Computer Communications and Networks

For other titles published in this series, go to
www.springer.com/series/4198

The **Computer Communications and Networks** series is a range of textbooks, monographs and handbooks. It sets out to provide students, researchers and non-specialists alike with a sure grounding in current knowledge, together with comprehensible access to the latest developments in computer communications and networking.

Emphasis is placed on clear and explanatory styles that support a tutorial approach, so that even the most complex of topics is presented in a lucid and intelligible manner.

Massimo Cafaro • Giovanni Aloisio
Editors

# Grids, Clouds and Virtualization

Springer

*Editors*
Dr. Massimo Cafaro
Dipartimento di Ingegneria
dell'Innovazione
Università del Salento
Via per Monteroni
73100 Lecce
Italy
massimo.cafaro@unisalento.it

Prof. Giovanni Aloisio
Dipartimento di Ingegneria
dell'Innovazione
Università del Salento
Via per Monteroni
73100 Lecce
Italy
giovanni.aloisio@unisalento.it

*Series Editor*
Professor A.J. Sammes, BSc, MPhil, PhD,
FBCS, CEng
Centre for Forensic Computing
Cranfield University
DCMT, Shrivenham
Swindon SN6 8LA
UK

ISSN 1617-7975
ISBN 978-0-85729-048-9 e-ISBN 978-0-85729-049-6
DOI 10.1007/978-0-85729-049-6
Springer London Dordrecht Heidelberg New York

British Library Cataloguing in Publication Data
A catalogue record for this book is available from the British Library

Library of Congress Control Number: 2010936502



*Cover design*: VTEX, Vilnius

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

# Preface

For more than a decade, the development of grid computing was driven by scientific applications. The need to solve large-scale, increasingly complex problems motivated research on grids systems. Many interesting problems have been solved with the help of grids, for instance, the nug30 Quadratic Assignment Problem.

This challenging optimization problem was posed in 1968 and requires, given a set of $n$ facilities, a set of $n$ locations, a distance specified for each pair of locations, and a flow (weight) specified for each pair of facilities (e.g., the amount of supplies transported between the two facilities), assigning all 30 facilities to the 30 different locations with the goal of minimizing the sum of the distances multiplied by the corresponding flows.

Despite its apparent simplicity, the problem is NP-Hard, and the number of possible assignments is extremely large, so that even if you could check a trillion assignments per second, this process would take over 100 times the age of the universe. However, once the algorithms and software necessary to tackle the previously unsolved problem on a computational grid were developed, solving the problem required nearly a week, with a computational endeavor involving more than 1,000 computational resources working simultaneously at eight institutions geographically distributed in different parts of the world.

The FightAIDS@Home project, which is based on the volunteered computing power of the World Community Grid, aims at testing candidate compounds against the variations (or "mutants") of HIV that can arise and cause drug resistance.

During November 2009, the project identified several fragments as new candidates for a novel binding site on the peripheral surface of HIV protease. These fragments docked well against the "exo site" and in vitro studies (i.e., "wet lab" experiments in test tubes) will assess their potencies. If these wet lab experiments produce promising results, then these fragments could form the foundation for the development of "allosteric inhibitors" of HIV protease (i.e., "flexibility wedges" that can disrupt the conformational changes that HIV protease must undergo in order to function). These allosteric inhibitors could represent a totally new class of anti-AIDS compounds.

These two examples clearly explain why scientists are now routinely supported in their research by grid infrastructures. But what about business and casual users?

Although projects such as BEinGRID have reported some successful business experiments that may profit from execution in grid environments, it appears that there is not a general business case for the grid. However, recent advances in virtualization techniques, coupled with the increased Internet bandwidth now available, led in 2007 to the concept of cloud computing. The emergence of this new paradigm is mainly based on its simplicity and the affordable price for seamless access to both computational and storage resources.

Virtualization enables cloud computing, providing the ability to run legacy applications on older operating systems, creation of a single system image starting from an heterogeneous collection of machines such as those traditionally found in grid environments, and faster job migration within different virtual machines running on the same hardware. For grid and cloud computing, virtualization is the key for provisioning and fair resource allocation. From the security point of view, since virtual machines run isolated in their sandboxes, this provides an additional protection against malicious or faulty codes.

Clouds provide access to inexpensive hardware and storage resources through very simple APIs, and are based on a pay-per-use model, so that renting these resources is usually much cheaper than acquiring dedicated new ones. Moreover, people are becoming comfortable with storing their data remotely in a cloud environment. Therefore, clouds are being increasingly used by scientists, small and medium sized enterprises, and casual users.

Grids, clouds, and virtualization are exciting technologies that are going to become prominent in the next few years; we expect a wide proliferation in their use, especially clouds since these distributed computing facilities are already accessible at a reasonable cost to many potential users. We also expect grids and clouds to play an ever increasing role in the field of scientific research. It is therefore necessary a thorough understanding of principles and techniques of these fields, and the main aim of this book is to foster awareness of the essential ideas by exploring current and future developments in the area.

