Metadata Management in Multi-Grids and Multi-Clouds

Daniel Espling[*]

LICENTIATE THESIS, SEPTEMBER 2011
DEPARTMENT OF COMPUTING SCIENCE
UMEÅ UNIVERSITY
SWEDEN

[*] Previously Henriksson.

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

espling@cs.umu.se

ISBN 978-91-7459-281-8
ISSN 0348-0542
UMINF 11.08

Printed by Print & Media, Umeå University, 2011

Abstract

Grid computing and *cloud computing* are two related paradigms used to access and use vast amounts of computational resources. The resources are often owned and managed by a third party, relieving the users from the costs and burdens of acquiring and managing a considerably large infrastructure themselves. Commonly, the resources are either contributed by different stakeholders participating in shared projects (grids), or owned and managed by a single entity and made available to its users with charging based on actual resource consumption (clouds). Individual grid or cloud sites can form collaborations with other sites, giving each site access to more resources that can be used to execute *tasks* submitted by users. There are several different models of collaborations between sites, each suitable for different scenarios and each posing additional requirements on the underlying technologies.

Metadata concerning the status and resource consumption of tasks are created during the execution of the task on the infrastructure. This metadata is used as the primary input in many core management processes, e.g., as a base for accounting and billing, as input when prioritizing and placing incoming task, and as a base for managing the amount of resources allocated to different tasks.

Focusing on management and utilization of metadata, this thesis contributes to a better understanding of the requirements and challenges imposed by different collaboration models in both grids and clouds. The underlying design criteria and resulting architectures of several software systems are presented in detail. Each system addresses different challenges imposed by cross-site grid and cloud architectures:

- The LUTSfed approach provides a lean and optional mechanism for filtering and management of usage data between grid or cloud sites.
- An accounting and billing system natively designed to support cross-site clouds demonstrates usage data management despite unknown placement and dynamic task resource allocation.
- The FSGrid system enables fairshare job prioritization across different grid sites, mitigating the problems of heterogeneous scheduling software and local management policies.

The results and experiences from these systems are both theoretical and practical, as full scale implementations of each system has been developed and analyzed as a part of this work. Early theoretical work on structure-based service management forms a foundation for future work on structured-aware service placement in cross-site clouds.

Populärvetenskaplig Sammanfattning

Grid computing och *cloud computing* är två besläktade metodiker för att komma åt och nyttja stora mängder datorresurser, exempelvis för att göra omfattande beräkningar och simuleringar eller till lagring av väldigt stora mängder data. Datorresurserna ägs och underhålls ofta av en tredje part, vilket besparar användarna kostnaderna och mödan att införskaffa och underhålla den stora infrastrukturen själva, speciellt som den stora mängden datorkraft oftast bara behövs under kortare perioder. Vanligtvis är resurserna antingen ägda av flera oberoende parter som deltar i gemensamma projekt (grid), eller ägda av en enda organisation och görs tillgängliga för allmänheten (eller en begränsad mängd användare) för att sedan debitera användare för datorkraften de faktiskt använder (clouds). Enskilda grids eller clouds kan samarbeta med andra aktörer för att få tillgång till än större mängder resurser som kan användas till att köra jobb åt användarna. Det finns flera olika samarbetsmodeller mellan aktörer som lämpar sig för olika tillfällen, och varje modell medför ytterligare krav på den underliggande tekniken.

När jobb körs på infrastrukturen skapas metadata, information om statusen hos jobbet och mängden resurser som förbrukas när jobbet körs. Dessa metadata är det huvudsakliga underlaget för flera interna processer i infrastrukturen. Exempelvis används det som bas för fakturering, som beslutsunderlag för att välja vilken ordning man ska prioritera jobb och som en indikation för när mängden resurser som tilldelats ett jobb behöver ökas eller minskas.

Med fokus på hanteringen och nyttandet av jobbmetadata bidrar denna avhandling till en djupare förståelse för de problem och krav som uppkommer i grids eller clouds som använder datorresurser från flera olika aktörer. Underliggande designkriterier och de resulterande arkitekturerna för flera mjukvarusystem presenteras i detalj. Varje system fokuserar på olika delar av de utmaningar som sammarbetsmodeller för grids och clouds medför:

- LUTSfed bidrar med filtrering och hantering av metadata mellan flera grids och clouds på ett minimalistisk och smidigt sätt.
- Ett system för bokföring och fakturering från grunden designat för att stödja flera clouds demonstrerar hur användningsdata kan hanteras utan kännedom om var jobben körs eller vetskap om hur mycket resurser jobbet kräver.

- FSGrid möjligör prioritering baserad på tidigare förbrukningsdata på ett enhetligt sätt över flera grids, oavsett skillnader i underliggande mjukvaror eller lokala policies.

Resultaten och erfarenheterna från dessa system är inte enbart teoretiska, eftersom fullskaliga implementationer av samtliga system har utvecklats och analyserats som en del av det här arbetet. Tidiga teoretiska resultat med fokus på placering av jobb i clouds där den interna strukturen hos jobbet tas i beaktning skapar en grund för vidare arbete inom ämnet.

Preface

This thesis contains an introduction to grid and cloud computing, with focus on meta-data management, and the below listed papers. The author changed surname from Henriksson to Espling just prior to printing this thesis, which is why the articles included in this thesis are printed under a different name than the thesis itself.

Paper I	E. Elmroth and D. Henriksson. Distributed Usage Logging for Federated Grids. *Future Generations Computer Systems*, 26(8):1215–1225, 2010.
Paper II	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 III	L. Larsson, D. Henriksson, and E. Elmroth. Scheduling and Monitoring of Internally Structured Services in Cloud Federations. In *Proceedings of IEEE ISCC 2011*, pages 173–178, 2011.
Paper IV	P-O. Östberg, D. Henriksson, and E. Elmroth. Decentralized, scalable, Grid Fairshare Scheduling (FSGrid). 2011. Submitted.

This research was conducted using the resources of the High Performance Computing Center North (HPC2N) and the UMIT research lab. Financial support has been provided by The Swedish Research Council (VR) under contract 621-2005-3667, by the European Community's Seventh Framework Programme ([FP7/2001-2013]) under grant agreement no. 215605 (RESERVOIR) and no. 257115 (OPTIMIS).

In addition to the publications included in the thesis, the following papers on related subjects has also been produced in the context of this work:

- 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).*

- 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.

- G. Katsaros, G. Gallizo, R. Kübert, T. Wang, J. O. 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)*, 2011. Accepted for publication.

Acknowledgments

First and foremost, I would like to thank my supervisor Erik Elmroth for creating (and maintaining) a pleasant, supportive, and inspiring research environment, and for always finding the time despite being a resource constantly subject to overbooking. I am also very grateful for the help and feedback given by my co-supervisor, Johan Tordsson, who took the time to give feedback on this thesis in mid July despite being on vacation and despite Tour de France running on TV.

A big thank you to all collaborators, colleagues in and outside our group, and coauthors of papers both within and outside the bounds of this thesis. You are too numerous to be mentioned by name, but interacting with the lot of you and sharing your views of things to solve shared problems is what makes this job interesting. We are also blessed with a very competent, kind, and understanding administrational staff, both for technical and non-technical tasks. Thank you for making our everyday working lives easier and for never backing down from challenges such as installing software we produce, or sorting my post-laundry traveling receipts.

A special thanks to Lars Larsson, my constant 2*vX* ally. Not only for daily company, support, and interesting discussions, but also for teaching me to leverage obscure tools and features, and for explaining countless times why things like *gqap* are perfectly sane commands to learn by heart.

Last but definitely not least I would like to thank my closest family and my friends for providing an outstanding environment to grow up, live, and hopefully grow old in. To my recently wedded wife Maria Espling, with whom I share everything (including the hassle of changing name halfway through a PhD): *Du är mitt guld också.*

Umeå, September 2011
Daniel Espling

Contents

Chapter 1

Introduction

Computing capacity available as a utility similar to water or electricity has been a vision for a very long time, with the predictions of John McCarty dating from the early sixties often seen as the starting point [62, 64]. Fifty years later there have been several incarnations of this paradigm, with the same underlying goal of computing capacity as a utility. Most often, the new paradigm does not entirely overlap with the previous paradigms in scope, leaving niches for several generations of paradigms to coexist.

Two of the most recent paradigms for computing as a utility are *grid computing* and *cloud computing*. We refer to the paradigms at large simply as *grids* and *clouds*, and use the terms *site* or *provider* to emphasize a single supplier in either paradigm. Work units sent to a grid are usually denoted *jobs* while those sent to a cloud are called *services*[1]. As cloud computing is a quite wide term (see Chapter 3), a cloud *service* can denote several different things. As most of this thesis focus on infrastructure management, we use service to denote self-contained work units supplied to infrastructure providers for execution. We also use the term *task*to denote both grid jobs and cloud services, and each term separately when referring only to either.

Grids and clouds are both fundamentally ways to group existing (heterogeneous) computer resources into an abstract pool of resources, and making those resources available to users as a virtual coherent infrastructure. Starting out with similar objectives, grids have evolved into reliable, high performing platforms mostly used for large-scale scientific computing while clouds has emerged as a remote hosting and execution option for many different kinds of software. Chapter 2 and Chapter 3 describe these paradigms in more detail.

