INTRODUCTION TO NUMERICAL METHODS FOR PARALLEL COMPUTERS

U. SCHENDEL

INTRODUCTION TO NUMERICAL METHODS FOR PARALLEL COMPUTERS

U. SCHENDEL
Professor of Numerical Mathematics and Computer Science
Freie Universität (Free University) of Berlin

Translator:
B. W. CONOLLY
Professor of Operational Research
Chelsea College, University of London

ELLIS HORWOOD LIMITED
Publishers · Chichester
Halsted Press: a division of
JOHN WILEY & SONS
New York · Chichester · Brisbane · Toronto

First published in 1984 by
ELLIS HORWOOD LIMITED
Market Cross House, Cooper Street, Chichester, West Sussex, PO19 1EB, England

The publisher's colophon is reproduced from James Gillison's drawing of the ancient Market Cross, Chichester.

Distributors:
Australia, New Zealand, South-east Asia:
Jacaranda-Wiley Ltd., Jacaranda Press,
JOHN WILEY & SONS INC.,
G.P.O. Box 859, Brisbane, Queensland 40001, Australia
Canada:
JOHN WILEY & SONS CANADA LIMITED
22 Worcester Road, Rexdale, Ontario, Canada.
Europe, Africa:
JOHN WILEY & SONS LIMITED
Baffins Lane, Chichester, West Sussex, England.
North and South America and the rest of the world:
Halsted Press: a division of
JOHN WILEY & SONS
605 Third Avenue, New York, N.Y. 10016, U.S.A.

British Library Cataloguing in Publication Data
Schendel, U.
Introduction to numerical methods for parallel computers. –
(Ellis Horwood series in mathematics and its applications)
1. Numerical analysis – Data processing
2. Parallel processing (Electronic computers)
I. Title II. Einfuhrung in die parallele Numerik. *English*
519.4 QA297

Library of Congress Card No. 84-10941

ISBN 0-85312-597-X (Ellis Horwood Limited)
ISBN 0-470-20091-X (Halsted Press)

Typeset by Ellis Horwood Limited.
Printed in Great Britain by R.J. Acford, Chichester

Table of Contents

Preface to the English Edition

In offering an English translation of the German original both author and translator hope that a wider audience for this fascinating subject will be captured. It seems still to be true that, whereas hardware descriptions of the advanced modern machines with which this book is concerned have a growing literature, the problems posed by the need to create a suitable body of accompanying mathematical software are not being tackled and documented quite so systematically. The subject invites the attention of all who are interested in the improved computational possibilities afforded by modern technology and we hope that the book will provide an introduction as well as documentation of existing methodology.

The opportunity has been taken to add to and expand some of the material as well as to correct misprints and errors. The additions include an enlarged treatment of eigenvalue procedures in Chapter 4, and two Appendices containing material which was not available when the book was written. The first Appendix supplements Chapters 1 and 2 by providing technical information about a number of particular machines and their performance characteristics in certain benchmark tests. The second Appendix gives a brief account of some parallelised algorithms for nonlinear optimisation.

Translator and author have enjoyed a pleasant and fruitful collaboration which has established a friendship as well as providing a rewarding professional experience. The author is very grateful to Professor B. W. Conolly for the translation of the book and for his improvements and to his colleagues Dipl. Math. J. Brandenburger and Dipl. Math. M. Schyska for reading the manuscript.

Berlin U. Schendel
London B. W. Conolly
1984

Foreword

This book is the result of a series of lectures at the Free University of Berlin, intended to provide an introduction to the principles of parallel numerical analysis. This branch of numerical mathematics has received its impetus and significance from the development of new computer architectures, in particular from the concept of the parallel computer, which has made it possible to tackle larger and more complex numerical problems than hitherto. The procedures available for the solution of such problems must often be 'parallelised' in such a way as to guarantee their successful operation on the corresponding parallel computer.

In this book certain recognised principles are set forth for the development of parallel numerical algorithms. Because no standard computer model for use in the development of parallel numerical methods in a unified theory has yet been developed, so far the various concepts proposed tend to depend heavily on the particular class of computer under discussion. The choice of material is confined by and large to those numerical procedures which form a part of the body of classical and well-tried methods in numerical analysis. On the other hand possible ways are discussed for the development of parallel algorithms for particular problems.

