^{(p+\frac{1}{2})}+f \ , \ p \geq 0 \ , \end{aligned} \tag{3.11}$$

where $u^{(0)}$ is a starting approximation and r is a positive constant called the