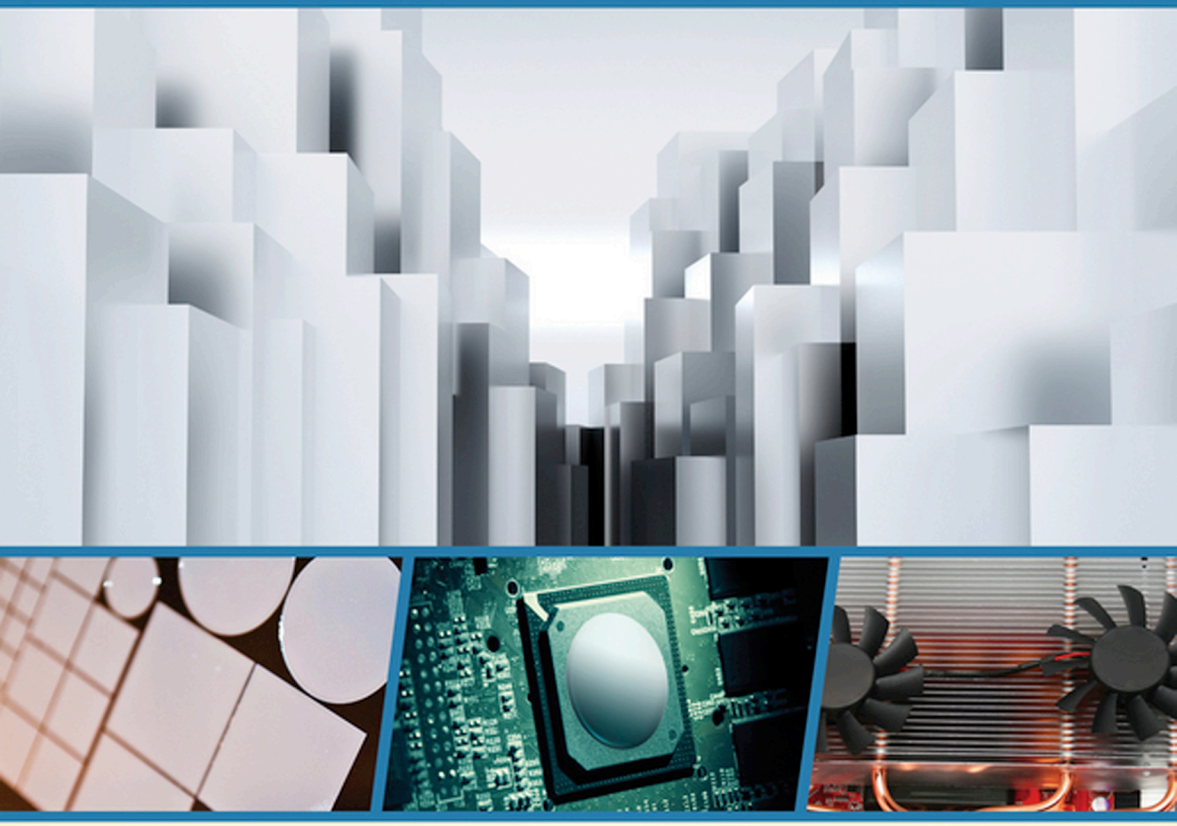
Wiley Series on Parallel and Distributed Computing

Albert Y. Zomaya, Series Editor



High-Performance Computing on Complex Environments

EDITED BY

Emmanuel Jeannot & Julius Žilinskas

WILEY

High-Performance Computing on Complex Environments

WILEY SERIES ON PARALLEL AND DISTRIBUTED COMPUTING

Series Editor: Albert Y. Zomaya

A complete list of titles in this series appears at the end of this volume.

High-Performance Computing on Complex Environments

Emmanuel Jeannot
Inria

Julius Žilinskas
Vilnius University

WILEY

Published by John Wiley & Sons, Inc., Hoboken, New Jersey.
Published simultaneously in Canada.

Library of Congress Cataloging in Publication Data:

Jeannot, Emmanuel.
High performance computing on complex environments / Emmanuel Jeannot, Julius Zilinskas.
pages cm
Includes bibliographical references and index.
ISBN 978-1-118-71205-4 (cloth)
1. High performance computing. I. Žilinskas, J. (Julius), 1973- II. Title.
QA76.88.J43 2014
004.1′1–dc23

2013048363

High-Performance Computing on Complex Environments / Emmanuel Jeannot and Julius Žilinskas
Printed in the United States of America.