Primary targets of this book are the development of a fundamental understanding of the principles of parallel numerical methods and preliminary guidance in the solution of problems. The mathematical apparatus is presented in such a way that scientists and engineers alike can be initiated into the subject.

I am grateful to Prof. Dr. Feilmeier (Braunschweig) and Prof. Dr. Sameh (Urbana, illinois) for many stimulating discussions, and to my former colleague Dipl. Math. K. Kupfernagel for his contributions and critical remarks. Finally I wish to express my special indebtedness to the Oldenbourg Verlag for their interest and excellent collaboration.

Berlin 1981 U. Schendel

1

Introduction

The model currently used to describe numerical methods in the context of digital computers is the well-known universal model of von Neumann. This is illustrated in Fig. 1.1.

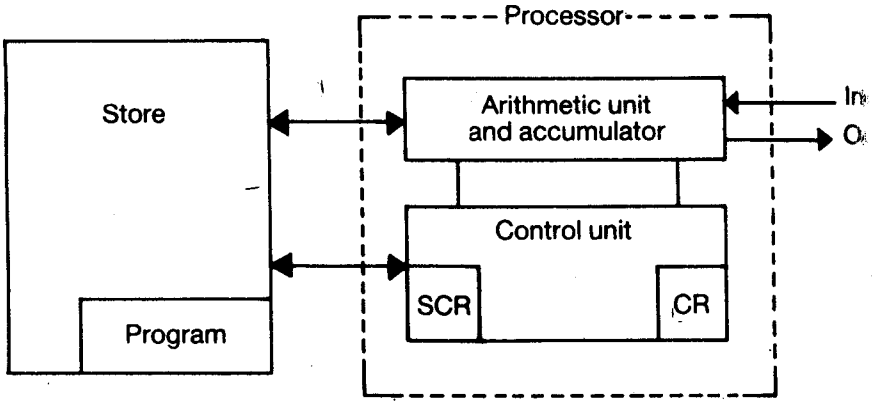

Fig. 1.1 – Universal computer

Legend

SCR = Sequence Control Register (contains the address of the next instruction).

CR = Control Register (contains a copy of the current instruction).

The universal computer possesses the following features:

(1) digital representation of variables;
(2) processing is carried out serially according to the basic operations of arithmetic and logic;
(3) the program is a coded version of the algorithms to be implemented and the data are held in the main store.

All further developments have taken this basic model as their starting point. In particular, the algorithms of numerical analysis are based on this type of computer model and entail a large number of elementary operations. Problems of rounding errors and their propagation, and also questions of numerical stability, turn out to be of great importance [1].

The performance of serial machines has improved enormously during the last twenty years as a result of new technology and improved design. Some features of parallelism have been introduced. These include:

(1) the organisation of input/output channels;
(2) overlapping in the execution of instructions;
(3) interleaved storage techniques.

These developments have nevertheless lagged behind the requirements of numerical methods, even at the programming level.

In the 1960s the possibility of truly parallel digital computers began to be discussed. Parallelism was to be interpreted in the widest sense, and the objective was to offer the user the full capability of such machines. Computers of this kind have indeed been available for some time, examples being the CDC STAR 100, the ILLIAC IV, STARAN, the CRAY-1, Texas Instruments TI ASC, IBM 2938, IBM 360/91, CDC 7600. However, there remains a need for a 'parallel numerical analysis', that is, a body of numerical mathematics which takes specific advantage of the possibilities offered by parallel computers. Of greater theoretical interest is the question of whether there exists a *maximal parallelism* for a given range of problems. Questions of this nature belong to the domain of complexity theory.

There follows a list of some of the pros and cons of parallel computers.

PRO

(1) Increase of computing speed
   (a) conventional structures are at their physical limits
   (b) a single computer with n processors is cheaper than n computers each having a single processor.
(2) Possibility of solving problems still too complex for serial machines. World-wide weather forecasting provides an example.
(3) The solution of problems which are by nature parallel. Operations with vectors and discrete simulation provide examples.
(4) Real time problems. Examples are provided by the processing of typographical data (picture and graphic processing), aerial survey.

