Grid Computing: Software
Environments and Tools

José C. Cunha
CITI Centre,
Department of Computer Science,
Faculty of Science and Technology
Universidade Nova de Lisboa
Quinta da Torre
2829-516 Caparica, Portugal
Email: jcc@di.fct.unl.pt

and

Omer F. Rana
School of Computer Science and Welsh e-Science Centre
Cardiff University, PO Box 916,
Cardiff CF24 3XF, UK
Email: o.f.rana@cs.cardiff.ac.uk
Preface

Grid computing combines aspects from parallel computing, distributed computing and data management, and has been playing an important role in pushing forward the state-of-the-art in computer science and information technologies. There is considerable interest in Grid computing at present, with a significant number of Grid projects being launched across the world. Many countries have started to implement their own Grid computing programmes – such as in the Asia Pacific region (including Japan, Australia, South Korea and Thailand), the European Union (as part of the Framework 5 and 6 programmes, and national activities such as the UK eScience programme), and the US (as part of the NSF CyberInfrastructure and the DDDAS programmes). The rising interest in Grid computing can be seen by the increase in the number of participants at the Global Grid Forum (http://www.gridforum.org/), as well as through regular sessions on this theme at several conferences.

Many existing Grid projects focus on deploying common infrastructure (such as Globus, UNICORE, and Legion/AVAKI). Such efforts are primarily aimed at implementing specialist middleware infrastructure that can be utilized by application developers, without providing any details about how such infrastructure can best be utilized. As Grid computing infrastructure matures, however, the next phase will require support for deploying and developing applications and associated tools and environments which can utilize this core infrastructure effectively. It is therefore important to explore software engineering themes which will enable computer scientists to address the concerns arising from the use of this middleware.

However, approaches to software construction for Grid computing are ad hoc at the present time. There is either deployment of existing tools not really meant for Grid environments, or tools that are not robust – and therefore not likely to be re-used in communities other than those within which they have been developed (examples include specialized libraries for BioInformatics and Physics, for instance). On the other hand, a number of projects are exploring the development of applications using specialist tools and approaches that have been explored within a particular research project, without considering the wider implications of using and deploying these tools. As a consequence, there is little shared understanding of the common needs of software construction, development, deployment and re-use. The main motivation for this book is to help identify what these common themes are, and to provide a series of chapters offering a more detailed perspective on these themes.

Recent developments in parallel and distributed computing: In the past two decades, advances in parallel and distributed computing allowed the development of many applications in Science and Engineering with computational and data intensive requirements. Soon it was realized that there was a need for developing generic software layers and integrated environments which could facilitate the problem solving process, generally in the context of a particular functionality. For
example, such efforts have enabled applications involving complex simulations with visualization and steering, design optimization and application behavior studies, rapid prototyping, decision support, and process control (both from industry and academia). A significant number of projects in Grid computing build upon this earlier work.

Recent efforts in Grid computing infrastructure have increased the need for high-level abstractions for software development, due to the increased complexity of Grid systems and applications. Grid applications are addressing several challenges which had not been faced previously by parallel and distributed computing: large scale systems allowing transparent access to remote resources; long running experiments and more accurate models; increased levels of interaction e.g. multi-site collaboration for increased productivity in application development.

**Distributed computing:** The capability to physically distribute computation and data has been explored for a long time. One of its main goals has been to be able to adapt to the geographical distribution of an application (in terms of users, processing or archiving ability). Increased availability and reliability of the systems architectures has also been successfully achieved through distribution of data and control. A fundamental challenge in the design of a distributed system has been to determine how a convenient trade-off can be achieved between transparency and awareness at each layer of its software architecture. The levels of transparency, as provided by distributed computing systems, has been (and will continue) to change over time, depending on the application requirements and on the evolution of the supporting technologies. The latter aspect is confirmed when we analyze Grid computing systems. Advances in processing and communication technologies have enabled the provision of cost-effective computational and storage nodes, and higher bandwidths in message transmission. This has allowed more efficient access to remote resources, supercomputing power, or large scale data storage, and opened the way to more complex distributed applications. Such technology advances have also enabled the exploitation of more tightly coupled forms of interactions between users (and programs), and pushed forward novel paradigms based on Web computing, Peer-2-Peer computing, mobile computing and multi-agent systems.

**Parallel computing:** The goal of reducing application execution time through parallelism has pushed forward many significant developments in computer system architectures, and also in parallel programming models, methods, and languages. A successful design for task decomposition and cooperation, when developing a parallel application, depends critically on the internal layers of the architecture of a parallel computing system, which include algorithms, programming languages, compilers and runtime systems, operating systems and computer system architectures. Two decades of research and experimentation have contributed to significant speedup improvements in many application domains, by supporting the development of parallel codes for simulation of complex models and for interpretation of large volumes of data. Such developments have been supported by
advanced tools and environments, supporting processing and visualization, computational steering, and access through distinct user interfaces and standardized application programming interfaces.

