COMPSTAT

Proceedings in
Computational Statistics

1988

Edited by
D. Edwards and N. E. Raun

Physica-Verlag
Heidelberg

COMPSTAT

Proceedings in Computational Statistics

8th Symposium held
in Copenhagen 1988

Edited by
D. Edwards and N. E. Raun

With 65 Figures

Physica-Verlag Heidelberg

David Edwards, NOVO, Clinical Research Department,
Novo Allé 9Q, DK-2880 Bagsværd, Denmark

Niels E. Raun, UNI-C, Danish Computing Centre for Research and
Education, Vermundsgade 5, DK-2100 Copenhagen Ø, Denmark

ISBN 3-7908-0411-8 Physica-Verlag · Heidelberg
ISBN 0-387-91336-X Springer-Verlag · New York

Printed in Germany

Cover: Erich Kirchner, Heidelberg
Offsetprinting: Kiliandruck Grünstadt
Bookbinding: T. Gansert GmbH, Weinheim-Sulzbach
7120/7130-543210

Preface

The papers assembled in this volume were presented at COMPSTAT 1988, the 8th biannual Symposium in Computational Statistics held under the auspices of the International Association for Statistical Computing.

The current impact of computers on the theory and practice of statistics can be traced at many levels: on one level, the ubiquitous personal computer has made methods for explorative data analysis and display, rarely even described in conventional statistics textbooks, widely available. At another level, advances in computing power permit the development and application of statistical methods in ways that previously have been infeasible. Some of these methods, for example Bayesian methods, are deeply rooted in the philosophical basis of statistics, while others, for example dynamic graphics, present the classical statistical framework with quite novel perspectives.

The contents of this volume provide a cross-section of current concerns and interests in computational statistics. A dominating topic is the application of artificial intelligence to statistics (and vice versa), where systems deserving the label "expert systems" are just beginning to emerge from the haze of good intentions with which they hitherto have been clouded. Other topics that are well represented include: nonparametric estimation, graphical techniques, algorithmic developments in all areas, projection pursuit and other computationally intensive methods.

COMPSTAT symposia have been held biannually since 1974. This tradition has made COMPSTAT a major forum for advances in computational statistics with contributions from many countries in the world. Two new features have been introduced at COMPSTAT '88. Firstly, the category of keynote papers has been introduced to highlight contributions judged to be of particular importance. Secondly, tutorial sessions in dynamic graphics (R. Becker), artificial intelligence in statistics (W. Gale) and graphical modelling (N. Wermuth) have been arranged, to satisfy the widespread interest in these new topics.

The programme committee, which consisted of E. B. Andersen, H. Caussinus, D. Edwards (chairman), D. Hand, T. Havránek, N. Lauro, F. van Nes and B. Streitberg, had the painful task of choosing 60 papers for publication in these proceedings, out of several hundred received. The criteria used were originality, accuracy and that the topics should have bearing on both statistics and computation.

The scientific programme consisted of the contributed, invited and keynote papers (collected in this volume) and short communications, posters and tutorials (collected elsewhere). Moreover, presentations and demonstrations of non-commercial software, an exhibition of commercial software, a book exhibition, and not least important an exhilarating social programme were arranged by the organizing committee, which consisted of P. Allerup, I. A. Larsen, A. Milhøj and N. E. Raun (chairman). The assistance of Lone Cramer is also gratefully acknowledged. The meeting was arranged by UNI•C, Danish Computing Centre for Research and Education and was sponsored by the Danish Research Council.