The idea of writing this book dates back to the highly successful Grids, Clouds and Virtualization Workshop that we organized in conjunction with the 4th International Conference on Grid and Pervasive Computing (GPC 2009) held in Geneva, 4–8 May 2009. We were contacted by Mr. Wayne Wheeler of Springer, and, after an insightful discussion, we agreed to serve as the editors for the book. Indeed, it is virtually impossible for a single person to write a book covering all of the important aspects of grids, clouds, and virtualization while maintaining the required depth, consistency, and appeal.

We invited many well-known and internationally recognized experts, asking them to contribute their expertise. The book delves into details of grids, clouds, and virtualization, guiding the reader through a collection of chapters dealing with key topics. The bibliography rather than being exhaustive, covers essential reference material. The aim is to avoid an encyclopedic approach since we believe that an attempt to cover everything will instead fail to convey any useful information to the interested readers, an audience including researchers actively involved in the field, undergraduate and graduate students, system designers and programmers, and IT policy makers.

The book may serve both as an introduction and as a technical reference. Our desire and hope is that it will be useful to many people familiarizing with the subject and will contribute to new advances in the field.

Lecce, Italy

Massimo Cafaro
Giovanni Aloisio

# Acknowledgements

Every book requires months of preparations, and this book is no exception. We would like to express our gratitude to the contributors for their participation in this project. Without their technical expertise, patience, and efforts, this book would not have been possible.

We are also indebted with the Springer editorial team for their cooperation efforts that made this book a reality. In particular, we are deeply grateful to Mr. Wayne Wheeler, Senior Editor, for his initial proposal and continuous encouragements. We serve as the editors of this book owing to his incredible skills and energy. Special thanks must also go to Mr. Simon Rees, Senior Editorial Assistant, for his dedication, support, and punctuality.

# Contents

# Contributors

**S. Ács** MTA SZTAKI, P.O. Box 63, 1518 Budapest, Hungary, acs@sztaki.hu

**Gagan Agrawal** Department of Computer Science and Engineering, The Ohio State University, Columbus, OH 43210, USA, agrawal@cse.ohio-state.edu

**Giovanni Aloisio** University of Salento, Lecce, Italy, giovanni.aloisio@unisalento.it

**Henri E. Bal** Department of Computer Science, Vrije Universiteit, De Boelelaan 1081A, 1081 HV Amsterdam, The Netherlands, bal@cs.vu.nl

**Massimo Cafaro** Dipartimento di Ingegneria dell'Innovazione, Università del Salento, Via per Monteroni, 73100 Lecce, Italy, massimo.cafaro@unisalento.it

**David Chiu** School of Engineering and Computer Science, Washington State University, Vancouver, WA 98686, USA, david.chiu@wsu.edu

**Marcos Dias de Assunção** INRIA, LIP Laboratory (UMR CNRS, INRIA, ENS, UCB), University of Lyon, 46 allée d'Italie, 69364 Lyon Cedex 07, France, marcos.dias.de.assuncao@ens-lyon.fr

**Ewa Deelman** University of Southern California, Marina del Rey, CA, USA, deelman@isi.edu

**Niels Drost** Department of Computer Science, Vrije Universiteit, De Boelelaan 1081A, 1081 HV Amsterdam, The Netherlands, niels@cs.vu.nl

**Miklos Erdely** University of Pannonia, Veszprem, Hungary, erdelyim@gmail.com

**Z. Farkas** MTA SZTAKI, P.O. Box 63, 1518 Budapest, Hungary, zfarkas@sztaki.hu

**Eduardo Huedo** Universidad Complutense de Madrid, 28040 Madrid, Spain, ehuedo@fdi.ucm.es

**Ceriel Jacobs** Department of Computer Science, Vrije Universiteit, De Boelelaan 1081A, 1081 HV Amsterdam, The Netherlands, ceriel@cs.vu.nl

**Shantenu Jha** Louisiana State University, Baton Rouge, 70803, USA, sjha@cct.lsu.edu

**Gideon Juve** University of Southern California, Marina del Rey, CA, USA, juve@usc.edu

**P. Kacsuk** MTA SZTAKI, P.O. Box 63, 1518 Budapest, Hungary, kacsuk@sztaki.hu

**Thilo Kielmann** Department of Computer Science, Vrije Universiteit, De Boelelaan 1081A, 1081 HV Amsterdam, The Netherlands, kielmann@cs.vu.nl

**M. Kozlovszky** MTA SZTAKI, P.O. Box 63, 1518 Budapest, Hungary, m.kozlovszky@sztaki.hu

**J. Lakshmi** SERC, Indian Institute of Science, Bangalore 560012, India, jlakshmi@serc.iisc.ernet.in

**Laurent Lefèvre** INRIA, LIP Laboratory (UMR CNRS, INRIA, ENS, UCB), University of Lyon, 46 allée d'Italie, 69364 Lyon Cedex 07, France, laurent.lefevre@inria.fr