10 9 8 7 6 5 4 3 2 1

To our colleague Mark Baker

Contents

Contributors

Alejandro Álvarez-Melcón, Technical University of Cartagena, Cartagena, Spain
Hamid Arabnejad, Universidade do Porto, Porto, Portugal
Henri E. Bal, VU University, Amsterdam, The Netherlands
Ranieri Baraglia, National Research Council of Italy, Pisa, Italy
Jorge G. Barbosa, Universidade do Porto, Porto, Portugal
Robert Basmadjian, Passau University, Passau, Germany
Gabriele Capannini, D&IT Chalmers, Göteborg, Sweden
Jesús Carretero, Universidad Carlos III of Madrid, Madrid, Spain
Raimondas Čiegis, Vilnius Gediminas Technical University, Vilnius, Lithuania
David Clarke, University College Dublin, Dublin, Ireland
Andrea Clematis, IMATI CNR, Genoa, Italy
Georges Da Costa, Toulouse University, Toulouse, France
Daniele D'Agostino, IMATI CNR, Genoa, Italy
Emanuele Danovaro, IMATI CNR, Genoa, Italy
Matjaž Depolli, Jožef Stefan Institute, Ljubljana, Slovenia
Kiril Dichev, University College Dublin, Dublin, Ireland
Niels Drost, Netherlands eScience Center, Amsterdam, The Netherlands
Jose J. Fernandez, National Centre for Biotechnology, National Research Council (CNB-CSIC), Madrid, Spain
Marc E. Frincu, West University of Timisoara, Timisoara, Romania
Javier Garcia, Universidad Carlos III of Madrid, Madrid, Spain
Ester M. Garzon, University of Almería, Almería, Spain
Domingo Giménez, University of Murcia, Murcia, Spain
Arturo Gonzalez-Escribano, Universidad de Valladolid, Valladolid, Spain
Torsten Hoefler, ETH Zürich, Zürich, Switzerland

José Ignacio Agulleiro, University of Almería, Almería, Spain
Aleksandar Ilic, Technical University of Lisbon, Lisbon, Portugal
Florin Isaila, Universidad Carlos III of Madrid, Madrid, Spain
Emmanuel Jeannot, Inria Bordeaux Sud-Ouest, Talence, France
Konstantinos Karaoglanoglou, Aristotle University of Thessaloniki, Thessaloniki, Greece
Helen Karatza, Aristotle University of Thessaloniki, Thessaloniki, Greece
Gabor Kecskemeti, University of Innsbruck, Innsbruck, Austria
Attila Kertesz, MTA SZTAKI Computer and Automation Research Institute, Budapest, Hungary
Timo van Kessel, VU University, Amsterdam, The Netherlands
Gregor Kosec, Jožef Stefan Institute, Ljubljana, Slovenia
Lukasz Kuczynski, Czestochowa University of Technology, Czestochowa, Poland
Alexey Lastovetsky, University College Dublin, Dublin, Ireland
Laurent Lefevre, INRIA, LIP Laboratory, Ecole Normale Superieure of Lyon, Lyon, France
Diego R. Llanos, Universidad de Valladolid, Valladolid, Spain
Dimitar Lukarski, Uppsala University, Uppsala, Sweden
Jason Maassen, Netherlands eScience Center, Amsterdam, The Netherlands
Sidi A. Mahmoudi, University of Mons, Mons, Belgium
Pierre Manneback, University of Mons, Mons, Belgium
Attila Cs. Marosi, MTA SZTAKI Computer and Automation Research Institute, Budapest, Hungary
Guillaume Mercier, Bordeaux Polytechnic Institute, Talence, France; Inria Bordeaux Sud-Ouest, Talence, France
Franco Maria Nardini, National Research Council of Italy, Pisa, Italy
Zsolt Nemeth, MTA SZTAKI Computer and Automation Research Institute, Budapest, Hungary
Maya Neytcheva, Uppsala University, Uppsala, Sweden
Ariel Oleksiak, Poznan Supercomputing and Networking Center, Poznan, Poland
Hector Ortega-Arranz, Universidad de Valladolid, Valladolid, Spain
Erencan Ozkan, Ankara University, Ankara, Turkey
Ozcan Ozturk, Bilkent University, Ankara, Turkey
Carlos Pérez-Alcaraz, University of Murcia, Murcia, Spain
Dana Petcu, West University of Timisoara, Timisoara, Romania
José-Ginés Picón, University of Murcia, Murcia, Spain
Jean-Marc Pierson, Toulouse University, Toulouse, France
Fernando D. Quesada, Technical University of Cartagena, Cartagena, Spain
Antonio J. Plaza, University of Extremadura, Caceres, Spain
Tomás Ramírez, University of Murcia, Murcia, Spain
Vladimir Rychkov, University College Dublin, Dublin, Ireland
Frank J. Seinstra, Netherlands eScience Center, Amsterdam, The Netherlands
Fabrizio Silvestri, National Research Council of Italy, Pisa, Italy
Leonel Sousa, Technical University of Lisbon, Lisbon, Portugal

FRÉDÉRIC SUTER, IN2P3 Computing Center, CNRS, IN2P3, Lyon-Villeurbanne, France
YURI TORRES, Universidad de Valladolid, Valladolid, Spain
SULEYMAN TOSUN, Ankara University, Ankara, Turkey
ROMAN TROBEC, Jožef Stefan Institute, Ljubljana, Slovenia
GHISLAIN LANDRY TSAFACK CHETSA, INRIA, LIP Laboratory, Ecole Normale Superieure of Lyon, Lyon, France
NATALIJA TUMANOVA, Vilnius Gediminas Technical University, Vilnius, Lithuania
FRANCISCO VAZQUEZ, University of Almería, Almería, Spain
MARCIN WOZNIAK, Czestochowa University of Technology, Czestochowa, Poland
ROMAN WYRZYKOWSKI, Czestochowa University of Technology, Czestochowa, Poland
ZIMING ZHONG, University College Dublin, Dublin, Ireland
JULIUS ŽILINSKAS, Vilnius University, Vilnius, Lithuania

Preface

High-performance computing (HPC) is an important domain of the computer science field. For more than 30 years, it has allowed finding solutions to problems and enhanced progress in many scientific and industrial areas, such as climatology, biology, geology, and drug design, as well as automobile and aerospace engineering. However, new technologies such as multicore chips and accelerators have forced researchers in the field to rethink most of the advances in the domain, such as algorithms, runtime systems, language, software, and applications.

