

Lecture Notes in Computer Science

771

Edited by G. Goos and J. Hartmanis

Advisory Board: W. Brauer D. Gries J. Stoer



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

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, re-use of illustrations, recitation, broadcasting, reproduction on microfilms or in any other way, and storage in data banks. Duplication of this publication or parts thereof is permitted only under the provisions of the German Copyright Law of September 9, 1965, in its current version, and permission for use must always be obtained from Springer-Verlag. Violations are liable for prosecution under the German Copyright Law.

© Springer-Verlag Berlin Heidelberg 1994
Printed in Germany

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.

Contents

I	Parallel Program Visualization	1
1	Introduction	3
1.1	Debugging of Parallel Programs	4
1.2	Approaches to Parallel Debugging	5
1.3	Performance Tuning	6
1.4	Visualization of Parallel Programs	7
2	Visualization Tools	9
2.1	Types of Visualization	9
2.2	Sequential and Parallel Visualization	10
2.3	Source Code Modification	11
2.4	Purpose of Visualization	12
2.4.1	Algorithm Animation	12
2.4.2	Visualization of Program Behavior	12
2.4.3	Visualization of Performance	13
2.4.4	Program Creation	14
2.4.5	Debugging	14
2.5	Time of Visualization	15
2.5.1	Real-Time Visualization	15
2.5.2	Post-mortem Visualization	15
2.6	Kind of Visualization	15
2.6.1	Visualization of Algorithms	15
2.6.2	Interconnection Topology	16
2.6.3	Mapping	16
2.6.4	Data Structures	16
2.6.5	Program Code	17
2.6.6	Program Behavior	17
2.6.7	Performance	18

3	Design Goals and Techniques	19
3.1	Design Goals	19
3.1.1	Ease of Understanding	19
3.1.2	Ease of Use	19
3.1.3	Portability	20
3.2	Demands and Suggestions	20
3.2.1	Multiple Views	20
3.2.2	State Cues	20
3.2.3	Static History	21
3.2.4	Amount of Input Data	21
3.2.5	Choosing Proper Input Data	21
3.2.6	Color Techniques	21
3.2.7	Audio Techniques	21
3.2.8	Scripts	22
3.3	Role of the Environment	22
3.4	Handling Information	22
3.5	Levels of Analysis	23
3.6	Continuous Display	23
3.7	Measurable Parameters	23
3.8	Visualization Primitives	24
3.9	Display Types	24
4	Available Software	27
4.1	BALSA	27
4.1.1	Animation of an Algorithm	27
4.1.2	User Facilities	28
4.2	Descendants of BALSA	28
4.2.1	BALSA-II	28
4.2.2	ZEUS	29
4.2.3	PASTIS	29
4.3	PARAGRAPH	30
4.4	TRACEVIEW	32
4.4.1	Session Management	32
4.4.2	Trace Management	33
4.4.3	View Management	33
4.4.4	Display Management	33
4.5	MAP and SHMAP	34
4.6	PIE	35
4.6.1	Modular Programming Metalanguage	35
4.6.2	The Program Constructor	36
4.6.3	The Implementation Assistant	37

4.6.4	Visualization using PIE	38
4.6.5	Future Plans	38
4.7	TOPSYS	39
4.7.1	Design Goals	39
4.7.2	TOPSYS Structure	39
4.7.3	Interface Definitions	40
4.7.4	Visualization Facilities	40
4.8	VIPS	41
4.8.1	Improved Version of VIPS	42
4.9	SMILI	43
4.10	Summary	45
5	PARAGRAPH	47
5.1	Execution Control	47
5.2	Utilization Displays	48
5.3	Communication Displays	52
5.4	Task Displays	55
5.5	Other Displays	57
5.6	Application-specific Displays	61
5.7	Options	61
II	Visualization of Parallel IDeC Methods	63
6	Parallel IDeC Methods	67
6.1	Introduction	67
6.2	The IDeC Procedure	68
6.2.1	The Defect Correction Principle	68
6.2.2	IDeC Meta-Algorithm	69
6.2.3	An IDeC Method for Parallelization	86
6.3	Parallelization of IDeC Methods	87
6.3.1	Categories of Parallelism	87
6.3.2	Principles of Parallelization	88
6.3.3	Parallelization of the IDeC Procedure	93
6.3.4	Step Size Control and its Effects	95
6.4	Load Distribution	97
6.5	IDeC Variant with Parallel Time Steps	99
6.5.1	Pros and Cons of IDeC _{time} Methods	100
6.6	IDeC Variant with Parallel Defect Corrections	101
6.6.1	Pros and Cons of IDeC _{order} Methods	104
6.7	IDeC Variant with Adaptive Parallel Defect Corrections	105

7	Performance Modelling and Evaluation of // IDeC Methods	107
7.1	Assessment of Parallel Algorithms	107
7.1.1	Principles of Algorithm Assessment	107
7.1.2	Performance Evaluation Criteria	110
7.1.3	Simulation of IDeC Methods	112
7.2	IDeC Variant with Parallel Time Steps	115
7.2.1	Validation	120
7.3	IDeC Variant with Parallel Defect Corrections	122
7.3.1	Validation	127
8	Representative Target Machines	129
8.1	Sequent Balance B21	129
8.2	Intel Hypercube iPSC/860	130
9	Trace File	131
9.1	PICL	131
9.1.1	Low-Level Primitives	132
9.1.2	High-Level Communication Routines	135
9.1.3	Execution Tracing	135
9.2	Trace File Generation	136
10	IDeC-Specific Displays	139
10.1	Display Categories	140
11	Shared Memory IDeC Methods	143
11.1	IDeC _{order} Methods	144
11.1.1	Optimal Parameters	144
11.1.2	Degenerated Parameters	153
11.2	IDeC _{time} Methods	160
11.2.1	Optimal Parameters	160
11.2.2	Degenerated Parameters	169
11.3	IDeC _{adaptive} Methods	177
11.3.1	Optimal Parameters	177
11.3.2	Degenerated Parameters	186
12	Distributed Memory IDeC Methods	191
12.1	IDeC _{order} Methods	192
12.1.1	Optimal Parameters	192
12.1.2	Degenerated Parameters	200

12.2	IDE _{C_{time}} Methods	206
12.2.1	Optimal Parameters	206
12.2.2	Degenerated Parameters	212
12.3	IDE _{C_{adaptive}} Methods	218
III	Visualization of Parallel Integration Methods	221
13	Parallel Integration	225
14	Simulated Target Machines	229
15	Trace File	231
15.1	DEFINE_USER_GANTT	231
15.2	DEFINE_USER_CLASSES	232
15.3	USER_GANTT	234
16	Integration-Specific Displays	235
16.1	Display Categories	236
17	Visualization of Parallel Integration Algorithms	237
17.1	Basic Algorithm	238
17.2	Message Combining	243
17.3	Reducing Idle Times	246
17.4	Choice of Granularity	250
	Bibliography	261
	Index	305