Enabling Technologies for Management of Distributed Computing Infrastructures

Daniel Espling

PhD Thesis, September 2013
Department of Computing Science
Umeå University
Sweden

Department of Computing Science
Umeå University
SE-901 87 Umeå, Sweden

espling@cs.umu.se

ISBN 978-91-7459-704-2
ISSN 0348-0542
UMINF 13.19

Printed by Print & Media, Umeå University, 2013

Abstract

Computing infrastructures offer remote access to computing power that can be employed, e.g., to solve complex mathematical problems or to host computational services that need to be online and accessible at all times. From the perspective of the infrastructure provider, large amounts of distributed and often heterogeneous computer resources need to be united into a coherent platform that is then made accessible to and usable by potential users. Grid computing and cloud computing are two paradigms that can be used to form such unified computational infrastructures.

Resources from several independent infrastructure providers can be joined to form large-scale decentralized infrastructures. The primary advantage of doing this is that it increases the scale of the available resources, making it possible to address more complex problems or to run a greater number of services on the infrastructures. In addition, there are advantages in terms of factors such as fault-tolerance and geographical dispersion. Such multi-domain infrastructures require sophisticated management processes to mitigate the complications of executing computations and services across resources from different administrative domains.

This thesis contributes to the development of management processes for distributed infrastructures that are designed to support multi-domain environments. It describes investigations into how fundamental management processes such as scheduling and accounting are affected by the barriers imposed by multi-domain deployments, which include technical heterogeneity, decentralized and (domain-wise) self-centric decision making, and a lack of information on the state and availability of remote resources.

Four enabling technologies or approaches are explored and developed within this work: (I) The use of explicit definitions of cloud service structure as inputs for placement and management processes to ensure that the resulting placements respect the internal relationships between different service components and any relevant constraints. (II) Technology for the runtime adaptation of Virtual Machines to enable the automatic adaptation of cloud service contexts in response to changes in their environment caused by, e.g., service migration across domains. (III) Systems for managing metadata relating to resource usage in multi-domain grid computing and cloud computing infrastructures. (IV) A global fairshare prioritization mechanism that enables computational jobs to be consistently prioritized across a federation of several decentralized grid installations.

Each of these technologies will facilitate the emergence of decentralized computational infrastructures capable of utilizing resources from diverse infrastructure providers in an automatic and seamless manner.

Populärvetenskaplig Sammanfattning

Den här avhandlingen handlar om två olika typer av system för att få ett stort antal datorer att kommunicera och samarbeta, *grid computing* och *cloud computing*.

Grid computing är en teknik för att koppla ihop ett stort antal datorer, servrar, för att kunna utföra beräkningsjobb som kräver mycket datorkraft. Exempel på sådana beräkningsjobb är att analysera experimentdata från forskningsområden som kemi, fysik eller biologi. Varje jobb skickas in till grid-systemet av respektive forskare (användare), och får ta en viss mängd tid och datorkraft. Hur mycket varje användare får använda grid-systemet bestäms normalt av en kommitté, som är utsedd att distribuera beräkningstimmarna för grid-systemet.

Namnet *grid computing* kommer från det engelska ordet *power grid* (elnätet), och det långsiktiga målet med grid-system är att användaren ska få tillgång till datorkraft från olika datorcenter utan att behöva bry sig om exakt varifrån datorkraften kommer. Servrarna på varje datorcenter kan se olika ut, exempelvis med olika sorters processorer, olika mycket minne, eller olika operativsystem (Windows, Linux, Mac OSX). Utmaningen med grid computing är att överbrygga de tekniska skillnaderna, och få tusentals olika datorer att samarbeta.

Cloud computing är en annan teknik för att förena datorresurser. Istället för beräkningsjobb som tar ett antal timmar så används cloud computing till datortjänster som inte är tidsbegränsade och som alltid förväntas finnas tillgängliga. Exempel på sådana tjänster är sökmotorer, sociala nätverk eller nyhetswebsidor. Till skillnad från grid computing så är cloud-system öppna för alla att använda, och man betalar för den datorkraft som man förbrukar.

Namnet *cloud computing* kommer från en tradition av att använda en bild av ett moln när man beskriver Internet utan att bry sig om de specifika datorerna som är inblandade. Precis som för grid-systemen så är målet med cloud computing att användaren inte ska behöva veta eller bry sig om var datorresurserna finns, vilken sorts servrar som används eller ens vilket företag som äger dem. Skillnaderna mellan grid- och cloud computing bottnar i de olika tillämpningarna (beräkningsjobb mot datortjänster), de ekonomiska modellerna (tilldelning av tid mot en öppen marknad för alla), samt underliggande tekniska skillnader i hur systemen är designade och byggda.

I avhandligen presenteras två nya tekniker som underlättar användadet av grid-system. Den första tekniken förbättrar prioriteringen av beräkningsjobb och får grid-systemet att bli mer rättvist, då användare som använt en mindre del av sin tilldelade

beräkningstid får en högre prioritet än de som använt stora delar av sin beräkningstid. Det gör att användningen av grid-systemet balanseras automatiskt. Den andra tekniken underlättar insamlandet av information från varje datorcenter om vilka jobb som har körts, av vem och hur längre. Informationen används exempelvis som underlag för kommitén för kommande tilldelningar av beräkningstid.

Förutom teknikerna för grid-system så presenterar den här avhandligen två nya tekniker för att underlätta distribuerad cloud computing. Den första nya cloud-tekniken gör det möjligt för användaren att grovt kunna specificera hur och var en datortjänst ska köra. Exempelvis kan användaren bestämma att tjänsten måste köras i ett visst land eller i en viss världsdel. Tekniken gör det möjligt för användaren att uttrycka krav eller önskemål, utan att kunna eller behöva specificera på precis vilken server som tjänsten ska köras. Den andra cloud-tekniken gör det möjligt för en datortjänst att automatiskt ta del av inställningar som är specifika för det datorcenter där tjänsten kör, och på så sätt förbättra samarbetet mellan tjänst och datorcenter.

Sammantaget så underlättar de här fyra utvecklade teknikerna användningen av sammankopplade stora datorsystem, både för beräkningsjobb och för datortjänster.

Preface

This thesis contains an introduction to the field and the papers listed below.

Paper I	L. Larsson, **D. Henriksson**, and E. Elmroth. Scheduling and monitoring of internally structured services in cloud federations. In *Proceedings of IEEE Symposium on Computers and Communications 2011*, pages 173–178, 2011.
Paper II	**D. Espling**, L. Larsson, W. Li, J. Tordsson, and E. Elmroth. Modeling and placement of structured cloud services. *Submitted*, 2013
Paper III	D. Armstrong, **D. Espling**, J. Tordsson, K. Djemame, and E. Elmroth. Runtime virtual machine recontextualization for clouds. In I. Caragiannis, et al., editors, *Euro-Par 2012: Parallel Processing Workshops*, volume 7640 of *Lecture Notes in Computer Science*, pages 567–576. Springer Berlin Heidelberg, 2013.
Paper IV	**D. Espling**, D. Armstrong, J. Tordsson, K. Djemame, and E. Elmroth. Contextualization: Dynamic configuration of virtual machines. *Submitted*, 2013.
Paper V	E. Elmroth and **D. Henriksson**. Distributed usage logging for federated grids. *Future Generations Computer Systems*, 26(8):1215–1225, 2010.
Paper VI	E. Elmroth, F. Galán, **D. Henriksson**, and D. Perales. Accounting and billing for federated cloud infrastructures. In *GCC '09: Proceedings of the 2009 Eighth International Conference on Grid and Cooperative Computing*, pages 268–275, Washington, DC, USA, 2009. IEEE Computer Society.
Paper VII	P.-O. Östberg, **D. Espling**, and E. Elmroth. Decentralized scalable fairshare scheduling. *Future Generation Computer Systems*, 29(1):130 – 143, 2013.
Paper VIII	**D. Espling**, P.-O. Östberg, and E. Elmroth. Integration and evaluation of decentralized fairshare prioritization (Aequus). *Submitted*, 2013

Note that the author changed surname from Henriksson to Espling in 2011.

The research presented herein was conducted using the resources of the High Performance Computing Center North (HPC2N) and the UMIT research lab. Financial

support was provided by the Swedish Research Council (VR) under contract 621-2005-3667, the Swedish Government's strategic research project eSSENCE, and by the European Community's Seventh Framework Programme ([FP7/2001-2013]) under grant agreements no. 215605 (RESERVOIR) and no. 257115 (OPTIMIS).

Publications by the author not included in the thesis:

- M. B. Yehuda, O. Biran, D. Breitgand, K. Meth, B. Rochwerger, E. Salant, E. Silvera, S. Tal, Y. Wolfsthal, J. Cáceres, J. Hierro, W. Emmerich, A. Galis, L. Edblom, E. Elmroth, **D. Henriksson**, F. Hernández, J. Tordsson, A. Hohl, E. Levy, A. Sampaio, B. Scheuermann, M. Wusthoff, J. Latanicki, G. Lopez, J. Marin-Frisonroche, A. Dörr, F. Ferstl, S. Beco, F. Pacini, I. Llorente, R. Montero, E. Huedo, P. Massonet, S. Naqvi, G. Dallons, M. Pezzé, A. Puliato, C. Ragusa, M. Scarpa, and S. Muscella. RESERVOIR - an ICT infrastructure for reliable and effective delivery of services as utilities. Technical report, IBM Haifa Research Laboratory, 2008.

- M. Lindner, F. Galán, C. Chapman, S. Clayman, **D. Henriksson**, and E. Elmroth. The Cloud Supply Chain: A Framework for Information, Monitoring, Accounting and Billing. In *2nd International ICST Conference on Cloud Computing (CloudComp 2010).*

- J. Tordsson, K. Djemame, **D. Espling**, G. Katsaros, W. Ziegler, O. Wäldrich, K. Konstanteli, A. Sajjad, M. Rajarajan, G. Gallizo, and S. Nair. Towards holistic cloud management. In D. Petcu and J. L. Vzquez-Poletti, editors, *European Research Activities in Cloud Computing*, pages 122–150. Cambridge Scholars Publishing, January 2012.

- G. Katsaros, G. Gallizo, R. Kübert, T. Wang, J. O. Fitó, and **D. Espling**. An integrated monitoring infrastructure for cloud environments. In *Cloud Computing and Services Science*, pages 149–164. Springer, 2012.

- G. Katsaros, J. Subirats, J. Oriol Fitó, J. Guitart, P. Gilet, and **D. Espling**. A service framework for energy-aware monitoring and VM management in clouds. *Future Generation Computer Systems*, 2012.

Acknowledgments

Time flies when you are having fun, and my time as a Ph.D. student both inside and outside of the office has seemingly passed by in seconds. There are a great number of people to thank for contributing to these outstanding years in Umeå:

I am very grateful to my supervisor, **Erik Elmroth**, for taking me on as a Ph.D. student and always being extremely supportive and friendly. Our group has grown quite a lot since I started in 2008, and despite the crew of your boat growing to four times its previous size, you still seem to find the time and enthusiasm to make everyone feel involved and important.

I would also like to thank my secondary advisor and colleague **Johan Tordsson** for always providing feedback, support, and friendly advice, and for all the interesting experiences we had during our FP7-projects. Thanks for shielding the rest of us from most of the administrative burden!

To **Lars Larsson**, my close friend and former office mate – thanks for the good times, both in D420 and around the globe. Even though the facilities in the new office are (way) nicer, I still miss sharing an office with you!

Thanks to my partner in crime, **Petter Svärd**, for the iOS adventures and for all the hockey-related activities and biased discussions. Now all we need is an app to make our team play better!

To all the members of our ever-growing research group (in no particular order) **Ahmed Ali-Eldin**, **Amardeep Mehta**, **Cristian Klein**, **Ewnetu Bayuh Lakew**, **Francisco Hernandez**, **Gonzalo Rodrigo Alvarez**, **Kosten Selome Tesfatsion**, **Lennart Edblom**, **Luis Tomás**, **Mina Sedaghat**, **Peter Gardfjäll**, **P-O Östberg**, and **Wubin Li**, thanks for creating an outstanding social and professional atmosphere and for contributing experiences and stories from so many different cultures and countries.

I am grateful to the **technical support and administrative staff** for their great work in helping us stay (somewhat) focused on our tasks. Special thanks are due to **Tomas Forsman** for all of his help with peripheral things that don't really form part of his job description, and to **Mats Johansson** for the exchange of "high-quality" literature.

Thanks to all the members of the Friday **fika group** for enhancing Fridays with a great variety of sweet delicacies. Similarly, I thank all of the **floorball players**, past and present, who participated in the Tuesday sessions that helped me to burn off the calories from said delicacies. To the White team for all of the great experiences and team-play, and to the Red team for hanging in there despite not winning more than two of the last ten seasons.

To all of the staff at the **Department of Computing Science**, **UMIT**, and **HPC2N** thanks for providing a great working environment.

Thanks to all of our collaborators in the RESERVOIR and OPTIMIS projects for all the exciting endeavors and for giving me the chance to travel around Europe. Special thanks to **Django Armstrong** for a wonderful collaboration and all the fun we had while performing it.

I already miss all of my co-students and friends from **C03** and its surroundings. Thanks for providing a great social environment during my first years in Umeå! I am also especially thankful to **Henrik Tegman** and **Per Westerlund** for all of our memorable and entertaining social activities.

To the **Zetterström**, **Mikaelsson** and **Larsson** families, thanks for being great friends. We will miss you guys when we move away!

I owe a special mention to the close friends from my childhood, **Peter Bomark**, **Tomas Eriksson**, and **Johan Öjemalm**, with whom my interest in technical subjects was sparked. Thanks to **Peter Sagebro**, **Lars Eldborn**, and everyone else who helped to ensure that I had my fill and more of sports and gaming-related activities during my spare time.

Thanks to **Solweig**, **Jonas**, **Sara**, **Frida**, **Niklas**, **John**, **Brita**, **Lennart**, and **Kai** for letting me become a part of the Palm family. I have felt welcome since the first day we met, and I appreciate being included in your traditions and activities.

I am hugely thankful to my loving family – **mom**, **dad**, and my grand parents **Aina**, **Östen**, and **Elin** for all of your love and support. Going north to visit any of you always feels like coming home. I am also very grateful to my brother **Tommy** for our shared experiences growing up and to you and **Kristin** for our time together here in Umeå. It has been great living in the same city, and I hope that we will live close to one-another in the future as well!

Finally I would like to thank my wife **Maria** for an amazing first decade together in Umeå. I am extremely grateful for everything we have experienced together since we met as first-year students, ranging from hardship to the euphoria of our wedding day. Thanks for all the love and support, work-related and otherwise. I am truly looking forward to the adventures of *Team Espling* that lie ahead!

Umeå, September 2013
Daniel Espling

Contents

1 Introduction **1**
1.1 Aims 2

2 Distributed Computing Infrastructures **3**
2.1 Grid Computing 4
2.1.1 Federations of Grids 6
2.2 Cloud Computing 7
2.2.1 Cloud Characteristics 8
2.2.2 Service Models 9
2.2.3 Deployment Models 11
2.2.4 Hybrid Deployment Models 13
2.2.5 Virtualization 17
2.2.6 Cloud Service Lifecyle 19

3 Management in Distributed Infrastructures **21**
3.1 Scheduling and Placement 21
3.1.1 Grid Scheduling 21
3.1.2 Cloud Placement 23
3.2 Monitoring 25
3.3 Elasticity 26
3.4 Accounting and Billing 27
3.5 Autonomic Computing 28

4 Thesis Contributions **31**
4.1 Service Structure 31
4.2 Contextualization 32
4.3 Accounting in Multi-domain Infrastructures 33
4.4 Decentralized Global Fairshare 34

5 Future Work **37**
5.1 Multi-domain Cloud Service Management 37
5.2 Multi-domain Accounting 39
5.3 Decentralized Global Fairshare 39

6 Outlook **41**

Paper I **67**

Paper II **77**

Paper III **91**

Paper IV **105**

Paper V **121**

Paper VI **133**

Paper VII **149**

Paper VIII **167**

Chapter 1

Introduction

The goal of making computing capacity available in the same way as utilities like water or electricity was arguably first put forward in the early sixties by John McCarty [86, 88]. Several paradigms based on this vision were introduced in the fifty years that followed, all of which were intended to increase the viability of supplying computing power as a utility. In most cases, the new paradigms do not overlap perfectly in scope with their predecessors, leaving niches that enable several paradigm generations to coexist.

Two of the most recently developed paradigms for computing as a utility are *grid computing* and *cloud computing*. In general, one can refer to systems based on these paradigms as *grids* and *clouds*, respectively, and use the terms *site* or *provider* to describe a single infrastructure supplier. Fundamentally, grids and clouds are both ways to group existing (heterogeneous) computer resources into an abstract pool of computing capacity, and to then make those resources available to users in the form of a coherent virtual infrastructure. Although the two paradigms were initially developed to fulfill similar functions, grids have evolved into reliable, high performance platforms that are mostly used for large-scale scientific computing *jobs*, while clouds are largely used as remote hosting and execution tools for various software packages that are often referred to as *services*. Chapter 2 describes grids and clouds in more detail.

Individual grids and clouds can be joined to form even larger resource pools through collaborations. These multi-domain environments pose additional challenges in terms of resource management, stemming from their technical heterogeneity, the need for decentralized decision making given that each domain may have different objectives, and the potential lack of information regarding the state and availability of remotely hosted resources. There are several different collaboration models (discussed in Section 2.2.3), each with their own set of challenges [74, 80]. The increasing scale and complexity of grids and clouds is creating a need for sophisticated management processes with minimal requirements in terms of human governance.

The contributions of this thesis relate to management processes for grids and clouds, and to the challenges of adapting and developing these processes to work in multi-domain environments. Paper I [137] and Paper II [77] describe the use of explicit service structuring to relate different components in a cloud service. Decisions related to management and placement (mapping service components to physical machines) can be based on the internal structure of the service to optimize execution during run-time. Paper III [15] and Paper IV [76] describe and discuss an emerging technology for run-time adaptation of cloud service components to a particular infrastructure where it is being hosted, allowing generically designed services to be automatically customized to suit their execution environments during run-time. Paper V [65] and Paper VI [69] present work on accounting and monitoring data management in multi-domain infrastructures. The development and evolution of a decentralized system for job prioritization in multi-domain grids (first introduced by Elmroth and Gardfjäll [66]) are presented in Paper VII [168] and Paper VIII [78].