Developments in parallel application development have also contributed to improvement in methods and techniques supporting the software life cycle, such as improved support for formal specification and structured program development, in addition to performance engineering issues. Component-based models have enabled various degrees of complexity, granularity, and heterogeneity to be managed for parallel and distributed applications — generally by reducing dependencies between different software libraries. For example, simulators and mathematical packages, data processing or visualization tools were wrapped as software components in order to be more effectively integrated into a distributed environment. Such developments have also allowed a clear identification of distinct levels of functionalities for application development and deployment: from problem specification, to resource management and execution support services. Developments in portable and standard programming platforms (such as those based on the Java programming language), have also helped in the handling of heterogeneity and interoperability issues.

In order to ease the computational support for scientific and engineering activities, integrated environments, usually called *Problem-Solving Environments (PSEs)* have been developed for solving classes of related problems in specific application domains. They provide the user interfaces and the underlying support to manage an increasingly complex life cycle of activities for application development and execution. This starts with the problem specification steps, followed by successive refinements towards component development and selection (for computation, control, and visualization). This is followed by the configuration of experiments, through component activation and mapping onto specific parallel and distributed computing platforms (including the set up of application parameters), followed by execution monitoring and control, possibly supported through visualization facilities.

As applications exhibit more complex requirements (intensive computation, massive data processing, higher degrees of interaction), many efforts have been focusing on easing the integration of heterogeneous components, and providing more transparent access to distributed resources available in wide-area networks, through (Web-enabled) portal interfaces.

*Grid computing:* When looking at the layers of a Grid architecture, they are similar to those of a distributed computing system:

1. User interfaces, applications and PSEs.
2. Programming and development models, tools and environments.
3. Middleware, services and resource management.
4. Heterogeneous resources and infrastructure.

However, researchers in Grid computing are pursuing higher levels of transparency, aiming to provide unifying abstractions to the end-user, with single
access points to pools of virtual resources. Virtual resources provide support for launching distributed jobs involving computation, data access and manipulation of scientific instruments, with virtual access to remote databases, catalogues and archives, as well as cooperation based on virtual collaboration spaces. In this view, the main distinctive characteristic of Grid computing, when compared to previous generations of distributed computing systems, is this (more) ambitious goal of providing increased transparency and “virtualization” of resources, over a large scale distributed infrastructure.

Indeed, ongoing developments within Grid computing are addressing the deployment of large scale application and user profiles, supported by computational Grids for high-performance computing, intelligent data Grids for accessing large datasets and distributed data repositories — all based on the general concept of “virtual organizations” which enable resource sharing across organizational boundaries. Recent interest in a “Grid Ecosystem” also places emphasis on the need to integrate tools at different software layers from a variety of different vendors, enabling a range of different solutions to co-exist for solving the same problem. This view also allows a developer to combine tools and services, and enables the use of different services which exist at the same software layer at different times. The availability of suitable abstractions to facility such a Grid Ecosystem still do not exist however.

Due to the above aspects, Grids are very complex systems, whose design and implementation involves multiple dimensions, such as large scale, distribution, heterogeneity, openness, multiple administration domains, security and access control, and dynamic and unpredictable behavior. Although there have been significant developments in Grid infrastructures and middleware, support is still lacking for effective Grid applications development, and to assist software developers in managing the complexity of Grid applications and systems. Such applications generally involve large numbers of distributed, and possibly mobile and intelligent, computational components, agents or devices. This requires appropriate structuring, interaction and coordination methods and mechanisms, and new concepts for their organization and management. Workflow tools to enable application composition, common ways to encode interfaces between software components, and mechanisms to connect sets of components to a range of different resource management systems are also required. Grid applications will access large volumes of data, hopefully relying upon efficient and possibly knowledge-based data mining approaches. New problem-solving strategies with adaptive behavior will be required in order to react to changes at the application level, and changes in the system configuration or in the availability of resources, due to their varying characteristics and behavior. Intelligent expert and assistance tools, possibly integrated in PSEs, will also play an increasingly important role in enabling the user-friendly interfacing to such systems.

As computational infrastructure becomes more powerful and complex, there is a greater need to provide tools to support the scientific computing community to make better use of such infrastructure. The last decade has also seen an unprecedented focus on making computational resources sharable (parallel
machines and clusters, and data repositories) across national boundaries. Significantly, the emergence of Computational Grids in the last few years, and the tools to support scientific users on such Grids (sometimes referred to as “eScience”) provides new opportunities for the scientific community to undertake collaborative, and multi-disciplinary research. Often tools for supporting application scientists have been developed to support a particular community (Astrophysics, Biosciences, etc), a common perspective on the use of these tools and making them more generic is often missing.