David Edwards • Niels E. Raun

Copenhagen

July 1988

Contents

Keynote Papers

Non-Parametric Estimation

Projection Pursuit

Graphical Techniques

Expert Systems

Languages and Packages

Computational Intensive Methods

Algorithms

Statistical Methods

Time Series

Statistical Data Bases and Survey Processing

Experimental Design

Econometric Computing

Late Arrivals

Address list of Authors

Contributors

Addresses of authors will be found at the end of this volume

Adorf, H.-M. 239
Alvoni, E. 369
Baldessari, B. 65
Bartlett, G. 291
Bell, E. 143
Belsley, D.A. 99
van den Berg, G.M. 149
Berzuini, C. 273
Bethlehem, J. G. 377
Boomsma, A. 335
Brillet, J.-L. 417
Brown, P. 207
le Calvé, G. 341
Catarci, T. 401
Ciampi, A. 267
Cleave, N. 207
Cox, M.A.A. 323
Cox, T.F. 323
Cwik, J. 439
Deville, J.C. 255
Dobrić, V. 135
Duijkers, T.J. 149
Duijsens, I.J. 149
Dutter, R. 219
Efron, B. 423
Eilers, P.H.C. 363
Elguero, E. 59
de Falguerolles, A. 81
Felsenstein, K. 349
Gallo, F. 65
Gilchrist, R. 213
Gordesch, J. 285
Green, M. 317
Grosbas, J.M. 255
Härdle, W. 17
Hájek, P. 117
Hastie, T. 279
Hietala, P. 163
Holmes-Junca, S. 59
Hougaard, P. 31
Iliakopoulos A. 73
Isogai, T. 87
Ivánek, J. 175
Jirků, P. 169
Jiroušek, R. 123
Keller, 377
Kříž, O. 123
Liski, E.P. 311
Luijben, T.C. 335
Marti-Recober, M. 357
Mathieu, J.R. 81
Mehta, C.R. 227
Messéan, A. 299
Molenaar, I.W. 187
Momirović, K. 135
Morin, A.M. 245
Murphy, B.P. 291
Muñoz-Gracia, M.P. 357
Murtagh, F. 239
Naeve, P. 193
Nelder, J.A. 107
Nordbotten, S. 181
Nummi, T. 93, 311
Nurhonen, M. 93
Okur, M.C. 329
Pötzelberger, K. 349

Statistical Data Bases and Survey Processing

Parallel Linear Algebra in Statistical Computations

G. W. Stewart*, Maryland

1. Introduction

The main problem in parallel computation is to get a number of computers to cooperate in solving a single problem. The word "single" is necessary here to exclude the case of processors in a system working on unrelated problems. Ideally we should like to take a problem that requires time T to solve on a single processor and solve it in time T/p on a system consisting of p processors. We say that a system is *efficient* in proportion as it achieves this goal.

In some statistical applications, like bootstrapping or simulations, this goal is easy to achieve. The reason is that the problems divide into independent subtasks, which can be run separately with the results being collected at the end. Although this should be gratifying to statisticians, such problems are not very interesting to people doing research in parallel computing.

Fortunately, there are large problems in regression analysis, signal processing, geodetics, etc. that could potentially benefit from efficient parallelization. For many of these, the heart of the computations is the numerical linear algebra. Consequently this paper is devoted to some of the issues in implementing parallel matrix algorithms.

Just as there is no single general architecture for parallel computers, there is no general theory of parallel matrix algorithms. The same sequential algorithm will be programmed one way on one system and in a completely different way on another. Since the number of potential architectures is very large [1,16], I have chosen to restrict this paper to three, for which commercial systems are available. They are SIMD systems, shared-memory systems, and message-passing systems. We shall treat each of these in the next three sections.

Given this paper's title, its focus on computer architectures requires explanation. When I sat down to write, I intended to stress statistics and parallel matrix computations. But as I proceeded, it became clear that the key to current research and practice lay in the machines themselves. Hence the change in emphasis.

2. SIMD Systems and Systolic Arrays

A *single-instruction, multiple-data* (SIMD [8]) system is a group of (usually simple) processors that execute the same sequence of instructions in lockstep under a global control. This results in a nontrivial computation because the instructions are executed with different data, which can pass from processor to processor to be combined with other data.

*This work was supported in part by the Air Force Office of Sponsored Research under grant AFOSR-82-0078.

Compstat 1988

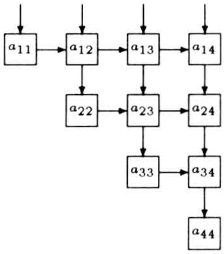
Originally a *systolic array* meant an array of special processors, acting in lockstep, through which data was pumped like blood through the heart [20]. Since the processors are not conceived to be programmable but do have the ability to perform different functions, systolic arrays are not, strictly speaking, SIMD systems. But neither system exists in a pure form, and the distinction has become blurred. Thus the term "systolic algorithm" is now used for algorithms that can be implemented on either system.

As an example, let us consider a systolic array to accumulate the cross-product matrix $A = X^{\mathrm{T}}X$ of an $n \times k$ regression matrix X. If we write A in the form

$$a_{ij} = x_{1i}x_{1j} + x_{2i}x_{2j} + \cdots + x_{ni}x_{nj} \tag{1}$$

we see that the problem is to extract the ith and jth elements from each row of X and add their products to a_{ij}. If we assign a processor to each element of A, then the problem becomes one of making sure that x_{ki} and x_{kj} arrive at the processor responsible for a_{ij} at the same time.

A systolic array for accumulating the upper half of A might be organized as follows.



Here the boxes stand for processors and the arrows indicate how data flows through the array. Each processor is associated with an element of the upper half of the cross-product matrix as shown in the figure. The rows of x are streamed through this processor array as follows. Each element x_{ij} enters the jth column of the array at the top. At each step it moves down one processor until it gets to the diagonal, at which point it begins moving across the jth row and eventually out of the computer.

Figure 1 shows the flow of data in greater detail. The numbers associated with the arrows are the subscripts of the elements of X that are about to enter the processor. When they enter, the processor multiplies them and adds them to the element of A for which it is responsible. Note that the flow of data is such that the appropriate elements of X end up at a processor at the right time.