1.1 Aims

Existing distributed computational infrastructures are commonly surrounded by technical, administrative, and political boundaries that complicate resource exchanges between different infrastructures. The aim of the work presented in this thesis is to explore, design, and develop new technologies for managing jobs and services in distributed computational infrastructures that will enable and facilitate the use of resources and infrastructures from multiple administrative domains.

Chapter 2

Distributed Computing Infrastructures

There is a significant demand for large-scale computational resources at all levels of modern society, ranging from the globe-spanning internet services used by multi-national companies and large scientific projects to the advanced simulation systems used in the design industry. These resources are traditionally hosted in-house, either in the form of computational *clusters* consisting of regular servers connected via high-speed network connections or as *super computers*, complex server infrastructures built using customized hardware.

Distributed computing offers an alternative to the in-house hosting and management of computer systems by offering access to a remote computational infrastructure. The remote infrastructure may either be hosted at a single location, or may be a virtual infrastructure composed of resources from several different physical locations and administrative domains. A multi-domain infrastructure can potentially incorporate more resources than would be feasible in a single location, and can offer benefits in terms of capacity, fault tolerance, and geographical dispersion. The main drawback of multi-domain infrastructure is the added complexity associated with hosting a system that spans regional, administrative, and often technological boundaries.

The distinction between grids and clouds is not perfectly defined, and the flexible definitions of both terms create considerable potential overlap. Moreover, the two technologies can be combined in some cases. For example, a deployment of the Sun Grid Engine (SGE) [94] on a cloud infrastructure is one of the use cases of the RESERVOIR project [185]. Other examples include the work of Keahey et al. [124], in which an abstract computing infrastructure is hosted across resources from several cloud providers.

However, despite the blurred boundaries between them, grids and clouds have become two separate paradigms with differing areas of focus:

- Grids are designed to support the sharing of pooled resources (normally high performance parallel computers) owned and administered by different organizations, primarily targeting users whose requirements exceed the capabilities of commodity hardware and may involve thousands of processor cores or hundreds of terabytes of storage.
- The development of cloud technologies is driven by economies of scale [198], since increasing the utilization of existing (often commodity) hardware resources reduces the operating expenses incurred by infrastructure providers, enabling them to offer hardware leasing at prices comparable to in-house hosting.

The different scopes of the two paradigms mean that they also differ in terms of their associated business models, architectures, and resource management needs. In the context of this thesis, the most important difference between them relates to the lifecycles of grid jobs and cloud services. In general, grid jobs are computational tasks executed on infrastructures with very high (combined) performance that give the user exclusive access to (a subset of) the available resources for a job until it is completed and then assign the resources to the next job in the queue. Cloud services, on the other hand, are expected to start almost immediately after they are submitted and to run until the service is explicitly canceled. The service runs on its assigned share of resources, which may increase or decrease during service execution. Conceptually, the difference between the ways in which the two resource types are managed is analogous to that between batch-processing (grids) and time-sharing [180] (clouds).

The following two sections provide a general overview of the two paradigms. For more in-depth comparisons of grids and clouds, see the work of Foster et al. [88], Sadashiv and Kumar [189], and Zhang et al. [242].

2.1 Grid Computing

Grid computing [86] is based on the interconnection of distributed and decentralized computational resources to form a cohesive infrastructure for large-scale computation. From its initial conception as a tool for offering general-purpose computational capacity as a utility, grid computing has evolved into a group of enabling technologies for large-scale scientific endeavors such as the LHC, the World-wide Telescope [210], and the Biomedical Informatics Research Network [100]. In many cases, grids serve both as tools for sharing raw computational resources and as mechanisms for sharing data from important scientific instruments.

As a concept, grid computing has grown to encompass many different tools for a variety of tasks, and can be seen as a group of related technologies rather than a single utility. Together with the fact that there is no absolute distinction between grids and other distributed environments, this has created some confusion regarding what should be regarded as a grid. While many definitions have been proposed [34, 206], the most widely used is that put forward by Foster [84]. Foster's definition takes the form of a three point checklist and states that a grid is 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".

Foster's definition is widely accepted but has not been standardized, and there are major grid efforts such as the grid supporting the Large Hadron Collider (LHC) [2] in which resources are under centralized control while still being referred to as a grid. The work presented in this thesis is based on a view that is very similar to Foster's definition, with a particular emphasis on the decentralized control of resources and the autonomy of participating sites.

The overall purpose of grid computing is to interconnect and unify resources that may be owned by different actors in different countries, have different physical characteristics (CPU frequency, CPU architecture, network bandwidth, disk space, etc.), and run different operating systems and software stacks. These resources are consumed by users, who are commonly organized into collaborating scientific communities that are known as Virtual Organizations [87].

A variety of middlewares [11, 19, 64, 85, 131, 207, 213, 222] are used as intermediate software layers for job submission and job management in grids. However, the vast set of different middlewares has created interoperability problems between the middlewares themselves [74], giving rise to an additional niche for software to ease the burden of working with systems that span multiple different middlewares [7, 41, 68, 97, 188].

Grid jobs can normally be seen as a self-contained bundle of computational jobs and input data that can be executed independently across different nodes to generate a set of output data. The jobs are batch-oriented and normally no user interaction with the job is required or even possible during their execution time, which limits the scope of applications suitable for execution on grids. The Job Submission Description Language (JSDL) [12] is a widely accepted standard for specifying job configuration properties such as hardware requirements, execution deadlines, and the sets of input and output files that are required or generated by the computations.

Grid resources are primarily used by specific research communities. This is due to a range of factors including their project-oriented business models, technical problems (which are often related to software dependencies), and interoperability issues [17, 88]. Despite their drawbacks, grids have enabled these communities to address problems that could would be intractable with other computational resources or scientific tools. A comprehensive overview of grid computing and its implications and uses in several fields (bioinformatics, medicine, astronomy, etc.) has been written by Foster and Kesselman [86]. Although this book was first published in 2004, the conceptual aspects of grids have not changed appreciably since.

In early 2013, there was a breakthrough in the search for the Higgs boson [1] that led to the successful identification of a key missing component from the Standard Model of physics [95]. The Worldwide LHC Computing Grid

(WLCG) [181, 235], which consists of more than 150 facilities spread over around 40 countries, was essential in analyzing and managing the vast quantities of data generated by the particle collision experiments, and grid computing is acknowledged to have been one of the key enabling technologies in the search for the Higgs boson. The WLCG is not a single, dedicated infrastructure but a federation of grids from all across the globe.

2.1.1 Federations of Grids

In a federation of grids [32, 89, 140, 173], the resources of several stand-alone grid infrastructures are made available for global use. Apart from the technical benefits of a united infrastructure, the formation of federations may be motivated by political objectives such as the consolidation of resources and the promotion of collaboration. Two notable initiatives that are aiming to unify Europe's national grids are EGEE [138, 139] and the European Grid Initiative [132]. Conway's adage [48] that the design of a computer system reflects the communication structure of the organization that produced it is applicable also to federated grids – unifying efforts in politics and governance often form the basis for unifying technological efforts.

The relationship between clusters, grids, and a federated grid is illustrated in Figure 1. Computational resources are joined together into clusters, and resources from one or more clusters can be united to form a grid. Resources from several grids in turn form a federation of grids. Individual resources are controlled by local resource management software at the cluster level. Local resource management systems manage grid-wide job admission and authentication, accounting of finished jobs, and the higher-level scheduling [72, 113] of grid jobs to resource sites. Grid systems also manage the inherent heterogeneity caused by uniting computational resources from several clusters. In a federation of grids, the autonomicity of each grid site is retained and all collaborating grids retain the functionality required to process and manage incoming grid jobs. The jobs executed as a part of the federation are normally submitted on the same premises as those of regular grid users.

As discussed by Field et al. [81], there are a number of challenges that arise from collaborations such as federated grids. Many of these challenges relate to interoperability and the heterogeneity that arises from the use of different grid middleware systems. Parts of this thesis concern meta-data management in federated grid infrastructures. This meta-data includes the usage records associated with jobs processed by the infrastructure, which are commonly used to monitor user resource consumption and schedule future jobs. When managing a federated grid, it is necessary to gather data from each collaborating site in order to establish a coherent picture of the flow of data within the federation. Paper V relates to the management of federated meta-data for accounting purposes, while Papers VII and VIII deal with the challenge of decentralizing the scheduling of incoming jobs across a multi-domain grid based on global usage quotas and previous usage.

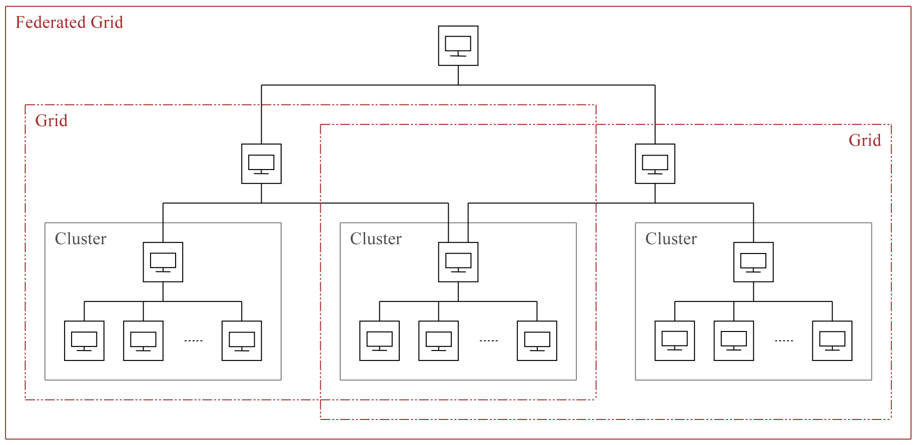

Figure 1: Overview of clusters, grids, and federated grids. Clusters span computational resources; grids span resources from one or more clusters and can be joined to form federated grids. Illustration from Östberg [167].

2.2 Cloud Computing

Cloud computing has emerged as a broad concept for the remote hosting and management of applications, platforms, or server infrastructure. The term *cloud computing* originates from the custom of representing computer networks with a drawing of a cloud and thereby concealing the resources' exact locations as well as the nature of the connections between them [192]. The same analogy applies to compute clouds; the location and other underlying details of the remote resources are abstracted and hidden from the users, who interact with resources running somewhere "in the cloud".

There are many different opinions on what constitutes a cloud and what distinguishes a cloud from a grid. During the last few years, the National Institute of Standards and Technology (NIST) [154] in the United States has attempted to progressly establish a unified definition of cloud computing to facilitate its characterization, and has actively solicited feedback on the draft definition from academics and various industrial organizations. The NIST definition has since become a de-facto standard, but it is important to recognize that it was established in a progressive fashion and builds on previous work. Notable early work on defining cloud computing was undertaken by a wide range of authors, including Weiss [229], Geelan [93], Vaquero et al. [224], Gruman and Knorr [102], Haaff [53], and McFedries [153]. The final NIST document [154] defines cloud computing as:

> *"... a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (networks, servers, storage, applications, and services) that can be rapidly*

provisioned and released with minimal management effort or service provider interaction."

This definition is general enough to encompass practically all cloud approaches. The model is further subdivided into a set of essential characteristics, service models, and deployment models, all of which are discussed below.

2.2.1 Cloud Characteristics

The essential characteristics of cloud computing as defined by the NIST [154] are reprinted verbatim below:

On-demand self-service. A consumer can unilaterally provision computing capabilities, such as server time and network storage, as needed automatically without requiring human interaction with each service provider.

Broad network access. Capabilities are available over the network and accessed through standard mechanisms that promote use by heterogeneous thin or thick client platforms (e.g., mobile phones, tablets, laptops, and workstations).

Resource pooling. The provider's computing resources are pooled to serve multiple consumers using a multi-tenant model, with different physical and virtual resources dynamically assigned and reassigned according to consumer demand. There is a sense of location independence in that the customer generally has no control or knowledge over the exact location of the provided resources but may be able to specify location at a higher level of abstraction (e.g., country, state, or data center). Examples of resources include storage, processing, memory, and network bandwidth.

Rapid elasticity. Capabilities can be elastically provisioned and released, in some cases automatically, to scale rapidly outward and inward commensurate with demand. To the consumer, the capabilities available for provisioning often appear to be unlimited and can be appropriated in any quantity at any time.

Measured service. Cloud systems automatically control and optimize resource use by leveraging a metering capability[1] at some level of abstraction appropriate to the type of service (e.g., storage, processing, bandwidth, and active user accounts). Resource usage can be monitored, controlled, and reported, providing transparency for both the provider and consumer of the utilized service.

Together, these characteristics establish the cloud as a self-service provisioning infrastructure whose managerial processes are automated and optimized without human intervention. All of the work presented in this thesis (whether

[1]Typically this is done on a pay-per-use or charge-per-use basis.

related to grid- or cloud computing) focuses on automization of managerial processes.

Early cloud definitions such as that proposed by Vaquero et al. [224] also considered the ability to guarantee capacity to consumers through Service Level Agreements (SLAs) to be a fundamental property of clouds. This requirement is not represented in the NIST definition, but there are implicit assumptions that the capabilities obtained from the provider correspond to the resources being provisioned. In the context of this work, SLAs or similar agreements are fundamental inputs for the automated scheduling processes that function as constraints and are used to differentiate between favorable and unfavorable system states. Scheduling and other management processes for distributed infrastructures relating to the characteristics listed above are discussed in Chapter 3.

2.2.2 Service Models

The automated provisioning of resources and services in a way that satisfies the essential cloud criteria can be done at several different layers of the software stack. Cloud computing offerings are commonly subdivided into three different service models:

Infrastructure as a Service (IaaS)
In IaaS solutions, hardware computing resources are made available to consumers as if they were running on dedicated, local machines. The impression of dedicated hardware is commonly achieved by utilizing hardware virtualization techniques, making it possible to host several virtualized systems on a single physical machine. Some notable IaaS providers are the Amazon Elastic Compute Cloud (EC2) [9], Rackspace [177], and Windows Azure [159].

Platform as a Service (PaaS)
Instead of offering access to (virtualized) hardware resources, PaaS systems offer deployments of applications or systems designed for a specific platform, such as a programming language or a custom software environment. PaaS systems include Google App Engine [98] and Saleforce's Force.com environment [230]. Ongoing projects such as 4Caast [91] and CumuloNimbo [118, 172] are developing new PaaS platforms aimed at simplifying the hosting of multi-tier applications and increasing the consistency and scalability of de-centralized service hosting, respectively.

Software as a Service (SaaS)
Web-based applications and services such as Microsoft Office Live [157], Google Apps [99] (not to be confused with App Engine), and the gaming platform OnLive [163] are available to online consumers without requiring them to install and manage the software locally. The software is instead hosted and managed on remote machines, making it possible to run

programs (including graphically intensive computer games) on remote servers instead of the local machine.

A common misconception is the assumption of an intrinsic relationship between different service models. For example, a PaaS system may be hosted on top of an IaaS infrastructure, but is not required to be so. This distinction is important because it enables us to reason about SaaS and PaaS systems without assuming underlying layers of cloud based infrastructures.

The cloud-related work presented in this thesis focuses on the management of clouds at the IaaS level. The work on service structure and contextualization (see Sections 4.1 and 4.2) may also be applicable at the PaaS level, and this area will be investigated more extensively in forthcoming studies.

2.2.2.1 IaaS Clouds

In this work, we identify three main actors who are relevant to cloud infrastructures (as shown in Figure 2): an *Infrastructure Provider* (IP), a *Service Provider* (SP), and the *End Users*. The IP owns and manages the physical resources and any supporting software that is required for infrastructure management. An SP provides a software service that is hosted by provisioning resources from one or more IPs. End Users are consumers of the service offered by the SP. Note that this separation is not present in the NIST definition, in which providers of both types are referred to as service providers. The separation between IPs and SPs is only directly relevant for IaaS clouds, but the distinction is important when discussing deployment models where the actors have different roles and perspectives.

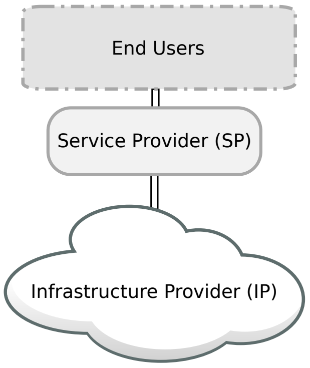

Figure 2: Three main actors for cloud IaaS: *Infrastructure Providers* make resources available to *Service Providers*, who in turn offer a software service to *End Users*.

Even though the actors are conceptually separate, a single organization or entity can fulfill more than one role at a time. For example, Amazon EC2 [9] is an IP that is used to host many different SPs. As of 2010, one of the services it hosts is the main Amazon bookstore website, for which Amazon is also the SP.

There is normally a many-to-one relationship between SPs and IPs: a single IP often hosts services from more than one SP at a time in a multi-tenant manner. Hardware virtualization techniques (see Section 2.2.5) are commonly used to isolate services and minimize interference. In some scenarios, a single SP may employ resources from more than one IP. Differences in the relationships between SPs and IPs create different cloud deployment models, as outlined in Section 2.2.3.

Security and privacy concerns are commonly seen as the main weaknesses of clouds [13, 88, 119]. Compared to grids, where access is usually preceded by face-to-face identity validations and certificate generation, clouds have a relaxed security model that is reminiscent of regular Web sites. It is common for clouds to use Web-based forms for sign up and management, and e-mails for password retrieval [88]. This relaxed security is very beneficial in terms of usability, but limits the trust of major companies considering using clouds for business-sensitive applications. While there is ongoing work on improving cloud security in general (see e.g. Christodorescu et al. [45] or Kandukuri et al. [119]), the use of privately hosted and managed clouds is one option for dealing with sensitive data while still gaining some of the benefits of conventional cloud systems.

Privacy concerns are often related to the physical location at which the data is stored. When using cloud-based storage platforms such as Dropbox [59], the underlying file data is stored on resources belonging to international IPs [58]. Some legislative bodies have prohibited the use of cloud-based storage solutions for governmental or other sensitive data, because the storage of such information at a remote location means that its confidentiality cannot be guaranteed. The work on structured services presented in this thesis (Papers I and II) provides a way of addressing this issue by using geographical placement constraints, which ensure that sensitive data is stored within a specified geographical region.