Further research and developments are therefore needed in several aspects of the software development process, including software architecture, specification languages and coordination models, organization models for large scale distributed applications, and interfaces to distributed resource management and execution services. The specification, composition, development, deployment, and control of the execution of Grid applications require suitable flexibility in the software life cycle, along its multiple stages, including application specification and design, program transformation and refinement, simulation and code generation, configuration and deployment, and the coordination and control of distributed execution. New abstractions, models and tools are required to support the above stages in order to provide a diversity of functionalities, such as:

- Specification and modelling of the application structure and behavior, with incremental refinement and composition, and allowing reasoning about global functional and non-functional properties.
- Abstractions for the organization of dynamic large scale systems.
- Representation and management of interaction patterns among components and services.
- Enabling of alternative mappings between the layers of the software architecture, supported by pattern or template repositories, that can be manipulated during the software development and execution stages.
- Flexible interaction with resource management, scheduling and discovery services for flexible application configuration and deployment, and awareness to Quality of Service.
- Coordination of distributed execution, with adaptability and dynamic reconfiguration.

Such types of functionalities will provide the foundations for building environments and frameworks, developed on top of the basic service layers that are provided by Grid middleware and infrastructures.

Outline of the book: The aim of this book is to identify software engineering techniques for Grid environments, along with specialist tools that encapsulate such techniques, and case studies that illustrate the use of these tools. With the emergence of regional, national and global programmes to establish Grid computing infrastructure, it is important to be able to utilize this infrastructure effectively. Specialist software is therefore necessary to both enable the deployment of applications over such infrastructure, and to facilitate software developers in constructing software components for such infrastructure. We feel the
second of these is a particularly important concern, as the uptake of Grid computing technologies will be restricted by the availability of suitable abstractions, methodologies, and tools.

This book will be useful for:

- **Software developers** who are primarily responsible for developing and integrating components for Grid environments.
- It will also be of interest to **application scientists and domain experts**, who are primarily users of the Grid software and need to interact with the tools.
- The book will also be useful for **deployment specialists**, who are primarily responsible for managing and configuring Grid environments.

We hope the book will contribute to increase the reader’s appreciation for:

- Software engineering and modelling tools which will enable better conceptual understanding of the software to be deployed across Grid infrastructure.
- Software engineering issues that must be supported to compose software components for Grid environments.
- Software engineering support for managing Grid applications.
- Software engineering lifecycle to support application development for Grid Environments (along with associated tools).
- How novel concepts, methods and tools within Grid computing can be put at work in the context of existing experiments and application case studies.

As many universities are now also in the process of establishing courses in Grid Computing, we hope this book will serve as a reference to this emerging area, and will help promote further developments both at university and industry. The chapters presented in this book are divided into four sections:

- **Abstractions**; chapters included in this section represent key modelling approaches that are necessary to enable better software development for deployment over Grid computing infrastructure. Without such abstractions, one is likely to see the continuing use of ad-hoc approaches.
- **Programming and Process**; chapters included in this section focus on the overall software engineering process necessary for application construction. Such a process is essential to channel the activity of a team of programmers working on a Grid application.
- **User Environments and Tools**; chapters in this section discuss existing application environments that may be used to implement Grid applications, or provide a discussion of how applications may be effectively deployed across existing Grid computing infrastructure.
- **Applications**; the final section provides sample applications in Engineering, Science and Education, and demonstrate some of the ideas discussed in other section with reference to specific application domains.

*José Cunha, Universidade Nova de Lisboa, Portugal*

*Omer F. Rana, Cardiff University, UK*
Contents

Abstractions

Chapter 1: Virtualisation in Grids: A Semantical Approach .................. XX
    Zsolt Nemeth and Vaidy Sunderam

Chapter 2: Using Event Models in Grid Design ................................. XX
    Anthony Finkelstein, Joe Lewis-Bowen, Giacomo Piccinelli, and Wolfgang
    Emerich

Chapter 3: Intelligent Grids .................................................... XX
    Xin Bai, Han Yu, Guoqiang Wang, Yongchang Ji, Gabriela M. Marinescu,
    Dan C. Marinescu, and Ladislau Bolloni

Programming and Process

Chapter 4: A Grid Software Process ........................................ XX
    Giovanni Aloisi, Massimo Caffaro, and Italo Epicoco

Chapter 5: Grid Programming with Java, RMI and Skeletons ............... XX
    Sergei Gorlatch and Martin Alt

User Environments and Tools

Chapter 6: A Review of Grid Portal Technology .......................... XX
    Maozhen Li and Mark Baker

Chapter 7: A Framework for Loosely Coupled Applications on Grid Environ-
ments ..................................................................................... XX
    Andreas Hoheisel, Thilo Ernst, and Uwe Der