**Ignacio M. Llorente** Universidad Complutense de Madrid, 28040 Madrid, Spain, llorente@dacya.ucm.es

**Andre Luckow** Louisiana State University, Baton Rouge, 70803, USA, aluckow@cct.lsu.edu

**Jason Maassen** Department of Computer Science, Vrije Universiteit, De Boelelaan 1081A, 1081 HV Amsterdam, The Netherlands, jason@cs.vu.nl

**A. Marosi** MTA SZTAKI, P.O. Box 63, 1518 Budapest, Hungary, atisu@sztaki.hu

**Andre Merzky** Louisiana State University, Baton Rouge, 70803, USA, andre@merzky.net

**Rubén S. Montero** Universidad Complutense de Madrid, 28040 Madrid, Spain, rubensm@dacya.ucm.es

**Rafael Moreno-Vozmediano** Universidad Complutense de Madrid, 28040 Madrid, Spain, rmoreno@dacya.ucm.es

**S.K. Nandy** SERC, Indian Institute of Science, Bangalore 560012, India, nandy@serc.iisc.ernet.in

**Anne-Cécile Orgerie** ENS Lyon, LIP Laboratory (UMR CNRS, INRIA, ENS, UCB), University of Lyon, 46 allée d'Italie, 69364 Lyon Cedex 07, France, annececile.orgerie@ens-lyon.fr

**A.J. Sammes**, Centre for Forensic Computing, Cranfield University, DCMT, Shrivenham, Swindon SN6 8LA, UK

**Saurabh Sehgal** Louisiana State University, Baton Rouge, 70803, USA

**Frank J. Seinstra** Department of Computer Science, Vrije Universiteit, De Boelelaan 1081A, 1081 HV Amsterdam, The Netherlands, fjseins@cs.vu.nl

**Jacopo Urbani** Department of Computer Science, Vrije Universiteit, De Boelelaan 1081A, 1081 HV Amsterdam, The Netherlands, jacopo@cs.vu.nl

**Timo van Kessel** Department of Computer Science, Vrije Universiteit, De Boelelaan 1081A, 1081 HV Amsterdam, The Netherlands, timo@cs.vu.nl

**Rob V. van Nieuwpoort** Department of Computer Science, Vrije Universiteit, De Boelelaan 1081A, 1081 HV Amsterdam, The Netherlands, rob@cs.vu.nl

**Ben van Werkhoven** Department of Computer Science, Vrije Universiteit, De Boelelaan 1081A, 1081 HV Amsterdam, The Netherlands, ben@cs.vu.nl

# Chapter 1
# Grids, Clouds, and Virtualization

**Massimo Cafaro and Giovanni Aloisio**

**Abstract** This chapter introduces and puts in context Grids, Clouds, and Virtualization. Grids promised to deliver computing power on demand. However, despite a decade of active research, no viable commercial grid computing provider has emerged. On the other hand, it is widely believed—especially in the Business World—that HPC will eventually become a commodity. Just as some commercial consumers of electricity have mission requirements that necessitate they generate their own power, some consumers of computational resources will continue to need to provision their own supercomputers. Clouds are a recent business-oriented development with the potential to render this eventually as rare as organizations that generate their own electricity today, even among institutions who currently consider themselves the unassailable elite of the HPC business. Finally, Virtualization is one of the key technologies enabling many different Clouds. We begin with a brief history in order to put them in context, and recall the basic principles and concepts underlying and clearly differentiating them. A thorough overview and survey of existing technologies provides the basis to delve into details as the reader progresses through the book.

## 1.1 Introduction

This chapter introduces and puts in context Grids, Clouds, and Virtualization [17]. Grids promised to deliver computing power on demand. However, despite a decade of active research, no viable commercial grid computing provider has emerged. On the other hand, it is widely believed—especially in the Business World—that HPC will eventually become a commodity. Just as some commercial consumers of electricity have mission requirements that necessitate they generate their own power,

M. Cafaro (✉) · G. Aloisio
University of Salento, Lecce, Italy
e-mail: massimo.cafaro@unisalento.it

G. Aloisio
e-mail: giovanni.aloisio@unisalento.it

M. Cafaro, G. Aloisio (eds.), *Grids, Clouds and Virtualization,*
Computer Communications and Networks,
DOI 10.1007/978-0-85729-049-6_1, © Springer-Verlag London Limited 2011

some consumers of computational resources will continue to need to provision their own supercomputers. Clouds are a recent business-oriented development with the potential to render this eventually as rare as organizations that generate their own electricity today, even among institutions who currently consider themselves the unassailable elite of the HPC business. Finally, Virtualization is one of the key technologies enabling many different Clouds. We begin with a brief history in order to put them in context, and recall the basic principles and concepts underlying and clearly differentiating them. A thorough overview and survey of existing technologies and projects provides the basis to delve into details as the reader progresses through the book.

