Grid vs Cloud – A Technology Comparison

Grid vs Cloud – Ein Technologischer Vergleich

Ivona Brandic, Schahram Dustdar, Vienna University of Technology

Summary Cloud Computing represents a novel and promising approach for implementing scalable ICT systems for individual-, communities-, and business-use relying on the latest achievements of diverse research areas, such as Grid computing, Service oriented computing, business processes, and virtualization. From the technological point of view Grid computing is considered as the most related predecessor technology of Cloud computing. Although, Cloud and Grid computing differ in many aspects, as for example, in the general idea of the provision of computational resource, which is in Clouds commercial based and in Grids community based, there are many similarities. In this paper we investigate the similarities and differences between Clouds and Grids by evaluating two successful projects, namely for the provision of native hight performance copmputing applications as Grid workflows and for the self-management of Cloud infrastructures. ►►► **Zusammenfassung** Cloud Computing ist eine neuartige Methode für die Implementierung skalierbarer ICT Systeme, die von den Individuen, Gemeinschaften und Geschäftsprozessen benutzt wird. Cloud Computing basiert auf diversen Technologien wie z. B. Grid Computing, Service-orientierte Architekturen, Geschäftsprozess-Management und Virtualisierung. Von dem technologischen Standpunkt wird Grid Computing als die wichtigste Vorläufertechnologie angesehen. Obwohl Grid und Cloud Infrastrukturen sich in vielen Aspekten unterscheiden, wie z. B. in der Idee wie die Computerressourcen angeboten werden – in Grid basiert dies auf der gemeinsamen Nutzung der Ressourcen in einer Gemeinschaft, in Clouds ist es kommerziell – haben beide Systeme auch sehr viele Gemeinsamkeiten. In dieser Arbeit untersuchen wir die Ähnlichkeiten und Unterschiede zwischen Grid und Cloud Systemen durch die Evaluierung von zwei erfolgreichen Projekten aus beiden Bereichen, nämlich eine Infrastruktur für die Bereitstellung der Hochleistungsapplikationen als Services für Grid und eine Cloud Infrastruktur für das Selbstmanagement der Cloud Applikationen.

Keywords D.2.11 [Software: Software Engineering: Software Architectures]; cloud computing, grid computing ►►►
Schlagwörter Software, Software Engineering, Software Architectures, cloud computing, grid computing

1 Introduction

Cloud computing represents a novel and promising approach for implementing scalable ICT systems for individual-, communities-, and business-use. Resources are pooled and offered on-demand with ubiquitous network access to rapidly configurable and elastic IT capabilities. Resources are delivered following three basic delivery models: access to applications (SaaS), provision of platforms to create applications (PaaS), and provision of infrastructures for processing, storage, and networking (IaaS). The key benefits of providing computing power using Clouds are:

1. avoidance of expensive computer systems configured to cope with peak performance;
2. pay-per-use solutions for computing cycles requested on-demand, and
3. avoidance of idle computing resources, resulting in novel business models.

Cloud computing can be defined as the convergence and evolution of several concepts from virtualization,

distributed application design, and enterprise IT management to enable a more flexible approach for deploying and scaling applications [1; 11; 12]. From the technological point of view Grid computing is considered as the most related predecessor technology of Cloud computing.

Grid computing provides concepts and tools for the provision of High Performance Computing (HPC) resources and applications as services that may be accessed transparently and on-demand. Major application area of Grid computing is interconnection and transparent use of computational resources for scientific and large scale applications. The main goal of *Grid computing* is to provide on demand access to HPC infrastructures by augmenting standardized protocols and services. Based on such an infrastructure pervasive access to geographically dispersed hardware, software, and information resources should be enabled. However, since HPC resources provided as Grid services are usually not under the control of end users non functional requirements of Grid applications turn out as the most challenging issues in utilization of HPC resources as services.

Although Cloud and Grid computing differ in many aspects as, for example, in the general idea of the provision of computational resource, which is in Clouds commercial based and in Grids community based, there are many similarities. One of the most important issues in both approaches is the management of non functional properties of the system.

Service provisioning in the Cloud as well as in Grid systems is based on Service Level Agreements (SLA) representing a contract signed between the customer and the service provider including the non-functional requirements of the service specified as Quality of Service (QoS). SLA considers obligations, service pricing, and penalties in case of agreement violations. Flexible and reliable management of SLA agreements is of paramount importance for both, Cloud and Grid users. On one hand, preventions of SLA violations ahead of time can avoid unnecessary penalties, provider has to pay in case of violations. Sometimes, simple actions like migrating VMs to available cores can prevent SLA violations. On the other hand, based on flexible and timely reactions to possible SLA violations, interactions with the users can be minimized increasing the chance for Cloud computing to take roots as a flexible and reliable form of on demand computing.

In this paper we evaluate the similarities and differences between the Grid and Cloud systems focusing on one and probably the most challenging aspect of both systems, namely the management of non functional user requirements. We reflect the past and current developments, similarities and differences, open research issues and lessons learned for the management of non functional requirements in Clouds and Grids based on two successful Grid and Cloud projects.

2 Background and Comparison Criteria

In this section we discuss the major similarities and differences between Grids and Clouds and present a criteria catalog for the detailed evaluation of the QoS and SLA management techniques for both technologies.

2.1 Grid Characteristics

With the advance of the Internet, Grid Computing emerged as a new state of the art technology for *resource sharing*. Especially in the area of HPC where high end devices, large scale computing resources, and networks are used for computational science, Grids represent an efficient solution for sharing of geographically distributed resources. Thereby, resource sharing is based on *preinstalled services* [8] following the concepts of stateful web services [13] as defined by Web Service Resource Framework (WSRF). WSRF provides a set of operations that web services may implement to become stateful. Thus, additional to the simple URI, which is sufficient to identify stateless service, stateful services consider a resource information that can be used by clients to communicate with services, store and retrieve data, start, stop and resume computations.