2.2.3 Deployment Models

There are several different deployment models for clouds, representing different relationships between an SP and one or more IPs. The NIST definition discusses three deployment models involving a single SP and a single IP, and classifies all more complex multi-participant deployment scenarios as *hybrid clouds*. In the context of this thesis, the distinction between different hybrid cloud models is important; a more detailed discussion of multi-participant deployment scenarios based on the work done in the RESERVOIR [183] and OPTIMIS [80] projects is provided in the later parts of this section.

The three scenarios involving a single SP and a single IP identified in the NIST definition are *public clouds*, *private clouds*, and *community clouds*. A public cloud (illustrated in Figure 3) is the baseline model for clouds, where one or more SPs share a publically available cloud infrastructure in a metered

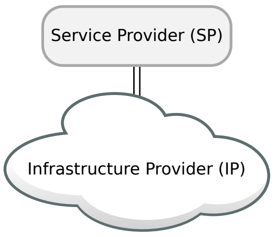

Figure 3: Public cloud deployment. The SP is one of many tenants on a publically available and remotely hosted infrastructure (the IP).

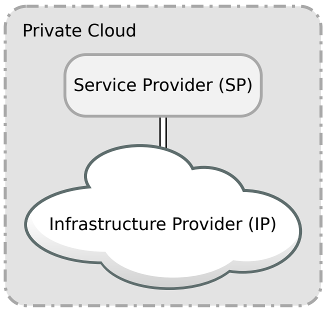

Figure 4: Private cloud deployment. The SP is assigned exclusive access to (parts of) a cloud infrastructure. The resources can either be hosted locally or as a designated part of a shared infrastructure.

and rapidly provisioned manner.

The NIST definition distinguishes *community clouds* from public clouds. Community clouds are clouds in which the infrastructure is dedicated to a specific community rather than being offered to the public for revenue reasons. Community clouds can be hosted either internally or externally. Briscoe and Marinos [39] discuss community clouds from a distributed perspective, whereby resources are provided by participants using a peer-to-peer [21] model rather than being centralized.

Private clouds, as shown in Figure 4, are cloud deployments hosted within the domain of an organization or at dedicated resources that are not made available for use by the general public [13]. Such deployments circumvent many of the security concerns related to hosting services in public clouds by keeping the data and computations within an isolated security domain. Virtual private clouds that rely on VPNs and cryptography have also been proposed [233].

Similarly to grids, the resources available to community- and private clouds need to be shared among the users in a fair manner rather than in the economical fashion of public clouds. This creates a different set of challenges in resource management – whereas public clouds can focus on maximizing utilization, private and community clouds ideally need to maximize utilization while retaining fairness. Public clouds sometimes employ different service levels where less prioritized services (such as Amazon Spot Instances [8]) can be dynamically neglected in favor of more highly prioritized ones, mitigating the problem of running low on resources. Another alternative for coping with a short supply of computational resources that can be used in private clouds involves offloading some of the workload to a remote IP by collaborating with external cloud providers under one of the hybrid cloud models.

2.2.4 Hybrid Deployment Models

The economical model of clouds is a key enabler for hybrid models; the rapid, self-serving provisioning of resources is conceptually identical regardless of whether the consumer is a regular SP or a remote IP. Therefore, hybrid clouds can be formed without the need for prior resource exchange agreements. In formal collaborations, however, the use of resources between IP sites may be governed by separate SLAs or framework agreements [38] that stipulate the terms of resource exchange between IPs. In hybrid cloud models, the IP site with whom the SP communicates is referred to as the *primary* site. Any other collaborating sites are referred to as *remote* sites. The control of the service and the responsibility towards the SP remain associated with the primary site regardless of where the service is executed, and the primary site is also responsible for ensuring that SLAs are maintained or compensated for.

Hybrids that incorporate elements from multiple deployment scenarios can be used to overcome the limitations that may be encountered in single provider usage scenarios. For example, to avoid the problem of finite resources in private clouds, such clouds may temporarily employ the resources of external public cloud providers. These *bursted private clouds* [203] combine the security and control advantages of private clouds and the seemingly endless scalability of public clouds. However, such deployments require very sophisticated placement policies to guarantee the integrity of the system, ensuring that only insensitive parts of the service are hosted on the public infrastructure. A bursted private cloud is illustrated in Figure 5.

Sotomayor et al. [203] outlined the general concept of a bursted private cloud and have provided an overview of the different cloud technologies and the extent of their support for this model. They have developed an open source software stack for cloud infrastructures known as OpenNebula [202], which can be used to create hybrid cloud solutions based on a private infrastructure and a set of cloud drivers that are used to burst specific tasks to various external providers such as Amazon EC2 [9] or ElasticHosts [63].

As is the case with grid computing, the use of multiple clouds introduces heterogeneity problems that can only be resolved through standardization. Native (hardware) virtualization is a vital first step towards standardization at the lowest hardware level. In addition, efforts have been made to create standardized and general formats for specifying virtual machines and virtual hard drives [55, 158] as well as general cloud Application Programming Interfaces

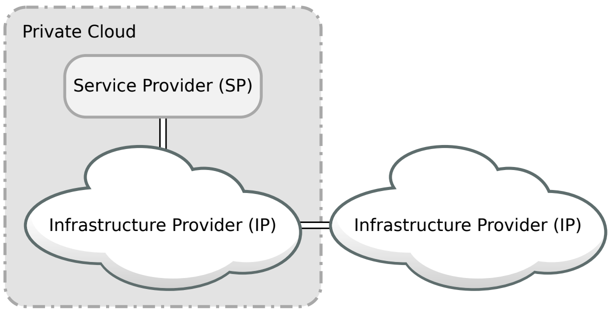

Figure 5: If needed, private clouds may outsource the execution of less sensitive tasks to a public cloud, creating a hybrid cloud system that is commonly referred to as a bursted private cloud.

(APIs) [54, 164, 165], but none of these standards has yet received general acceptance. Recent work on cross-infrastructure abstraction layers [26, 82, 232] has produced unifying software layers that hide the specifics of the underlying cloud infrastructures to enable cloud services to be designed and built in a unified way. However, these technologies have also not been widely taken up and further standardization efforts will probably be required to establish a consensus in terms of which abstraction layer technology will be adopted.

Compatibility issues aside, there are a number of operational challenges imposed by the use of hybrid clouds. Since each site retains complete autonomy, including over things such as policies and objectives, the internal workings of each site are largely obscured to the other sites that are participating in the collaboration. This means that each site only has detailed knowledge of its own local resources with at best incomplete information regarding the state of the other sites. Service management decisions across clouds must therefore be based on probabilities and statistics rather than complete information. Another challenge that is not encountered with public or private clouds is that sites participating in collaborations may experience external events that affect the state of their services and the availability of infrastructural resources. For example, a remote site may trigger the withdrawal of services running on the remote infrastructure, forcing the primary site to re-plan the distribution of tasks across the remaining available infrastructure.

2.2.4.1 Federated Clouds

Federations of clouds (Figure 6) are formed at the IP level, making it possible for infrastructure providers to make use of remote resources without involving or notifying the SP that owns the service. Gaining access to more resources is not the only potential benefit of placing VMs in a remote cloud. Other motivations include fault tolerance, economical incentives, and potentially the ability to satisfy technical or non-technical criteria (such as those relating to geographical

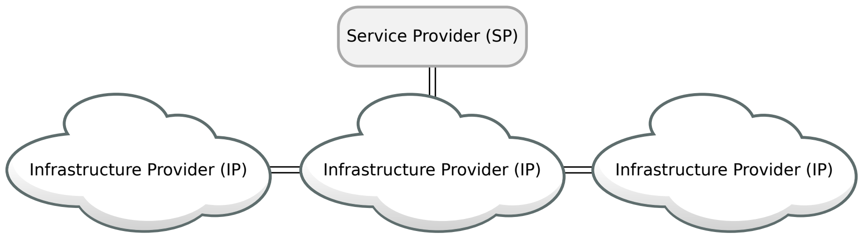
Service Provider (SP)
Infrastructure Provider (IP)
Infrastructure Provider (IP)
Infrastructure Provider (IP)

Figure 6: Federated cloud deployment. The SP interacts with the primary IP, which in turn may offload parts of the workload to one or more remote IPs.

location) [182], which might not be possible using local infrastructure.

The provisioning of remote resources through federations can be done concurrently across several remote sites, using factors such as cost, energy efficiency, and previous experience to decide which resources to use [80]. In some cases, a service may be passed along from a remote site for execution at a third party site, creating a chain of federations. As each participant in the chain is only aware of the closest collaborating sites, special care has to be taken with VM management and information flow in such scenarios [71].

The RESERVOIR project [182, 183, 184, 238] focuses on creating and validating the concept of cloud federations across several infrastructure providers. One of the contributions of this thesis is early work on cloud accounting for multi-site collaborations such as federations (Paper IV). This work was conducted as part of the RESERVOIR project, and a more detailed description of the contribution can be found in Section 5.2. Other results from this project include the design and creation of Virtual Application Networks (VANs) [105]. These overlay networks extend the ideas of authors such as Tsugawa and Fortes [220] and offer one way of enabling VMs that form a part of an internal private network to be migrated to another site within a federation without being disconnected. These VANs can be used to manage monitoring information for services that span several cloud sites, as discussed in Section 3.2.

2.2.4.2 Split Cloud Deployment

In a split cloud deployment the SP interacts directly with several different IP infrastructures. In this case, which is illustrated in Figure 7, the SP is responsible for planning, initiating, and monitoring the execution of services running on different IPs. Any interoperability issues must be detected and managed by the SP, which may limit the range of sites that can be used in a multi-cloud deployment. This is sometimes also referred to as a *Multi-cloud* scenario [80].

The automatic selection and management of different alternatives using *brokers* is a well known approach for distributed infrastructures such as grid computing [75, 126]. As shown by Ferrer et al. [80] and Tordsson et al. [219],

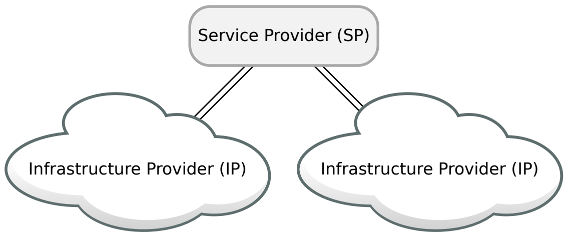

Figure 7: In a split cloud deployment, the SP itself may control and decide the deployment of a service using several different IPs.

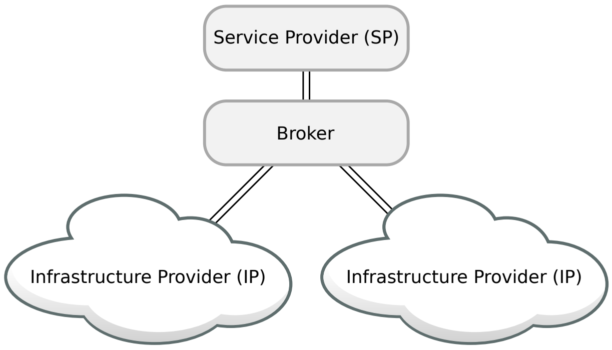

Figure 8: A dedicated broker component is used by the SP to simplify the deployment and management process across several infrastructures.

brokers can also be used as intermediate components when an SP interacts with several cloud IPs. In this case, illustrated in Figure 8, the broker is placed between the SP and the IPs. In fact, the broker may act as an SP to the IP and as an IP to the SP, transferring the complexity of dealing with multiple simultaneous cloud deployments to the broker itself [80]. Tordsson et al. [219] provide an overview of the process and discuss their own practical experiences of cloud brokering, including the quantitative performance gains that have been achieved by brokering resources belonging to different cloud providers.

The aim of the OPTIMIS [80] project is to create a toolkit of components that is sufficiently flexible to support any deployment scenario, including federated and split cloud deployments. In a scenario such as a brokered split deployment, the IP that will be used is not selected until the service is deployed, and subsequent optimization procedures initiated by the broker may even change the IP used for hosting during run-time. This requires the services to be designed and constructed in a general way to ensure maximum compatibility with most IPs. The work on cloud service contextualization presented in this thesis (Papers III and IV) was conducted as part of the OPTIMIS project and resulted in the development of a technique that allows parts of cloud services to be dynamically adapted to their current execution environment. For more information on service contextualization, see Section 2.2.6. The contributions to service contextualization presented in this thesis are discussed in Section 4.2.

2.2.5 Virtualization

Hardware virtualization techniques [20, 175] provide tools for dynamically segmenting physical hardware, making it possible to run several different *Virtual Machines* (VMs) [194] on a single unit of physical hardware simultaneously. Each VM is a self-contained unit, including an operating system, and booting a VM is very similar to powering on a normal desktop computer. The physical resources are subdivided, managed, and made available to the VMs through a *Hypervisor* (also known as a VM Monitor [175]).

The concept of virtualization dates back to the late 1960s but remained largely unused for quite some time until it became the subject of renewed interest in the late 1990s. The oft-cited reason for this delay is that the widespread x86 processor technology that dominated the market during the intervening period was cumbersome and less well suited to virtualization than its predecessors. Another important factor is that processors became cheap enough to warrant increasing the number of computers when additional machines were required rather than focusing on virtualization [128]. Efficient methods for software-based virtualization on x86 platforms were developed in the late 1990s, and processors with hardware support for virtualization became available in the mid 2000s [3, 35].

Virtualization is the underlying packaging and abstraction technology for most IaaS clouds, and there are several initiatives aimed at enabling the use of virtualization in HPC and grid computing. For example, Keahey et al. [122] recommend the use of VMs in grids as a way of more fully satisfying quality of service requirements and facilitating portability between execution environments. Haizea [201] is a scheduling framework that uses VMs as a tool to maximize utilization while still supporting advance reservations by suspending and resuming VMs. In this way, gaps in the execution schedule between jobs can be utilized by resuming a previously suspended VM and executing that VM for a short period of time. In their analysis and comparison of virtualization technologies for HPC, Walters et al. [227] identify four different categories of virtualization:

Full Virtualization Uses a hypervisor to fully emulate system hardware, making it possible to run unmodified guest operating systems at the expense of performance. Well known implementations include VirtualBox [228], Parallels Desktop [171], and Microsoft Virtual PC [110].

Native Virtualization Native virtualization makes use of hardware support in processors to make the costly translations of instructions from full virtualization in hardware instead of software. Known technologies includes KVM [128], Xen [20], and VMware [225].

Paravirtualization In Paravirtualization [231], the operating system in the VM is modified to make use of an API provided by the hypervisor to achieve better performance than full virtualization. Xen [20] and VMware [225] are two well established technologies that support Paravirtualization.

Operating System-level Virtualization Unix-based virtualization systems such as OpenVZ [166] can provide operating system-level virtualization without hypervisors by running several user instances sharing a single kernel.

Virtualization techniques in different categories are generally incompatible, and for paravirtualization there may be interoperability issues even between different versions of the same hypervisor technology. For IaaS clouds, the most common virtualization techniques are native virtualization and paravirtualization. Hardware support allows native virtualization to perform at almost the same level as paravirtualization, keeping the losses imposed by virtualization down to a couple of percent [3, 20].

As discussed by Rosenblum [186] there are several benefits of using virtualization in system management, but the most important property of VMs in the context of this thesis is that a VM can be migrated (moved from one physical host to another) either by pausing it and resuming it on another host or by moving it without suspension using *live migration* [36, 46, 209]. This is a key enabler for run-time VM management as it allows the re-optimization of VM placement across physical resources without significantly affecting the running services.

In essence, virtualization offers a means to break some of the basic assumptions related to traditional server hosting. With virtualization, the assumption that the physical location where a server system is hosted remains constant throughout the run of the system no longer holds since the VM may be migrated during run-time. Similarly, the amount of resources assigned to a (virtual) machine is not constant, and can change dynamically during run-time using elastic service provisioning, a property that is commonly referred to as *elasticity* (see Section 3.3).

2.2.5.1 VM Instantiation

In most IaaS clouds, VMs form the basic computational units to be executed on the infrastructure. Some providers have a set of predefined VM sizes that can be used on the infrastructure, while other providers allow their consumers to specify the desired size of their VM more freely [96]. To assign more resources to a cloud service, more VMs belonging to that service can be started on the infrastructure. Normally, each VM *instance* is based on a corresponding VM *template* (or type)[2]. Each template contains a pre-installed operating system along with the applications needed to fulfill a specific role in the service, and the number of

[2]These terms are not to be confused with what Amazon EC2 [9] defines as "Instance Types", which are predefined hardware configurations of VMs.

running instances of each template can vary over time. In theory, the number of resources assigned to each service component can be varied independently by adjusting the number of running instances of the corresponding template. In practice, however, the load on one type of service component is likely to be correlated to the load on other related types of service components.

Compared to preparing a unique VM for each instance, the approach based on instantiating from a template has a number of advantages:

- The number of potential instances is not limited by the number of prepared VM images.
- Updates to the operating system and applications running inside the VM can be applied at a single place.
- Storage and network loads are reduced because only a single template corresponding to each service component needs to be managed.

The work presented in Paper I and Paper II relates to how the relationships between different service components (such as a worker node and a load balancer) can be modeled and expressed explicitly. The resulting model supports relationships both between VM templates (types) and between individual instances based on the same template, and can be used as a source of input for placement decisions in infrastructures that span multiple domains. Section 3.1 provides more background information on service placement, while the contributions presented in this thesis are discussed in Section 4.1.

2.2.6 Cloud Service Lifecyle

The lifecycle of a cloud service can conceptually be subdivided into a set of phases as shown in Table 2.1. In the *Definition phase*, which is performed offline by developers, the cloud service is developed and packaged, e.g. into VM templates. A *service manifest* [90] containing all of the meta data required for the service is created at this stage. This is followed by the *Deployment phase*, during which the manifest and templates are submitted for execution to a suitable service provider. As a part of the deployment, a predefined number of VM instances spawned from the templates are initiated. Finally, the service is monitored and managed by the infrastructure in the *Operations phase*, during which the infrastructure may alter the deployment and constitution of the service according to predefined rules.

The separation between the phases in this model is not strict, and different parts of the service may be in different phases at any given time. For example, data from running VM instances are gathered during the operations phase, and as a result the number of resources assigned to the service may be varied using elastic service provisioning by instantiating new VM instances. The new instances pass through the deployment phase individually and independently of each other and any already running components.

Table 2.1: Life-cycle phases of a cloud service.