It is expected that a high-end supercomputer will be able to deliver several hundreds of petaflops (1 petaflop is 10^{15} floating-point operations per second) in 5 years from now. However, this will require mastering several challenges, such as energy efficiency, scalability, and heterogeneity.

Better and efficient parallel computers will enable solving problems at a scale and within a timeframe that has not been reached so far. These modern hierarchical and heterogeneous computing infrastructures are hard to program and use efficiently, particularly for extreme-scale computing. Consequently, none of the state-of-the-art solutions are able to efficiently use such environments. Providing tools for the whole software stack will allow programmers and scientists to efficiently write new program that will use most of the available power of such future complex machines.

COST Action IC0805 "Open European Network for High-Performance Computing on Complex Environments" (ComplexHPC) was devoted to heterogeneous and hierarchical systems for HPC, and is aimed at tackling the problem at every level (from cores to large-scale environments) and providing new integrated solutions for large-scale computing for future platforms. The duration of ComplexHPC Action was May 2009–June 2013. The goal of COST Action was to establish a European research network focused on high-performance heterogeneous computing to address the whole

range of challenges posed by these new platforms, including models, algorithms, programming tools, and applications. Indeed, some of the most active research groups in this area are in Europe. The network has contributed to exchanging information, identifying synergies, and pursuing common research activities, thereby reinforcing the strength of these groups and the leadership of Europe in this field. This book presents the results of COST Action. The chapters are written by expert participants of the Action.

This book is intended for scientists and researchers working in the field of HPC. It will provide advanced information for the readers already familiar with the basics of parallel and distributed computing. It may also be useful for PhD students and early stage researchers in computer science and engineering. It will also be of help to these young researchers to get a deep introduction to the related fields.

This book would not have been possible without the efforts of the contributors in preparing the respective chapters, and we would like to thank them for timely submissions and corrections. We would also like to thank Prof. Albert Zomaya for giving us the opportunity to publish this book in the "Wiley Series on Parallel and Distributed Computing." We would also like to thank Simone Taylor, Director, Editorial Development, John Wiley & Sons, Inc., and the editorial team for their patience and guiding us through the publication of this book. We would also like to thank COST for the support that enabled the publication.

Delft, Netherlands
May, 2013

E. Jeannot and J. Žilinskas

ESF provides the COST Office through an EC contract

COST is supported by the EU RTD Framework programme

COST–the acronym for European Cooperation in Science and Technology–is the oldest and widest European intergovernmental network for cooperation in research. Established by the Ministerial Conference in November 1971, COST is presently used by the scientific communities of 36 European countries to cooperate in common research projects supported by national funds.

The funds provided by COST–less than 1% of the total value of the projects–support the COST cooperation networks (COST Actions) through which, with EUR 30 million per year, more than 30 000 European scientists are involved in research having a total value which exceeds EUR 2 billion per year. This is the financial worth of the European added value which COST achieves.

A "bottom up approach" (the initiative of launching a COST Action comes from the European scientists themselves), "à la carte participation" (only countries interested in the Action participate), "equality of access" (participation is open also to the scientific communities of countries not belonging to the European Union) and "flexible structure" (easy implementation and light management of the research initiatives) are the main characteristics of COST.

As precursor of advanced multidisciplinary research COST has a very important role for the realisation of the European Research Area (ERA) anticipating and complementing the activities of the Framework Programmes, constituting a "bridge" towards the scientific communities of emerging countries, increasing the mobility of researchers across Europe and fostering the establishment of "Networks of Excellence" in many key scientific domains such as: Biomedicine and Molecular Biosciences; Food and Agriculture; Forests, their Products and Services; Materials, Physical and Nanosciences; Chemistry and Molecular Sciences and Technologies; Earth System Science and Environmental Management; Information and Communication Technologies; Transport and Urban Development; Individuals, Societies, Cultures and Health. It covers basic and more applied research and also addresses issues of pre-normative nature or of societal importance.

Web: http://www.cost.eu

PART I

Introduction

1

Summary of the Open European Network for High-Performance Computing in Complex Environments

Emmanuel Jeannot
Inria Bordeaux Sud-Ouest, Talence, France

Julius Žilinskas
Vilnius University, Vilnius, Lithuania

In this chapter, we describe the COST Action IC0805 entitled "Open European Network for High-Performance Computing on Complex Environments." This Action had representation from more than 20 countries and lasted from 2009 to 2013. We outline the scientific focus of this Action, its organization, and its main outcomes. The chapter concludes by presenting the structure of the book and its different chapters.