Other relevant paradigms are, e.g., High Performance Computing (HPC) [44] and High Throughput Computing (HTC) [111]. HPC systems focus on running parallel jobs on centralized, dedicated hardware with very high performance in terms of, e.g., computational speed and network latency. HTC on the other hand focuses on maximizing the use of distributed, widely heterogeneous, and

[1]Not to be confused with Web Services [30] as a technology.

unreliable resources not for the sake of a single job but for the general system as a whole. Even though, from a management perspective, HPC and HTC avoids many of the challenges of grids and clouds covered by this thesis, concepts such as those in Paper I (accounting data management) and Paper IV (decentralized fairshare scheduling) can be applied to HPC and HTC environments as well.

Individual grids and clouds can be joined into even bigger pools of resources through collaborations. These multi-grid and multi-cloud environments pose additional challenges for the management of submitted tasks, and several different collaboration models with unique challenges exists [55, 57]. One such collaboration model is *federations* of grids or clouds, where a single grid or cloud may utilize resources from other sites, commonly as part of bilateral resource exchange agreements. For grids, large projects such as the Large Hadron Collider (LHC) [108] has outgrown the capacity of any single grid and require cross-grid solutions to cope with the high resource demand. Similarly, clouds form collaborations to cope with surges in demand when local resources are not sufficient, giving the impression of clouds as endless pools of resources. In some cases, the collaborating cloud may in turn outsource the execution to a third cloud site, creating a chain of delegation from the originating site to the site where the task is finally executed. Clients for grids and clouds should be kept unaware and unconcerned about whether the infrastructure is part of a collaboration or not, and will normally not be aware of on which collaborating site a submitted task is finally executed (as long as the job does not have explicit restrictions on placement). Therefore, the underlying infrastructure itself must deal with any heterogeneity or additional complexity imposed by the collaborative environment, for example the task metadata management.

Metadata concerning, e.g., the resource consumption or duration of a task are collected during (or after) the execution of a task. This metadata has to be collected and managed equally regardless of if the task executes locally or at a collaborating site, as the data is commonly used as basis for many internal processes in both grids and clouds. The process of collecting, sharing, and managing run time information about a task is called *monitoring*. Grids normally only use monitoring information regarding the state of physical resources, and utilize job metadata generated upon job completion for tasks such as accounting, billing, and job scheduling. Clouds typically rely solely on run time monitoring data for internal management processes, as cloud services does not have a fixed execution time.

The focus of thesis is how to collect, manage, and utilize task metadata in different collaboration models of grids and clouds. The thesis investigates how these fundamental tasks are affected by the barriers imposed by collaborations such as federations, e.g., technical heterogeneity, distributed and (site-wise) self-centric decision making, and incomprehensive information on the state and availability of remote resources. Papers I and II focus primarily on the collection and management of task metadata, while papers III and IV focus on how to utilize the task metadata for resource allocations in clouds and grids.

The following summarizing chapters presented prior to the papers provides a general introduction and context to topics relevant to the presented papers: Chapter 2 presents a basic overview of grid computing.Chapter 3 describes cloud computing, including a detailed explanation of the infrastructure management of clouds and several different collaboration scenarios. Chapter 4 presents an overview of task metadata management in both grids and clouds. Papers are summarized in Chapter 5 and potential directions for future work are outlined in Chapter 6 before the bibliography finalizes the summarizing chapters.

Chapter 2

Grid Computing

The foundation of an open networking structure that would later emerge into the Internet was laid by the National Science Foundation (NSF) [123] back in 1986, when the NFSNET backbone was built to connect five supercomputers in the U.S. [62, 107]. Twenty five years later the Internet has evolved into a general utility used by more than two billion people [119]. Meanwhile, grid computing [62] has emerged as a technology and paradigm focusing on the original intent of the Internet – interconnecting resources to form supercomputers.

The analogy between the Internet and grid computing runs deep. The Internet started out as several isolated networks (for example CSNET [35] and ARPANET [2]) only available to specific research communities [107]. Since then, it has evolved into a ubiquitous, unified, and commonly available communications utility. Grid computing stems from the vision of offering computer resources as easily and transparently as electricity using the power grid (and hence the name), while in reality the concept of *The Grid* is still at the stage of early Internet; existing grids are isolated networks targeting specific communities, primarily used for large-scale research projects.

Grid computing as a concept has grown vast enough to encompass many different tools for many different tasks, becoming a group of related technologies rather than a single unified utility. This, and the fact that there is no absolute definition distinguishing grids from other distributed environments, leads to some confusion on what should be considered a grid. Among many definitions [21, 36, 155], the most commonly used definition by Foster [60] comes in the form of a three point checklist, defining grids as systems 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 not standardized, and there are major grid efforts (such as the LHC Computing Grid (LCG) [97]) that groups resources under centralized control while still being referred to as a grid. The view on grids underlying the work presented in this thesis is very similar to Foster's definition, with emphasis on decentralized control of resources and autonomy of participating sites.

Since the initial vision of offering general-purpose computational capacity as a utility, grid computing has evolved into a tool mostly used to enable infrastructure for large-scale scientific projects, such as the Large Hadron Collider (LHC) [108], the World-wide Telescope [159], and the Biomedical Informatics Research Network [73]. In many cases, grids are not only means to share raw computational resources but also makes it possible to share data from important scientific instruments. The project-oriented business model, technical problems (often related to software dependencies), and interoperability issues are a some reasons why the use of grid resources are mostly restricted to specific scientific communities [11, 64]. For these communities, however, grids have made it possible to address problems previously out of reach in terms of computational resource requirements or available scientific tools. A comprehensive overview of grid computing and its implications and uses in several fields (bioinformatics, medicine, astronomy, etc.) is given by Foster and Kesselman [62]. Although this book dates from 2004, the conceptual aspects of grids have not changed notably since.

2.1 Grid as an Infrastructure

The overall purpose of grid computing is to interconnect resources which 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 commonly organized in collaborating scientific communities, Virtual Organizations [63].

A wide variety of grid middlewares including [8, 13, 47, 61, 97, 156, 161, 167] are used as intermediate software layers for job submission and job management in grids. The vast set of different middlewares has created interoperability problems between the middlewares themselves [55], creating an additional niche for software to ease the burden to work with different middlewares [5, 51, 70, 144, 170].

Grid jobs can normally be seen as a self-contained bundle of computational jobs and input data which 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 execution time, which limits the scope of applications suitable for execution on grids. For non-trivial jobs, however, there is commonly considerable amounts of inter-process communication required during job execution. The Job Submission Description Language (JSDL) [9] is a widely accepted standard for specifying job configuration properties such as hardware requirements, execution deadlines, and sets of input and output file required or generated by the computations.

When running a job on a grid, the first step is to select which of the available resources to execute on. This can either be done manually by the user, or by the support of a *resource broker* [26, 54, 99]. Once a suitable resource has been selected, the job is submitted for execution to the local scheduler of that resource. Common technologies for local resource scheduling includes Maui [84] and SLURM [181]. 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 [146].

The lack of user interactions 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 allows users to reserve specific execution times if required, most often at the expense of overall resource utilization due to creation of small unusable gaps prior to the start-time of the reserved jobs [149]. Backfilling techniques [121, 154] are commonly used to increase resource utilization, and may also be used to mitigate the loss of utilization caused by reservations. There are many different strategies to grid job scheduling, some focusing on, e.g, scheduling for the benefit of a single application [18], optimizing the job wait time [77], optimizing the total system throughput [82], avoiding starvation[1], or to offer advance reservations. An early overview and performance comparison of grid scheduling techniques can be found in [79].

Another parameter commonly used in scheduling is *fairness*. The concept, originating from [88], is commonly used in scheduling to take previous consumption and user shares into account, prioritizing jobs for users higher if that user has a lot of unspent shares. There are several approaches to fairshare scheduling in grids, e.g., [38, 43, 45, 49, 94, 96]. The definition of fairness varies between the different approaches, some measuring the total resource utilization, others the number of accepted jobs or the number of missed deadlines per user [129]. All approaches uses some historical utilization data as input in the scheduling process.

A modern batch system scheduler can be configured in many ways to strive towards one or more objectives, normally using weighed combinations of several parameters. The scheduler prioritizes the jobs dynamically and submits jobs for execution on the local resources. After job completion, a *usage record* [112] is generated with metadata concerning the job. This information is subsequently used for internal grid process such as accounting and fairshare scheduling. For more information about metadata management, see Chapter 4.

2.2 Federated Grids

As mentioned in the Internet analogy at the start of the chapter, grids emerged as isolated islands similarly to the early isolated networks now made a part of the unified Internet. The initial vision of grid was a wide spanning resource network functioning as a utility, and there are several efforts to create *federations* of grids [20, 65, 106, 132], where grids unifies (parts of) their resources for

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

common use while still retaining full control over the local infrastructure. For example, the Swedish and Norwegian national grids [125, 150] are two of the actors contributing resources to the Nordic Data Grid Facility (NDGF) [124] consortium. Even though the resources are acquired, owned, and managed by each national grid, a subset of the jobs executed on these resources are run on the behalf of NDGF. In a federation of grids, each site must remain a fully functional autonomous grid in itself, unlike regular computational resources constituting a normal grid which may rely on common grid functionality in order to function. Therefore, federated grids require fully decentralized, but interoperable, solutions in particular for scheduling and metadata management.

The motivations behind federations of grids are not only technical, but often economical or political to consolidate resources and promote collaborations. For instance, EGEE (originally Enabling Grids for E-science in Europe) project [105] is a series of projects initiated by the European Union to create a wide spanning computational grid infrastructure based mainly on the gLite [104] middleware. European Grid Initiative (EGI) [98] is a substantial European initiative to further unify national grids across Europe, largely continuing on the EGEE effort but with a significant focus on seamless interoperability and integration of several different underlying technologies.