CON

(1) Poor utilisation of the machine (a management problem).
(2) Complicated organisation of the data (parallel access).

Great difficulty arises from the fact that at present there is no standard model for parallel systems. This matter will be considered further in Chapter 2 when we examine what are called SIMD and MIMD models.

It is of the utmost importance in parallel numerical analysis to be able to assess the speed gain expected from the operation of p processors in parallel. For this purpose an operational measure S, called the *speed-up* ratio, is introduced. It is defined by

$$S = \frac{\text{Computing time on a serial machine}}{\text{Computing time using a parallel machine}}. \tag{1.1}$$

To obtain a valid result it is essential, for a given problem, to compare the *best* serial algorithm available with the *best* parallel algorithm, even if these are different. According to Stone [2] S has essentially four possible forms:

S	*Examples*
(a) $S = kp$	Matrix calculations, discretisation.
(b) $S = kp/\log_2 p$	Sorting, tridiagonal linear systems, linear recurrence formulae, evaluation of polynomials.
(c) $S = k \log_2 p$	Searching.
(d) $S = k$	Certain nonlinear recurrences and compiler operations.

Here k is a machine-dependent quantity such that $0 < k < 1$ (and is near to unity), and p is the number of processors. The classification of particular types of problem according to one of (a) to (d) is, however, provisional until they can be identified within the framework of complexity theory. The performance limits attainable by a reasonable parallel numerical procedure lie between (b) and (c). The computing time required for an algorithm for a parallel computer is both unknown and machine-dependent. To be able to assess the merits of a parallel algorithm one needs to count the number of time unit steps needed. For this purpose it is convenient to introduce the idealised notion that during such a time unit step exactly one arithmetical operation can be carried out in the parallel mode (see Chapter 4). Let T_p be the number of time unit steps required by a parallel algorithm designed to utilise p ($\geqslant 1$) processors. Then T_1 is the time needed by the corresponding serial algorithm. In order to measure the attributes of parallel operation the following parameters are introduced.

Definitions

The *speed-up* factor of a parallel algorithm by comparison with its serial counterpart is defined by

$$S_p = T_1/T_p \quad (\geqslant 1),$$

(see also 1.1), and the *efficiency* by

$$E_p = S_p/p \quad (\leqslant 1). \tag{1.2}$$

E_p measures the *utilisation* of the parallel machine. The longer processors are idle, or carry out extra calculations introduced through the parallelisation of the problem, the smaller becomes E_p.

To compare two parallel algorithms for the same problem the following measure of effectiveness F_p is introduced:

$$F_p = S_p/C_p \ , \tag{1.3}$$

where

$$C_p = pT_p \tag{1.4}$$

measures the 'cost' of the algorithm. Note that

$$F_p = S_p/(pT_p) = E_p/T_p = E_pS_p/T_1 \leqslant 1 \ . \tag{1.5}$$

F_p is thus a measure both of speed-up and efficiency. A parallel algorithm can accordingly be regarded as effective if it maximises F_p.

The following simple example is given for illustration. Suppose that it is required to form the sum

$$A = \sum_{1}^{16} a_i \ .$$

To carry this out with a single processor sequentially requires 15 additions. If we take the number of additions as a measure of the time needed, then normalising by assuming the time for a single addition to be the unit, we have $T_1 = 15$. If two processors were available we would form the two sums

$$b_1 = a_1 + a_2 + \ldots + a_8 \ , \qquad b_2 = a_9 + a_{10} + \ldots + a_{16}$$

simultaneously, requiring seven time units, and then form

$$c = b_1 + b_2$$

at the next stage, requiring a further time unit. Thus A would be obtained in $T_2 = 7 + 1 = 8$ time units. If now we had three processors A could be formed in the following three stages:

$$b_1 = a_1 + a_2 + a_3 + a_4 + a_5, \quad b_2 = a_6 + a_7 + a_8 + a_9 + a_{10},$$

$$b_3 = a_{11} + a_{12} + a_{13} + a_{14} + a_{15};$$

$$c_1 = b_1 + b_2, \quad c_2 = b_3 + a_{16};$$

$$d_1 = c_1 + c_2 = A \ .$$

This requires $T_3 = 4 + 1 + 1 = 6$ time units. Given four and eight processors the following procedures and their corresponding times are feasible.