Definition Phase	**Deployment Phase**	**Operations Phase**
Develop	Select Provider	Monitor / Optimize
Compose	Deploy	Execute
Configure	Contextualize	Recontextualize

VM instances that are started from the same template are identical, but some settings typically need to be dynamically configured for each instance. The process of configuring each instance automatically is called *contextualization*.

2.2.6.1 Contextualization

Contextualization [14, 122, 123, 221] is a process that allows newly started VM instances to be adapted and dynamically configured on a per-instance basis. The configuration can involve assigning a unique network address to a VM, or providing applications running within the VM with data that was unknown when the template was constructed. For example, it may be necessary to supply a worker node with the IP-address where a load balancing component is running, which will not be known until the load balancer component is actually deployed. In multi-domain clouds, the provider selection process that is performed during the deployment phase can be done on a per-instance basis (although it may be limited by placement constraints), and this may result in VM instances of the same service (originating from the same VM template) running on different infrastructures. Contextualization makes it possible to adjust the VM to suit any specific conditions required by any given service provider.

Contextualization is usually performed during the boot process of a VM instance. As a result of service (or infrastructure) optimization during the operations phase, some instances may be migrated from one physical host to another, which may invalidate the configuration and adaptation work done during the boot-time contextualization stage. Since the migration of VMs does not cause the VM to reboot, a new round of contextualization is not triggered.

Papers III and IV outline the concept of *recontextualization*, a technology for performing perform run-time reconfiguration and adaptation of VM instances. Recontextualization enables individual VM instances to be updated during the Operations phase, at any point chosen by the VM hypervisor. This allows the network configuration of a VM that has been migrated to a new physical environment to be updated automatically post-migration. Recontextualization can also be used to provide context-aware applications running inside the service with context-based events, making it a potentially key supporting technology for future context-aware cloud services. The contributions relating to VM recontextualization that are contained within this thesis are discussed in Section 4.2.

Chapter 3

Management in Distributed Infrastructures

3.1 Scheduling and Placement

The assignment of incoming workloads to physical resources is a fundamental management task for any computational infrastructure, and the correctness and capability of the algorithms used can greatly affect system resource utilization. The process of assigning incoming jobs or services to available resources is usually referred to as *Scheduling* (for grids) or *Placement* (for clouds).

3.1.1 Grid Scheduling

Grid computing jobs are typically self-contained applications that operate on a specific set of input data. Each job is assigned a maximum execution time upon job submission, and this expected duration is used to plan the execution of subsequent jobs on the same resource. A conceptual illustration of grid scheduling, inspired by the work of Klusáček and Rudová [130], is shown in Figure 9. A large set of input data can often be subdivided into smaller parts, allowing many thousands of independent grid jobs to be started in parallel. Jobs that are easily subdivided into independent tasks that can easily run in parallel are usually described as being *embarrassingly parallel* [101] and are particularly suitable for execution on a grid. More complex workflows involving jobs with internal dependencies can also be submitted but require more advanced workload control processes [25, 70, 141, 212, 223] and scheduling heuristics [191].

The first step towards grid job execution is to select which of the available infrastructures should be used. This can either be done manually by the user or with the support of a *resource broker* [42, 73, 133]. Once a suitable resource has been selected, the job is submitted to the local scheduler of that resource for execution. Common technologies for local resource scheduling

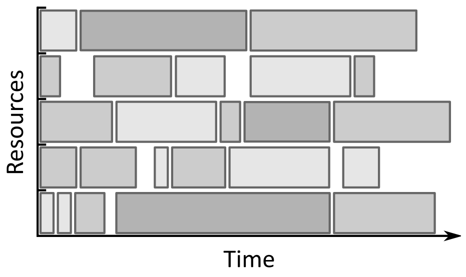

Figure 9: Conceptual view of grid scheduling.

include Maui [115] and SLURM [239]. In contrast to the local scheduler, the broker does not have full control over the resources and must rely on best-effort scheduling of jobs [193].

The lack of user interaction makes it possible for a grid to schedule (and re-schedule) jobs, as there are normally no strict restrictions on when the job should run. Advance reservations allow users to reserve specific execution times if required, typically at the expense of overall resource utilization due to the creation of unusable gaps prior to the start-time of the reserved jobs [199]. Backfilling techniques [160, 205] are commonly used to increase resource utilization, and may also be used to mitigate the loss of utilization caused by reservations [200]. There are many different strategies for grid job scheduling with different emphases: some focus on scheduling for the benefit of a single application [28] while others aim to optimize the job wait time [104] or the total system throughput [111], to avoid starvation[3], or to offer advance reservations. An overview and performance comparison of various grid scheduling techniques can be found in [106].

Scheduling is a significant challenge in federations of grids [32, 52, 75], especially when dealing with non-trivial jobs because the correct execution of a parallel job often means that the job has to be executed in parallel across different sites. A common approach for managing independent jobs in a federation is to first perform grid-level scheduling to assign each job to a grid, and then delegate resource scheduling decisions to the underlying grid infrastructures [72, 113, 140]. An alternative solution involves a hierarchical grid organization [24, 114]. In this case, the local grid sees other grids within the federation as very large local resources with special characteristics, and outsources job execution to those grids using standard job management interfaces.

Fairshare scheduling [50, 60, 61, 66, 121, 127, 129] is a widely used approach in which the scheduler tries to distribute computational resources between users according to predefined usage shares. The scheduler compares each user's historical resource consumption to their predefined resource allocation and then prioritizes their incoming jobs based on the difference between the two. The data

[3]Starvation occurs when some jobs are constantly neglected in favor or other jobs, starving them of resources.

used in the fairshare process are typically gathered from usage records, which are either obtained by querying the underlying accounting system or received directly from the infrastructure. The accuracy and availability of the usage records and the delay before the historical usage data is made available to the fairshare scheduler directly affects the system's performance and convergence time.

The work presented in Paper VII and Paper VIII concerns the design and evaluation of a decentralized approach to fairshare scheduling that is designed for multi-domain grid deployments, in which global historical usage patterns and a globally defined usage share are used for prioritization at each local site.

3.1.2 Cloud Placement

In contrast to grid jobs, cloud services are longer running applications that do not have definite completion times and whose resource requirements may change over time. A cloud service is expected to run until it is manually canceled, and the resource consumption of each service component is bounded by the size of the VM. A conceptual illustration of cloud placement of services with dynamically changing resources is shown in Figure 10.

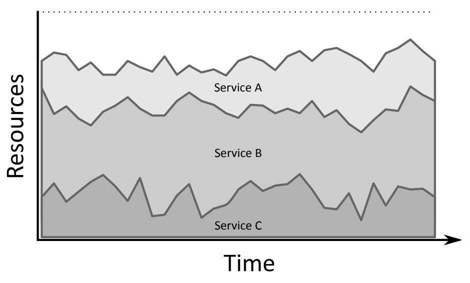

Figure 10: Conceptual view of cloud placement.

Unlike grids, which are primarily composed and managed as collaborative non-profit infrastructures, most cloud infrastructures are commercial. These two different environments create different management requirements; whereas grids tend to focus on user prioritization and fairness, most cloud infrastructures have objectives related to revenue maximization.

The revenue derived from a cloud is directly related to resource utilization. In the basic cloud model, the size of each VM is limited to a set of predefined (and compatible) values. These in turn determine the amount of CPU, memory, and storage assigned to each VM. In addition, the size of each VM remains constant throughout its lifespan. This makes it easy to achieve high and non-fluctuating levels of utilization.

Dynamic VM sizing approaches [96, 169, 178] have been introduced with the aim of relaxing the limitations on VM constitution in cloud environments and enabling the allocation of each resource type (i.e. CPU, memory, and storage space) to be specified independently of the others. It is not straightforward to maximize utilization while executing VMs with non-standard resource usage patterns, especially if their resource requirements change dynamically. Such systems therefore require more sophisticated management processes. As an alternative way of addressing this problem, the overbooking strategies used in other application areas [31, 103, 187, 208] are being adapted for use in cloud placement [156, 218]. These approaches tend to rely on the assumption that it is very unlikely for all of the available resources to be requested simultaneously [103].

The operating costs (and environmental impact) of cloud infrastructures can be minimized by reducing the number of physical machines that are used to host the current workload by consolidating running VMs onto a smaller set of physical resources, allowing parts of the physical infrastructure to be shut down [49, 161, 170]. However, services running on shared infrastructure may experience performance degradation, and the level of interference between services that will occur in such cases is not always easy to predict [49, 214].

SLAs specify the terms of resource provisioning on clouds. These agreements stipulate a minimal quantity of resources that must be assigned to each service, and may dictate how the SP is to be compensated if the agreement is not upheld by the IP. This presents an important challenge in cloud placement: it is necessary to achieve high resource utilization (to maximize revenue) across a set of active physical resources and to ensure that one allocates enough resources to run all of the desired services in a way that minimizes overall resource allocation (to minimize operational costs) while upholding all SLAs that have been signed in order to avoid having to pay compensation and potentially suffering a loss of reputation or business [151].

Multi-site cloud deployments (as described in Section 2.2.4) allow a cloud IP to also utilize resources from other infrastructures. Although remote resources are likely to have higher operating costs than local ones, the extra flexibility provided by this approach can make it possible to accept more services on the local infrastructure, with *bursting* used to transfer part of the load to a remote cloud if resources start running low in order to avoid breaking SLAs.

To enable VM consolidation and bursting, individual VMs can be moved from one physical host to another by means of VM migration. However, the relationships between the different VMs that comprise a given service are generally not known to the IP, and the performance of the service may be degraded significantly if a fundamental service component (such as the database back-end of a multi-tier application) is migrated to a remotely hosted physical machine [197]. Some of the work presented in this thesis relates to the development of a method for the explicit structuring of cloud services that expresses the internal relationships between service components in a way that can be understood by cloud placement processes when performing VM placement or migration. Some preliminary work on scheduling heuristics incorporating these constraints has also been conducted. This work is presented in Paper I and Paper II, and is further discussed in Section 4.1.

3.2 Monitoring

Monitoring is the process of gathering information about the infrastructure and/or hosted services during run time. In grid systems, monitoring is usually done to ensure the health, performance, and status of the infrastructure resources [226, 241]. This information is subsequently used for fault detection and recovery, to predict resource performance, and to tune the system for better performance [215]. While grid monitoring has some bearing on run-time management processes, it is beyond the scope of this thesis.

In cloud computing, monitoring of running services is a fundamental task because monitoring data is the primary input used for decision making in other management processes. Consequently, the lack of compatible monitoring systems is one of the main sources of incompatibility in cross-site clouds [13, 137]. A complicating factor to consider in multi-domain cloud scenarios is that more than one site might be interested in the monitoring data produced for a given service, and the set of interested domains may change dynamically during run-time in parallel with the dynamic changes in the set of IPs that contribute resources to the service's hosting.

Three different kinds of monitoring data are used in clouds:

- Infrastructure specific measurements showing the health and utilization of physical resources. The need to monitor the state of infrastructural resources is not unique to cloud computing, and the tools that are used for general purpose system monitoring (such as Nagios [23], Ganglia [150], or collectd [83]) can be used also in cloud contexts.

- VM resource consumption data. Measurements of resource consumption for individual VMs running on the hardware can be obtained by communicating with the VM hypervisor, or by using tools (such as the libvirt [33] API) that are capable of operating across several different hypervisors. The VM information is commonly used to assess the fulfillment of SLAs or as an input for elasticity and service profiling.

- Monitoring values specific to the service in question. These Key Performance Indicators (KPIs) [185] are normally only available from inside the service software itself, and might constitute values such as the current number of active sessions to a Web based application or the number of concurrent transactions in a database system. These values can be used to perform automatic elasticity and service performance optimization on metrics central to the service designer.

The measurement and management of KPI data from inside the service itself presents a number of interesting problems that have not yet been well studied [120]. Some cloud solutions (such as RESERVOIR [183]) have a strong separation between service management and the VM itself, in the sense that the VM is unaware of the exact location of the management components, and the management components at the higher levels of the stack are unaware of the location of the VM. This *location unawareness* [65, 71, 105] is a gray-box approach [16] to service management and will determine which techniques can be used to make service-specific data available to the cloud infrastructure from inside the VMs.

The Lattice framework [47] provides a solution for service level monitoring that uses customized virtual networks (VANs [105]) to pass measurements from inside the VMs to the infrastructure without external network access. In this solution, the functionality of the network broadcast directive is overridden and used for monitoring purposes. This solution would not be viable in the absence of customized virtual networks, so its applicability is limited.

An alternative based on the File System in User Space (FUSE) concept [211] has been outlined by Elmroth and Larsson [71]. In this solution, FUSE is used to create a small application that simulates a hard drive partition. File system calls to the partition (such as writes) result in a method call inside the application, where the complexity of externalizing the data from the VM to the infrastructure can be managed. However, this does not address the problem of actually externalizing the data in cases where the locations of some management components are unknown.

The management of KPI monitoring data in multi-domain clouds is one of the use cases for the work on contextualization and recontextualization presented in Paper III and Paper IV. The suggested approach (which is also partially based on FUSE) allows the infrastructure provider to dynamically configure and reconfigure the target endpoint for KPI monitoring information both during deployment and during run-time. One of the system's usage goals was to allow the infrastructure to automatically redirect the data flow to a local monitoring endpoint in response to a VM being migrated in from another infrastructure. This redirection ensures that the data passes through the site-local monitoring process while also relaying the information back to any interested collaborating sites.

3.3 Elasticity

The ability to quickly request or release resources in response to variations in the load associated with a given service is one of the most prominent features of cloud computing. *Elasticity* [5, 10, 13, 145, 155] is the process of automating these capacity adjustments and transferring the decision making responsibility from human administrators to processes that are run by either the SP or the IP. By specifying a set of *Elasticity Rules* [185] and including these rules in the service manifest [90], the rules for scaling a service are made into an integral part of the service itself. These rules can be used to specify things such as the maximum number of users that may be served by each VM instance, which can in turn be used together with reactive or predictive models to calculate the number of required instances [6].

There are two types of elasticity, *horizontal elasticity* and *vertical elasticity*. Horizontal elasticity is the capacity to increase or decrease the number of VM instances of a certain type in response to changes in the current load [6]. Vertical elasticity is the capacity to dynamically increase or decrease the hardware resources (such as the amount of RAM or number of CPUs) assigned to one or more VM(s) [237]. Horizontal elasticity puts additional strain on the application running inside the VMs because the system itself must synchronize incoming tasks between different instances by load balancing. Vertical elasticity, on the other hand, requires the operating system and the application(s) running inside the VM to be capable efficiently using a dynamically changing resource allocation, which may include a variable quantity of RAM or number of processor cores.

Automatic scaling based on elasticity imposes certain requirements on other management processes. For example, elasticity is often based on KPI data gathered from inside the applications that comprise the service. The underlying monitoring system must therefore support the collection and management of KPI data. In multi-domain clouds, meta-data relevant to the service must also be collected from all infrastructures on which service components are being hosted. This requires the ability to exchange monitoring data between all of the relevant infrastructure domains in a mutually compatible format. Such exchanges must be prompt and comprehensive because losses of monitoring data or delays in its transmission can significantly reduce the system's elasticity.

3.4 Accounting and Billing

Accounting systems are responsible for metering and managing records of resource consumption by users in grids or clouds. In grids, a usage record [148] for a job is created once the job has finished executing. The usage record contains general meta-data about the job such as start and finishing time stamps, and may also contain a summary of the job's combined resource consumption in terms of, e.g., the amount of data transferred within the network. In federations of grids, the accounting data generated upon job completion is usually important both for the originating grid site, the executing grid site, and possibly any consortium or organization that is involved in linking these resources together. Managing usage records in such environments is the subject of Paper V.

Grid deployments are commonly based on collaborative sharing models where the usage data are converted into abstract currencies [22, 92, 174]. Abstract credits are awarded to users through an out-of-band application procedure, in which, e.g., a steering committee allocates credits to different projects based on scientific merit. These credits can then be exchanged for computing time on the infrastructure. There are numerous other approaches, economical models, and architectures that can be used in grid management. These are commonly based on auctions or other market-based schemes and have been discussed at length in the literature [18, 40, 43, 67, 135, 240]. However, they are beyond the scope of this thesis. Nakai and Van Der Wijngaart [162] have presented an in-depth economical analysis of the viability and expectations of market-based systems for grid scheduling, showing that they are not generally applicable and may not afford the desired outcomes.

Cloud systems normally rely on run time monitoring of service resource consumption as a basis for accounting. For multi-domain cloud scenarios, the aggregation of data from different sites is usually managed by the underlying monitoring system because accounting is not the only internal cloud process that depends on the aggregation of raw monitoring data.

In public clouds, users are free to request as many resources as they require in the short term, paying only for the resources they are currently requesting. In such systems, the accounting data (which are derived from monitoring data) are used as inputs in the *billing* process, which converts the hardware usage data into a monetary cost based on a given pricing scheme. The two major payment models used in clouds are *prepaid* and *postpaid*, with postpaid being the most common. Both terms are used in the same manner as in the mobile-phone industry. Under the prepaid model, credits are purchased in advance and consumed in accordance with resource consumption. This offers greater control over the maximum costs but running out of credits may cause the service to stop executing. Under the postpaid model, the consumer is billed at regular intervals for their usage in a preceding period of time. This may result in unexpectedly high levels of resource consumption in some cases (typically due to poorly configured elasticity) but presents no risk of running out of credits and hence disturbing service execution. Paper VI presents an accounting system designed for use in distributed cloud environments that support both prepaid and postpaid payment models.

Deployment scenarios such as those involving bursted private clouds or cloud federations offer potentially unlimited amounts of hardware resources because unknown amounts of resources from collaborating sites can be used during service execution. In theory, this means that very large quantities of accounting (and monitoring) data may be generated by the running services. Ensuring that the local infrastructure can scale to accommodate this potentially unlimited amount of data is a significant challenge. Accounting data is commonly treated as financial data, which means that there are strict regulations governing its storage and management; among other things, it must typically be kept for long periods of time (at least ten years in some jurisdictions). This creates an important resource provisioning problem. Fully scalable solutions for such data have yet to be fully developed, but some preliminary work on this subject has been presented by Das et al. [51] and Lindner et al. [146].

3.5 Autonomic Computing

Autonomic computing, a discipline originating from IBM's autonomic computing initiative [112] in 2001, aims to develop computing systems that are capable of self-management during run-time. Autonomic computing has been defined as: “Computing systems that can manage themselves given high-level objectives from administrators.” [125].