High-Performance Computing on Complex Environments, First Edition.
Edited by Emmanuel Jeannot and Julius Žilinskas.

1.1 INTRODUCTION AND VISION

In recent years, the evolution and growth of the techniques and platforms commonly used for high-performance computing (HPC) in the context of different application domains has been truly astonishing. While parallel computing systems have now achieved certain maturity thanks to high-level libraries (such as ScaLAPACK) or runtime libraries (such as MPI), recent advances in these technologies pose several challenging research issues. Indeed, current HPC-oriented environments are extremely complex and very difficult to manage, particularly for extreme-scale application problems.

At the very low level, the latest generation CPUs are made of multicore processors that can be general-purpose or highly specialized in nature. On the other hand, several processors can be assembled into a so-called symmetrical multiprocessor (SMP) which can also have access to powerful specialized processors, such as graphics processing units (GPUs), that are now increasingly being used for programmable computing resulting from their advent in the video-game industry, which has significantly reduced their cost and availability. Modern HPC-oriented parallel computers are typically composed of several SMP nodes interconnected by a network. This kind of infrastructure is hierarchical and represents a first class of heterogeneous system in which the communication time between two processing units is different, depending on whether the units are on the same chip, on the same node, or not. Moreover, current hardware trends anticipate a further increase in the number of cores (in a hierarchical way) inside the chip, thus increasing the overall heterogeneity, even more toward building extreme-scale systems.

At a higher level, the emergence of heterogeneous computing now allows groups of users to benefit from networks of processors that are already available in their research laboratories. This is a second type of infrastructure where both the network and the processing units are heterogeneous in nature. Specifically, here the goal is to deal with networks that interconnect a (often high) number of heterogeneous computers that can significantly differ from one another in terms of their hardware and software architecture, including different types of CPUs operating at different clock speeds and under different design paradigms, and also different memory sizes, caching strategies, and operating systems.

At the high level, computers are increasingly interconnected together throughout wide area networks to form large-scale distributed systems with high computing capacity. Furthermore, computers located in different laboratories can collaborate in the solution of a common problem. Therefore, the current trends of HPC are clearly oriented toward extreme-scale, complex infrastructures with a great deal of intrinsic heterogeneity and many different hierarchical levels.

It is important to note that all the heterogeneity levels mentioned above are tightly linked. First of all, some of the nodes in computational distributed environments may be multicore SMP clusters. Second, multicore chips will soon be fully heterogeneous with special-purpose cores (e.g., multimedia, recognition, networking) and not only

GPUs mixed with general-purpose ones. Third, these different levels share many common problems such as efficient programming, scalability, and latency management. Hence, it is very important to conduct research targeting the heterogeneity at all presented hardware levels. Moreover, it is also important to take special care of the scalability issues, which form a key dimension in the complexity of today environment. The extreme scale of this environment comes from every level:

1. *Low Level*: number of CPUs, number of cores per processor;
2. *Medium Level*: number of nodes (e.g., with memory);
3. *High Level*: distributed/large-scale (geography dispersion, latency, etc.);
4. *Application*: extreme-scale problem size (e.g., calculation-intensive or data-intensive).

In 2008, the knowledge on how to efficiently use program or scale applications on such infrastructures was still vague. This was one of the main challenges that researchers wanted to take on. Therefore, at that time, we decided to launch the COST Action for high-performance and extreme-scale computing in such complex environments entitled "*Open European Network for High-Performance Computing in Complex Environments*." The main reasons were as follows:

- There was a huge demand in terms of computational power for scientific and data-intensive applications;
- The architectural advances offered the potential to meet the application requirements;
- None of the state-of-the-art solutions in HPC at that time allowed exploitation to this potential level;
- Most of the research carried out in this area was fragmented and scattered across different research teams without any coordination.

COST[1] was indeed an appropriate framework for the proposed Action. The main goal of this Action was to overcome the actual research fragmentation on this very hot topic by gathering the most relevant European research teams involved in all the scientific areas described above (from the CPU core to the scientific applications) and coordinate their research.

Summarizing, this project within the COST framework allowed us to expect some potential benefits such as high-level scientific results in the very important domain of high-performance and extreme-scale computing in complex environment; strong coordination between different research teams with significant expertise on this subject; a better visibility of the European research in this area; and a strong impact on other scientists and high-performance applications.

[1]European Cooperation in Science and Technology: http://www.cost.eu.

1.2 SCIENTIFIC ORGANIZATION

1.2.1 Scientific Focus

The expected scientific impacts of the project were to encourage the specific community to focus research on hot topics and applications of interest for the EU, to propagate the collaboration of research groups with the industry, to stimulate the formation of new groups in new EU countries, and to facilitate the solution of highly computationally demanding scientific problems as mentioned above. For this, the groups involved in this Action collaborated with several scientific and industrial groups that could benefit from the advances made by this Action, and prompted the incorporation of new groups to the network.

To achieve the research tasks, different leading European research teams participated in the concrete activities detailed in Section 1.3.