## 1.2 A Bit of History

The history of Grids and Clouds may be traced back to the 1961 MIT Centennial, when John McCarthy, a pioneer in mathematical theory of computation and artificial intelligence and the inventor of the Lisp programming language, first exposed the idea of utility computing: "...If computers of the kind I have advocated become the computers of the future, then computing may someday be organized as a public utility just as the telephone system is a public utility...The computer utility could become the basis of a new and important industry."

Indeed, owing to the huge costs and complexity of provisioning and maintaining a data center, in the next two decades many large organizations (primarily banks) rented computing power and storage provided by mainframe computers geographically spread in the data centers of IBM and other providers. Meanwhile, mini, micro, and personal computers appeared on the market. During early 1980s, the majority of the organizations acquired affordable personal computers and workstations. This was perceived as the end of utility computing until the next decade.

In 1992, Charlie Catlett and Larry Smarr introduced the concept of metacomputing in their seminal paper [50]. The term metacomputing refers to computation on a virtual supercomputer assembled connecting together different resources like parallel supercomputers, data archives, storage systems, advanced visualization devices, and scientific instruments using high-speed networks that link together these geographically distributed resources. The main reason for doing so is because it enables new classes of applications [14, 39] previously impossible and because it is a cost-effective approach to high-performance computing. Metacomputing proved to be feasible in several experiments and testbeds, including the I-WAY experiment [19, 25] and in the Globus Gusto testbed [26].

The new applications were initially classified as follows:

- desktop supercomputing;
- smart instruments;
- collaborative environments;
- distributed supercomputing.

Desktop supercomputing included applications coupling high-end graphics capabilities with remote supercomputers and/or databases; smart instruments are scientific instruments like microscopes, telescopes, and satellites requiring supercomputing power to process the data produced in near real time. In the class of collaborative environments there were applications in which users at different locations could interact together working on a supercomputer simulation; distributed supercomputing finally was the class of applications requiring multiple supercomputers to solve problems otherwise too large or whose execution was divided on different components that could benefit from execution on different architectures.

The challenges to be faced before metacomputing could be really exploited were identified as related to the following issues:

- scaling and selection;
- unpredictable structure;
- heterogeneity;
- dynamic behavior;
- multiple administrative domains.

Interestingly (from a research perspective), these are still relevant today. Scaling is a concern, because we expect that grid and cloud environments in the future will become even larger, and resources will be selected and acquired on the basis of criteria such as connectivity, cost, security, and reliability. These resources will show different levels of heterogeneity, ranging from physical devices to system software and schedulers policies; moreover, traditional high-performance applications are developed for a single supercomputer whose features are known a priori, e.g., the latency of the interconnection network; in contrast, grid and cloud applications will run in a wide range of environments, thus making impossible to predict the structure of the computation. Another concern is related to the dynamic behavior of the computation [51], since we cannot, in general, be assured that all of the system characteristics stay the same during the course of computation, e.g., the network bandwidth and latency can widely change, and there is the possibility of both network and resource failure. Finally, since the computation will usually span resources geographically spread at multiple administrative domains, there is not a single authority in charge of the system, so that different scheduling policies and authorization mechanisms must be taken into account.

In the same years, a middleware project called Legion [33] promoted the grid object model. Legion was designed on the basis of common abstractions of the object-oriented model: everything in Legion was an object with a well-defined set of access method, including files, computing resources, storage, etc. The basic idea was to expose the grid as a single, huge virtual machine with the aim of hiding the underlying complexity to the user. The middleware proved to be useful in several experiments, including a distributed run of an ocean model and task-farmed computational chemistry simulations. However, it became immediately apparent that the majority of the people in academia were not fond of the grid object model; consequently, the attention shifted toward the use of the Globus Toolkit, which a few years later provided an alternative software stack and became quickly the de facto standard for grid computing.

The Globus Toolkit, initially presented as a metacomputing environment [24], was one of the first middleware solutions really designed to tackle the issues related to large-scale distributed computing, and its usefulness in the context of metacomputing was demonstrated in the Gusto testbed; among the distributed simulations that have been run using Globus, there was SF-Express [16], the largest computer simulation of a military battle involving at the time more that 100,000 entities. Even the EuropeanData Grid project, in charge of building a European middleware for grid computing, did so leveraging many of the Globus components. These efforts led to the gLite middleware [40], which is going to be used for the analysis of experimental results of the Large Hadron Collider, the particle accelerator at CERN that recently started operations in Geneva. The middleware was also actively tested in the context of the EGEE project [41], which provided the world largest computational grid testbed to date. Another European project developed a different middleware, Unicore [21], targeting mainly HPC resources. A similar endeavor, devoted to the implementation of grid middleware for high-performance computing, took place in Japan starting in 2003 with the National Research Grid Initiative (NAREGI) [43] and culminating in 2008 with the release of the middleware.