The increasing scale and complexity of distributed infrastructures such as grids or clouds has created a need for automated management of resources and services because manual management of large-scale resources is ineffective, expensive, and error-prone. The vision of autonomic computing includes full automatization of the application provisioning life-cycle, including automatic and informed system updates, reconfiguration, post-upgrade regression testing, and possibly automatic downgrading of the system to enable automatic detection of failures related to attempted updates [125]. Dobson et al. [56] have reviewed the progress made during the first decade of autonomic computing. Their overview clearly shows that although considerable progress has been made in this area and the need for autonomic management has grown substantially since 2001, there are many key challenges that remain to be addressed.

Autonomic computing can be conceptually subdivided into four main functional areas [125]:

Self-Configuration
Automatic configuration of components during run-time according to high-level policies.

Self-Healing
Automatic discovery and correction of both software and hardware faults during run-time.

Self-Optimization
Automatic monitoring and control of resources to ensure optimal performance and function with respect to the defined requirements.

Self-Protection
Proactive identification and protection from arbitrary attacks or failures.

The terms *self-management* or *self-* management* are commonly used when discussing autonomic processes in general. Such processes are commonly modeled using a four step procedure that is referred to as MAPE. In this model, the system Monitors the managed element, Analyzes the monitored data, and finally Plans and Executes corrective measures. This is complemented by a Knowledge Base that stores and manages the information used in each step [116, 125].

There has been considerable work on the application of self-* concepts to cloud computing at several levels of the software stack. The most common approach is to employ self-optimization techniques to modify the placement of VMs on physical resources to reach higher-level objectives such as high utilization or SLA enactment. Work in this area has been presented by various authors. Maurer et al. [152] developed and evaluated designs for the knowledge base used in the MAPE loop that rely on both case-based reasoning and policies that are specified as rules, with the aim of achieving high utilization while adhering to the terms of the relevant SLAs. Wood et al. [234] presented a similar approach in which run-time optimization techniques are used to avoid hot spots in resource utilization by re-sizing VMs and creating new mappings between VMs and physical resources through migration. Hirofuchi et al. [108, 109] demonstrated the use of a postcopy migration [107, 134] extension to KVM that considerably reduces the time required for VM migration, allowing for reaction-based self-optimization with an expected live-migration time of about one second. This allows VM instances to share physical machines with very high rates of overbooking during periods of low activity, while also enabling the system to rapidly migrate VMs away to dedicated servers as their activity increases in order to remain compliant with SLA terms. Shrivastava et al. [197] introduced the concept of application-aware VM migration, whereby the knowledge base is augmented with information on inter-VM dependencies and the underlying network topology to minimize network traffic during migration.

The work on service structure and (re)contextualization presented within this thesis relates to the development of enabling technologies for self-* processes. Service structure definitions extend the knowledge base and enable self-* processes to use service structure data in decision-making. Among other things, this makes it possible to properly account for service performance dependencies and thereby substantially improve component placement. (Re)contextualization in turn offers a mechanism that can be used as a trigger and data transfer tool for self-* operations that affect software running inside a VM. This topic is further discussed in Section 4.2.

Chapter 4

Thesis Contributions

The work presented in this thesis includes four enabling technologies and methods for multi-domain grid and cloud infrastructures:

- The use of explicit definitions of cloud service structure as inputs for placement and management processes, to ensure that the resulting placement respects internal service relations and constraints.
- Technology for runtime adaptation of Virtual Machines to enable the contexts of cloud service to automatically adapt to changes in the environment, e.g., as a result of service migration across domains.
- Systems for managing meta-data relating to resource usage in multi-domain grid computing and cloud computing infrastructures.
- A global fairshare prioritization mechanism that enables computational jobs to be consistently prioritized across federations consisting of several decentralized grid installations.

4.1 Service Structure

As previously discussed, multi-domain cloud deployments offer advantages in terms of flexibility, scale of available resources, and economy. Much work on technical interoperability will be required to facilitate their general use and development, along with new approaches to service management that can be applied across multiple administrative and technical domains.

Together with the contributions of Larsson and Elmroth [137] and Hadas et al. [105], the work presented in Papers I and V of this thesis defined the concept of location unawareness. It is assumed that a location unaware service is not aware of and does not track the infrastructure on which is it running. Therefore, all updates to the operations of the VM that are necessitated by changes in its placement must be managed by the external cloud management processes rather than the VM itself. In multi-domain clouds, location unawareness also stipulates that a local cloud deployment should not be concerned with the precise remote location on which a given service component is running and cannot affect the placement of the service component on the remote infrastructure. In effect, location unawareness is a way of formalizing the cloud service developer's inability to predict the placement of individual service components within the cloud, and extends this uncertainty to include the relationship between different IPs.

In Paper I [137], we present a technique for mitigating the uncertainty associated with location unawareness by allowing the service owner to impose constraints on how the different parts of the service are deployed. The paper presents a hierarchical model for expressing the internal structure of a cloud service, including placement constraints in the form of geographical or intra-componental affinities [37]. A service-structure aware VM placement algorithm can use this information to deploy the service to the infrastructure in a way that accounts for both its internal structural requirements and any explicit placement constraints that have been selected. When migrating parts of the service to another host, local or remote, it may be necessary to simultaneously migrate several VMs in order to remain compliant with these requirements and constraints. Paper I also describes a placement model that respects the user-specified constraints, and extends this model with a heuristic for determining which parts of the service are most suitable for migration by considering which other parts of the service would be affected by each potential relocation.

Paper II [77] presents an extension of our model for service structure and placement that formalizes its hierarchical graph structure and demonstrates how it can be converted into mathematical placement constraints. These constraints can be interpreted by placement algorithms during service placement. A formal mathematical model for placement optimization that incorporates the constraints is presented, and the viability of the approach was demonstrated by using it in a series of simulation-based experimental evaluations. In the simulations, 15300 randomly constructed test cases with varying amounts of background load, affinity constraints, and anti-affinity constraints were analyzed to determine the number of viable placements that could be identified in each case and the number of time-outs. The results demonstrate that the placement optimization model is insensitive to the presence of background load, and that affinity is the most restrictive factor in this test setup.

4.2 Contextualization

Contextualization is closely related to location unawareness. Because a service developer cannot know the location at which the service will be executed in advance, the service must be designed in a generic manner without specific ties to any IP. Contextualization is normally done as part of the VM boot process, which makes the migration of contextualized instances from one infrastructure to another problematic – because the VM is not rebooted, the contextualization of the VM instance will not be modified to match the new infrastructure.

Paper III [15] introduces and defines the concept of recontextualization, a run-time reconfiguration process for VM instances that does not require restarts or downtime. To demonstrate the viability of the approach, a mechanism for recontextualization is presented, implemented, and evaluated. The suggested mechanism employs virtual media devices rather than network access to pass information from the hypervisor side to the VM. This enables contextualization to be used for network configuration management as well as VM migration. Using KPI monitoring as an example, we demonstrate how the end-point address used for monitoring can be dynamically updated inside the VM at the discretion of the hypervisor. Among other things, this allows an infrastructure to which a VM instance has recently been relocated to pass KPI measurements through the local monitoring system. In the evaluation process, the overhead imposed by recontextualization was determined by measuring the time required to complete specific tasks with and without the feature enabled. Overall, recontextualization increased VM migration times by 18% when using the KVM or Xen hypervisors. For KVM, most of the extra time was spent outside the bounds of the recontextualization component and the time penalty was likely due to the extra overhead imposed by preparing for migration with virtual devices attached. For Xen, the device management functionality available through libvirt proved insufficient and had to be replaced with sub-process calls at the system level. This workaround is likely to have increased the time required for migration. In essence, while there is room for performance improvements, this work demonstrates the viability of recontextualization as a technology.

A unified approach that integrates contextualization and recontextualization is presented in Paper IV [76]. By employing a custom file system implemented using FUSE [211], several different and partially overlapping data sets can be layered on top of one-another. This makes it possible to construct a hierarchy of configuration data in which the default settings from the development phase are dynamically replaced with contextualization settings that are introduced during VM deployment or recontextualization settings that are made available by the IP during run-time. The paper describes an application-level use case featuring a distributed file system that shows how recontextualization can be used to optimize application performance. The example shows how a file system client node can be dynamically reconfigured after migration to use the closest node within a distributed file system server to improve performance.

4.3 Accounting in Multi-domain Infrastructures

The establishment of monitoring and accounting solutions with broad compatibility is a key challenge in the development of multi-site infrastructure for both grids and clouds [13, 81, 137]. Monitoring and accounting data are essential for grid and cloud management because they are used as inputs in many other management processes. For cloud computing, monitoring data must be collected on a per-service basis from all infrastructures that are involved in hosting the service of interest. The monitoring data are used as inputs for several internal management processes, such as elasticity and accounting. In the accounting case, monitoring data form the basis of accounting and billing between the SP and the IP, and potentially also between any IPs involved in hosting the service.

Paper V [65] describes studies on accounting and billing in federated cloud environments. The paper considers cloud-related problems that are not encountered in traditional grid and cluster environments. These include the problem of accounting for services with a dynamic number of sub-components where the precise number of sub-components is not known by the accounting system and that of accounting for services in which the placement of sub-components in the federated infrastructure is also dynamic and unknown. A set of requirements for accounting and billing systems in federated clouds is formulated based on the studied use cases and general non-functional requirements. Existing accounting systems for grid and cluster systems are evaluated based on these requirements, but no existing alternative is found to fully support the set of requirements imposed by this environment. Consequently, a new architecture for a cloud focused accounting and billing subsystem is proposed.

The implications of multi-domain grids with regard to accounting data management are studied in Paper VI [69]. In this paper, a set of different multi-domain usage scenarios are used as a starting point for the extraction of a general set of requirements. It is shown that the setup of the multi-domain grid and the addition of new jobs must be done in a way that is non-intrusive towards existing grid installations that are already in production. A design for a light-weight component that controls the flow of usage information between different parts of the collaboration is then presented and evaluated. This process is made non-intrusive and optional by reusing existing read and write interfaces from the data management components, and provides support for different levels of cardinality (one-to-many, many-to-one, many-to-many) in usage record sharing within a collaboration. We demonstrate that the component can be used to realize different usage scenarios by configuring and deploying it in different ways, without affecting the operation of data management components that are running in different parts of the collaboration.

4.4 Decentralized Global Fairshare

As discussed in Section 3.1.1, the scheduling of jobs in multi-domain grids is normally performed in several steps. The first step involves the selection of a suitable sub-infrastructure for the execution of a job and the submission of the job to that location for execution. The job is then placed in the job queue of the local resource management system, at which point another round of scheduling and prioritization is performed within the sub-infrastructure before the job is assigned physical resources and executed.

Paper VII [168] presents the design, construction, and functional evaluation of Aequus, a grid-wide support system for job prioritization based on fairshare allocation that is based on the work of Elmroth and Gardfjäll [66]. The proposed system is a distributed stand-alone system that can be used by job schedulers to externalize the fairshare prioritization process. The paper presents a distributed tree-based policy model for specifying user shares hierarchically, making it possible for a project to subdivide its own share of usage into specific shares on a per-user and/or per-sub-project basis without requiring the involvement of administrators from the executing sites. The paper also describes an algorithm for prioritizing user jobs based on predefined user shares and historical usage data, and the decentralized architecture used to realize the system is described in detail. The overall system behavior and its ability to accurately prioritize jobs in different scenarios is evaluated in detail, showing that the system is capable of achieving grid-wide fairshare even in the presence of dynamically changing policies and run-time site failures.

The work on decentralized fairshare scheduling is extended in Paper VIII [78]. This paper demonstrates how the system can be integrated with existing local resource management systems (SLURM and Maui) and presents an evaluation of the system's performance in a fully integrated environment. Workload modeling techniques [57, 79, 143, 147] are used to generate input data for system evaluation. By varying the statistical distribution parameters for the underlying workload models, the system is evaluated using a range of different workloads. Some workloads are dominated by periodical usage patterns and some are dominated by a bursty usage pattern in which a large portion of the jobs are submitted in a short period of time. The system's behavior is shown to be consistent and stable regardless of the type of workload imposed.

Chapter 5

Future Work

The studies presented in this thesis can be grouped into three main categories:

- The use of service structure data and contextualization as tools for multi-domain cloud service management.
- Accounting for multi-domain infrastructures.
- Decentralized global fairshare.

This chapter presents potential directions for future investigations in each of these areas.

5.1 Multi-domain Cloud Service Management

Both service structure and contextualization are enabling technologies for service deployment in multi-domain clouds. The use of service structure data enables properties of the service known only by the service designer to be taken into consideration during placement decisions, and can mitigate increases in latency that would otherwise be associated with multi-cloud deployments. Contextualization enables the adaptation of the service to new environments, and allows for closer interaction between VMs and the hosting infrastructure, even in cases where the technical particularities of the hosting infrastructure are not known at deployment time or change during runtime.

A promising future use case for service structure data is to complement existing VM migration techniques with information on inter-VM relationships as demonstrated by Shrivastava et al. [197]. These authors use information on the communication between different components and the derived communication patterns as one of several factors when optimizing performance. Relying on the known service structure instead of run-time measurements would reduce the strain on monitoring within the infrastructure and facilitate the preemptive optimization of service component placement rather than reactionary adjustment.

The geographical locality of data placement and computational provisioning in collaborative clouds raises concerns relating to performance, fault-tolerance, and legislative aspects [37, 117]. Some public clouds offer coarse-grained control of service placement by allowing the user to chose a region for deployment [142]. These regions are usually very large (for instance, Europe is normally treated as a single region), and the regional divisions probably reflect the locations of physical resources and data centres. The ability to express structure and placement constraints explicitly could make future cloud infrastructures suitable platforms for the deployment of a new group of services that rely on global deployment while still retaining some control over how and where the service is deployed.

An interesting technology for multi-cloud deployments is the ability to dynamically and automatically split a service manifest into several sub-manifests [144]. This enables brokers of cloud resources to manage the deployment of a service across multiple infrastructure without prior support from the service developer. A natural extension of this service decomposition would be to incorporate data on the internal structure of the relevant services in order to provide a heuristic for service subdivision.

As described in Section 3.5, autonomic computing [116, 125] is a vision of computing systems that are self-managing given high-level objectives from administrators. The work on service structure and (re-)contextualization presented in this thesis can be regarded as enabling technologies for autonomic computing. Service structure can be used for self-optimization and self-configuration at the infrastructure level to fine-tune the deployment of the service. Contextualization is a fundamental supporting system for self-configuration at the infrastructure level, allowing VM instances to be migrated to a new infrastructure during run-time. Such migrations may be triggered by other self-* processes such as self-optimization. As illustrated by the distributed file system use-case, our work on recontextualization can convey changes to the service context from the IP to the applications inside the service. This enables the use of self-* process even within the service itself.

One vision for future clouds involves a global cloud infrastructure with no technical and administrative boundaries, allowing cloud services to be dynamically relocated to any of the constituent infrastructures without requiring prior agreements or technical preparation [29, 30, 62]. The development of such a unified global infrastructure would represent a very significant step towards fulfilling the vision of computing as a utility and amplify the advantages of multi-domain cloud deployments. For example, the proximity- and geographically-aware placement of cloud services [4, 176] would make it possible to offer new kinds of IT services that leverage data that are distributed geographically around the globe. Placement constraints and awareness of internal service structure, and the ability to adapt the service to new execution environments using contextualization, would be extremely useful in such scenarios.

The combination of service structure data and (re)contextualization enables some interesting possibilities for self-configuration. For example, service structure data can be used to determine which networks (internal or external) each VM instance should be connected to. Once this has been done, the required network settings can be applied inside the VM using (re)contextualization.

5.2 Multi-domain Accounting

The work on cloud accounting presented in Paper V was performed in the early days of cloud computing, and the outcome is one of the first accounting and billing systems designed for clouds. Compared to previous systems used in grids, the accounting systems for clouds have to be more extensively integrated into the management processes of the cloud infrastructure. For example, if a prepaid service is running low on credits, the infrastructure may force the service to scale down, or stop completely until more credits are available.

The increasing size and complexity of cloud infrastructures makes the process of multi-domain monitoring very challenging [44], and the management of monitoring information is likely to be one of the key challenges in the establishment of a unified cloud infrastructure. Recent developments in cloud accounting include efforts to create fully decentralized models for accounting and billing [136, 236], which may help to mitigate the problem of massively scaled infrastructures. Future work in multi-domain accounting includes further development of decentralized accounting models.

5.3 Decentralized Global Fairshare

The Aequus system presented in Paper VII and Paper VIII enables the global enforcement of fairshare policies and also incorporates global logging and management of (summarized) usage information. The performance and convergence rates of Aequus are calculated based on the assumption of a short turn-around time between the completion of a job and the system becoming aware that the job is complete. The use of speculative job execution times defined by users to estimate job resource consumption has been shown to have a great impact on the system's convergence rate [66] because it greatly reduces the severity of the fluctuations caused by overcompensation. Tomás et al. [217] have achieved similar results by integrating Aequus with the SA-Layer scheduling framework [216], considerably reducing the number of jobs that must be run in the system before fairshare converges. The incorporation of tools for evaluating the impact of recently submitted jobs into local resource management systems such as SLURM and Maui would be an interesting task for future work in this area.

In recent years, the energy costs associated with running large computational clusters have become a significant issue, to the point where operational costs rival the cost of investment in new hardware [27]. Reducing the energy consumption of compute clusters is a growing challenge that affects both the economical and ecological aspects of large-scale computing [13]. Resource control systems can be designed with different objectives in mind; in the cloud case, this is sometimes referred to as holistic management [195]. Most local resource management systems are designed to maximize utilization by allowing free resources to be utilized by the most highly ranked users, even if said users have already exceeded their designated allocations. By incorporating the ability to dynamically start and suspend physical servers [190, 204] (and/or using dynamic voltage frequency scaling [196]) with fairshare, local resource schedulers can be extended to work towards economical and environmentally friendly objectives instead. For example, the amount of running physical resources could be dynamically adjusted to favor users that are currently running jobs on the infrastructure, suspending free resources to save energy rather than executing tasks for users that have already exceeded their allocation. It would perhaps be interesting to develop the global fairshare system for grid allocation that is currently offered by Aequus to also account for fairness in terms of energy savings between resource sites.

Chapter 6

Outlook

The cross-domain utilization of computing resources offers a range of interesting technical opportunities ranging from increased scales of operation to economical advantages and the ability to spread services across diverse geographical locations. It also offers the potential for dramatically increasing the energy efficiency of (multi-domain) cloud computing, which may be more important than all of the other advantages combined.