1.2.2 Working Groups

Four working groups were set up to coordinate the scientific research:

- numerical analysis for hierarchical and heterogeneous and multicore systems;
- libraries for the efficient use of complex systems with emphasis on computational library and communication library;
- algorithms and tools for mapping and executing applications onto distributed and heterogeneous systems;
- applications of hierarchical-heterogeneous systems.

It is important to note that these working groups targeted vertical aspects of the architectural structure outlined in the previous section. For instance, the Action's goal was to carry out work on numerical analysis at the multicore level, at the heterogeneous system level, as well as at the large-scale level. The last working group (Applications) was expected to benefit from research of the other three groups.

1.3 ACTIVITIES OF THE PROJECT

To achieve the goal of this Action, the following concrete activities were proposed. The main goal was to promote collaboration through science meetings, workshops, schools, and internships. This allowed interchange of ideas and mobility of researchers.

1.3.1 Spring Schools

The goal was to provide young researchers with a good opportunity to share information and knowledge and to present their current research. These schools contributed to the expansion of the computing community and spread of EU knowledge.

1.3.2 International Workshops

The goal of these meetings was to take the opportunity during international conferences to meet the attendees and other researchers by co-locating workshops.

1.3.3 Working Groups Meetings

The scientific work plan was divided among different working groups. Each working group had substantial autonomy in terms of research projects. A leader nominated by the Management Committee led each working group. Members of a given working group met once or twice a year to discuss and exchange specific scientific issues and problems.

1.3.4 Management Committee Meetings

These meetings were devoted to the organization of the network and ensured the scientific quality of the network.

1.3.5 Short-Term Scientific Missions

The goal of short-term scientific missions (STSMs) was to enable visits by early stage researchers to foreign laboratories and departments. This was mainly targeted at young researchers to receive cross-disciplinary training and to take advantage of the existing resources. The goal was to increase the competitiveness and career development of those scientists in this rapidly developing field through cutting-edge collaborative research on the topic.

1.4 MAIN OUTCOMES OF THE ACTION

We believe that this COST Action was a great success. It gathered 26 European countries and 2 non-COST countries (Russia and South Africa). We have held 12 meetings and 2 spring schools. Fifty-two STSMs have been carried out. We have a new FP7 project coming from this Action (HOST). We have edited a book, and more than 100 papers have been published thanks to this Action.

We have set up an application catalog that gathers applications from the Action members. Its goal is to gather a set of HPC applications that can be used as test cases or benchmarks for researchers in the HPC field. The applications catalog is available at https://complexhpc-catalogue.bordeaux.inria.fr.

In total, the Action gathered more than 250 participants over the four years of the project.

We have sent a survey to the Action members. From this survey, it clearly appears that one of the greatest successes of the Action is the continuous strengthening of the network for many of its members both in terms of research teams and research domains. Many STSMs have been done through new network connections. Spring schools are seen as a major success, as they helped many young researchers to share

and exchange knowledge and gain new connections. Many PhD theses have been defended during the course of the Action, and some of the management committee members have been invited on the defense board of some of these PhDs. Moreover, many presentations given during the meeting are considered very useful and have opened new research directions for other attendees.

We had four goals in this Action:

1. to train new generations of scientists in high-performance and heterogeneous computing;
2. to overcome research fragmentation, and foster HPC efforts to increase Europe's competitiveness;
3. to tackle the problem at every level (from cores to large-scale environment);
4. vertical integration to provide new integrated solutions for large-scale computing for future platforms.

Goal 1 has exceeded our expectations. The spring schools have been a great success. We had many STSMs, and the number of early stage researchers attending the meeting was always very high. We had great response from young researchers.

Goal 2 has also been achieved satisfactorily. Thanks to the Action, many joint researches have been carried out, and we have created a nice network of researchers within our Action. Moreover, many top-level publications have been made thanks to the Action.

Goal 3 has also been achieved. We have scientific results that cover the core level and the distributed infrastructure, as well as results that cover the intermediate layers. This is due to the fact that the consortium was made of researchers from different areas. This was very fruitful.

Goal 4 has not been achieved. The main reason is the fact that providing integrated solutions requires more research and development than a COST Action can provide. It goes far beyond the networking activities of COST Action.

1.5 CONTENTS OF THE BOOK

This book presents some of the main results, in terms of research, of the COST Action presented in this chapter. We are very proud to share this with the interested reader. We have structured the book according to the following parts in order to have a good balance between each part:

1. Numerical Analysis for Heterogeneous and Multicore Systems (Chapters 2, 3, and 4);
2. Communication and Storage Considerations in High-Performance Computing (Chapters 5, 6, 7, and 8);
3. Efficient Exploitation of Heterogeneous Architectures (Chapters 9, 10, 11, and 12);
4. CPU + GPU coprocessing (Chapters 13, 14, and 15);

5. Efficient Exploitation of Distributed Systems (Chapters 16 and 17);
6. Energy Awareness in High-Performance Computing (Chapters 18, 19, and 20);
7. Applications of Heterogeneous High-Performance Computing (Chapters 21, 22, and 23).