There was a pressure to grid-enable many existing legacy products. Among the schedulers, we recall here Condor [52], Nimrod [10], Sun Grid Engine [30], and Platform Computing LSF [56]. The Condor project begun in 1988 and is therefore one of the earliest cycle scavenging software available. Designed for loosely coupled jobs, and to cope with failures, it was ported to the grid through a Globus extension aptly named Condor-G. The Nimrod system was designed to manage large parameter sweep jobs, in which the same executable was run each time with a different input. Since the jobs are independent, a grid is clearly a good fit for this situation. The Nimrod-G system, extended once again through the Globus Toolkit, was also one of the earliest projects in which the concepts of grid economy appeared. In addition to several other criteria, the user could also take into account the cost of the whole simulation when submitting a job. The system was able to charge the CPU cycles distinguishing HPC resources from traditional, off-the-shelf ones. Sun Grid Engine, an open source project led by Sun, started in 2000. The software was originally based on the Gridware Codine (COmputing in DIstributed Network Environments) scheduler. It was grid-enabled and is now being cloud-enabled. Finally, LSF was one the first commercial products offering support for grid environments through its extension named LSF MultiCluster.

The business and commercial world recognized the impact and the potential of grid computing, whose usefulness was demonstrated in hundreds of projects. However, almost all of the projects were driven by people involved in academia and targeting mainly scientific applications. Very few projects, notably the BEinGrid [20] one, were in charge of demonstrating the use of grid computing for business oriented goals. BEinGrid, with its sample of 25 business experiments, was also successful in the implementation and deployment of Grid solutions in industrial key sectors. Nonetheless, grids are still not appealing to the business world, and this is reflected in the lack of current commercial grid solutions. Platform Computing provided years ago a version of the Globus Toolkit which is now unsupported and

not for sale. Legion was initially sold by Applied Meta, then by Avaki Corporation which was acquired in 2005 by Sybase, Inc. What remains of the software is now used by the company's customers to share enterprise data. Entropia, Inc., a company founded in 1997, sold distributed computing software for CPU scavenging until 2004.

The idea of harnessing idle CPUs worldwide was first exploited by the SETI@home project [11], launched in 1996. Soon a number of related projects including GIMPS [44], FightAIDS@Home [47], Folding@home [13] arose. GIMPS, the Great Internet Mersenne Prime Search, was started in January 1996 to discover new world-record-size Mersenne primes. A Mersenne prime is a prime of the form $2^P - 1$. The first Mersenne primes are 3, 7, 31, 127 (corresponding to $P = 2, 3, 5, 7$). There are only 47 known Mersenne primes, and GIMPS has found 13 of the 47 Mersenne primes ever found during its 13-year history. FightAIDS@Home uses distributed computing exploiting idle resources to accelerate research into new drug therapies for HIV, the virus that causes AIDS; FightAIDS@Home made history in September 2000 when it became the first biomedical Internet-based grid computing project. Proteins, in order to carry out their function, must take on a particular shape, also known as a fold. One of the Folding@home goals is to simulate protein folding in order to understand how proteins fold so quickly and reliably, and to learn about what happens when this process goes awry (when proteins misfold).

The initial release of the Apple Xgrid [35, 36] technology was also designed for independent, embarrassingly parallel jobs, and was later extended to support tightly coupled parallel MPI jobs. A key requirement of Xgrid was simplicity: everyone is able to setup and use an hoc grid using Xgrid, not just scientists. However, only Apple resources running the Mac OS X operating system may belong to this grid, although third-party software (an Xgrid Linux agent and a Java-based one) not supported or endorsed by Apple allows deploying heterogeneous grids.

## 1.3 Grids

It is interesting to begin by reviewing some of most important definitions of grid computing given during the course of research and development of the field. The earliest definition given in 1998 by Foster and Kesselman [37] is focused around on-demand access to computing, data, and services: a computational grid is a hardware and software infrastructure that provides dependable, consistent, pervasive, and inexpensive access to high-end computational capabilities. Two years later, the definition was changed to reflect the fact that grid computing is concerned with coordinated resource sharing and problem solving in dynamic, multi-institutional virtual organizations. In the latest definitions of Foster [23, 27], a grid is described respectively as an infrastructure for coordinated resource sharing and problem solving in dynamic, multi-institutional virtual organizations and as a system that coordinates resources that are not subject to centralized control, using standard, open, general-purpose protocols and interfaces to deliver nontrivial qualities of service. We will

argue later, when comparing grids and clouds, that delivering nontrivial qualities of service is very difficult using a distributed approach instead of a centralized one.

Grids are certainly an important computing paradigm for science, and there are many different scientific communities facing large-scale problems simply too big to be faced even on a single, powerful parallel supercomputer. After more than a decade of active research and development in the field, all of the people expected grids to become a commercial service; however, this has not happened yet. Therefore, we ask ourselves: is there a general business case for grids?