Chapter 8: Toward GRIDLE: A Way to Build Grid Application Searching
Through an Ecosystem of Components ........................................ XX
    Diego Puppin, Fabrizio Silvestri, Salvatore Orlando, and Domenico
    Laforenza

Chapter 9: Programming, Composing, Deploying for the Grid ........... XX
    Laurent Baduel, Françoise Baude, Denis Caromel, Arnaud Contes, Fabrice
    Huet, Matthieu Morel, and Romain Quilici

Chapter 10: ASSIST as a Research Framework for High-performance Grid
Programming Environments .................................................... XX
    Marco Aldinucci, Massimo Coppola, Marco Vanneschi, Corrado Zoccolo and
    Marco Danelutto
Chapter 11: A Visual Programming Environment for Developing Complex Grid Applications .................................................. XX
   Antonio Congiuasta, Domenico Tullia, and Paolo Trunfio

Applications

Chapter 12: Solving Computationally Intensive Engineering Problems on the Grid using Problem Solving Environments ................ XX
   Christopher Goodyer and Martin Berzins

Chapter 13: Design Principles for a Grid-Enabled Problem Solving Environment to be used by Engineers ........................ XX
   Graeme Pound and Simon Cox

Chapter 14: Towards the Utilisation of Grid Computing in Electronic Learning XX
   Victor Pankratius and Gottfried Vossen
List of Contributors

Marco Aldimucci$^1,2$, Massimo Coppola$^1,2$, Marco Danelutto$^2$, Marco Vanneschi$^2$, Corrado Zoccolo$^2$
1 Dipartimento di Informatica, Università di Pisa, Italy
2 Istituto di Scienza e Tecnologie della Informazione, CNR, Pisa, Italy

Giovanni Aloisio, Massimo Cafaro, and Italo Epicoco
Center for Advanced Computational Technologies, University of Lecce, Italy

Laurent Baduel, Françoise Baude, Denis Caromel, Arnaud Contes, Fabrice Huet, Matthieu Morel, and Romain Quilici
OASIS: Joint Project CNRS / INRIA / University of Nice Sophia – Antipolis, INRIA 2004, route des Lucioles - B.P. 93 - 06902 Valbonne Cedex, France

Xin Bai$^1$, Han Yu$^1$, Guoqiang Wang$^1$, Yongchang Ji$^1$, Gabriela M. Marinescu$^1$, Dan C. Marinescu$^1$, and Ladislau Bölöni$^2$
1 School of Computer Science, University of Central Florida, P.O.Box 162362, Orlando, Florida 32816-2362, USA
2 Department of Electrical and Computer Engineering University of Central Florida, P.O.Box 162450, Orlando, Florida 32816-2450, USA

Antonio Congiusta$^1,2$, Domenico Talia$^1,2$, and Paolo Trunfio$^2$
1 ICAR-CNR, Institute of the Italian National Research Council, Via P. Bucci, 41c, 87036 Rende, Italy
2 DEIS - University of Calabria, Via P. Bucci, 41c, 87036 Rende, Italy

Anthony Finkelstein, Joe Lewis-Bowen, and Giacomo Piccinelli
Department of Computer Science, University College London, Gower Street, London, WC1E 6BT, UK

Christopher E. Goodyer$^1$ and Martin Berzins$^1,2$
1 Computational PDEs Unit, School of Computing, University of Leeds, Leeds, UK
2 SCI Institute, University of Utah, Salt Lake City, Utah, USA

Sergei Gorlatch and Martin Alt
Westfälische Wilhelms-Universität Münster, Germany
Andreas Hohesel, Thilo Ernst, and Uwe Der
Fraunhofer Institute for Computer Architecture and Software Technology
(FIRST), Kekuléstr. 7, D-12489 Berlin, Germany

Maozhen Li¹ and Mark Baker²
¹ Department of Electronic and Computer Engineering, Brunel University
Uxbridge, UB8 3PH, UK
² The Distributed Systems Group, University of Portsmouth
Portsmouth, PO1 2EG, UK

Zsolt Németh¹ and Vaidy Sunderam²
¹ MTA SZTAKI Computer and Automation Research Institute
H-1518 Budapest, P.O. Box 63, Hungary
² Math & Computer Science, Emory University, Atlanta, GA 30322, USA

Victor Pankratius¹ and Gottfried Vossen²
¹ AIFB Institute, University of Karlsruhe, D-76128 Karlsruhe, Germany
² ERCIS, University of Münster, D-48149 Münster, Germany

Graeme Pound and Simon Cox
School of Engineering Sciences, University of Southampton, Southampton,
SO17 1BJ, UK

Diego Puppin¹, Fabrizio Silvestri¹, Salvatore Orlando², Domenico
Laforenza¹
¹ Institute for Information Science and Technologies, ISTI - CNR, Pisa, Italy
² Università di Venezia, Ca’ Foscari, Venezia, Italy