$p = 4$

$$\left.\begin{aligned} &b_1 = a_1 + \ldots + a_4, \quad b_2 = a_5 + \ldots + a_8, \quad b_3 = a_9 + \ldots + a_{12}, \\ &b_4 = a_{13} + \ldots + a_{16} \end{aligned}\right\} \quad (3)$$

$$c_1 = b_1 + b_2, \quad c_2 = b_3 + b_4 \quad (1)$$

$$d_1 = c_1 + c_2 \quad (1)$$

$$T_4 = 5.$$

The numbers in ( ) mean the needed time units.

$p = 8$

$$b_1 = a_1 + a_2, \quad b_2 = a_3 + a_4, \ldots, b_8 = a_{15} + a_{16} \quad (1)$$

$$c_1 = b_1 + b_2, \quad c_2 = b_3 + b_4, \ldots, c_4 = b_7 + b_8 \quad (1)$$

$$d_1 = c_1 + c_2, \quad d_2 = c_3 + c_4 \quad (1)$$

$$A = d_1 + d_2 \quad (1)$$

$$T_8 = 4.$$

A table of the performance measures can now be constructed

p	T_p	C_p	S_p	E_p	$F_pT_1=S_pE_p$
1	15	15	1	1	1
2	8	16	1.88	0.94	1.76
3	4	18	2.5	0.83	2.08
4	5	20	3	0.75	2.25
8	4	32	3.75	0.47	1.76

The table shows that with increasing p, S_p increases steadily while E_p decreases. F_pT_1, however, has a maximum when p=4 which indicates that p=4 is the optimal choice of number of processors for this calculation. However, as we saw, the algorithms are different in detail for each p and the variations in S_p, E_p and F_p are partly the consequence of different parallel algorithms for the same problem. The parameters introduced above give one measure for the assessment of a parallel algorithm. Other aspects for consideration are stability and the analysis of errors. Of the greatest importance, however, is to recognise which problems already possess a parallel character, and which can be 'parallelised'.

2

Possible Computer models

Considerable difficulties surround the problem of formulating a standard machine model for parallel numerical methods, yet such a model is needed for theoretical purposes as well as to provide a background for the the development of algorithms.† The situation is quite different from that met in classical numerical analysis where von Neumann's universal computer model provides a basis. Figure 2.1 shows a general model of a parallel computer. This is, however, still too general for detailed statements to be made concerning the functioning of the computer. In particular it is too general for the development of algorithms. For further complementary discussion see Appendix 1.

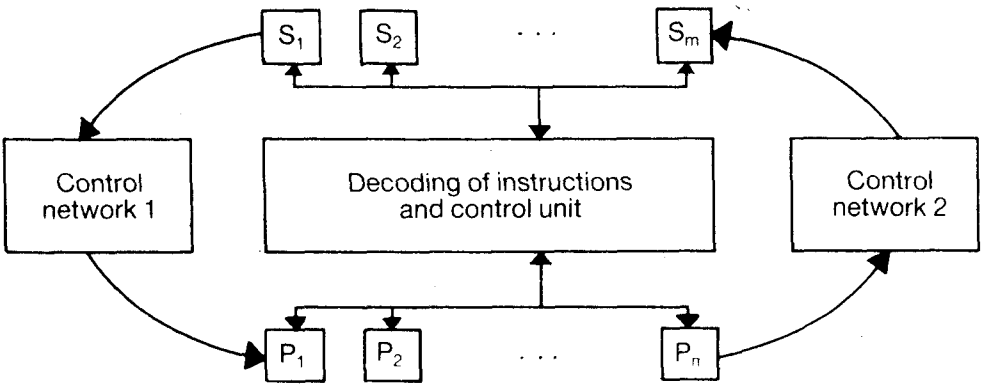

Fig. 2.1 – General configuration of a parallel computer with different levels of Parallelism.
S: stores; P: processors

Note: Parallelism is possible
(1) within the control unit;
(2) among the processors;
(3) among the stores;
(4) in the data central networks.

† Useful discussions of modern high performance machines are given by Ibbett [3], Zakharov [50] and Hockney [51]. This is, however, somewhat technical for a beginner.