To answer this question, we recall that it is now clear that grid computing benefits specific classes of applications and that the technology itself, while powerful, is probably not yet simple enough to be released commercially. This is reflected in the following sentence of Frank Gillette, Forester: "None of us have figured out a simple way to talk about (grid) . . . because it isn't simple." And, when something takes an hour or more to be explained, it certainly cannot be sold easily. Therefore, application providers may prefer to integrate the required middleware into their applications or take advantage of existing Platform and DataSynapse solutions. The issue raised by the complexity of grids proves to be a formidable barrier that we must overcome if we want grids to become a fact of life that everyone uses and nobody notices. This has been recently remarked in a position paper by Fox and Pierce [29] in which the authors discuss the fact that "Grids meet Too much Computing, Too much Data and never Too much Simplicity."

On the other hand, early testbeds and recent grids deployment have clearly shown the potential of grid computing. In particular, loosely coupled parallel applications, parameter sweep studies and applications described by a workflow assembling and orchestrating several different components are viable candidates for grid computing. Example applications include those from the High Energy Physics (HEP) community, falling in the parameter sweep class, and those from the bioinformatics community, falling in the workflow class. Typical loosely coupled parallel applications are exemplified by climate simulations, in which the initial functional decomposition of the problem is well suited for execution on grids linking together distributed supercomputers; the blocks identified by the initial decomposition (atmosphere, land, sea, ice, etc.) are then run on those supercomputers and exchange data infrequently.

When the vendors realized that this kind of applications were far beyond the common needs of the majority of the people, the attention shifted to the emerging field of cloud computing. Here, in contrast to grid computing, the emphasis was put on simplicity as the ingredient to make clouds easily accessible.

## 1.4 Clouds

It is rather difficult to precisely define what clouds and cloud computing are, especially taking into account the many possible different uses. We recall here that the same also happened in the context of grids. The concept as we all know it today was introduced in late 2007 [54]. Among the many definitions that were given since then, we report the following ones:

- Gartner: Cloud computing is a style of computing where massively scalable IT-related capabilities are provided "as a service" across the Internet to multiple external customers;
- Forrester: A pool of abstracted, highly scalable, and managed infrastructure capable of hosting end-customer applications and billed by consumption;
- IBM: An emerging computing paradigm where data and services reside in massively scalable data centers and can be ubiquitously accessed from any connected devices over the internet.

How clouds are actually perceived by the people appears to be much broader in scope. The following five definitions [48] expose several perspectives on this subject.

1. Cloud computing refers (for many) to a variety of services available over the Internet that deliver compute functionality on the service provider's infrastructure (e.g., Google Apps, Amazon EC2, or Salesforce.com). A cloud computing environment may actually be hosted on either a grid or utility computing environment, but that does not matter to a service user;
2. - Cloud computing = Grid computing. The workload is sent to the IT infrastructure that consists of dispatching masters and working slave nodes. The masters control resource distributions to the workload (how many slaves run the parallelized workload). This is transparent to the client, who only sees that workload has been dispatched to the cloud/grid and results are returned to it. The slaves may or may not be virtual hosts;
   - Cloud computing = Software-as-Service. This is the Google Apps model, where apps are located "in the cloud," i.e., somewhere in the Web;
   - Cloud computing = Platform-as-Service. This is the Amazon EC2 et al. model, where an external entity maintains the IT infrastructure (masters/slaves), and the client buys time/resources on this infrastructure. This is "in the cloud" in so much that it is across the Web, outside of the organization that is leasing time off it;
3. The cloud simply refers to the move from local to service on the Web. From storing files locally to storing them in secure scalable environments. From doing apps that are limited to GB spaces to now apps that have no upper boundary, from using Microsoft Office to using a Web-based office. Somewhere in 2005–2008 storage online got cheaper and more secure than storing locally or on your own server. This is the cloud. It encompasses grid computing, larger databases like Bigtable, caching, always accessible, failover, redundant, scalable, and all sorts of things. Think of it as a further move into the Internet. It also has large implications for such battles as static vs. dynamic, RDBMS vs. BigTable and flat data views. The whole structure of business that relies on IT infrastructure will change, programmers will drive the cloud, and there will be lots of rich programmers at the end. It is like the move from mainframe to personal computers. Now you have a personal space in the clouds;
4. Grid and Cloud are not exclusive of each other... Our customers view it this way: Cloud is pay for usage (i.e., you do not necessarily own the resources), and

Grid is how to schedule the work, regardless where you run it. You can use a cloud without a grid and a grid without a cloud. Or you can use a grid on a cloud;

