Lecture Notes in Computer Science 1220
Edited by G. Goos, J. Hartmanis and J. van Leeuwen

Peter Brezany

Input/Output Intensive Massively Parallel Computing

Language Support, Automatic Parallelization, Advanced Optimization, and Runtime Systems

Springer

Author

Peter Brezany
Institute for Software Technology and Parallel Systems
University of Vienna
Liechtensteinstrasse 22, A-1090 Vienna, Austria
E-mail: brezany@par.univie.ac.at

Cataloging-in-Publication data applied for

Die Deutsche Bibliothek - CIP-Einheitsaufnahme

Brezany, Peter:
Input output intensive massively parallel computing : language support, automatic parallelization, advanced optimization, and runtime sytems / Peter Brezany. - Berlin ; Heidelberg ; New York ; Barcelona ; Budapest ; Hong Kong ; London ; Milan ; Paris ; Santa Clara ; Singapore ; Tokyo : Springer, 1997
(Lecture notes in computer science ; Vol. 1220)
ISBN 3-540-62840-1 kart.

CR Subject Classification (1991): D.3, D.1.3, D.4.2, C.1.2, F.1.2, B.3.2, D.4.4, G.1

ISSN 0302-9743
ISBN 3-540-62840-1 Springer-Verlag Berlin Heidelberg New York

Typesetting: Camera-ready by author
SPIN 10549551 06/3142 – 5 4 3 2 1 0 Printed on acid-free paper

To my wife Jarmila, children Zuzana and Jozef, my mother Zuzana, and in memory of my father Jozef

Preface

Massively parallel processing is currently the most promising answer to the quest for increased computer performance. This has resulted in the development of new programming languages, and programming environments and has significantly contributed to the design and production of powerful massively parallel supercomputers that are currently based mostly on the distributed-memory architecture.

Traditionally, developments in high-performance computing have been motivated by applications in which the need for high computational power clearly dominated the requirements put on the input/output performance. However, the most significant forces driving the development of high-performance computing are emerging applications that require a supercomputer to be able to process large amounts of data in sophisticated ways. Hardware vendors responded to these new application requirements by developing highly parallel massive storage systems.

However, after a decade of intensive study, the effective exploitation of massive parallelism in computation and input/output is still a very difficult problem. Most of the difficulties seem to lie in programming existing and emerging complex applications for execution on a parallel machine.

The efficiency of concurrent programs depends critically on the proper utilization of specific architectural features of the underlying hardware, which makes automatic support of the program development process highly desirable. Work in the field of programming environments for supercomputers spans several areas, including the design of new programming languages and the development of runtime systems that support execution of parallel codes and supercompilers that transform codes written in a sequential programming language into equivalent parallel codes that can be efficiently executed on a target machine. The focus of this book is just in these areas; it concentrates on the automatic parallelization of numerical programs for large-scale input/output intensive scientific and engineering applications. The principles and methods that are presented in the book are oriented towards typical distributed-memory architectures and their input/output systems.

The book addresses primarily researchers and system developers working in the field of programming environments for parallel computers. The book will also be of great value to advanced application programmers wishing to gain insight into the state of the art in parallel programming.

Since Fortran plays a dominant role in the world of high-performance programming of scientific and engineering applications, it has been chosen as the basis for the presentation of the material in the text.

For full understanding of the contents of the book it is necessary that the reader has a working knowledge of Fortran or a similar procedural high-level programming language and a basic knowledge of machine architectures and compilers for sequential machines.

The book's development In writing this book, I utilized the results of my work achieved during research and development in the European Strategic Program for Research and Development in Information Technology (ESPRIT), in particular, ESPRIT Projects GENESIS, PPPE, and PREPARE. Most of the methods and techniques presented in the book have been successfully verified by a prototype or product implementation or are being applied in on-going projects. Topics related to parallel input/output have been the basis for the proposals of new research projects that start at the University of Vienna this year.

The material of the book has been covered in courses at the University of Vienna given to students of computer science and in the Advanced Course on Languages, Compilers, and Programming Environments given to advanced developers of parallel software.

Contents of the book Each chapter begins with an overview of the material covered and introduces its main topics with the aim of providing an overview of the subject matter. The concluding section typically points out problems and alternative approaches, discusses related work, and gives references. This scheme is not applied if a chapter includes extensive sections. In this case, each section is concluded by a discussion of related work. Some sections present experimental results from template codes taken from real applications, to demonstrate the efficiency of the techniques presented.

Chapter 1 provides motivation, a brief survey of the state of the art in programming distributed-memory systems, and lists the main topics addressed in the book. Input/Output requirements of the current Great Challenge applications are illustrated in three examples which are both I/O and computational intensive: earthquake ground motion modeling, analysis of data collected by the Magellan spacecraft, and modeling atmosphere and oceans.

Chapter 2 specifies a new parallel machine model that reflects the technology trends underlying current massively parallel architectures. Using this machine model, the chapter further classifies the main models used for programming distributed-memory systems and discusses the programming style associated with each model.

The core of the book consists of chapters 3–7. While the first chapter deals with programming language support, the subsequent three chapters show how programs are actually transformed into parallel form and specify requirements on the runtime system. Chapter 7 develops new concepts for an advanced runtime support for massively parallel I/O operations.

Chapter 3 describes Vienna Fortran 90, a high-performance data-parallel language that provides advanced support both for distributed computing and the operations on files stored in massively parallel storage systems. In this chapter the presentation is mainly focused on the language extensions concerning parallel I/O.

Chapter 4 first describes the principal tasks of automatic parallelization of regular and irregular in-core programs and then addresses several important optimization issues. In-core programs are able to store all their data in main memory.

Chapter 5 deals with basic compilation and optimizations of explicit parallel I/O operations inserted into the program by the application programmer.

Chapter 6 treats the problem of transforming regular and irregular out-of-core Vienna Fortran 90 programs into out-of-core message-passing form. Out-of-core programs operate on significantly more data (large data arrays) that can be held in main memory. Hence, parts of data need to be swapped to disks.

Compilation principles and methods are presented in chapters 5 and 6 in the context of the *VFCS (Vienna Fortran Compilation System)*.

Chapter 7 proposes an advanced runtime system referred to as *VIPIOS (VIenna Parallel Input/Output System)* which is based on concepts developed in data engineering technology.

Chapter 8 (conclusion) presents some ideas about the future development of programming environments for parallel computer systems.

Acknowledgements Many people deserve thanks for their assistance, encouragement, and advice. In particular, I thank Hans Zima for giving me the opportunity to work on the ambitious research and development projects and for his encouragement. I am grateful to Michael Gerndt (KFA Jülich), Erich Schikuta, and Thomas Mück (both from the Department for Data Engineering of the University of Vienna), Joel Saltz (University of Maryland), and Alok Choudhary (Northwestern University, Evanston) for many stimulating discussions and suggestions, and to Viera Sipkova for the implementation of several parallelization methods introduced in this book within the Vienna Fortran Compilation System. Thanks also go to Minh Dang and Ka Heng Wong for carrying out many performance measurements to demonstrate the efficiency of the techniques presented in the book.

It is also a pleasure to thank colleagues from GMD Bonn, GMD FIRST Berlin, GMD Karlsruhe, Parsytec Aachen, TNO Delft, ACE Amsterdam, TU Munich, INRIA Rennes, Steria Paris, University of Liverpool, and University of Southhampton with whom I have cooperated in the ESPRIT projects GENESIS, PREPARE, and PPPE.

January 1997 *Peter Brezany*

Table of Contents