Interoperability between different grid deployments is a considerable challenge. Field et al. [58] present a comprehensive overview of challenges in grid collaborations, based on their experiences from work on the EGEE project and co-chairing the Grid Interoperation Now (GIN) [136] efforts. The authors describe several approaches to achieve technical interoperability, and conclude that standardization efforts is the best way to achieve technical interoperability despite demonstrating that enforcing standardization is a time consuming and non-trivial task [58]. Field et al. also emphasize the need to not only consider technical difficulties, but also the differences in operational processes which may prevent seamless interoperability [58]. Task metadata management and compatible monitoring are two of the challenges highlighted by Field et al. that are also within the scope of this thesis.

The TeraGyroid project [132] also presents experiences from federated resource usage. In this project they execute tasks on resources belonging to the US TeraGrid [28] and the UK e-Science Grid [80]. They found that they had to port and configure the application to each resource on the grids on which it should be run, and also had to spend considerable efforts to persuade site administrators in both grids to accept certificates issued by the other party [132].

Boghosian et al. [20] provide invaluable insights on the challenges and advantages of grid federations. In this project, the efforts off three different groups are united to create a federated environment to execute applications which are not embarrassingly parallel. Similarly to the TeraGyriod project [132], these groups spent large efforts on interoperability at the user and middleware layers, saying that the "...the probability of success is likely to decrease exponentially with every additional independent grid.". They also state that "Interoperation between Grids today requires much more than just tedious manual effort; it requires almost heroic effort.", Boghosian et al. found that the primary barrier was not technical, but rather "... the varying levels of evolution and maturity of the constituent Grids." as a result of differences in purposes, priorities, and expertise of the collaborating sites [20].

One of the biggest challenges in federated grids is scheduling [20, 40, 56], especially of non-trivial jobs as the correct execution of a parallel job often means that the job has to be executed in parallel across different sites. The way in which jobs are shared between a set of grids decides the structure and relations of grids within a federation. Fundamental work on distributed scheduling for independent tasks is presented in [106], using meta-schedulers to schedule a common queue of jobs in and between different grids. Other solutions are based on hierarchically organizing grids [17, 83]. Here, a local grid can regard another grid as a very large local resource with special characteristics, and outsource job execution to another grid using standard interfaces.

De Assunção et al. outline the InterGrid [40], a solution based on inter-grid routing analogous to connecting different ISP networks [40, 116], and provides a good overview on the challenges associated with a unified grid. Unfortunately, there are no indications of implementations or practical evaluations of this approach.

Chapter 3

Cloud Computing

Cloud computing has emerged as a broad concept for remote hosting and management of applications, platforms, or server infrastructure, while still offering interactions with remote resources as if they where provisioned locally. The term *cloud computing* originates from the custom of representing computer (or telephone) networks using a drawing of a cloud, hiding the exact location of where things are located or how they are connected. The same analogy applies to computational clouds; the location and other underlying details of remote resources are abstracted and hidden from the user, and the resources are available "on the cloud".

Similarly to grids, cloud computing lacks a crisp and commonly accepted definition and there are many different views (e.g. [64, 68, 75, 176]) as to what constitutes a cloud, and what differs a cloud from a grid (see Section 3.3). Two of the most commonly used definitions originates from the National Institute of Standards and Technology (NIST) [166], and Vaquero et al. [169]. NIST defines [115] cloud computing as:

> *"... a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., 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 different cloud approaches, while the one by Vaquero et al. [169] has additional (non-strict) conditions of Service Level Agreements (SLAs) that guarantees capacity to consumers:

> *"Clouds are a large pool of easily usable and accessible virtualized resources (such as hardware, development platforms and/or services). These resources can be dynamically reconfigured to adjust to a variable load (scale), allowing also for an optimum resource utilization.*

This pool of resources is typically exploited by a pay-per-use model in which guarantees are offered by the Infrastructure Provider by means of customized SLAs."

The above definitions overlap to a large extent, focusing on easy, on-demand access to hardware, application platforms, or services with low delays in the release and provisioning of additional resources. Both definitions employs three widespread service models / scenarios to subdivide the area of cloud computing into subareas:

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 system on the same physical host. Some examples of IaaS providers includes Amazon Elastic Compute Cloud (EC2) [7], Rackspace [135], and VMware vCloud Express [172].

Platform as a Service (PaaS)
Instead of offering access to (virtualized) hardware resources, PaaS systems offers deployment of applications or systems designed for a specific platform, such as a programming language or a custom software environment. PaaS systems includes Google App Engine [71], Saleforce's Force.com environment [177], and upcoming projects such as 4Caast [1], CumuloNimbo [37, 131], and Contrail [120], all supported by the European Seventh Framework Programme.

Software as a Service (SaaS)
Web-based applications including, e.g., Microsoft Office Live [117], Google Apps [72] (not to be confused with App Engine), and the gaming platform OnLive [126] are available to consumers online without the need to install and manage the software locally. The software is instead hosted and managed on remote machines, making it possible to run software (including graphic intensive computer games) on remote servers instead of the local machine.

Of these subareas, SaaS and PaaS are normally developed and maintained by a single administrative unit while IaaS sometimes makes use of resources from several different clouds (similarly to federation of grids). Therefore, the remainder of this thesis focuses on IaaS concepts of clouds, and more specifically on the implications imposed by considering and utilizing resources from more than one infrastructure provider. However, many of the managerial concepts described in Chapter 4 can be applied to, e.g., PaaS and SaaS environments as well.

3.1 Virtualization

Hardware virtualization techniques [14, 134] provide means of dynamically segmenting the physical hardware, making it possible to run several different *Virtual Machines* (VMs) on the same physical hardware at the same time. Each VM is a self contained unit, including an operating system, and booting a VM is very much like powering on a normal desktop computer. The physical resources are subdivided, managed, and made available to the executing VMs through a *Hypervisor* (also called VM Monitor).

The concept of virtualization dates from the late 1960s but have been largely unused for quite some time, until it gained renewed interest in the late 1990s. The oft cited reason is that the widespread x86 processor technology was cumbersome and impractical to virtualize compared to its predecessors, and also became cheap enough to increase the number of computers instead of focusing on virtualization [95]. The late 1990s saw efficient software-based virtualization of the x86 platform, and hardware support for virtualization in processors was released in the mid 2000s [3, 22].

Virtualization is the underlying packaging and abstraction technology for basically all IaaS clouds, and there are also several initiatives for using virtualization in HPC and grid computing. For example, Keahey et al. [90] suggest using VMs in grids to, e.g, better meet quality of service demands and provide easier portability between execution environments. Haizea [151] is a scheduling framework utilizing VMs as a tool to maximize utilization while still supporting advance reservations by suspending and resuming VMs. This way,small gaps between jobs can be utilized by resuming a previously suspended VM. An analysis and comparison of virtualization technologies for HPC is presented by Walters et al. [174].

There are several different technologies for virtualization, which Walters et al. [174] present and organize into four different categories:

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 [175], Parallels Desktop [130], and Microsoft Virtual PC [81].

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 [95], Xen [14], and VMware [171].

Paravirtualization In Paravirtualization [178], the operating system in the virtual machine [147] is modified to make use of an API provided by the hypervisor to achieve better performance than full virtualization. Xen [14] and VMware [171] are two well established technologies supporting Paravirtualization.

Operating System-level Virtualization Unix based virtualization systems such as OpenVZ [128] 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 might be interoperability issues even between different versions of the same hypervisor technology. The hardware support makes native virtualization perform almost at the same level as paravirtualization, keeping the losses imposed by virtualization at a couple of percent [3, 14].

There are several benefits of using virtualization in system management (see, e.g., [142]), but the most important ones in the context of this thesis are: VMs are self contained systems, making it possible to execute the VM on all compatible hypervisors; VMs can be paused and resumed; and VMs can be migrated (moved) either by pausing them and resuming them on another host or by moving them without suspending them. Migration a VM without (non-neglectable) downtime is known as *live migration* [32]. There are several schemes for optimizing the migration process, and live migration of VMs can be done with marginal downtime [23, 32, 158]. Being able to execute VMs on remote hosts without severe software dependencies, and the ability to relocate VMs without major effort or downtime forms the core of multi-site cloud computing concept.

3.2 Cloud as an Infrastructure

The starting point of cloud computing as an infrastructure is arguably Amazon [7] offering the provisioning of their resources to anyone, without the need of any application process or long-term commitments, and charging users only for the resources they actually consume.

The quick provisioning of resources makes it possible for consumers to adapt their current resource requirements with very short delays by starting up or stopping VMs according to their needs. To avoid having to customize large amounts of VMs individually, a VM *template* (or type) is often used to start up several identical *instances*[1].

When starting several instances of VMs it is the responsibility of the software running inside each VM to synchronize with the other running instances, for example by registering with a load balancer. Some configuration settings, such as the IP of the load balancer, cannot be encoded into the template itself, either because it is not available until run time or because it needs to be unique for each VM instance. The process of configuring each instance automatically is called *contextualization* [90, 91, 165]. Contextualization is usually performed just prior to booting a VM, and pausing or resuming (or migrating) a VM does not cause another round of contextualization.

[1]These terms are not to be confused with "Instance Types“, which are predefined hardware configurations of VMs offered by, e.g., Amazon EC2 [7].

There are three main actors involved in cloud infrastructures, illustrated in Figure 1. The *Infrastructure Provider* (IP) owns and manages the physical resources and any supporting software that is required for infrastructure management. The *Service Provider* (SP) is responsible for the contents the service itself, installing and managing the software running inside the VMs. End users are the consumers of the service offered by the SP.