5. I typically break up the idea of cloud computing into three camps:

   - Enablers. These are companies that enable the underlying infrastructures or the basic building blocks. These companies are typically focused on data center automation and or server virtualization (VMware/EMC, Citrix, BladeLogic, RedHat, Intel, Sun, IBM, Enomalism, etc.);
   - Providers (Amazon Web Services, Rackspace, Google, Microsoft). The ones with the budgets and know-how to build out global computing environments costing millions or even billions of dollars. Cloud providers typically offer their infrastructure or platform. Frequently, these As-a-Service offerings are billed and consumed on a utility basis;
   - Consumers. On the other side of the spectrum, I see the consumers companies that build or improve their Web applications on top of existing clouds of computing capacity without the need to invest in data centers or any physical infrastructure. Often these two groups can be one in the same such as Amazon (SQS, SDB, etc.), Google (Apps), and Salesforce (Force). But they can also be new startups that provide tools and services that sit on top of the cloud (Cloud management).

   Cloud consumers can be a fairly broad group including just about any application that is provided via a Web-based service like a Webmail, blogs, social network, etc. Cloud computing from the consumer point of view is becoming the only way you build, host, and deploy a scalable Web application.

On the other hand, the main findings of the Cloud BoF held at OGF22, Cambridge, MA, on 27 Feb 2008 were the following ones:

- Clouds are “Virtual Clusters” (“Virtual Grids”) of possibly “Virtual Machines”;
- They may cross administrative domains or may “just be a single cluster”; the user cannot and does not want to know;
- Clouds support access (lease of) computer instances;
- Instances accept data and job descriptions (code) and return results that are data and status flags;
- Each Cloud is a “Narrow” (perhaps internally proprietary) Grid;
- When does Cloud concept work:
  - Parameter searches, LHC style data analysis, ...
  - Common case (most likely success case for clouds) versus corner case?
- Clouds can be built from Grids;
- Grids can be built from Clouds;
- Geoffrey Fox: difficult to compare grids and clouds because neither term has an agreed definition;
- Unlike grids, clouds expose a simple, high-level interface;
- There are numerous technical issues:
  - performance overhead, cost, security, computing model, data-compute affinity, schedulers and QoS, link clouds (e.g., compute-data), ...;

– What happens when a cloud goes down? What about interoperability of clouds? Standardization? Is it just another service?

A recent report [12] present an in-depth view of cloud computing. In what follows we will try to make things easier to understand, summarizing our perspective. In order to do so, we begin discussing the main features and characteristics of clouds as currently deployed and made available to the people. The main characteristic of cloud computing certainly is its focus on virtualization. Clouds are succeeding owing to the fact that the underlying infrastructure and physical location are fully transparent to the user.

Beyond that, clouds also exhibit excellent scalability allowing users to run increasingly complex applications and breaking the overall workload into manageable pieces served by the easily expandable cloud infrastructure. This flexibility is a key ingredient, and it is very appealing to the users. Clouds can adapt dynamically to both consumer and commercial workloads, providing efficient access through a Service Oriented Architecture to a computing infrastructure delivering dynamic provisioning of shared compute resources.

An attempt to provide a comprehensive comparison of grids and clouds is of course available [28]. However, we now compare and contrast grids and clouds, in order to highlight what we think are key differences. Grids are based on open standards; the standardization process happens in organizations such as the Open Grid Forum, OASIS, etc. Clouds, in contrast, do not provide standardized interfaces. Especially commercial clouds solutions are based on proprietary protocols, which are not disclosed to the scientific community. A related aspect is that of interoperability. While in grid environments interoperability has become increasingly important, and many efforts are devoted to this topic [15], clouds are not interoperable and will not be in the short-term period. Vendors have no interest at the moment to provide interoperability among their cloud infrastructures.

Almost all of the currently deployed grids have been publicly funded and operated. This is of course a very slow process, when compared to clouds that are instead privately funded and operated. Many companies have already invested and continue to invest a huge amount of money to develop cloud technologies; however, considering the limited amount of funding available to scientists, we must remark the excellent results in the field of grid computing.

Examining how grids are operated, it is easy to see that, since the beginning, there was a design requirement to build grid infrastructure tying together distributed administrative domains, possibly geographically spread. On the contrary, clouds are managed by a single company/entity/administrative domain. Everything is centralized, the infrastructure is hosted in a huge centre, and only the clients are actually geographically distributed. The architecture is basically client–server.

We note here that, despite its simplicity, the client–server architecture is still the most widely used in the commercial world, owing to the fact that it is very hard to beat hosted/managed services with regard to performance and resiliency: these services are currently geographically replicated and hugely provisioned and can guarantee/meet a specific Service Level Agreement. Highly distributed architectures, including peer-to-peer systems, while in principle are theoretically more

resistant to threats such as Denial of Service attacks etc., still do not provide the same performance guarantee. For instance, a P2P Google service is deemed to be impossible [42] (although Google uses internally their own P2P system, based on the concept of MapReduce [18]), and people who do it for business, today do not do it peer-to-peer, with the notable exception of the Skype service [49].