Computing has become a major factor in global energy consumption. The total resource consumption of the Internet is very hard to define and measure but is estimated to represent 10% of the worlds total energy consumption [179], with data centres consuming 1-2% by themselves [149]. However, the servers at data centres are very energy efficient compared to the (often less modern) in-house alternatives, so by making existing businesses replace their in-house servers with more energy efficient hosting in data centres, the energy consumption for hosting such businesses could be reduced by up to 87% [149].

By inventing and developing new technologies for the cloud, we may be able to create new incentives for existing businesses to make the transition to more energy-efficient hosting. Arguably, one of the major benefits that cloud hosting could offer is access to servers and resources that are spread all around the world, enabling even small and medium-sized enterprises to design and create fully distributed and geographically-aware services, a possibility that is currently limited to major companies with their own global infrastructure such as Google or Facebook.

As of 2013, there is no publicly available globe-spanning computing infrastructure that can be used for hosting. However, there are several independent hosting alternatives that, if combined, would be able to cover a significant part of the globe. By enabling and improving the use of resources across multiple hosting domains, we will hopefully be able to take the first steps toward a unified global infrastructure, which could provide great technological benefits and improvements in energy efficiency.

Bibliography

[1] G. Aad, T. Abajyan, B. Abbott, J. Abdallah, S. Abdel Khalek, A. Abdelalim, O. Abdinov, R. Aben, B. Abi, M. Abolins, et al. Observation of a new particle in the search for the standard model higgs boson with the atlas detector at the lhc. *Physics Letters B*, 2012.

[2] G. Aad, E. Abat, J. Abdallah, A. Abdelalim, A. Abdesselam, O. Abdinov, B. Abi, M. Abolins, H. Abramowicz, E. Acerbi, et al. The ATLAS experiment at the CERN large hadron collider. *Journal of Instrumentation*, 3(08):S08003, 2008.

[3] K. Adams and O. Agesen. A comparison of software and hardware techniques for x86 virtualization. In *Proceedings of the 12th international conference on Architectural support for programming languages and operating systems*, pages 2–13. ACM, 2006.

[4] S. Agarwal, J. Dunagan, N. Jain, S. Saroiu, A. Wolman, and H. Bhogan. Volley: Automated data placement for geo-distributed cloud services. In *NSDI*, pages 17–32, 2010.

[5] A. Ali-Eldin, M. Kihl, J. Tordsson, and E. Elmroth. Efficient provisioning of bursty scientific workloads on the cloud using adaptive elasticity control. In *Proceedings of the 3rd workshop on Scientific Cloud Computing*, pages 31–40. ACM, 2012.

[6] A. Ali-Eldin, J. Tordsson, and E. Elmroth. An adaptive hybrid elasticity controller for cloud infrastructures. In *Network Operations and Management Symposium (NOMS), 2012 IEEE*, pages 204–212. IEEE, 2012.

[7] G. Allen, K. Davis, T. Goodale, A. Hutanu, H. Kaiser, T. Kielmann, A. Merzky, R. Van Nieuwpoort, A. Reinefeld, F. Schintke, et al. The grid application toolkit: toward generic and easy application programming interfaces for the grid. *Proceedings of the IEEE*, 93(3):534–550, 2005.

[8] Amazon.com, Inc. Amazon EC2 Spot Instances. `http://aws.amazon.com/ec2/spot-instances/`, September 2013.

[9] Amazon.com, Inc. Amazon Elastic Compute Cloud. http://aws.amazon.com/ec2, September 2013.

[10] G. Ananthanarayanan, C. Douglas, R. Ramakrishnan, S. Rao, and I. Stoica. True elasticity in multi-tenant data-intensive compute clusters. In *Proceedings of the Third ACM Symposium on Cloud Computing*, page 24. ACM, 2012.

[11] D. Anderson. BOINC: A system for public-resource computing and storage. In *5th IEEE/ACM International Workshop on Grid Computing*, pages 4–10, 2004.

[12] A. Anjomshoaa, F. Brisard, M. Drescher, D. Fellows, A. Ly, A. S. McGough, D. Pulsipher, and A. Savva. Job Submission Description Language (JSDL) specification, version 1.0. http://www.ogf.org/documents/GFD.136.pdf, September 2013.

[13] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, et al. A view of cloud computing. *Communications of the ACM*, 53(4):50–58, 2010.

[14] D. Armstrong, K. Djemame, S. Nair, J. Tordsson, and W. Ziegler. Towards a Contextualization Solution for Cloud Platform Services. In *IEEE Third International Conference on Cloud Computing Technology and Science (CloudCom), 2011*, pages 328–331. IEEE, 2011.

[15] D. Armstrong, D. Espling, J. Tordsson, K. Djemame, and E. Elmroth. Runtime virtual machine recontextualization for clouds. In I. Caragiannis, M. Alexander, R. Badia, M. Cannataro, A. Costan, M. Danelutto, F. Desprez, B. Krammer, J. Sahuquillo, S. Scott, and J. Weidendorfer, editors, *Euro-Par 2012: Parallel Processing Workshops*, volume 7640 of *Lecture Notes in Computer Science*, pages 567–576. Springer Berlin Heidelberg, 2013.

[16] A. C. Arpaci-Dusseau and R. H. Arpaci-Dusseau. Information and control in gray-box systems. In *ACM SIGOPS Operating Systems Review*, volume 35, pages 43–56. ACM, 2001.

[17] H. Bal, C. de Laat, S. Haridi, K. Jeffery, J. Labarta, D. Laforenza, P. Maccallum, J. Mass, L. Matyska, T. Priol, et al. Next Generation Grid (s) European Grid Research 2005–2010. *Information Society Technologies, European Commission, Expert Group Rep*, 2003.

[18] M. Balazinska, H. Balakrishnan, and M. Stonebraker. Contract-based load management in federated distributed systems. In *Proceedings of the 1st conference on Symposium on Networked Systems Design and Implementation-Volume 1*. USENIX Association, 2004.

[19] J. Baldassari, D. Finkel, and D. Toth. SLINC: A Framework for Volunteer Computing. In S. Zheng, editor, *Proceedings of the 18th IASTED International Conference on Parallel and Distributed Computing and Systems*, 2006.

[20] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield. Xen and the art of virtualization. In *SOSP '03: Proceedings of the nineteenth ACM symposium on Operating systems principles*, pages 164–177. ACM, October 2003.

[21] D. Barkai. *Peer-to-Peer Computing: technologies for sharing and collaborating on the net.* Intel Press, 2001.

[22] A. Barmouta and R. Buyya. GridBank: A Grid Accounting Services Architecture (GASA) for Distributed Systems Sharing and Integration. In *Workshop on Internet Computing and E-Commerce, Proceedings of the 17th Annual International Parallel and Distributed Processing Symposium (IPDPS 2003), IEEE Computer Society Press, USA, April*, pages 22–26, 2003.

[23] W. Barth. *Nagios: System and Network Monitoring.* No Starch Press, San Francisco, CA, USA, 2nd edition, 2008.

[24] C. Baumbauer, S. Goasguen, and S. Martin. Bouncer: A globus job forwarder. In *Proc. 1st TeraGrid Conf*, 2006.

[25] A. Belloum, E. Deelman, and Z. Zhao. Scientific workflows. *Scientific Programming*, 14(3–4), 2006.

[26] M. Ben-Yehuda, M. D. Day, Z. Dubitzky, M. Factor, N. Har'El, A. Gordon, A. Liguori, O. Wasserman, and B.-A. Yassour. The turtles project: Design and implementation of nested virtualization. In *Proceedings of the 9th USENIX conference on Operating systems design and implementation*, pages 1–6. USENIX Association, 2010.

[27] A. Berl, E. Gelenbe, M. Di Girolamo, G. Giuliani, H. De Meer, M. Q. Dang, and K. Pentikousis. Energy-efficient cloud computing. *The Computer Journal*, 53(7):1045–1051, 2010.

[28] F. Berman, R. Wolski, H. Casanova, W. Cirne, H. Dail, M. Faerman, S. Figueira, J. Hayes, G. Obertelli, J. Schopf, et al. Adaptive computing on the grid using AppLeS. *IEEE Transactions on Parallel and Distributed Systems*, 14(4):369–382, 2003.

[29] D. Bernstein, E. Ludvigson, K. Sankar, S. Diamond, and M. Morrow. Blueprint for the intercloud-protocols and formats for cloud computing interoperability. In *Fourth International Conference on Internet and Web Applications and Services, 2009. ICIW'09.*, pages 328–336. IEEE, 2009.

[30] D. Bernstein, D. Vij, and S. Diamond. An intercloud cloud computing economy-technology, governance, and market blueprints. In *SRII Global Conference (SRII), 2011 Annual*, pages 293–299. IEEE, 2011.

[31] G. Birkenheuer, A. Brinkmann, and H. Karl. The gain of overbooking. In E. Frachtenberg and U. Schwiegelshohn, editors, *Job Scheduling Strategies for Parallel Processing*, volume 5798 of *LNCS*, pages 80–100. 2009.

[32] B. Boghosian, P. Coveney, S. Dong, L. Finn, S. Jha, G. Karniadakis, and N. Karonis. Nektar, spice and vortonics: Using federated grids for large scale scientific applications. In *Challenges of Large Applications in Distributed Environments, 2006 IEEE*, pages 34–42. IEEE, 2006.

[33] M. Bolte, M. Sievers, G. Birkenheuer, O. Niehörster, and A. Brinkmann. Non-intrusive virtualization management using libvirt. In *Proceedings of the Conference on Design, Automation and Test in Europe*, pages 574–579. European Design and Automation Association, 2010.

[34] M. L. Bote-Lorenzo, Y. A. Dimitriadis, and E. Gómez-Sánchez. Grid characteristics and uses: a grid definition. pages 291–298, February 2004.

[35] J. S. Bozman and G. P. Chen. Optimizing Hardware for x86 Server Virtualization. White Paper.

[36] R. Bradford, E. Kotsovinos, A. Feldmann, and H. Schiöberg. Live wide-area migration of virtual machines including local persistent state. In *Proceedings of the 3rd international conference on Virtual execution environments*, pages 169–179. ACM, June 2007.

[37] I. Brandic, S. Pllana, and S. Benkner. High-level composition of QoS-aware grid workflows: an approach that considers location affinity. In *Workflows in Support of Large-Scale Science, 2006. WORKS'06.*, pages 1–10. IEEE, 2006.

[38] D. Breitgand, A. Marashini, and J. Tordsson. Policy-Driven Service Placement Optimization in Federated Clouds. Technical Report H-0299, IBM Research Report, 2011.

[39] G. Briscoe and A. Marinos. Digital ecosystems in the clouds: towards community cloud computing. In *Third IEEE International Conference on Digital Ecosystems and Technologies, 2009. DEST'09.*, pages 103–108. IEEE, 2009.

[40] J. Brunelle, P. Hurst, J. Huth, L. Kang, C. Ng, D. Parkes, M. Seltzer, J. Shank, and S. Youssef. Egg: An extensible and economics-inspired open grid computing platform, 2006.

[41] R. Buyya. Grid Economy Comes of Age: Gridbus Technologies for Service-Oriented Cluster and Grid Computing. In *Proceedings of the 2 nd IEEE International Conference on Peer-to-Peer Computing (P2P 2002), Linkoping, Sweden, Sept*, pages 5–7, 2002.

[42] R. Buyya, D. Abramson, and J. Giddy. Nimrod/g: An architecture for a resource management and scheduling system in a global computational grid. In *The Fourth International Conference on High Performance Computing in the Asia-Pacific Region, 2000. Proceedings.*, volume 1, pages 283–289. IEEE, 2000.

[43] R. Buyya, D. Abramson, and S. Venugopal. The grid economy. *IEEE Press*, 93;3:698–714, March 2008.

[44] C. Canali and R. Lancellotti. Automatic virtual machine clustering based on bhattacharyya distance for multi-cloud systems. In *Proceedings of the 2013 international workshop on Multi-cloud applications and federated clouds*, pages 45–52. ACM, 2013.

[45] M. Christodorescu, R. Sailer, D. Schales, D. Sgandurra, and D. Zamboni. Cloud security is not (just) virtualization security: a short paper. In *Proceedings of the 2009 ACM workshop on Cloud computing security*, pages 97–102. ACM, 2009.

[46] C. Clark, K. Fraser, S. Hand, J. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield. Live migration of virtual machines. In *Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation-Volume 2*, pages 273–286. USENIX Association, May 2005.

[47] S. Clayman, A. Galis, C. Chapman, G. Toffetti, L. Rodero Merino, L. Vaquero, K. Nagin, and B. Rochwerger. Monitoring Service Clouds in the Future Internet. In *Towards the Future Internet - Emerging Trends from European Research*, pages 115–126, Amsterdam, The Netherlands, 2010. IOS Press.

[48] M. E. Conway. How do committees invent. *Datamation*, 14(4):28–31, 1968.

[49] A. Corradi, M. Fanelli, and L. Foschini. VM Consolidation: a Real Case Based on OpenStack Cloud. *Future Generation Computer Systems*, In Press.

[50] E. Dafouli, P. Kokkinos, and E. Varvarigos. Fair Execution Time Estimation Scheduling in Computational Grids. *Distributed and Parallel Systems*, pages 93–104, 2008.

[51] S. Das, S. Agarwal, D. Agrawal, and A. El Abbadi. Elastras: An elastic, scalable, and self managing transactional database for the cloud. 2009.

[52] M. De Assunção, R. Buyya, and S. Venugopal. InterGrid: A case for internetworking islands of Grids. *Concurrency and Computation: Practice and Experience (CCPE)*, 20(8):997–1024, 2008.

[53] B. de Haaff. Cloud computing - the jargon is back! *Cloud Computing Journal*, August 2008. Electronic Magazine, article available at `http://cloudcomputing.sys-con.com/node/613070`.

[54] Distributed Management Task Force. Cloud Management Standards. `http://www.dmtf.org/standards/cloud`, September 2013.

[55] Distributed Management Task Force, Inc. Open Virtualization Format Specification. DMTF 0243 (Standard), Feb. 2009.

[56] S. Dobson, R. Sterritt, P. Nixon, and M. Hinchey. Fulfilling the vision of autonomic computing. *Computer*, 43(1):35–41, 2010.

[57] A. B. Downey and D. G. Feitelson. The elusive goal of workload characterization. *ACM SIGMETRICS Performance Evaluation Review*, 26(4):14–29, 1999.

[58] I. Drago, M. Mellia, M. M. Munafo, A. Sperotto, R. Sadre, and A. Pras. Inside dropbox: understanding personal cloud storage services. In *Proceedings of the 2012 ACM conference on Internet measurement conference*, IMC '12, pages 481–494, New York, NY, USA, 2012. ACM.

[59] Dropbox Inc. Dropbox. `https://www.dropbox.com`, September 2013.

[60] C. Dumitrescu, M. Wilde, and I. Foster. A model for usage policy-based resource allocation in grids. *Sixth IEEE International Workshop on Policies for Distributed Systems and Networks, 2005.*, pages 191 – 200, June 2005.

[61] C. L. Dumitrescu and I. Foster. Usage Policy-Based CPU Sharing in Virtual Organizations. In *GRID '04: Proceedings of the 5th IEEE/ACM International Workshop on Grid Computing*, pages 53–60, Washington, DC, USA, 2004. IEEE Computer Society.

[62] A. Edmonds, T. Metsch, D. Petcu, E. Elmroth, J. Marshall, and P. Ganchosov. FluidCloud: An open framework for relocation of cloud services. In *Proceedings of the 5th USENIX conference on Hot Topics in Cloud Ccomputing*. USENIX Association, 2013. To appear.

[63] ElasticHosts Ltd. ElasticHosts. `http://www.elastichosts.com/`, September 2013.

[64] M. Ellert, M. Grønager, A. Konstantinov, B. Kónya, J. Lindemann, I. Livenson, J. L. Nielsen, M. Niinimäki, O. Smirnova, and A. Wäänänen. Advanced resource connector middleware for lightweight computational Grids. *Future Generation Computer Systems*, 27(2):219–240, 2007.

[65] E. Elmroth, F. Galán, D. Henriksson, and D. Perales. Accounting and Billing for Federated Cloud Infrastructures. In *GCC '09: Proceedings of the 2009 Eighth International Conference on Grid and Cooperative Computing*, pages 268–275, Washington, DC, USA, 2009. IEEE Computer Society.

[66] E. Elmroth and P. Gardfjäll. Design and evaluation of a decentralized system for Grid-wide fairshare scheduling. In H. Stockinger et al., editors, *First International Conference on e-Science and Grid Computing*, pages 221–229. IEEE CS Press, 2005.

[67] E. Elmroth, P. Gardfjäll, O. Mulmo, and T. Sandholm. An OGSA-Based Bank Service for Grid Accounting Systems. In J. Dongarra et al., editors, *Applied Parallel Computing. State-of-the-art in Scientific Computing*, volume 3732 of *Lecture Notes in Computer Science*, pages 1051–1060. Springer-Verlag, 2005.

[68] E. Elmroth, P. Gardfjäll, A. Norberg, J. Tordsson, and P.-O. Östberg. Designing general, composable, and middleware-independent Grid infrastructure tools for multi-tiered job management. In T. Priol and M. Vaneschi, editors, *Towards Next Generation Grids*, pages 175–184. Springer-Verlag, 2007.

[69] E. Elmroth and D. Henriksson. Distributed Usage Logging for Federated Grids. *Future Generations Computer Systems*, 26(8):1215–1225, 2010.

[70] E. Elmroth, F. Hernández, and J. Tordsson. A light-weight Grid workflow execution engine enabling client and middleware independence. In R. Wyrzykowski et al., editors, *Parallel Processing and Applied Mathematics, Lecture Notes in Computer Science vol. 4967*, pages 754–761. Springer-Verlag, 2008.

[71] E. Elmroth and L. Larsson. Interfaces for Placement, Migration, and Monitoring of Virtual Machines in Federated Clouds. In *Eighth International Conference on Grid and Cooperative Computing (GCC 2009)*, pages 253–260, Los Alamitos, CA, USA, August 2009. IEEE Computer Society.

[72] E. Elmroth and J. Tordsson. An interoperable, standards-based Grid resource broker and job submission service. In H. Stockinger et al., editors, *First International Conference on e-Science and Grid Computing*, pages 212–220. IEEE CS Press, 2005.