Even though the actors are conceptually separate, the same organization may of course both own the infrastructure, host services on the infrastructure, and be the end users of their own service. There is also a many-to-many relation between the SPs and IPs, and a single IP normally hosts services from many SPs in a multitennant manner (using the isolation of VMs to keep them from interfering each other). Similarly, a single SP may run services (or even parts of services) on several IPs.

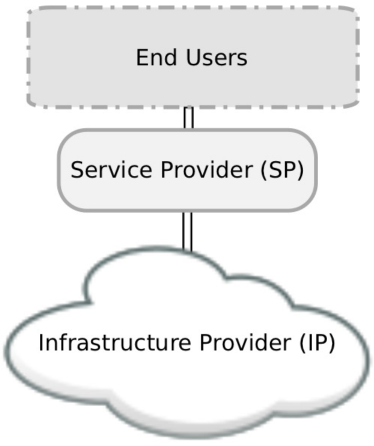


Figure 1: Three main actors for cloud IaaS: the Infrastructure Provider(s) make resources available to Service Provider(s), who in turn offer a software service to End Users.

The IaaS service model is normally offered by the IP, but may have supporting functionality running in the SP. The software running inside the service managed by the SP may consist any type of software, which may (or may not) be other flexible platforms such as PaaS or SaaS solutions. Notably, PaaS or SaaS systems are not required to be hosted on underlying IaaS infrastructures by the service providers, but the variation in resource requirements of PaaS or SaaS systems lends itself well to such solutions. Similarly, SaaS systems may (or may not) be hosted with the support of an underlying PaaS system.

From a resource management perspective, deploying a service to an IP is very much like starting a normal computer application – its lifetime and usage patterns are unknown to the underlying operating system, but the system is still responsible for managing and multitasking different applications without detailed instructions from the user. In an operating system, less prioritized tasks are often neglected in favor of higher prioritized ones, mitigating the problem of insufficient available resources. Similarly, some cloud vendors makes use of less prioritized instances (such as Amazon's Spot Instances [6]) to increase the utilization when the system is not under heavy load. When resources are running low, the IP can may either free up resources by stopping less prioritized services, or by outsourcing the executions of some VMs to other IPs (see Section 3.4).

Security and privacy concerns are commonly seen as the main limiting factor of clouds as a general utility [64, 85]. Compared to grids, where access usually is preceded by face-to-face identity validations and certificate generation, clouds has a relaxed security model reminiscent of regular Internet sites, using Web based forms for sign up and management, and emails for password retrievals [64]. This relaxed security is a great benefit in terms of usability, but limits the trust of major companies considering using clouds for business-sensitive applications. While the ongoing work on cloud security is progressing (see e.g., [31, 85]), privately hosted and managed clouds has become an option for dealing with sensitive data while still gaining some benefits from the cloud computing paradigm.

Early results of scientific computing using clouds are presented in [89], although most of the results are based on "clouds" where a user has to apply by email for the free execution of a VM during short period of time (hours). The lack of quick on-demand provisioning, the need for manual interactions with the providers prior to execution, and the lack of a utility based business model makes it highly debatable whether the systems used in [89] should be considered clouds at all, or rather an extension of the authors earlier published work on Virtual Workspaces in grids [90].

3.3 Grids and Clouds Compared

While both technologies can be seen as enabling technologies to utilize all kinds of computational infrastructure, the main differences are primarily not about technical solutions; as already mentioned, the utilization of virtualized environments to ease deployment and execution for tasks was known in grids before the cloud era [90]. Instead, clouds and grids have emerged as two different paradigms due to approaching the vision of computing capacity as a utility from different angles.

- Grids are designed to support sharing of pooled resources (normally high performing parallel computers) owned and administered by different organizations, primarily targeting users with hardware requirements surpassing the capacity of commodity hardware (e.g. thousands of processor cores or hundreds of terabytes of storage).

- The development of clouds as a technology is driven by economies of scale [148], where the increased utilization of existing (often commodity) hardware resources offers lower operational expenses for the infrastructure providers, which in turn makes it possible for such providers to offer hardware leasing at prices comparable to in-house hosting.

The differences in scope between the paradigms cause considerable differences in e.g. business models, architecture, and resource management. In the context of this thesis, the most interesting differences are those between grid jobs and cloud services, including how resources are provisioned to the supplied tasks. More in-depth comparisons between clouds and grids can be found in, e.g., [64].

Grid jobs by nature are computational jobs executed on infrastructures with very high (combined) performance, granting exclusive access to resources for the job until it is completed before assigning the resources to the next job in the queue. The capacity requirements and execution time of grid jobs are normally known beforehand, and used as input in job scheduling. Cloud services, on the other hand, are expected to start almost immediately after they are submitted and to run without a fixed execution time 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 way resources are managed is analogous to time-sharing [137] (grids) vs. space-sharing [157] (clouds) in operating systems.

The extensive use of VMs in cloud computing also means that the delays for starting up and terminating jobs are greater than those of grid computing, as VMs adds quite a bit of overhead in data transfer and start-up times. To generalize, grids are inherently more suitable for applications with high demands on stability and performance by guaranteeing them exclusive access to resources over a short period of time. Clouds are more suitable for less critical long-running tasks suitable for execution on public or shared hardware, and normally offers support for scaling up and down the amount of allocated resources according to the current needs.

The boundaries between grids and clouds are not absolute and generous definitions of either terms creates a large potential overlap. The technologies can also be used in combination. For example, deployment of the Sun Grid Engine (SGE) [69] in a cloud infrastructure is one of the use cases of the RESERVOIR project (see Section 3.4)[141], showing the plausibility of utilizing the flexibility of clouds to host a grid middleware. To make use of the flexibility of the infrastructure, the SGE was deployed using a master VM for job distribution and several instances of worker VMs for job execution, adapting the amount of worker nodes according to the amount of jobs waiting to be executed [33].

Another effort to run cluster software on IaaS infrastructure presented by Keahey et al. is called Sky Computing [92]. In this approach, resources from three different Universities are combined into ”Virtual Clusters”. Hadoop [179] and Message Passing Interface (MPI) [74] cluster software is hosted on the different VMs, creating a cluster utilizing resources from three university sites.

3.4 Cloud Collaborations

Similarly to federations of grids, clouds can be joined together in different collaboration models to take advantage of the joint infrastructure. While the main advantage of federated grids is the increased capacity, clouds may also take advantage of collaborations to, e.g., offer geographical redundancy or execute services at geographically advantageous locations otherwise outside the available infrastructure. The economical model of clouds gives rise to several different forms of collaborations, described in Section 3.4.1. In some scenarios, a cloud may provision resources from one or more remote cloud(s) using the regular client interfaces, removing the need for prior resource exchange agreements.

In the basic case, the SP interacts with a single IP and is kept unaware of whether the IP uses resources as a part of a collaboration or not. In collaborative cases, the original IP site where the service was submitted is referred to as the *primary* site, while any collaborating sites are the *remote* sites. The control of the service and responsibility towards the SP remains in the primary site regardless of where the service is actually executed, and the primary site is also responsible for ensuring that SLAs are maintained or compensated for. To be able to utilize remote resources, the use of resources between IP sites may be governed by separate SLAs or *framework agreements* [24], stipulating the terms of resource exchange between IPs.

As with grid computing, the use of several clouds introduces a lot of heterogeneity problems that ultimately only can be resolved using standardization efforts. Native (hardware) virtualization is a first major step to standardization on the lowest hardware level. There are also efforts to create standardized and general formats for specifying virtual machines and virtual hard drives [42, 118] and general cloud APIs [34, 41, 113, 127], but neither standard has yet emerged as a generally accepted candidate.

VM incompatibility issues aside, there are a number of operational challenges imposed by the used of collaborative clouds. Since each site retains its full autonomy, and its own policies and objectives, the internal workings of each site are largely obscured to other sites in the collaborations. This means each site only has details available regarding local resources, and at best incomplete information regarding the state in other sites. Service provisioning across clouds therefore has to be based on probabilities and statistics rather than complete information. Another challenge not present in single clouds is that sites participating in collaborations may have external events affecting the state of a service and resource availability of the infrastructure. For example, a remote site may place services on the infrastructure of the primary site, or force the withdrawal of VMs running on the infrastructure of the remote site and therefore forcing the primary site to re-plan the placement across the infrastructure.

The RESERVOIR (Resources and Services Virtualization without Barriers) [138, 139, 140, 180] project focus on creating and validating the concept of cloud federations across several infrastructure providers through several use cases, including running SGE [69] and SAP [145] applications on the federated infrastructure. One of the results of the project is the design and creation of Virtual Application Networks (VANs) [78]. These overlay networks, extending previous work from e.g. [164], offers one solution to allow VMs being a part of internal private networks to be migrated to other sites in the federation without being disconnected. These VANs can be used to manage monitoring information for services spanning several cloud sites, see Section 4.1. The RESERVOIR project also outlines a common specification for cloud services [66, 139] to facilitate interoperability, made by extending the standard Open Virtualization Format [42].

The OPTIMIS [57] project targets the creation of a toolkit of components able to (among other things) support multiple cloud scenarios without extensive changes to the software itself. The project [57] also outlines interesting conflicts of interest between the different actors (SPs and IPs). For example, the ambition of the IP to maximize profit is usually contradictory to the SP ambitions of hosting services at a low cost without neglecting the service performance.