Chapter 2 discusses the redesign of the iterative solution algorithm in order to efficiently execute them on heterogeneous architectures. Chapter 3 studies the performance of a meshless numerical partial differential equation (PDE) solver, parallelized with OpenMP. The results depend on the way the computations are distributed and the way the cache is used. Chapter 4 presents the development of three parallel numerical algorithms for the solution of parabolic problems on graphs with a theoretical and experimental study of their scalability.

Chapter 5 surveys different techniques for mapping processes to computing units in order to optimize communication cost and reduce execution time. Chapter 6 offers a comprehensive overview of how to implement topology- and performance-aware collective communications. Chapter 7 analyzes the many-core architecture using a new model (K-model) in order to estimate the complexity of a given algorithm designed for such an architecture. Chapter 8 presents a scalable I/O storage system for the hierarchical architecture of Blue Gene computers featuring buffering and asynchronous I/O.

Chapter 9 describes algorithmic techniques for offline scheduling of independent workflows in order to satisfy user's quality of service. Chapter 10 investigates the advantage of using modern heterogeneous architecture for the efficient implementation of the Reed–Solomon erasure code. Chapter 11 analyzes the factors that enable the development of efficient parallel programs on modern many-core parallel architecture. Chapter 12 studies efficient solutions for electromagnetism applications in clusters of CPU + GPU nodes.

Chapter 13 describes how the functional performance model can be used to optimize the performance of scientific applications for heterogeneous and hierarchical platform. Chapter 14 presents algorithms for multilevel load-balancing on multicore and multi-GPU environments. Chapter 15 faces the all-pair shortest path problem for sparse graph. Different scheduling strategies are studied to efficiently solve such problems on heterogeneous systems.

Chapter 16 surveys different resource management systems and scheduling algorithms for HPC for clouds. Chapter 17 discusses different approaches for performing resource discovery in large-scale distributed systems.

Chapter 18 focuses on how to optimize and adapt software solution to improve energy efficiency in the context of HPC application. Chapter 19 studies energy-aware scheduling policies for three scenarios of federated cloud dealing with energy awareness. Chapter 20 explores the use of heterogeneous chip multiprocessors for network security and strategy to improve energy consumption in such contexts.

Chapter 21 describes the "jungle computing paradigm," which consists in gathering a complex hierarchical collection of heterogeneous computing hardware with an application to hyperspectral remote sensing. Chapter 22 presents a new model for image and video processing based on parallel and heterogeneous platforms in order

to improve the performance of the application when dealing with high-definition images. Chapter 23 applies load-balancing techniques to efficiently execute tomographic reconstruction using hybrid GPU + CPU systems.

As you can see, this covers a large spectrum of results and topics on HPC and heterogeneous systems.

We wish you a fruitful and enjoyable time with this book.

ACKNOWLEDGMENT

This publication is supported by COST.

PART II

Numerical Analysis for Heterogeneous and Multicore Systems

2

On the Impact of the Heterogeneous Multicore and Many-Core Platforms on Iterative Solution Methods and Preconditioning Techniques

Dimitar Lukarski and Maya Neytcheva
Uppsala University, Uppsala, Sweden

Computer simulations are now broadly recognized as a third branch of research, complementing theory and experimental work. The significant increase of available computing power has enabled tackling very large scale, challenging, real-life problems and opening new possibilities for revolutionary breakthrough results in science and engineering. At the same time, the complexity of the computer architecture has risen to levels where it is possible to achieve its full computing power only after careful redesigning of existing algorithms and developing novel computational and

High-Performance Computing on Complex Environments, First Edition.
Edited by Emmanuel Jeannot and Julius Žilinskas.

communication strategies. In this chapter, we discuss this issue for a class of methods, broadly used in scientific computations—the iterative solution methods.

2.1 INTRODUCTION

For many years, the potential of available, serial as well as parallel, computer resources has been growing hand in hand with the need to numerically solve increasingly larger models of real-life problems. During the past decades, it has been recognized that, together with theoretical development and laboratory or field experiments, computation has become a third branch of research. Scientific computing is today's driving force behind the progress in the most challenging and demanding problems we attempt to solve. As examples, we mention turbulent combustion with complex chemistry, atmospheric dynamics, laser fusion, medical imaging, detailed modeling of the human heart, and artificial brain simulation with over a million neurons, to name a few.

In recent years, we have been witnessing a change in the means to increase the computational power which has a strong impact on how that power should be utilized via the algorithms used in scientific computations. Therefore, we briefly describe the phases in performing numerical simulations. Consider a complex physical phenomenon, described as a set of, usually coupled, processes that develop in space and time, which we want to study, analyze, and predict. It is assumed that the simulation requires a large amount of computer resources in terms of memory and computation.

The process of performing the numerical simulations can be split into the following steps:

I Mathematical model: Describes the phenomenon continuously in time and space in terms of mathematical relations, most often as coupled ordinary or partial differential equations. These equations depend on various problem parameters, such as thermal conductivity, capacitance, material properties, and so on.