[73] E. Elmroth and J. Tordsson. A Grid resource broker supporting advance reservations and benchmark-based resource selection. In J. Dongarra, K. Madsen, and J. Waśniewski, editors, *Applied Parallel Computing - State of the Art in Scientific Computing, Lecture Notes in Computer Science vol. 3732*, pages 1061–1070. Springer-Verlag, 2006.

[74] E. Elmroth and J. Tordsson. Grid Resource Brokering Algorithms Enabling Advance Reservations and Resource Selection Based on Performance Predictions. *Future Generation Computer Systems. The International Journal of Grid Computing: Theory, Methods and Applications*, 24(6):585–593, 2008.

[75] E. Elmroth and J. Tordsson. A standards-based Grid resource brokering service supporting advance reservations, coallocation and cross-Grid interoperability. *Concurrency Computat.: Pract. Exper.*, 21(18):2298–2335, 2009.

[76] D. Espling, D. Armstrong, J. Tordsson, K. Djemame, and E. Elmroth. Contextualization: Dynamic configuration of virtual machines. 2013. Submitted.

[77] D. Espling, L. Larsson, W. Li, J. Tordsson, and E. Elmroth. Modeling and placement of structured cloud services. 2013. Submitted.

[78] D. Espling, P.-O. Östberg, and E. Elmroth. Integration and evaluation of decentralized fairshare prioritization (Aequus). 2013. Submitted.

[79] D. Feitelson. Workload modeling for computer systems performance evaluation. *Book Draft, Version 0.37*, 2012.

[80] A. Ferrer, F. Hernández, J. Tordsson, E. Elmroth, A. Ali Eldin, C. Zsigri, R. Sirvent, J. Guitart, R. Badia, K. Djemame, W. Ziegler, T. Dimitrakos, S. Nair, G. Kousiouris, K. Konstanteli, T. Varvarigou, B. Hudzia, A. Kipp, S. Wesner, M. Corrales, N. Forgó, T. Sharif, and C. Sheridan. OPTIMIS: a Holistic Approach to Cloud Service Provisioning. *Future Generation Computer Systems*, 28:66 – 77, 2011.

[81] L. Field, E. Laure, and M. Schulz. Grid deployment experiences: Grid interoperation. *Journal of Grid Computing*, 7(3):287–296, 2009.

[82] A. Fishman, M. Rapoport, E. Budilovsky, and I. Eidus. HVX: Virtualizing the cloud. In *Proceedings of the 5th USENIX conference on Hot Topics in Cloud Computing*, page To appear. USENIX Association, 2013.

[83] F. Forster. collectd. `http://collectd.org/`, September 2013.

[84] I. Foster. What is the grid? A three point checklist. *GRID today*, 1(6):32–36, 2002.

[85] I. Foster. Globus Toolkit Version 4: Software for Service-Oriented Systems. *Journal of Computer Science and Technology*, 21(4):513–520, 2006.

[86] I. Foster and C. Kesselman. *The Grid: Blueprint for a new computing infrastructure.* Morgan Kaufmann, 2004.

[87] I. Foster, C. Kesselman, and S. Tuecke. The anatomy of the grid: Enabling scalable virtual organizations. *International Journal of High Performance Computing Applications*, 15(3):200–222, 2001.

[88] I. Foster, Y. Zhao, I. Raicu, and S. Lu. Cloud computing and grid computing 360-degree compared. In *Grid Computing Environments Workshop, 2008. GCE'08*, pages 1–10. IEEE, 2008.

[89] P. Fowler, S. Jha, and P. Coveney. Grid-based steered thermodynamic integration accelerates the calculation of binding free energies. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 363(1833):1999, 2005.

[90] F. Galán, A. Sampaio, L. Rodero-Merino, I. Loy, V. Gil, and L. M. Vaquero. Service Specification in Cloud Environments Based on Extensions to Open Standards. In *Proceedings of the Fourth International ICST Conference on COMmunication System softWAre and middlewaRE*, COMSWARE '09, pages 19:1–19:12, New York, NY, USA, 2009. ACM.

[91] S. Garcia-Gomez, M. Escriche-Vicente, P. Arozarena-Llopis, M. Jiménez-Gañán, F. Lelli, Y. Taher, J. Biro, C. Momm, A. Spriestersbach, J. Vogel, G. Le Jeune, A. Giessmann, F. Junker, M. Dao, S. Carrie, J. Niemoller, and D. Mazmanov. 4CaaSt: Comprehensive management of cloud services through a PaaS. In *2012 IEEE 10th International Symposium on Parallel and Distributed Processing with Applications (ISPA)*, pages 494–499, 2012.

[92] P. Gardfjäll, E. Elmroth, L. Johnsson, O. Mulmo, and T. Sandholm. Scalable Grid-wide capacity allocation with the SweGrid Accounting System (SGAS). *Concurrency and Computation: Practice and Experience*, 20(18):2089–2122, 2008.

[93] J. Geelan. Twenty One Experts Define Cloud Computing. *Virtualization*, August 2008. Electronic Magazine, article available at `http://virtualization.sys-con.com/node/612375`.

[94] W. Gentzsch. Sun grid engine: Towards creating a compute power grid. In *First IEEE/ACM International Symposium on Cluster Computing and the Grid, 2001. Proceedings.*, pages 35–36. IEEE, 2001.

[95] S. Glashow. Partial Symmetries of Weak Interactions. *Nucl.Phys.*, 22:579–588, 1961.

[96] D. Gmach, J. Rolia, and L. Cherkasova. Selling t-shirts and time shares in the cloud. In *Proc. of Intl. Symposium on Cluster, Cloud and Grid Computing (CCGrid)*, pages 539–546, 2012.

[97] T. Goodale, S. Jha, H. Kaiser, T. Kielmann, P. Kleijer, G. Von Laszewski, C. Lee, A. Merzky, H. Rajic, and J. Shalf. SAGA: A Simple API for Grid

Applications. High-level application programming on the Grid. *Computational Methods in Science and Technology*, 12(1):7–20, 2006.

[98] Google Inc. Google App Engine. `http://code.google.com/appengine/`, September 2013.

[99] Google, Inc. Google Apps. `http://www.google.com/apps/`, September 2013.

[100] J. Grethe, C. Baru, A. Gupta, M. James, B. Ludaescher, M. Martone, P. Papadopoulos, S. Peltier, A. Rajasekar, S. Santini, et al. Biomedical informatics research network: building a national collaboratory to hasten the derivation of new understanding and treatment of disease. *Studies in health technology and informatics*, 112:100–110, 2005.

[101] R. Gruber, V. Keller, P. Kuonen, M.-C. Sawley, B. Schaeli, A. Tolou, M. Torruella, and T.-M. Tran. Towards an intelligent grid scheduling system. In *Parallel Processing and Applied Mathematics*, pages 751–757. Springer, 2006.

[102] G. Gruman and E. Knorr. What cloud computing really means. *InfoWorld*, April 2008. Electronic Magazine, available at `http://www.infoworld.com/d/cloud-computing/what-cloud-computing-really-means-031`.

[103] R. Guerin, H. Ahmadi, and M. Naghshineh. Equivalent capacity and its application to bandwidth allocation in high-speed networks. *IEEE Journal on Selected Areas in Communications*, 9(7):968–981, 1991.

[104] F. Guim and J. Corbalan. A job self-scheduling policy for HPC infrastructures. In *Job Scheduling Strategies for Parallel Processing*, pages 51–75. Springer, 2008.

[105] D. Hadas, S. Guenender, and B. Rochwerger. Virtual Network Services For Federated Cloud Computing. Technical Report H-0269, IBM Technical Reports, Nov. 2009.

[106] V. Hamscher, U. Schwiegelshohn, A. Streit, and R. Yahyapour. Evaluation of job-scheduling strategies for grid computing. *Grid Computing GRID 2000*, pages 191–202, 2000.

[107] M. R. Hines and K. Gopalan. Post-copy based live virtual machine migration using adaptive pre-paging and dynamic self-ballooning. In *Proceedings of the 2009 ACM SIGPLAN/SIGOPS international conference on Virtual execution environments*, pages 51–60. ACM, 2009.

[108] T. Hirofuchi, H. Nakada, S. Itoh, and S. Sekiguchi. Enabling instantaneous relocation of virtual machines with a lightweight vmm extension. In *10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing (CCGrid), 2010*, pages 73–83. IEEE, 2010.

[109] T. Hirofuchi, H. Nakada, S. Itoh, and S. Sekiguchi. Reactive consolidation of virtual machines enabled by postcopy live migration. In *Proceedings of the 5th international workshop on Virtualization technologies in distributed computing*, pages 11–18. ACM, 2011.

[110] J. Honeycutt. Microsoft Virtual PC 2004 Technical Overview. *Microsoft, Nov*, 2003.

[111] B. Hong and V. Prasanna. Distributed adaptive task allocation in heterogeneous computing environments to maximize throughput. In *Parallel and Distributed Processing Symposium, 2004. Proceedings. 18th International*, page 52. IEEE, 2004.

[112] P. Horn. Autonomic computing: IBM's perspective on the state of information technology. 2001.

[113] E. Huedo, R. Montero, and I. Llorente. The gridway framework for adaptive scheduling and execution on grids. *Scalable Computing: Practice and Experience*, 6(3), 2001.

[114] E. Huedo, R. Montero, and I. Llorente. A recursive architecture for hierarchical grid resource management. *Future Generation Computer Systems*, 25(4):401–405, 2009.

[115] D. Jackson, Q. Snell, and M. Clement. Core Algorithms of the Maui Scheduler. In D. Feitelson and L. Rudolph, editors, *Job Scheduling Strategies for Parallel Processing*, volume 2221 of *Lecture Notes in Computer Science*, pages 87–102. Springer Berlin / Heidelberg, 2001.

[116] B. Jacob, R. Lanyon-Hogg, D. K. Nadgir, and A. F. Yassin. A practical guide to the IBM autonomic computing toolkit, 2004.

[117] K. Jeffery and B. Neidecker-Lutz, editors. *The Future Of Cloud Computing, Opportunities for European Cloud Computing Beyond 2010.* European Commission, Information Society and Media, January 2010.

[118] R. Jimenez-Peris, M. Patiño-Martinez, K. Magoutis, A. Bilas, and I. Brondino. Cumulonimbo: A highly-scalable transaction processing platform as a service. *ERCIM News*, 89, 2012.

[119] B. Kandukuri, V. Ramakrishna Paturi, and A. Rakshit. Cloud security issues. In *2009 IEEE International Conference on Services Computing*, pages 517–520. IEEE, 2009.

[120] G. Katsaros, G. Gallizo, R. Kübert, T. Wang, J. Oriol Fito, and D. Henriksson. A Multi-level Architecture for Collecting and Managing Monitoring Information in Cloud Environments. In *CLOSER 2011: International Conference on Cloud Computing and Services Science (CLOSER)*, Noordwijkerhout, The Netherlands, May 2011.

[121] J. Kay and P. Lauder. A fair Share scheduler. *Commun. ACM*, 31(1):44–55, 1988.

[122] K. Keahey, I. Foster, T. Freeman, and X. Zhang. Virtual workspaces: Achieving quality of service and quality of life in the Grid. *Scientific Programming*, 13(4):265–276, 2005.

[123] K. Keahey and T. Freeman. Contextualization: Providing one-click virtual clusters. In *IEEE Fourth International Conference on eScience, 2008. eScience'08.*, pages 301–308. IEEE, 2008.

[124] K. Keahey, M. Tsugawa, A. Matsunaga, and J. Fortes. Sky computing. *Internet Computing, IEEE*, 13(5):43 –51, September – October 2009.

[125] J. O. Kephart and D. M. Chess. The vision of autonomic computing. *Computer*, 36(1):41–50, 2003.

[126] A. Kertész and P. Kacsuk. A taxonomy of grid resource brokers. *Distributed and Parallel Systems*, pages 201–210, 2007.

[127] K. H. Kim and R. Buyya. Fair resource sharing in hierarchical virtual organizations for global grids. In *GRID '07: Proceedings of the 8th IEEE/ACM International Conference on Grid Computing*, pages 50–57, Washington, DC, USA, 2007. IEEE Computer Society.

[128] A. Kivity, Y. Kamay, D. Laor, U. Lublin, and A. Liguori. KVM: The Linux Virtual Machine Monitor. In *Proceedings of the Linux Symposium*, volume 1, pages 225–230, 2007.

[129] S. D. Kleban and S. H. Clearwater. Fair Share on High Performance Computing Systems: What Does Fair Really Mean? In *CCGRID '03: Proceedings of the 3st International Symposium on Cluster Computing and the Grid*, page 146, Washington, DC, USA, 2003. IEEE Computer Society.

[130] D. Klusáček and H. Rudová. Efficient grid scheduling through the incremental schedule-based approach. *Computational Intelligence*, 27(1):4–22, 2011.

[131] J. Knobloch and L. Robertson. LHC computing Grid technical design report. `https://lcg-archive.web.cern.ch/lcg-archive/TDR/LCG_TDR_v1_04.pdf`, September 2011.

[132] D. Kranzlmüller. The future european grid infrastructure - Roadmap and challenges. In *Proceedings of the ITI 2009 31st International Conference on Information Technology Interfaces, 2009. ITI'09.*, pages 17–20. IEEE, 2009.

[133] K. Krauter, R. Buyya, and M. Maheswaran. A taxonomy and survey of grid resource management systems for distributed computing. *Software: Practice and Experience*, 32(2):135–164, 2002.

[134] H. Lagar-Cavilla, J. Whitney, A. Scannell, P. Patchin, S. Rumble, E. De Lara, M. Brudno, and M. Satyanarayanan. SnowFlock: rapid virtual machine cloning for cloud computing. In *Proceedings of the 4th ACM European conference on Computer systems*, pages 1–12. ACM, 2009.

[135] K. Lai, L. Rasmusson, E. Adar, L. Zhang, and B. Huberman. Tycoon: An implementation of a distributed, market-based resource allocation system. *Multiagent and Grid Systems*, 1(3):169–182, 2005.

[136] E. B. Lakew, F. Hernandez-Rodriguez, L. Xu, and E. Elmroth. Management of distributed resource allocations in multi-cluster environments. In *Performance Computing and Communications Conference (IPCCC), 2012 IEEE 31st International*, pages 275–284. IEEE, 2012.

[137] L. Larsson, D. Henriksson, and E. Elmroth. Scheduling and Monitoring of Internally Structured Services in Cloud Federations. In *Proceedings of IEEE Symposium on Computers and Communications 2011*, pages 173–178, 2011.

[138] E. Laure, S. Fisher, A. Frohner, C. Grandi, P. Kunszt, A. Krenek, O. Mulmo, F. Pacini, F. Prelz, J. White, et al. Programming the Grid with gLite. *Computational Methods in Science and Technology*, 12(1):33–45, 2006.

[139] E. Laure and B. Jones. Enabling Grids for e-Science: The EGEE Project. *Grid computing: infrastructure, service, and applications*, page 55, 2009.

[140] K. Leal, E. Huedo, and I. Llorente. A decentralized model for scheduling independent tasks in Federated Grids. *Future Generation Computer Systems*, 25(8):840–852, 2009.

[141] K. Lee, N. W. Paton, R. Sakellariou, E. Deelman, A. A. A. Fernandes, and G. Mehta. Adaptive workflow processing and execution in pegasus. *Concurrency and Computation: Practice and Experience*, 21(16):1965–1981, 2009.

[142] A. Li, X. Yang, S. Kandula, and M. Zhang. CloudCmp: Comparing Public Cloud Providers. In *Internet Measurement Conference*, 2010.

[143] H. Li, D. Groep, and L. Wolters. Workload characteristics of a multi-cluster supercomputer. In *Job Scheduling Strategies for Parallel Processing*, pages 33–53. Springer, 2005.

[144] W. Li, P. Svärd, J. Tordsson, and E. Elmroth. A general approach to service deployment in cloud environments. In *Second International Conference on Cloud and Green Computing (CGC), 2012*, pages 17–24. IEEE, 2012.

[145] H. C. Lim, S. Babu, and J. S. Chase. Automated control for elastic storage. In *Proceedings of the 7th international conference on Autonomic computing*, pages 1–10. ACM, 2010.

[146] M. Lindner, F. Galán, C. Chapman, S. Clayman, D. Henriksson, and E. Elmroth. The cloud supply chain: A framework for information, monitoring, accounting and billing. In *2nd International ICST Conference on Cloud Computing (CloudComp 2010)*, 2010.

[147] U. Lublin and D. G. Feitelson. The workload on parallel supercomputers: modeling the characteristics of rigid jobs. *Journal of Parallel and Distributed Computing*, 63(11):1105–1122, 2003.

[148] R. Mach, R. Lepro-Metz, B. Hamilton, S. Jackson, and L. McGinnis. Usage Record Format Recommendation. *Draft Rec-UR-Usage, Global Grid Forum, Usage Record WG, March*, 2005.

[149] E. Masanet, A. Shehabi, L. Ramakrishnan, J. Liang, X. Ma, B. Walker, V. Hendrix, and P. Mantha. The energy efficiency potential of cloud-based software: A US case study. 2013.

[150] M. L. Massie, B. N. Chun, and D. E. Culler. The Ganglia Distributed Monitoring System: Design, Implementation And Experience. *Parallel Computing*, 30:2004, 2003.

[151] M. Maurer, I. Brandic, and R. Sakellariou. Enacting SLAs in clouds using rules. In *Euro-Par 2011 Parallel Processing*, pages 455–466. Springer, 2011.

[152] M. Maurer, I. Brandic, and R. Sakellariou. Adaptive resource configuration for cloud infrastructure management. *Future Generation Computer Systems*, 2012.

[153] P. McFedries. The cloud is the computer. *IEEE Spectrum Online*, August 2008. Electronic Magazine, available at `http://www.spectrum.ieee.org/aug08/6490`.

[154] P. Mell and T. Grance. The NIST definition of cloud computing. *NIST special publication*, 800:145, 2011.

[155] S. Meng, L. Liu, and V. Soundararajan. Tide: achieving self-scaling in virtualized datacenter management middleware. In *Proceedings of the 11th International Middleware Conference Industrial track*, pages 17–22. ACM, 2010.

[156] X. Meng, C. Isci, J. Kephart, L. Zhang, E. Bouillet, and D. Pendarakis. Efficient resource provisioning in compute clouds via VM multiplexing. In *Proc. of the Intl. Conference on Autonomic Computing (ICAC)*, pages 11–20, 2010.