3.4.1 Cloud Computing Scenarios

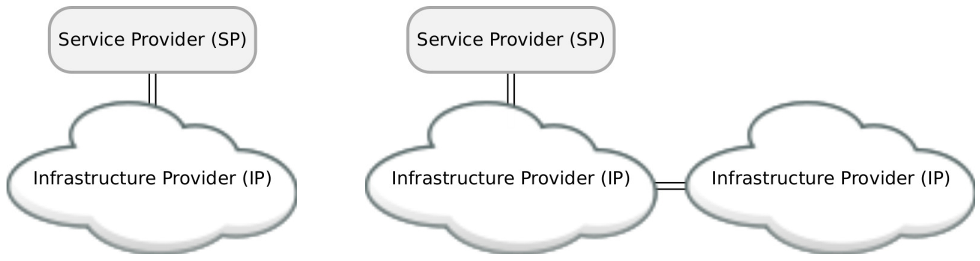
The relation between different clouds in collaboration is commonly modeled as different deployment scenarios [57, 139], depending on the type of interactions between the different sites in the collaboration. We divide the scenarios into three main categories, *federated clouds*, *multi-clouds*, and *private clouds*, each described and illustrated in the coming subsections. Different scenarios can also be combined into *hybrid clouds*, with *bursted private clouds* commonly used as an example. Note that all collaboration scenarios are *multi-clouds* in the sense that they span more than one cloud. The term is used in this more general sense in the title of this thesis, but used in a more specific case in this subsection to describe a specific collaboration scenario. This is done in order to stay in line with, e.g., [57].

Figure 2a shows a simplified model of a standard cloud which is used as the starting point when describing the other deployment scenarios. As previously mentioned in Chapter 3, a single IP normally hosts the services of several SPs, although only a single SP is shown in the illustrations.

Federated Clouds

Federations of clouds (Figure 2b) are formed at the IP level, making it possible for infrastructure providers to make use of remote resources without involving or notifying the SP owning the service. Gaining access to more resources is not the only potential benefit of placing VMs in a remote cloud. Other reasons include fault tolerance, economical incentives, or the ability to meet technical or non-technical constraints (such as geographical location) [138] which would not be possible within the local infrastructure.

Provisioning of remote resources through federations can be done with several remote sites at the same time, using factors such as cost, energy efficiency, and previous performance to decide which resources to use [57]. 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


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

(a) A standard cloud deployment.

(b) Two cloud IPs form a federation.

Figure 2: The illustration on the left shows a standard cloud scenario, where one or more SPs are using the resources of a single IP. In the federated case, shown on the right, an IP may employ other IPs to host (parts of) the running services without involving the SP.

aware of the closest collaborating sites, special care has to be taken in the VM management and information flow in such scenarios [53].

Multi-Clouds

The scenario where the SP itself is involved in moving and prioritizing between different IP offerings is called a *Multi-cloud* [57] scenario. In this case, illustrated in Figure 3, the SP is responsible for planning, initiating, and monitoring the execution of services running on different IPs. Any interoperability issues has to be detected and managed by the SP, affecting the set of sites which can be used for multi-cloud deployments.

The automatic selection and management of different alternatives using *brokers* is a well known approach for, e.g., grid computing [56, 93]. As shown in [57, 163], brokers can also be used as an intermediate component in multi-cloud scenarios. In this case, illustrated in Figure 4, the broker is placed between the SP and the IP. The broker may act as an SP to the IP and as an IP to the SP, containing a lot of the complexity of multi-cloud deployments within the broker itself [57].

Tordsson et al. [163] provide an overview and practical experiences of cloud brokering, including quantified results of performance gained from the brokering of resources belonging to different cloud providers.

Private Clouds

Private clouds, shown in Figure 5a are cloud deployments hosted within the domain of an organization or a company not made available for use by the general public [10]. Such deployments circumvents many of the security concerns related to hosting services in public clouds by keeping the execution within the same security domain, while still offering a computational infrastructure to internal users.

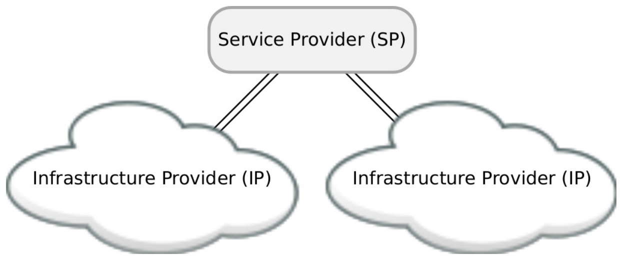


Figure 3: In multi-cloud scenarios, the SP itself may control and decide the deployment of a service using several different IPs.

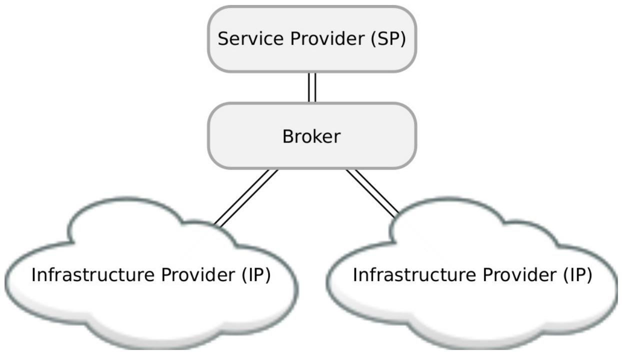


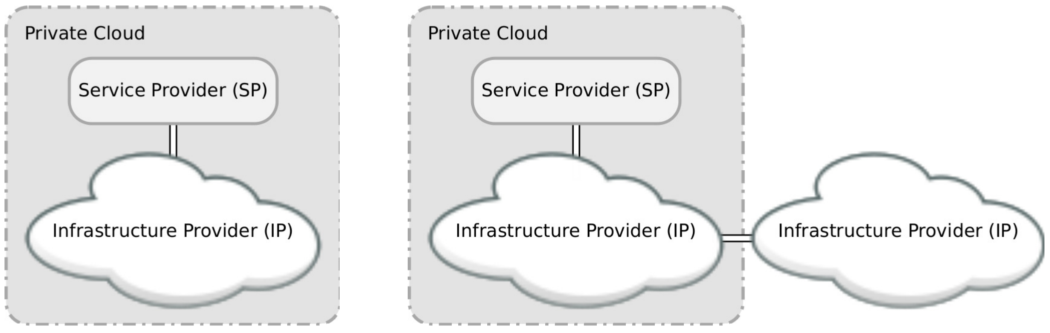
Figure 4: In brokered multi-cloud, a dedicated broker component is used by the SP to simplify the deployment and management process.

Similarly to grids, private clouds only have a finite set of resources and therefore the infrastructure must at some point, prioritize, enqueue, or reject service requests in order to satisfy SLA agreements [153]. It is also likely that private clouds are based on collaboration models between peers rather than pay-per-use alternatives. This creates a need for a service model closer to that of grids than public clouds, and so far there has been little focus in literature on the specific challenges of private clouds.

Hybrid Clouds

Hybrids between different scenarios can be used to overcome limitations of single 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* (described in e.g., [153]) offers a combination of the security and control advantages of private clouds and the seemingly endless scalability of public clouds, but requires very sophisticated placement policies to guarantee the integrity of the system. The relation between private and bursted private clouds is illustrated in Figure 5.

Sotomayor et al. [153] outline the general concepts of hybrid clouds and provides an overview of different cloud technologies and their support for hybrid models. In their work, OpenNebula [152] is used to create hybrid cloud solutions


Private Cloud
Service Provider (SP)
Infrastructure Provider (IP)
Private Cloud
Service Provider (SP)
Infrastructure Provider (IP)
Infrastructure Provider (IP)

(a) Private cloud. (b) Bursted private cloud (hybrid)

Figure 5: Private clouds offer stronger guarantees on control and security as the whole infrastructure can be administered within the same security domain. If needed, private clouds may have less sensitive tasks be executed on a public cloud instead, forming a hybrid cloud scenario commonly referred to as bursted private cloud.

based on a private infrastructure and a set of cloud drivers used to burst to different external providers such as Amazon EC2 [7] or ElasticHosts [46].

Chapter 4

Task Metadata Management

The primary focus of this thesis is the collection, management, and use of task metadata in distributed and multi-provider infrastructures such as grids and clouds. Previous chapters have introduced the fundamental concepts of the main paradigms, including different collaboration models, and this chapter outlines internal infrastructure procedures related to task metadata.

The task metadata contains information about, e.g., the duration, status, and resource consumption of a running task, and forms the primary source of feedback for different internal procedures in the infrastructure. The following sections covers gathering and managing of task metadata, and describes different internal grid and cloud infrastructure processes using the metadata as the primary input.

4.1 Monitoring

Monitoring is the process of gathering information about infrastructure or a service during run time. In grid systems, the focus of monitoring lies on the health, performance, and status of the infrastructure resources [173, 183]. This information is subsequently used for fault detection and recovery, prediction of resource performance, and also to tune the system for better performance [162]. Grid monitoring is slightly out of scope regarding task metadata management, as monitoring is normally not performed regarding the grid jobs themselves (see [183] for a comprehensive overview of grid monitoring). Instead, metadata concerning the result and status of a grid job is collected once the job has terminated (in the shape of usage records), regardless of if the job succeeded to complete successfully or not. Creation and management of these records are further discussed in Section 4.2.