It is instructive to look at this system from the point of view of an individual processor, say processor `(I,J)`. If we refer to the values in the communication links to the north, south, east and west by **northx**, **southx**, **eastx**, and **westx**, and **a** denotes the current value of a_{IJ}, then a program for the `(I,J)`-processor might read as follows.

```
if (I == J)
   westx = northx;
a = a + northx*westx;
eastx = westx;
southx = northx;
```

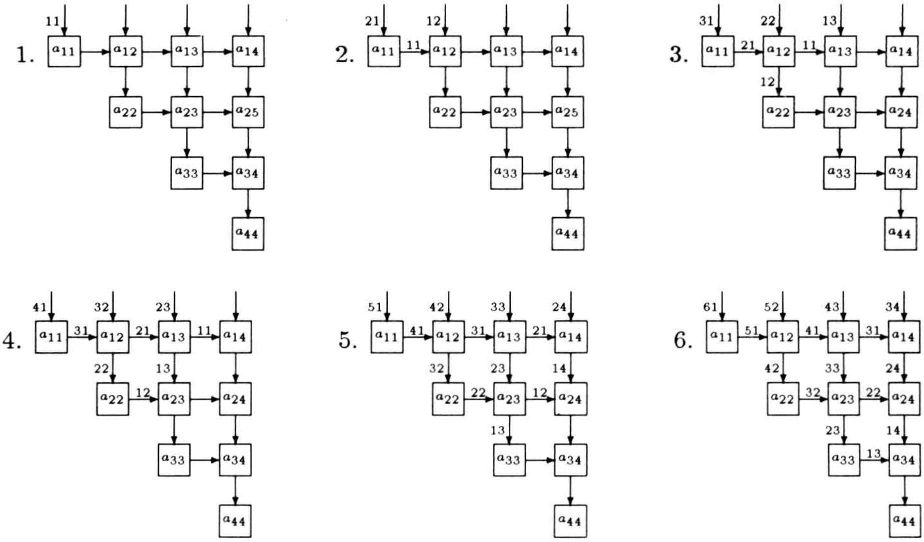

1.
11
a_{11} a_{12} a_{13} a_{14} a_{22} a_{23} a_{25} a_{33} a_{34} a_{44}
2.
21 12 11
a_{11} a_{12} a_{13} a_{14} a_{22} a_{23} a_{25} a_{33} a_{34} a_{44}
3.
31 22 13 21 11 12
a_{11} a_{12} a_{13} a_{14} a_{22} a_{23} a_{24} a_{33} a_{34} a_{44}
4.
41 32 23 31 21 11 22 13 12
a_{11} a_{12} a_{13} a_{14} a_{22} a_{23} a_{24} a_{33} a_{34} a_{44}
5.
51 42 33 24 41 31 21 32 23 14 22 12 13
a_{11} a_{12} a_{13} a_{14} a_{22} a_{23} a_{24} a_{33} a_{34} a_{44}
6.
61 52 43 34 51 41 31 42 33 24 32 22 23 14 13
a_{11} a_{12} a_{13} a_{14} a_{22} a_{23} a_{24} a_{33} a_{34} a_{44}

Figure 1: Flow of Data for $A = X^{\mathrm{T}}X$

Here it is understood that one iteration of the code is performed each time the controller signals the system to advance a step.

There are three things to note about this code. First, it is not strictly SIMD, since the processors on the diagonal behave differently from the others. However, a single program suffices for all processors, a situation sometimes tagged SPMD (single program, multiple data). Second, the code is local. Each processor knows only about its own variables and its input and output. Third, communication is explicit; the code specifies where the data comes from and where it goes to. These characteristics, which programs for systolic systems share with message-passing systems, make for code that is not obviously linked to its task. Certainly it is not easy to recognize equation (1) in the above program. Nonetheless, there is a certain satisfaction in designing systolic algorithms for matrix computations to judge from the number that have been published (e.g., see [3,5,21,29,30]).

When systolic arrays were first proposed, it was hoped that they would provided inexpensive, special-purpose processing for a variety of applications. Things have not worked out this way. The array above is a toy that solves a 4 × 4 problem. To accumulate a 100 × 100 matrix one would require 5,050 processors—nontrivial processors that can perform floating-point arithmetic. Moreover, one cannot afford to build such a big system for a single application; the processors must also be programmable, which increases their complexity. The end result of these considerations is the WARP computer, a linear systolic array of high-performance processors [2]. By all accounts it is effective, but it is neither simple nor cheap.

On the other hand, general purpose SIMD machines have been built and run on a variety of problems. Their main advantage is that they can bring large numbers of simple processors to bear on single problems. They are very effective with simple algorithms that proceed in short, repetitive bursts of computations. Their main disadvantage is their inflexibility. They are tedious to code, even for highly structured problems like computing