II Discrete model: Because of the high complexity of the continuous model, analytical solutions are in general not available. Therefore, we pose the task to compute the solution in a number of discrete points in time and space, thus discretizing the mathematical model. This can be accomplished using various techniques. Space discretization can be done using finite differences, finite elements, finite volumes, boundary elements, and so on. Similarly, in time, various explicit or implicit time-stepping procedures can be utilized. In addition to the model parameters, here additional discretization parameters are introduced, usually denoted as h in space and τ in time. The discrete model is expressed in terms of linear or nonlinear algebraic systems of equations which have to be solved. Depending on the problem, but also on the discretization techniques, the matrices associated with the algebraic systems can be dense or sparse, symmetric or nonsymmetric, and so on. As nonlinear systems are most

often solved via linearization, we consider from now on only linear systems that are also large and sparse.

III The linear systems arising in Step II have to be solved by a proper solution method—direct or iterative. Because of the targeted large-sized problems and the lesser demands on computer resources, we consider iterative solvers only. The iterative methods may introduce yet other method parameters which increase further the dimension of the parameter space.

IV Computer implementation: To enable computer simulations, the numerical methods have to be implemented on some computer platform.

V Visualization and verification: This step is also of importance, but it is not considered here any further.

When performing numerical simulations, we deal with two major concerns. The first concern is *robustness* with respect to model, discretization, and method parameters. Robustness is understood in the sense that the numerical efficiency of the iterative method chosen in Step III should not depend on changes in the parameters. For example, the number of iterations should not increase uncontrollably when h decreases. The *numerical efficiency*, related to fast convergence rate, can be seen also as an element of the robustness of the method. The second concern is the *efficiency of the implementation* in Step IV. It is based on a programming model (such as shared or distributed memory model), programming language, and a particular computer platform.

It has been recognized that, in order to achieve fast, accurate, and reliable results from computer simulations, sufficient knowledge is required for all the above steps, in particular Steps II, III, and IV, and awareness of the interplay among them. By choosing one or another discretization method, we may influence the structure and the properties of the arising matrices; by choosing a particular solution method we may ensure robustness with respect to the various problem, discretization, and method parameters; but we may sacrifice the amount of internal parallelism. Knowledge about the computer architecture on which the simulations are to be run may influence the choice of the solution method, which in turn has to be combined with the requirements of accuracy and robustness.

With the ongoing radical shift in the computer architecture toward multicore and many-core computational units, the importance of the above arguments becomes even stronger. The new technology based on multicore and many-core devices provides higher performance capabilities both in terms of computational power (GFlop/s) and memory bandwidth (GB/s). The available and easily accessible power enables scientists to tackle larger problems with higher resolution, providing in this way a better understanding of the world.

The radical shift is clearly seen when we compare the supercomputers available 10 years ago with the personal computers of today. All supercomputers in 2003 contained less than 10,000 cores[1]. By the end of 2004, the situation was still the

[1] Top500 http://www.top500.org/statistics/efficiency-power-cores/.

same, with two exceptions. At present, the computer landscape is very different. Not only the Top500 leaders have over 500,000 cores. Currently, NVIDIA delivers GPU (graphical processing unit) cards with more than 2500 cores per device (see GPU NVIDIA K20X[2]). With an improved power supply, four of these cards can be installed in a standard personal computer and thus one can obtain a system with more than 10,000 cores, which is a commodity at our desktop.

In order to achieve fast and reliable performance of the iterative methods, it becomes crucial to reconsider and redesign the implementation of well-known algorithms as well as to gain a deeper insight into what the expected performance is of the most common solution techniques on multicore heterogeneous computer platforms.

The focus of this chapter is to show how iterative methods can be performed efficiently on highly parallel, heterogeneous platforms. We present various methods and examples to show how this can be done, which includes mathematical description, as well as hardware-specific aspects.

2.2 GENERAL DESCRIPTION OF ITERATIVE METHODS AND PRECONDITIONING

We briefly discuss basic iterative techniques as well as two of the most often used projection-based methods—the conjugate gradient (CG) method [1], the generalized minimal residual (GMRES) method [2], and the multigrid (MG) method [3]. We also describe the defect-correction technique [4] as an illustration of an approach particularly suitable for solving linear systems on heterogeneous computers.

2.2.1 Basic Iterative Methods

Consider the solution of the linear system

$$A\mathbf{x} = \mathbf{b}, \tag{2.1}$$

where $A \in \mathbb{R}^{n\times n}$ is a nonsingular matrix, so Equation (2.1) has a unique solution. The matrix A is large and sparse, and therefore the number of nonzero elements, $nnz(A)$, is proportional to the size of the matrix, n; that is, $nnz(A) = O(n)$.

Finding the solution to Equation (2.1) is equivalent to finding the root of the equation

$$\mathbf{b} - A\mathbf{x} = \mathbf{0}. \tag{2.2}$$

One straightforward way to introduce simple iterative methods is to rewrite (2.2) as a fixed-point iteration, namely

$$\begin{aligned} &\text{for some given } \mathbf{x}^{(0)}, \text{ iterate} \\ &\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + (\mathbf{b} - A\mathbf{x}^{(k)}),\ k = 0, 1, \ldots \text{ until convergence.} \end{aligned} \tag{2.3}$$

[2]NVIDIA K20 specification http://www.nvidia.com/object/tesla-servers.html

The computational procedure (2.3) defines a basic stationary iterative method. The computation cost per iteration involves one matrix–vector multiplication and two vector updates, and is clearly $O(n)$. Such a method, however, usually exhibits too slow a convergence, which manifests itself in unacceptably many iterations. In some cases, convergence may not even be achieved.