Monitoring of running services is fundamental in clouds as monitoring data is the primary input used in most internal management procedures. The lack of compatible monitoring is one of the main incompatibility hurdles of cross-site clouds [10, 103]. There are three different kinds of monitoring data used in clouds, measurements from the infrastructure, the hypervisor, or from within the service itself:

- Infrastructure specific measurements showing the health and utilization of physical resources. Monitoring the state of infrastructure resources is not a specific problem for cloud computing, and the same tools used for general purpose system monitoring (such as Nagios [16], Ganglia [114], or collectd [59]) can be used also in these contexts.
- Data concerning the resource consumption of individual VMs running on the hardware can be obtained by communicating with the VM hypervisor, or by using tools (such as the libvirt [109] API) that are capable of operating across several different hypervisors. The VM information is commonly used to perform the fulfillments of SLAs or as input to elasticity and service profiling.
- Service specific Key Performance Indicators (KPIs) are used to measure and manage monitoring values specific to the service. These values 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, e.g., elasticity.

Measuring and managing monitoring KPIs from inside the service itself is an interesting problem that is not yet well studied [86]. Some cloud solutions (such as RESERVOIR [139]) have a strong separation between service management and the VM itself, in the sense that the VM is unaware of the location of the management components, and the management components are unaware of the location of the VM. This *location unawareness* [48, 53, 78] has a great influence on what techniques can be used to make the service specific data available to the cloud infrastructure from inside the VMs.

An important factor to consider in cloud collaborations is that more than one site might be interested in the monitoring data produced for a given service. For this reason, naive solutions such as sending the data from inside the VM to an external internet endpoint cannot be used in, e.g., federation scenarios, as the data would not be visible to the infrastructure on the remote site[1]. There is also no guarantee that all VMs of a service has external network access [78]. Instead, the monitoring data has to flow back from the executing site to the primary site through any intermediaries (if any).

The Lattice framework [33] presents a solution for service level monitoring based on customized virtual networks (VANs [78]) to pass measurements from inside the VMs to the infrastructure on the outside without external network

[1]Recall that VMs are not re-contextualized when they are migrated.

access. In this solution, the functionality of the network broadcast directive is overridden and used for monitoring tasks instead. However, without the customized virtual networks this solution would not be possible, and so this is not a generally applicable alternative.

An alternative based on File System in User Space (FUSE) [160] is outlined in [53]. In this solution, FUSE is used to create a small application that simulates a hard drive partition. File system calls (such as writes) result in a normal programmer controlled method call in the application, and the complexity of externalizing the data can be hidden inside the FUSE based application. The problems of actually externalizing the data without knowing the location of some management component remains unsolved, however.

An architecture and implementation of a service oriented monitoring framework for use in cloud infrastructures is presented in [87]. This approach does not seem to consider the problem imposed by service level management nor federations, but instead focuses on monitoring of information from different sources and for use in real-time applications.

4.2 Accounting and Billing

Accounting systems are responsible for metering and managing records on resource consumption by users in grids or clouds. In grids, a *Usage Record* [112] for a job is usually created once the job has finished executing. The usage record contains a lot of general metadata about the job, such as when it was started and finished, and may also contain a summary of the combined resource consumption of a job in terms of, e.g., amount of data transferred on the network. Cloud systems normally rely on run time monitoring of service resource consumption as a basis for accounting.

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 linking these resources together. Managing usage records in such environments is the subject of Paper I [52]. For cross-site cloud computing, the aggregation of data from different site is usually managed by the underlying monitoring system, as accounting is not the only internal cloud process depending on the aggregated raw monitoring data.

One of the major differences between grids and clouds is the underlying economical model, which can be clearly linked to the origins of each paradigm and to the niches they occupy today. For grids, the most common solutions are based collaborative sharing models where the usage data is converted to abstract currencies [15, 67, 133]. Abstract credits are normally awarded to users through an out-of-band application procedure, in which 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 suggestions on how to achieve economical models and architectures for use in grids, commonly based on auctions or other market-based schemes, some of which can be found in [12, 25, 27, 50, 101, 182]. Nakai and Van Der Wijngaart [122] presents an in-depth economical analysis of the feasibility and expectations of markets in grid scheduling, proving that the use of markets is not generally applicable and may not lead to the desired outcomes [122].

Many grid accounting systems also support converting the abstract currencies to real monetary units (at least by easily extending the core mechanisms), but real economical models for grid usage has never been widely adopted. One reason could be that the allocation of abstract credits means that stakeholders can partly affect the utilization of the infrastructure. The use of real money could mean that smaller projects could be constantly outbid by other consumers, preventing them from utilizing the common infrastructure.

In public clouds, users are free to request as much resources as they require on the short term, and paying only for the resources they are currently requesting. In such systems, the accounting data (based on monitoring) is used as input in the *billing* process, converting the hardware measurements to real monetary bills using different pricing schemes.

The two major payment models used in clouds are *prepaid* and *postpaid*, used in the same manner as in the mobile-phone industry. Prepaid, where credits are purchased in advance and consumed in accordance with resource consumption, offers greater control over the maximum costs but running out of credits may cause the service to stop executing. Postpaid, where the consumer is billed at regular intervals for the previous usage, is more sensitive to unexpected amounts of resource consumption, but does not risk running out and hence disturbing the service execution.

Many cloud providers employ overbooking strategies [143] and sell more resources than is actually available, relying on probabilistic models that not all resources are be requested at the same time [76]. However, overbooking strategies ultimately leads to increased amounts of broken SLAs, and each broken SLA generates compensations to the SP. Therefore, dealing with both costs and compensations is a major requirements for accounting in clouds. Birkenheuer et al. [19] show that overbooking schemes are valid options and can achieve a 20% increase in profit even when considering compensations for broken SLAs.

Deployment scenarios such as bursted private clouds or cloud federations offers seemingly unlimited hardware resources, as there may always be resources available at collaborating sites. In theory, this means that also the amount of accounting (and monitoring) data generated by services in the cloud is unlimited. Accounting data is commonly considered financial data, with means there are high demands on storing and managing such data over a long period of time (at least ten years in some jurisdictions). This creates a resource provisioning problem for the management of accounting data similar to the problem addressed by cloud computing itself. Totally scalable solutions for such data has not yet been fully established, but initial work on this subject can be found in, e.g., [39, 110].

4.3 Scheduling and Placement

The process of assigning incoming tasks to available resources, usually denoted *Scheduling* for grids and *Placement* for clouds (although scheduling is sometimes used also for cloud services), is one of the internal processes often relying on task metadata for future decisions.

In grid computing, fairshare scheduling [38, 45, 49] is a wide-spread approach where the scheduler tries to distribute computational resources according to predefined usage shares. The scheduler normally operate on aggregated task metadata for each user and compares the previous usage to the users predefined allocation of resources, using the difference between promised and utilized resources as a factor for prioritizing incoming jobs. The data used in the fairshare process is usually based on usage records, obtained either by querying the underlying accounting system or by receiving such records straight from the infrastructure. The accuracy and availability of usage records and the delay before the data is made available to the fairshare scheduler directly affects the performance and convergence time in the system. Preliminary results in quantifying the relation between task metadata management and job convergence is presented in Paper IV, and further evaluation of these factors are part of future work.

Similarly to grid scheduling, cloud placement can be focused on several objectives and the objectives of each autonomous site may be different [24]. Even within a single site there might be several conditions to consider, and there is often a trade-off between multiple factors such as maximizing the utilization of the infrastructure while minimizing the risk of breaking SLAs. Currently, the amount of broken SLAs seems to be the primary means of measuring the suitability of a cloud deployment. The placement problem takes very different forms in different cloud scenarios. The limited resources in private clouds creates a need for similar solutions as employed in grid computing, as the total amount of requested capacity will be larger than the available resources at some point [153]. In public clouds, the resources are seemingly unlimited and solutions of the placement problem for an IP can focus on optimizing the revenue while minimizing the risk of breaking SLAs [24]. Hybrid scenarios such as bursted private clouds have different challenges as the utilization of the limited local resources must be balanced with the higher costs (and insecurity) of the public resources. As shown by Van den Bossche et al. [168], approaches which perform very well in public clouds may perform drastically worse in bursted private cloud settings due to very large differences in the required time to find an optimal solution when considering also the internal resources. Similarly to fairshare scheduling in grids, the quality and availability of monitoring data and the delays imposed by collaborations is likely to have a great impact on cloud placement, but quantification and further analysis of these areas are subject to future work.

4.4 Elasticity

The ability to quickly request or release resources in response to the current load of a service is one of the most prominent features of cloud computing. *Elasticity* is the process of automating the decisions for when to scale up or down and transfer the decision making from human administrators to processes running in SP or in the infrastructure. By specifying a set of *Elasticity Rules* [141] and include the rules in the service manifest [66], the rules for scaling a service becomes an integral part of the service itself. The rules can be used to specify, e.g., how many users can be served by each VM instance, which may be used in combination with reactive or predictive models to calculate the number of required instances [4].

There are two types of elasticity, *horizontal elasticity* and *vertical elasticity* [4]. In horizontal elasticity, the number of VM instances of a certain type is increased or decreased to correlate with the current load. In vertical elasticity, the amount of hardware resources assigned to one or more VM(s) (such as the amount of RAM or number of CPUs) is dynamically increased or decreased. Horizontal elasticity puts additional strain on the application running inside the VMs, as the system itself must synchronize the tasks between the different instances. Vertical elasticity, on the other hand, requires that the operating system and application running inside the VM is capable of making efficient use of, e.g., a dynamic amount of available RAM.

The elasticity process is normally based on monitoring data concerning the hardware consumption of the VM, or on KPIs monitored from inside the application itself. To shorten the reaction time, elasticity requires up do date measurements regarding the state or KPIs of each VM regardless of where in the (cross-site) infrastructure each VM is running. Normally, the time required for instantiation of new VMs is a few minutes, but recent efforts by, e.g., Lagar-Cavilla et al. [100] has shown that new VMs can be started up in the matter of seconds using techniques similar to *fork* system calls. To avoid becoming the bottleneck, performance is a key requirement for monitoring solutions designed to support rapid elasticity.