Coming to the main purpose of grid and cloud systems, it is immediately evident that for grid systems, the main raison d'etre is resource sharing. Cloud systems are instead intended to provide seamless access to a huge, scalable infrastructure. Given the different purpose they serve, it comes to no surprise that these systems target different classes of applications. As we have already noted, grids are well suited for scientific research and for the needs of high-end users. Clouds are mainly used today for data analysis, information processing, and data mining.

We conclude this section recalling that Many Task Computing (MTC) and High Throughput Computing (HTC) service providers and resource service providers can benefit from the economies of scale that clouds can deliver [53], making this kind of infrastructure appealing to the different stakeholders involved in.

## 1.5 Virtualization

The Virtual Machine (VM) concept [45] dates back to the 1960s; it was introduced by IBM as a mean to provide concurrent, interactive access to their mainframe computers. A VM was an instance of the physical machine and gave users the illusion of accessing the physical machine directly. VMs were used to transparently enable time-sharing and resource-sharing on the (at the time) highly expensive hardware. Each VM was a fully protected and isolated copy of the underlying system. Virtualization was thus used to reduce the hardware costs on one side and to improve the overall productivity by letting many more users work on it simultaneously. However, during the course of years, the hardware got cheaper and simultaneously multiprocessing operating systems emerged. As a consequence, VMs were almost extinct in 1970s and 1980s, but the emergence of wide varieties of PC-based hardware and operating systems in 1990s revived virtualization ideas.

Virtualization technologies represent a key enabler of cloud computing, along with the recent advent of Web 2.0 and the increased bandwidth availability on the Internet. The most prominent feature is the ability to install multiple OS on different virtual machines on the same physical machine. In turn, this provides the additional benefits of overall cost reduction owing to the use of less hardware and consequently less power. As a useful side effect, we also note here that virtualization generally leads to increased machine utilization. The main aim of virtualization technologies is to hide the physical characteristics of computing resources from the way in which other systems, applications, or end users interact with those resources. In this book, Lakshmi et al. [38] propose an end-to-end system virtualization architecture and thoroughly analyze it.

Among the many benefits provided by virtualization, we recall the ability to run legacy applications requiring an older platform and/or OS, the possibility of creating

a single system image starting from an heterogeneous collection of machines such as those traditionally found in grid environments, and faster job migration within different virtual machines running on the same hardware. For grid and cloud computing, virtualization is the key for provisioning and fair resource allocation. From the security point of view, since virtual machines run isolated in their sandboxes, this provides an additional protection against malicious or faulty codes.

Besides computing, storage may be virtualized too. Through the aggregation of multiple smaller storage devices characterized by attributes such as performance, availability, capacity and cost/capacity, it becomes possible to present them as one or more virtual storage devices exhibiting better performance, availability, capacity, and cost/capacity properties. In turn, this clearly enhances the overall manageability of storage and provides better sharing of storage resources.

A virtualization layer provides the required infrastructural support exploiting lower-level hardware resources in order to create multiple independent virtual machines that are isolated from each other. This layer, traditionally called Virtual Machine Monitor (VMM), usually sits on top of the hardware and below the operating system. Virtualization as an abstraction can take place at several different levels, including instruction set level, Hardware Abstraction Layer (HAL), OS level (system call interface), user-level library interface, and the application level.

Virtualizing the instruction set requires emulation of the instruction set, i.e., interpreting the instructions in software. A notable example is Rosetta [9], which is a lightweight dynamic translator for Mac OS X distributed by Apple. Its purpose is to enable legacy applications compiled for the PowerPC family of processors to run on current Apple systems that use Intel processors. Other examples include Bochs [1], QEMU [2], and BIRD [46].

HAL level virtual machines are based on abstractions lying between a real machine and an emulator; in this case a virtual machine is an environment created and managed by a VMM. While emulator's functionalities provide a complete layer between the operating system or applications and the hardware, a VMM is in charge of managing one or more VMs, and each VM in turn provides facilities to an OS or application as if it is run in a normal environment, directly on the hardware. Examples include VMware [3], Denali [55], Xen [4], Parallels [5], and Plex86 [6].

Abstracting virtualization at the OS level requires providing a virtualized system call interface; this involves working on top of the OS or at least as an OS module. Owing to the fact that system calls are the only way of communication from user-space processes to kernel-space, the virtualization software can control user-space processes by managing this system interface. Additionally, besides system's library, usually applications also link code provided by third-party user-level libraries. Virtualizing this library interface can be done by controlling the communication link between the applications and the rest of the system through well-defined API hooks. Therefore, it is possible to expose a different implementation using the same set of APIs. As an example, WINE HQ [7] and CrossOver [8] support running Windows applications respectively on Unix and Mac OS X.

Virtualization at the application is not based on the insertion of a virtualization layer in the middle. Instead, this kind of abstraction is implemented as an application