Aiming at accelerating the convergence of the iterative process has led to the idea to involve some method parameter, replacing the simple iteration (2.3) by

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \tau \mathbf{r}^{(k)}, \quad \text{or} \quad \mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \tau_k \mathbf{r}^{(k)}, \tag{2.4}$$

where $\mathbf{r}^{(k)} = \mathbf{b} - A\mathbf{x}^{(k)}$ is the *residual* at the kth iteration and τ or τ_k are some properly chosen method parameters. In Equation (2.4), the method parameters to tune are scalars. Of course, nothing prevents us from replacing them with a properly chosen matrix, referred to in the sequel as P; thus we consider

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + P^{-1}\mathbf{r}^{(k)}, \ k = 0, 1, \ \ldots \text{ untill convergence.} \tag{2.5}$$

As will be discussed later, P can also vary during the iterative process. For simplicity, now we consider that it is some explicitly given nonsingular matrix.

It is easy to see that Equation (2.5) is obtained by replacing the original system $A\mathbf{x} = \mathbf{b}$ by the transformed system

$$P^{-1}A\mathbf{x} = P^{-1}\mathbf{b},$$

and applying the fixed-point scheme to it. In this case, the iterative scheme becomes

$$\begin{cases} \mathbf{r}^{(k)} = \mathbf{b} - A\mathbf{x}^{(k)}, \\ P\mathbf{d}^{(k)} = \mathbf{r}^{(k)}, \\ \mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \mathbf{d}^{(k)}. \end{cases} \tag{2.6}$$

The scheme (2.6) has a higher computational complexity than that of Equation (2.4), since a solution of a system with the matrix P is required at each iteration. Clearly, the achieved decrease in the number of iterations must be significant enough to compensate for the extra cost per iteration.

We see from Equation (2.6) that the choice $P = A$ would lead to a procedure that converges within one iteration. However, the computational cost would be unacceptably high, similar to that of a direct solution method. Clearly, P should satisfy some conditions so that we can achieve faster convergence, keeping at the same time the overall computational costs of the whole iterative method as low as possible.

The matrix P is referred to as a *preconditioner* to A. We consider next some well-known choices of P, leading to a family of classical iterative methods which are based on the so-called matrix splitting technique.

Intuitively, P has to be related to A. Consider the following splitting of A,

$$A = P - R,$$

where P is nonsingular and R can be seen as an error matrix. Then,

$$P^{-1}A = P^{-1}(P - R) = I - P^{-1}R,$$

where I is the identity matrix of proper order.

The matrix $B = P^{-1}R$ is referred to as the *iteration matrix* and is used in theoretical derivations to show the convergence of the corresponding iterative method, as well as to estimate its rate of convergence (see [5] for details). Using the splitting, we rewrite Equation (2.5) as follows:

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + P^{-1}(\mathbf{b} - A\mathbf{x}^{(k)}) = P^{-1}\mathbf{b} + P^{-1}R\mathbf{x}^{(k)}$$

or

$$P\mathbf{x}^{(k+1)} = R\mathbf{x}^{(k)} + \mathbf{b}.$$

Let A be represented in the following way, $A = D - L - U$ where D, L, and U are the diagonal, the strictly lower triangular, and the strictly upper triangular part of A, respectively. Table 2.1 shows some classical iterative schemes, based on the latter splitting of A.

For more details on the convergence of these methods, refer to [5].

The common characteristic of these methods is the simplicity of their implementation. Here, P is a diagonal or a triangular matrix, and the degree of parallelism is related to the sparsity structure of the underlying matrices.

The bottleneck of these methods is their slow convergence. Their importance has not been lost, however. Today, they are mostly used as subsolvers in more advanced iterative techniques described in Sections 2.2.2 and 2.5. Because of their low arithmetic cost and ease of implementation, they are important ingredients in the so-called projection methods.

2.2.2 Projection Methods: CG and GMRES

The idea behind the projection methods is that the original problem of huge dimension (easily of tens or even hundreds of millions of degrees of freedom) is projected over a subspace of much smaller dimension. An approximate solution is sought in that smaller subspace and then projected back to the original large space. When the

TABLE 2.1 Classical Iterative Schemes Based on Matrix Splitting

Method	P	R	Scheme
Jacobi iteration	D	$L + U$	$D\mathbf{x}^{(k+1)} = (L + U)\mathbf{x}^{(k)} + \mathbf{b}$
Gauss–Seidel backward	$D - U$	L	$(D - U)\mathbf{x}^{(k+1)} = L\mathbf{x}^{(k)} + \mathbf{b}$
Gauss–Seidel forward	$D - L$	U	$(D - L)\mathbf{x}^{(k+1)} = U\mathbf{x}^{(k)} + \mathbf{b}$
SOR	$D - \omega L$	$\omega U + (1 - \omega)D$	$(D - \omega L)\mathbf{x}^{(k+1)} = (\omega U + (1 - \omega)D)\mathbf{x}^{(k)} + \mathbf{b}$