Chapter 5

Summary of the Papers

The publications in this thesis focus on different aspects of task data management, including collecting of task data, management of the data within a distributed grid or cloud environment, and how the collected data can be used in different internal procedures such as accounting, billing, or job scheduling.

5.1 Paper I

Paper I [52] investigates how task data collected across autonomous nodes in different distributed grid or cloud usage scenarios can be managed and shared to other parties in the collaboration. The scenarios considered in this paper are hierarchies of grids, mutual grid collaborations, and federations of clouds.

In the paper, we identify a set of requirements from the different usage scenarios, and use these requirements to evaluate several different approached to task data sharing in these environments. This process results in the implementation and evaluation of a light-weight component (LUTSfed) controlling the flow of usage information between different parts of the collaboration. This process is made non-intrusive and optional by reusing existing read and write interfaces of the data management components, and adds support for different cardinality (one-to-many, many-to-one, many-to-many) in usage sharing. We demonstrate how the LUTSfed component can be used to realize the three different usage scenarios by configuring and deploying the component in different ways, without affecting the operation of the already running data management components in different parts of the collaboration.

The performance of the stand-alone LUTSfed component is evaluated in different scenarios, including relations of different cardinality between the number of source and target components and the performance losses inferred by using the LUTSfed component.

5.2 Paper II

Paper II [48] investigates accounting and billing in the federated cloud environments introduced in the RESERVOIR project. The paper is based on two new use cases not present in traditional grid and cloud environments; accounting for cloud services for which the number of sub-components is dynamic and unknown to the accounting system, and accounting for cloud services in which also the placement of sub-components in the federated infrastructure is dynamic and unknown.

A set of requirements for an accounting and billing system in federated clouds is formulated based on the use cases and general non-functional requirements. Existing grid accounting systems are evaluated based on these requirements, but no existing alternative is found to fully support the set of requirements imposed by this environment. Instead, a new architecture for a cloud focused accounting and billing subsystem is proposed. This new architecture is designed from the start to fulfill the requirements imposed by the federated cloud environment. Paper II describes the proposed architecture is detail.

5.3 Paper III

Paper III [103] provides a unified view on a set of managerial challenges present in cloud federations, namely representation, placement, and monitoring of cloud services. A cloud service may constitute of several different subsystem, such as internal networks and shared file storage nodes, and may also have different requirements and restrictions for placement in different parts of the service. Paper III presents a model for expressing the internal structure of cloud services including, e.g., geographical or intra-component affinities (placement restrictions).

When performing placement of cloud services, the placement restrictions expressed in the service structure must be considered. Migrating (moving) parts of the service to another host may require cascading migrations in order to adhere to the specified affinities. We present a model for placement that abides the specified constraints, and extend this model with a heuristic to determine which parts of the service that are suitable for migration.

The placement process is partly based on monitoring data collected from different parts of the cloud federation. The paper presents a data distribution architecture based on semantic metadata annotations that may be used to bridge the gap imposed by monitoring systems in different parts of the cloud federations.

5.4 Paper IV

Paper IV [129] presents the design and functional evaluation of a grid-wide support system for job prioritization based on fairshare allocations, based on earlier work published in [49].

The proposed system is a distributed, stand-alone, support-system usable by job schedulers to externalize the fairshare prioritization procedure. 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 per user and/or sub-project. The paper describes an algorithm for prioritizing user jobs based on predefined user shares and historical usage data. Finally, 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 demonstrated, showing that system is capable of achieving grid-wide fairshare also in the presence of dynamically changing policies and run-time site failures. Performance results show that the convergence rate of the system is greatly affected by delays in data updates, highlighting the relation between task metadata management and system performance.

Chapter 6

Future Work

As discussed in this thesis, task metadata management is a fundamental task in both grids and clouds and the data is used as the primary input to many different processes. The following sections present categories of future work related to the topics described in the thesis.

6.1 Service Monitoring

As outlined in Section 4.1, the problem of monitoring of internal service data is an open but important problem in cloud computing. The data extracted from the application running in the VM is commonly used in, e.g., accounting and elasticity, and data extraction must be coherent and with low delays regardless of if the site is participating in a collaboration or not. A general solution for extracting information from inside the VM and making it available to the cloud infrastructure is one potential area for future work, possibly considering the aid of the hypervisor software itself by employing specific system calls similar to those used in paravirtualization.

6.2 Accounting and Billing

Research on accounting and billing is so far focused on IaaS clouds, quantifying resource consumption in similar ways as in grid computing. A possible future work direction is to investigate if and how these systems would have to evolve to be applicable also to PaaS and SaaS environments.

Private (and hybrid) clouds are popular alternatives for hosting sensitive applications, but so far accounting and billing for these kinds of clouds have not been thoroughly explored. The usage models of private clouds is most likely closer to those of a collaborative grid than a public cloud, and the monetary based compensation system used in public clouds may therefore not be applicable. Similarly, bursted private clouds probably has to incorporate limitations on how much external resources may be provisioned, and by which users [168].

As briefly mentioned in Section 4.2, the amount of accounting and billing data in scenarios such as bursted private clouds or cloud federation can potentially be unlimited. Since this data is commonly required to be stored and managed for a long period of time provisioning resources for management of accounting and billing data is a resource scaling problem yet unsolved. This problem is briefly discussed in [110], outlining record data aggregation and scalable database back-ends as two possible approaches to this problem. However, further work is required to determine the implications on data consistency and durability if using scalable database back-ends such as ElasTras [39], Cassandra [102] or BigTable [29].

6.3 Fairshare Scheduling

Future work on fairshare scheduling in grids as outlined in Paper IV [129] includes more in-depth analysis of different algorithms for calculating fairness based on the historical usage and user allocations. Different algorithms and different settings of parameters such as the amount of historical data to consider is likely to have a large impact on the behavior of the system.

Early results on the impact of task metadata management on the accuracy of fairshare is presented in Paper IV, and further analysis and quantification within this area is subject to future work. As outlined in [49] the inclusion of estimated times for running jobs in the fairshare process may also have a great impact on the accuracy and convergence rate of the system, and is one possible avenue for further studies especially with regard to the additional requirements on metadata management associated with dealing also with running jobs.

Further integration work with different scheduler software and evaluation of the system performance over a long time in a real deployment is also part of future work.

Bibliography

[1] 4CaaSt project. Morfeo 4CaaSt. `http://4caast.morfeo-project.org/`, September 2011.

[2] J. Abbate. From ARPANET to INTERNET: A history of ARPA-sponsored computer networks, 1966-1988. 1994.

[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] A. Ali-Eldin, J. Tordsson, and E. Elmroth. An Adaptive Hybrid Elasticity Controller for Cloud Infrastructures. 2011. Submitted.

[5] 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.

[6] Amazon.com, Inc. Amazon EC2 Spot Instances. `http://aws.amazon.com/ec2/spot-instances/`, September 2011.

[7] Amazon.com, Inc. Amazon Elastic Compute Cloud. `http://aws.amazon.com/ec2`, September 2011.

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

[9] 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 2011.

[10] M. Armbrust, A. Fox, R. Griffith, A. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, et al. Above the clouds: A berkeley view of cloud computing. *EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2009-28*, 2009.

[11] 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.

[12] 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*, pages 15–15. USENIX Association, 2004.

[13] 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.

[14] 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.

[15] 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.

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

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

[18] 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. *Parallel and Distributed Systems, IEEE Transactions on*, 14(4):369–382, 2003.

[19] G. Birkenheuer, A. Brinkmann, and H. Karl. The gain of overbooking. In *Job Scheduling Strategies for Parallel Processing*, pages 80–100. Springer, 2009.

[20] 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.

[21] M. Bote-Lorenzo, Y. Dimitriadis, and E. Gómez-Sánchez. Grid characteristics and uses: a grid definition. In *Grid Computing*, pages 291–298. Springer, 2004.

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

[23] R. Bradford, E. Kotsovinos, A. Feldmann, and H. Schiöberg. Live wide-area migration of virtual machines including local persistent state. In *VEE '07: Proceedings of the 3rd international conference on Virtual execution environments*, pages 169–179. ACM, June 2007.

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

[25] 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.

[26] R. Buyya, D. Abramson, and J. Giddy. Nimrod/G: An architecture for a resource management and scheduling system in a global computational grid. In *hpc*, page 283. Published by the IEEE Computer Society, 2000.

[27] R. Buyya, D. Abramson, and S. Venugopal. The grid economy. *Proceedings of the IEEE*, 93(3):698–714, 2005.

[28] C. Catlett. The philosophy of TeraGrid: building an open, extensible, distributed TeraScale facility. In *CCGrid*, page 8. Published by the IEEE Computer Society, 2002.

[29] F. Chang, J. Dean, S. Ghemawat, W. Hsieh, D. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. Gruber. Bigtable: A distributed storage system for structured data. In *Proceedings of the 7th USENIX Symposium on Operating Systems Design and Implementation (OSDI'06)*, 2006.

[30] R. Chinnici, J. Moreau, A. Ryman, and S. Weerawarana. Web services description language (wsdl) version 2.0 part 1: Core language. *W3C working draft*, 26, 2004.

[31] 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.

[32] C. Clark, K. Fraser, S. Hand, J. G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield. Live Migration of Virtual Machines. In *Proceedings of the 2nd ACM/USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, pages 273–286. ACM, May 2005.

[33] 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, The Netherlands, 2010. IOS Press.

