Lecture Notes in Computer Science 771
Edited by G. Goos and J. Hartmanis

Gerald Tomas Christoph W. Ueberhuber

Visualization of Scientific Parallel Programs

Springer-Verlag
Berlin Heidelberg New York
London Paris Tokyo
Hong Kong Barcelona
Budapest

Series Editors

Gerhard Goos
Universität Karlsruhe
Postfach 69 80
Vincenz-Priessnitz-Straße 1
D-76131 Karlsruhe, Germany

Juris Hartmanis
Cornell University
Department of Computer Science
4130 Upson Hall
Ithaca, NY 14853, USA

Authors

Gerald Tomas
Christoph W. Ueberhuber
Institute for Applied and Numerical Mathematics, Technical University Vienna
Wiedner Hauptstraße 8-10/115, A-1040 Wien, Austria

CR Subject Classification (1991): D.2.2, D.1.3, D.1.7, G.1.0, G.1.4, G.1.7, G.4

ISBN 3-540-57738-6 Springer-Verlag Berlin Heidelberg New York
ISBN 0-387-57738-6 Springer-Verlag New York Berlin Heidelberg

CIP data applied for

Typesetting: Camera-ready by author
SPIN: 10131031 45/3140-543210 - Printed on acid-free paper

Preface

The substantial effort of parallel programming is only justified if the resulting codes are adequately efficient. In this sense, all types of performance tuning are extremely important to parallel software development. With parallel programs, performance improvements are much more difficult to achieve than with conventional (sequential) programs. One way to overcome this inherent difficulty is to bring in *graphical* tools.

When trying to visualize parallel programs, one is faced with a vast amount of relevant literature consisting of more than 100 articles in journals or conference proceedings and dozens of software systems.

This book pursues two major goals: first, to cover recent developments in parallel program visualization techniques and tools, most of which have not yet been dealt with by text books; second, to demonstrate the application of specific visualization techniques and software tools to scientific parallel programs.

For this purpose, two prototypical problem areas have been chosen: (i) *solution of initial value problems of ordinary differential equations* – a notoriously difficult problem with respect to parallelization, and (ii) *numerical integration* – a problem which is seemingly easy to parallelize. In these two fields the advantages of parallel program visualization are demonstrated. One representative software system – PARAGRAPH – is used to show how visualization techniques can contribute to experimental performance assessment and improvement.

Synopsis

Part I of this book is a general section which introduces the topic of parallel program visualization and gives an overview of current techniques and available software tools.

Part II describes a particular class of parallel methods for the numerical solution of initial value problems of ordinary differential equations.

Iterated Defect Correction (IDeC) is an acceleration technique which iteratively improves numerical approximations to the solution of differential equations. The initial numerical approximation is obtained by using a (one-step or multi-step) discretization method.

The family of IDeC methods allows various kinds of parallelization. The most important design decision has to be made with respect to load distribu-

tion. The calculations associated with the basic method and the solution of the neighboring problems can be allocated to the available processors in different ways. The investigation of the relative merits of these variants using the parallel program visualization system PARAGRAPH is described in Part II of this book.

Part III deals with parallel integration algorithms. These algorithms have attracted the interest of researchers ever since the early days of multiprocessor systems. This interest can be attributed to the beneficial property of additivity with respect to the partitioning of the region of integration into pairwise disjoint subregions. Nevertheless, scalable parallel integration algorithms with high efficiency are not easy to obtain. This is particularly true for problem adaptive algorithms.

The most troublesome feature of the otherwise attractive globally adaptive integration schemes is their inherently sequential nature. Major algorithmic modifications and extensions are necessary to make efficient implementations possible on parallel computers. Even more significant is the effort needed to obtain satisfactory performance over a wide range of target machines.

Parallel program visualization helps to find satisfactory load distribution schemes which are instrumental to the utilization of the available power of a parallel computer. In particular, dynamic load distribution mechanisms are often much easier to assess when their effect can be studied in pictorial representations.

Acknowledgements

We would like to thank our colleagues Roman Augustyn, Michael Karg and Arnold Krommer at the Technical University of Vienna for having produced some of the material included in Parts II and III.

We are also grateful to Michael Heath (University of Illinois) and Martin Schönhacker (Technical University of Vienna) for their valuable comments on the manuscript.

We wish to thank the members of the Institute for Statistics and Computer Science of the University of Vienna, especially Gabriele Kotsis, for providing many of the abstracts in the bibliography.

Finally, we wish to express our gratitude to the Austrian Science Foundation for continued financial support under Grant No. S 5304–PHY.

January 1994 GERALD TOMAS and CHRISTOPH UEBERHUBER

Contents