[157] Microsoft Corporation. Microsoft Office Live. `http://www.officelive.com`, September 2011.

[158] Microsoft Corporation. Virtual Hard Disk Image Format Specification, September 2013.

[159] Microsoft Inc. Windows Azure: Microsoft's Cloud Platform. `http://www.windowsazure.com/en-us/`, September 2013.

[160] A. W. Mu'alem and D. G. Feitelson. Utilization, predictability, workloads, and user runtime estimates in scheduling the IBM SP2 with backfilling. *IEEE transactions on parallel and distributed systems*, 12(6):529–543, 2001.

[161] S. Murugesan. Harnessing green IT: Principles and practices. *IT professional*, 10(1):24–33, 2008.

[162] J. Nakai and R. F. Van Der Wijngaart. Applicability of markets to global scheduling in grids. *NAS Report*, pages 03–004, 2003.

[163] OnLive, Inc. OnLive.com. `http://www.onlive.com`, September 2013.

[164] Open Cloud Manifesto. Open cloud manifesto. `http://www.opencloudmanifesto.org/`, September 2013.

[165] Open Grid Forum OCCI-WG. Open Cloud Computing Interface. `http://www.occi-wg.org/`, September 2013.

[166] OpenVZ project team. OpenVZ Wiki. `http://www.openvz.org`, September 2013.

[167] P.-O. Östberg. *Virtual infrastructures for computational science: software and architectures for distributed job and resource management.* PhD thesis, Umeå University, Department of Computing Science, March 2011.

[168] P.-O. Östberg, D. Espling, and E. Elmroth. Decentralized scalable fairshare scheduling. *Future Generation Computer Systems*, 29(1):130 – 143, 2013.

[169] P. Padala, K.-Y. Hou, K. G. Shin, X. Zhu, M. Uysal, Z. Wang, S. Singhal, and A. Merchant. Automated control of multiple virtualized resources. In *Proceedings of the 4th ACM European conference on Computer systems*, pages 13–26. ACM, 2009.

[170] P. Padala, X. Zhu, Z. Wang, S. Singhal, K. G. Shin, et al. Performance evaluation of virtualization technologies for server consolidation. *HP Labs Technical Report*, 2007.

[171] Parallels. Parallels Optimized Computing. `http://www.parallels.com/eu/`, September 2013.

[172] F. Perez-Sorrosal, M. Patiño-Martinez, R. Jimenez-Peris, and B. Kemme. Elastic si-cache: consistent and scalable caching in multi-tier architectures. *The VLDB Journal—The International Journal on Very Large Data Bases*, 20(6):841–865, 2011.

[173] S. Pickles, R. Blake, B. Boghosian, J. Brooke, J. Chin, P. Clarke, P. Coveney, N. González-Segredo, R. Haines, J. Harting, et al. The TeraGyroid experiment. In *Proceedings of the Workshop on Case Studies on Grid Applications at GGF*, volume 10, page 2004, 2004.

[174] R. Piro, A. Guarise, and A. Werbrouck. An Economy-based Accounting Infrastructure for the DataGrid. In *Proceedings of the 4th International Workshop on Grid Computing (GRID2003)*, 2003.

[175] G. Popek and R. Goldberg. Formal requirements for virtualizable third generation architectures. *Communications of the ACM*, 17(7):412–421, 1974.

[176] H. Qian and Q. Wang. Towards proximity-aware application deployment in geo-distributed clouds. *Advances in Computer Science and its Applications*, 2(3):416–424, 2013.

[177] Rackspace, US Inc. Rackspace Cloud. `http://www.rackspace.com/cloud/`, September 2013.

[178] J. Rao, X. Bu, C.-Z. Xu, and K. Wang. A distributed self-learning approach for elastic provisioning of virtualized cloud resources. In *IEEE 19th International Symposium on Modeling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS), 2011*, pages 45–54. IEEE, 2011.

[179] T. Renzenbrink. How much electricity does the internet use? `http://www.techthefuture.com/technology/how-much-electricity-does-the-internet-use/`, September 2013.

[180] D. Ritchie and K. Thompson. The UNIX time-sharing system. *Communications of the ACM*, 17(7):365–375, 1974.

[181] L. Robertson. Computing services for lhc: From clusters to grids. In R. Brun, F. Carminati, and G. Galli Carminati, editors, *From the Web to the Grid and Beyond*, The Frontiers Collection, pages 69–89. Springer Berlin Heidelberg, 2012.

[182] B. Rochwerger, D. Breitgand, A. Epstein, D. Hadas, I. Loy, K. Nagin, J. Tordsson, C. Ragusa, S. C. E. Levy, A. Maraschini, P. M. H. Muñoz, G. Toffetti, and M. Villari. RESERVOIR : When one cloud is not enough. *IEEE Computer*, 44(3):44–51, 2011.

[183] B. Rochwerger, D. Breitgand, E. Levy, A. Galis, K. Nagin, I. Llorente, R. Montero, Y. Wolfsthal, E. Elmroth, J. Caceres, M. Ben-Yehuda, W. Emmerich, and F. Galán. The RESERVOIR model and architecture for open federated cloud computing. *IBM Journal of Research and Development*, 53(4), 2009. Paper 4.

[184] B. Rochwerger, C. Váquez, D. Breitgand, D. Hadas, M. Villari, P. Massonet, E. Levy, A. Galis, I. Llorente, R. Montero, Y. Wolfsthal, K. Nagin, L. Larsson, and F. Galán. An Architecture for Federated Cloud Computing. *Cloud Computing*, 2010.

[185] L. Rodero-Merino, L. Vaquero, V. Gil, F. Galán, J. Fontán, R. Montero, and I. Llorente. From infrastructure delivery to service management in clouds. *Future Generation Computer Systems*, 26(8):1226–1240, 2010.

[186] M. Rosenblum. The reincarnation of virtual machines. *Queue*, 2(5):34–40, 2004.

[187] M. Rothstein. An airline overbooking model. *Transportation Science*, 5(2):180, 1971.

[188] M. Russell, P. Dziubecki, P. Grabowski, M. Krysiński, T. Kuczyński, D. Szjenfeld, D. Tarnawczyk, G. Wolniewicz, and J. Nabrzyski. The vine toolkit: A java framework for developing grid applications. *Parallel Processing and Applied Mathematics*, pages 331–340, 2008.

[189] N. Sadashiv and S. D. Kumar. Cluster, grid and cloud computing: A detailed comparison. In *Sixth International Conference on Computer Science & Education (ICCSE), 2011*, pages 477–482. IEEE, 2011.

[190] M. Sakai. ACPI sleep control, July 24 2001. US Patent 6,266,776.

[191] R. Sakellariou and H. Zhao. A hybrid heuristic for dag scheduling on heterogeneous systems. In *Parallel and Distributed Processing Symposium, 2004. Proceedings. 18th International*, page 111. IEEE, 2004.

[192] J. Scanlon and B. Wieners. The internet cloud. *The Industry Standard, Tech. Rep*, 1999.

[193] J. Schopf. Ten actions when Grid scheduling. In J. Nabrzyski, J. Schopf, and J. Węglarz, editors, *Grid Resource Management State of the art and future trends*, chapter 2. Kluwer Academic Publishers, 2004.

[194] L. Seawright and R. MacKinnon. VM/370-A Study of Multiplicity and Usefulness. *IBM Systems Journal*, 18(1):4–17, 1979.

[195] M. Sedaghat, F. Hernández, and E. Elmroth. Unifying cloud management: Towards overall governance of business level objectives. In *11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid), 2011*, pages 591–597. IEEE, 2011.

[196] G. Semeraro, G. Magklis, R. Balasubramonian, D. H. Albonesi, S. Dwarkadas, and M. L. Scott. Energy-efficient processor design using multiple clock domains with dynamic voltage and frequency scaling. In *High-Performance Computer Architecture, 2002. Proceedings. Eighth International Symposium on*, pages 29–40. IEEE, 2002.

[197] V. Shrivastava, P. Zerfos, K.-w. Lee, H. Jamjoom, Y.-H. Liu, and S. Banerjee. Application-aware virtual machine migration in data centers. In *INFOCOM, 2011 Proceedings IEEE*, pages 66–70. IEEE, 2011.

[198] J. Silvestre. Economies and diseconomies of scale. *The New Palgrave: A Dictionary of Economics*, 2:80–84, 1987.

[199] W. Smith, I. Foster, and V. Taylor. Scheduling with Advance Reservations. In *14th International Parallel and Distributed Processing Symposium*, pages 127–132, 2000.

[200] Q. Snell, M. Clement, D. Jackson, and C. Gregory. The performance impact of advance reservation meta-scheduling. In D. G. Feitelson and L. Rudolph, editors, *Job Scheduling Strategies for Parallel Processing: IPDPS 2000 Workshop, JSSPP 2000, LNCS 1911*, pages 137–153. Springer-Verlag, 2000.

[201] B. Sotomayor, K. Keahey, and I. Foster. Combining Batch Execution and Leasing Using Virtual Machines. In *HPDC - The ACM/IEEE International Symposium on High Performance Distributed Computing*, 2008.

[202] B. Sotomayor, R. Montero, I. Llorente, I. Foster, and F. de Informatica. Capacity leasing in cloud systems using the OpenNebula engine. *Cloud Computing and Applications*, 2008, 2008.

[203] B. Sotomayor, R. S. Montero, I. M. Llorente, and I. Foster. Virtual infrastructure management in private and hybrid clouds. *IEEE Internet Computing*, 13:14–22, 2009.

[204] S. Srikantaiah, A. Kansal, and F. Zhao. Energy aware consolidation for cloud computing. In *Proceedings of the 2008 conference on Power aware computing and systems*, volume 10. USENIX Association, 2008.

[205] S. Srinivasan, R. Kettimuthu, V. Subramani, and P. Sadayappan. Characterization of backfilling strategies for parallel job scheduling. In *International Conference on Parallel Processing Workshops, 2002. Proceedings.*, pages 514–519. IEEE, 2002.

[206] H. Stockinger. Defining the grid: a snapshot on the current view. *The Journal of Supercomputing*, 42(1):3–17, 2007.

[207] A. Streit, D. Erwin, T. Lippert, D. Mallmann, R. Menday, M. Rambadt, M. Riedel, M. Romberg, B. Schuller, and P. Wieder. UNICORE - from project results to production grids. In L. Grandinetti, editor, *Grid Computing: The New Frontiers of High Performance Processing, Advances in Parallel Computing 14*, pages 357–376. Elsevier, 2005.

[208] J. Subramanian, S. Stidham, and C. J. Lautenbacher. Airline yield management with overbooking, cancellations, and no-shows. *Transportation Science*, 33(2):147–167, 1999.

[209] P. Svärd, B. Hudzia, J. Tordsson, and E. Elmroth. Evaluation of delta compression techniques for efficient live migration of large virtual machines. In *Proceedings of the 7th ACM SIGPLAN/SIGOPS international conference on Virtual execution environments*, pages 111–120. ACM, 2011.

[210] A. Szalay and J. Gray. The world-wide telescope. *Science*, 293(5537):2037–2040, 2001.

[211] M. Szeredi. Filesystem in userspace. `http://fuse.sourceforge.net/`, September 2013.

[212] I. Taylor, M. Shields, I. Wang, and A. Harrison. The Triana workflow environment: architecture and applications. In I. Taylor et al., editors, *Workflows for e-Science*, pages 320–339. Springer-Verlag, 2007.

[213] D. Thain, T. Tannenbaum, and M. Livny. Distributed computing in practice: The Condor experience. *Concurrency Computat. Pract. Exper.*, 17(2–4):323–356, 2005.

[214] O. Tickoo, R. Iyer, R. Illikkal, and D. Newell. Modeling virtual machine performance: challenges and approaches. *ACM SIGMETRICS Performance Evaluation Review*, 37(3):55–60, 2010.

[215] B. Tierney, R. Aydt, D. Gunter, W. Smith, M. Swany, V. Taylor, and R. Wolski. A grid monitoring architecture. 2002.

[216] L. Tomás, A. C. Caminero, C. Carrión, and B. Caminero. Network-aware meta-scheduling in advance with autonomous self-tuning system. *Future Generation Computer Systems*, 27(5):486 – 497, 2011.

[217] L. Tomás, P.-O. Östberg, B. Caminero, C. Carrión, and E. Elmroth. An adaptable in-advance and fairshare meta-scheduling architecture to improve grid qos. In *Proceedings of the 2011 IEEE/ACM 12th International Conference on Grid Computing*, pages 220–221. IEEE Computer Society, 2011.

[218] L. Tomás and J. Tordsson. Improving cloud infrastructure utilization through overbooking. In *Proceedings of the ACM Cloud and Autonomic Computing Conference*. ACM, 2013. To appear.

[219] J. Tordsson, R. S. Montero, R. Moreno-Vozmediano, and I. M. Llorente. Cloud brokering mechanisms for optimized placement of virtual machines across multiple providers. *Future Generation Computer Systems*, 28(2):358–367, 2012.

[220] M. Tsugawa and J. Fortes. A virtual network (ViNe) architecture for grid computing. In *Parallel and Distributed Processing Symposium, 2006. IPDPS 2006. 20th International*, page 10. IEEE, 2006.

[221] Ubuntu Community. CloudInit - Community Ubuntu Documentation. `https://help.ubuntu.com/community/CloudInit`, September 2013.

[222] H. Using Windows. Server 2008 job scheduler. *Microsoft Corporation, Published: June*, 2008.

[223] W. M. P. van Der Aalst, A. H. M. Ter Hofstede, B. Kiepuszewski, and A. P. Barros. Workflow patterns. *Distributed and parallel databases*, 14(1):5–51, 2003.

[224] L. M. Vaquero, L. Rodero-Merino, J. Cáceres, and M. Lindner. A break in the clouds: towards a cloud definition. *SIGCOMM Comput. Commun. Rev.*, 39(1):50–55, 2009.

[225] VMware. VMware VMotion: Live migration of virtual machines without service interruption datasheet. `http://www.vmware.com/files/pdf/VMware-VMotion-DS-EN.pdf`, September 2013.

[226] A. Waheed, W. Smith, J. George, and J. Yan. An infrastructure for monitoring and management in computational grids. *Languages, Compilers, and Run-Time Systems for Scalable Computers*, pages 619–628, 2000.

[227] J. Walters, V. Chaudhary, M. Cha, S. Guercio Jr, and S. Gallo. A Comparison of Virtualization Technologies for HPC. In *22nd International Conference on Advanced Information Networking and Applications*, pages 861–868. IEEE, 2008.

[228] J. Watson. Virtualbox: bits and bytes masquerading as machines. *Linux Journal*, 2008(166):1, 2008.

[229] A. Weiss. Computing in the clouds. *NetWorker*, 11(4):16–25, 2007.

[230] C. D. Weissman and S. Bobrowski. The design of the force.com multitenant internet application development platform. In *Proceedings of the 35th SIGMOD international conference on Management of data*, SIGMOD '09, pages 889–896, New York, NY, USA, 2009. ACM.

[231] A. Whitaker, M. Shaw, S. Gribble, et al. Denali: Lightweight virtual machines for distributed and networked applications. Technical report, Citeseer, 2002.

[232] D. Williams, H. Jamjoom, and H. Weatherspoon. The xen-blanket: Virtualize once, run everywhere. In *Proceedings of the 7th ACM European conference on Computer Systems*, pages 113–126. ACM, 2012.

[233] T. Wood, A. Gerber, K. Ramakrishnan, P. Shenoy, and J. Van der Merwe. The case for enterprise-ready virtual private clouds. In *Proceedings of the 2009 conference on Hot topics in cloud computing*, page 4. USENIX Association, 2009.

[234] T. Wood, P. Shenoy, A. Venkataramani, and M. Yousif. Sandpiper: Black-box and gray-box resource management for virtual machines. *Computer Networks*, 53(17):2923–2938, 2009.

[235] Worldwide LHC Computing Grid. `http://wlcg.web.cern.ch`. September 2013.

[236] L. Xu, E. B. Lakew, F. Hernandez-Rodriguez, and E. Elmroth. A scalable accounting solution for prepaid services in cloud systems. In *IEEE Ninth International Conference on Services Computing (SCC), 2012*, pages 81–89. IEEE, 2012.

[237] L. Yazdanov and C. Fetzer. Vertical scaling for prioritized vms provisioning. In *Second International Conference on Cloud and Green Computing (CGC), 2012*, pages 118–125. IEEE, 2012.

[238] M. B. Yehuda, O. Biran, D. Breitgand, K. Meth, B. Rochwerger, E. Salant, E. Silvera, S. Tal, Y. Wolfsthal, J. Cáceres, J. Hierro, W. Emmerich, A. Galis, L. Edblom, E. Elmroth, D. Henriksson, F. Hernández, J. Tordsson, A. Hohl, E. Levy, A. Sampaio, B. Scheuermann, M. Wusthoff, J. Latanicki, G. Lopez, J. Marin-Frisonroche, A. Dörr, F. Ferstl, S. Beco, F. Pacini, I. Llorente, R. Montero, E. Huedo, P. Massonet, S. Naqvi, G. Dallons, M. Pezzé, A. Puliato, C. Ragusa, M. Scarpa, and S. Muscella. RESERVOIR - an ICT infrastructure for reliable and effective delivery of services as utilities. Technical report, IBM Haifa Research Laboratory, 2008.

[239] A. Yoo, M. Jette, and M. Grondona. SLURM: Simple Linux Utility for Resource Management. In D. Feitelson, L. Rudolph, and U. Schwiegelshohn, editors, *Job Scheduling Strategies for Parallel Processing*, volume 2862 of *Lecture Notes in Computer Science*, pages 44–60. Springer Berlin / Heidelberg, 2003.

[240] J. Yu, S. Venugopal, and R. Buyya. A market-oriented grid directory service for publication and discovery of grid service providers and their services. *The Journal of Supercomputing*, 36(1):17–31, 2006.

[241] S. Zanikolas and R. Sakellariou. A taxonomy of grid monitoring systems. *Future Generation Computer Systems*, 21(1):163–188, 2005.

[242] S. Zhang, X. Chen, S. Zhang, and X. Huo. The comparison between cloud computing and grid computing. In *International Conference on Computer Application and System Modeling (ICCASM), 2010*, volume 11, pages V11–72. IEEE, 2010.