[34] Cloud Computing Interoperability Forum. Unified Cloud Interface Project. `http://www.cloudforum.org/`, September 2011.

[35] D. Comer. The computer science research network CSNET: A history and status report. *Communications of the ACM*, 26(10):747–753, 1983.

[36] CoreGRID. CoreGRID annual report 2007. `http://www.coregrid.net/mambo/content/view/310/301/`, September 2011.

[37] CumuloNimbo project team. CumuloNimbo: Highly Scalable Transactional Multi-Tier PaaS Main menu. `http://www.cumulonimbo.eu/`, July 2011.

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

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

[40] 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.

[41] Distributed Management Task Force. Cloud Management Standards. `http://www.dmtf.org/standards/cloud`, September 2011.

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

[43] N. Doulamis, E. Varvarigos, and T. Varvarigou. Fair Scheduling Algorithms in Grids. *IEEE Transactions on Parallel and Distributed Systems*, 18:1630–1648, 2007.

[44] K. Dowd. *High performance computing.* O'Reilly & Associates, Inc., 1993.

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

[46] ElasticHosts Ltd. ElasticHosts. `http://www.elastichosts.com/`, September 2011.

[47] M. Ellert, M. Grønager, A. Konstantinov, B. Konya, J. Lindemann, I. Livenson, J. Nielsen, M. Niinimäki, O. Smirnova, and A. Wäänänen. Advanced Resource Connector middleware for lightweight computational Grids. *Future Generation computer systems*, 23(2):219–240, 2007.

[48] 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.

[49] 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.

[50] 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.

[51] 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.

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

[53] 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.

[54] 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.

[55] 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.

[56] 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.

[57] A. J. Ferrer, F. Hernández, J. Tordsson, E. Elmroth, A. Ali-Eldin, C. Zsigri, R. Sirvent, J. Guitart, R. M. Badia, K. Djemame, W. Ziegler, T. Dimitrakos, S. K. 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. 2011. Accepted.

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

[59] Florian Forster. collectd. `http://collectd.org/`, September 2011.

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

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

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

[63] 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.

[64] 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.

[65] 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.

[66] 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.

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

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

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

[70] 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.

[71] Google Inc. Google App Engine. `http://code.google.com/appengine/`, September 2011.

[72] Google, Inc. Google Apps. `http://www.google.com/apps/`, September 2011.

[73] 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.

[74] W. Gropp, E. Lusk, and A. Skjellum. Using MPI: portable parallel programming with the message passing interface. 1999.

[75] 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`.

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

[77] 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.

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

[79] 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.

[80] T. Hey and A. Trefethen. The UK e-science core programme and the grid. *Future Generation Computer Systems*, 18(8):1017–1031, 2002.

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

[82] 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.

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

[84] 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.

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

[86] G. Katsaros, G. Gallizo, R. Kübert, T. Wang, J. O. 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)*. Accepted for publication.

[87] G. Katsaros, G. Kousiouris, S. Gogouvitis, D. Kyriazis, and T. Varvarigou. A service oriented monitoring framework for soft real-time applications. In *Service-Oriented Computing and Applications (SOCA), 2010 IEEE International Conference on*, pages 1–4. IEEE.

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

[89] K. Keahey, R. Figueiredo, J. Fortes, T. Freeman, and M. Tsugawa. Science clouds: Early experiences in cloud computing for scientific applications. *Cloud computing and applications*, 2008, 2008.

[90] 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.

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

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

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

[94] 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.

[95] 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.

[96] 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.

[97] J. Knobloch and L. Robertson. LHC computing Grid technical design report. `http://lcg.web.cern.ch/LCG/tdr/`, September 2011.

[98] D. Kranzlmuller. The future European Grid Infrastructure - Roadmap and challenges. In *Information Technology Interfaces, 2009. ITI'09. Proceedings of the ITI 2009 31st International Conference on*, pages 17–20. IEEE, 2009.

[99] K. Krauter, R. Buyya, and M. Maheswaran. A taxonomy and survey of Grid resource management systems for distributed computing. *Softw. Pract. Exper.*, 32(2):135–164, 2002.

[100] 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.

[101] 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.

[102] A. Lakshman and P. Malik. Cassandra-A Decentralized Structured Storage System. In *Workshop on Large Scale Distributed Systems and Middleware (LADIS)*, 2009.

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

[104] 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.

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

[106] 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.

[107] B. Leiner, V. Cerf, D. Clark, R. Kahn, L. Kleinrock, D. Lynch, J. Postel, L. Roberts, and S. Wolff. A brief history of the Internet. *Internet Society*, 10, 2003.

[108] LHC Project Webpage. `http://lhc.web.cern.ch/lhc/`, September 2011.

[109] libvirt development team. libvirt: The virtualization api. `http://libvirt.org/`, September 2011.

[110] 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).*

[111] M. Livny, J. Basney, R. Raman, and T. Tannenbaum. Mechanisms for high throughput computing. *SPEEDUP journal*, 11(1):36–40, 1997.

[112] 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.

[113] Manifesto, O.C. Open Cloud Manifesto. *Availabe online: www. opencloudmanifesto. org/*, 20, 2009.

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

[115] P. Mell and T. Grance. The NIST definition of cloud computing. *National Institute of Standards and Technology*, 53(6), 2009.

[116] C. Metz. Interconnecting ISP networks. *Internet Computing, IEEE*, 5(2):74–80, 2001.

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

[118] Microsoft Corporation. Virtual Hard Disk Image Format Specification, September 2011.

[119] Miniwatts Marketing Group. World Internet Usage Statistics News and World Population Stats. `http://www.internetworldstats.com/stats.htm`, September 2011.

[120] C. Morin. Open computing infrastructures for elastic services: contrail approach. In *Proceedings of the 5th international workshop on Virtualization technologies in distributed computing*, pages 1–2. ACM, 2011.

[121] 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.

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

[123] National Science Foundation. US National Science Foundation (NSF). `http://www.nsf.gov/`, September 2011.

[124] Nordic Data Grid Facility. `http://www.ndgf.org/`, September 2011.

[125] NorGrid. `http://www.norgrid.no/`, September 2011.

[126] OnLive, Inc. OnLive.com. `http://www.onlive.com`, September 2011.

[127] Open Grid Forum OCCI-WG. Open Cloud Computing Interface. `http://www.occi-wg.org/`, September 2011.

[128] OpenVZ project team. OpenVZ Wiki. `http://www.openvz.org`, September 2011.

[129] P.-O. Östberg, D. Henriksson, and E. Elmroth. Decentralized, scalable, Grid Fairshare Scheduling (FSGrid). 2011. Submitted.

[130] Parallels. Parallels Optimized Computing. `http://www.parallels.com/eu/`, September 2011.

[131] 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*, pages 1–25.

[132] 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.

[133] 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.

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

[135] Rackspace, US Inc. Rackspace Cloud. `http://www.rackspace.com/cloud/`, September 2011.

[136] M. Riedel, E. Laure, T. Soddemann, L. Field, J. Navarro, J. Casey, M. Litmaath, J. Baud, B. Koblitz, C. Catlett, et al. Interoperation of world-wide production e-science infrastructures. *Concurrency and Computation: Practice and Experience*, 21(8):961–990, 2009.

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

[138] 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*, 2011. Accepted.

[139] 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.

[140] 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.

[141] 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.

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

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

[144] 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.

[145] SAP. SAP Enterprise Resource Planning. http://www.sap.com/erp, visited April 2011, September 2011.

[146] 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.

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

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

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

[150] SNIC. SweGrid - The Swedish GRID Initiative. http://www.snic.vr.se/projects/swegrid, September 2011.

[151] 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.

[152] 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.

[153] 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.

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

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

[156] 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.

[157] K. Suzaki and D. Walsh. Implementing the Combination of Time Sharing and Space Sharing on AP/Linux. In *Job Scheduling Strategies for Parallel Processing*, pages 83–97. Springer, 1998.

[158] 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.

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

[160] M. Szeredi. Filesystem in userspace. `http://fuse.sourceforge.net/`, September 2011.

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

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

[163] J. Tordsson, R. Montero, R. Vozmediano, and I. Llorente. Cloud brokering mechanisms for optimized placement of virtual machines across multiple providers. 2010. Submitted for journal publication.

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

[165] Ubuntu Community. CloudInit - Community Ubuntu Documentation. `https://help.ubuntu.com/community/CloudInit`, September 2011.

[166] U.S. Department of Commerce. National Institute of Standards and Technology. `http://www.nist.gov`, September 2011.

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

[168] R. Van den Bossche, K. Vanmechelen, and J. Broeckhove. Cost-Optimal Scheduling in Hybrid IaaS Clouds for Deadline Constrained Workloads. In *2010 IEEE 3rd International Conference on Cloud Computing*, pages 228–235. Ieee, 2010.

[169] 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.

[170] S. Venugopal, R. Buyya, and L. Winton. A Grid service broker for scheduling e-science applications on global data Grids. *Concurrency Computat.: Pract. Exper.*, 18(6):685–699, May 2006.

[171] VMWARE. VMware VMotion: Live migration of virtual machines without service interruption datasheet. `http://www.vmware.com/files/pdf/VMware-VMotion-DS-EN.pdf`, September 2011.

[172] VMware, Inc. VMware vCloud Express. `http://www.vmware.com/solutions/cloud-computing/public-cloud/vcloud-express.html`, September 2011.

[173] 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.

[174] 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.

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

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

[177] 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.

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

[179] T. White. *Hadoop: The Definitive Guide.* Yahoo Press, 2010.

[180] 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.

[181] 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.

[182] 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.

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