The collaborative nature Grids led to the emergence of multiple organizations that function as one unit through the use of their shared competencies and resources for the purpose of one or more identified goals. Thus, administration of resources in Grids is solved with the concept of *virtual organization* representing a dynamic set of individuals and/or institutions aligned with a set of resource-sharing rules and conditions to solve a specific (research) goal. Thus, organizations and individuals belonging to a specific virtual organization may share resources for a specific time frame to achieve certain research goal. Grid technologies have been successfully used for the establishment of computational testbeds in eScience or for various *large scale and parallel computations* as for example in the EGEE [15] or myGrid [16] projects. myGrid project produced a set of tools designed to help e-Scientists modeling and execution of in silico experiments. The tools support creation of e-laboratories and have been used in domains such as systems biology, social science, music, astronomy, multimedia and chemistry. The Enabling Grids for E-science (EGEE) project was funded by the European Commission developing a Grid infrastructure and service Grid infrastructure which was available to scientists 24 hours-a-day for the execution of their experiments.

Although Grid computing seems to be obsolete today, several technologies and concepts relevant for the today's Cloud infrastructures appeared as an outcome of different Grid projects. Major outcomes and achievements from the Grid era are sophisticated models for the management of batch systems, which eventually led to the development of the virtualization middleware like Eucalyptus [11]. One such example is Oracle Grid Engine [14] representing a distributed resource management (DRM) system that

manages the distribution of users' workloads to available compute resources.

Since the idea of Grid computing was sharing of geographically distributed resource following the rules of virtual organization, guarantees to rely on promised resources availability became indispensable. Thus, QoS giving non trivial qualities on service execution became a crucial part of the Grid computing infrastructures. QoS is negotiated before the service usage and is expressed by means of SLAs representing popular formats for the specification of QoS. Similar to the virtualization technologies, SLAs became one of the major enabling technologies for the development of Cloud Computing infrastructures.

However, missing market mechanisms for open and dynamic Grids and lack of appropriate tools and technological solutions for the management of SLAs are some of the most important reasons why Grid computing did not succeed as a new state of the art technology for on demand computing [4].

2.2 Characteristics of the Clouds

Cloud computing facilitates on-demand and scalable resource provisioning as services in a pay-as-you-go manner thereby making resources available at all times from every location. From the technological point of view Cloud computing does not represent a new technology, rather it is considered to be a novel form how existing technologies are used to achieve efficient resource pooling and resource management. Thus, Cloud computing relies on existing technologies like Grid computing, service oriented computing, virtualization, Web 2.0, and similar by augmenting them and combining them to achieve Cloud related goals like elasticity and energy efficiency.

On-demand resource provision requires *massive scalability* achieved through pooling of resources to data centers and by applying various Cloud engineering approaches for insourcing or outsourcing of resources to handle peaks and slopes (e. g., Cloud storming and Cloud bursting or Cloud federation [12]). *Virtualization* technology separating computation and technology from the hardware layer was the key concept which facilitated on demand provision of computational resources for arbitrary users. In Grids the relationships are established offline, usually between academic institutes, and SLAs are used to guarantee certain performance metrics. However, Clouds fully rely on the online relationships, which are established on-demand and on a case by case basis for the certain business case. This also led to new research challenges for the development of the QoS and *Service Level Agreements*, which are beyond just guaranteeing some performance metrics. It considers trust establishment, compliance management and appropriate *market mechanisms* where potential Cloud users and providers can meet each other.

While Cloud computing represents a promising technology for the operation of next generation ICT infrastructures on the one hand, it exhibits a high rate of energy waste due to the characteristics of Clouds, like virtualization overhead and massive scalability [17]. Thus, with the advance of Cloud technologies novel *energy efficient* resource provisioning models appeared considering data center consolidation, intelligent workload prediction, knowledge and speculative SLA management [7] revealing novel challenges for the implementation of Cloud infrastructures.

2.3 Major Differences between Grids and Clouds

In the following we summarize the major differences between Grids and Clouds.

Business Models. While in Grid business models are usually based on bilateral agreements between academic institutions, provision of resource in Clouds requires more differentiated business models as discussed next. Currently, we observe several types of business models ranging from resource providers who only provide computing resources (e. g., Amazon, Tsunamic Technologies), over SaaS providers who sell their own resources together with their own software services (e. g., GoogleApps, Salesforce.com) to companies that attempt to run a mixed approach, i. e., they allow users to create their own services but at the same time offer their own services (Sun N1 Grid, Microsoft Azure).

Resource Management. Resource management represents another major difference between Grids and Clouds. While Grids rely on batch systems, utilization of virtualization technologies represents the resource management solution for the Clouds.

Resource Provision Models. As already discussed in previous sections Grid resource provisioning models are based on virtual organisations where the relationships are established offline. In Clouds usage of SLAs, compliance, and trust management is essential.

Resource Availability. In Grids resource sharing relies on the best effort manner, sometimes resources are not available and sometimes there are plenty of resources which are idle. Clouds rely on massive elasticity in Clouds. Challenging issues in Clouds are to find the balance between wasting resources due to the virtualization overhead and standby modes of devices on the one hand, and pooling of resources to facilitate efficient consumption of resources and reducing energy consumption on the other.

3 Sample Projects

Our comparison is based on two reference projects: *Amadeus* for Grid QoS management and *FoSII* for management of self-adaptable Cloud infrastructures.

GRID: Amadeus. Figure 1 shows the architecture of the Amadeus environment used to manage execution of Grid workflows. Amadeus has been successfully utilized for the

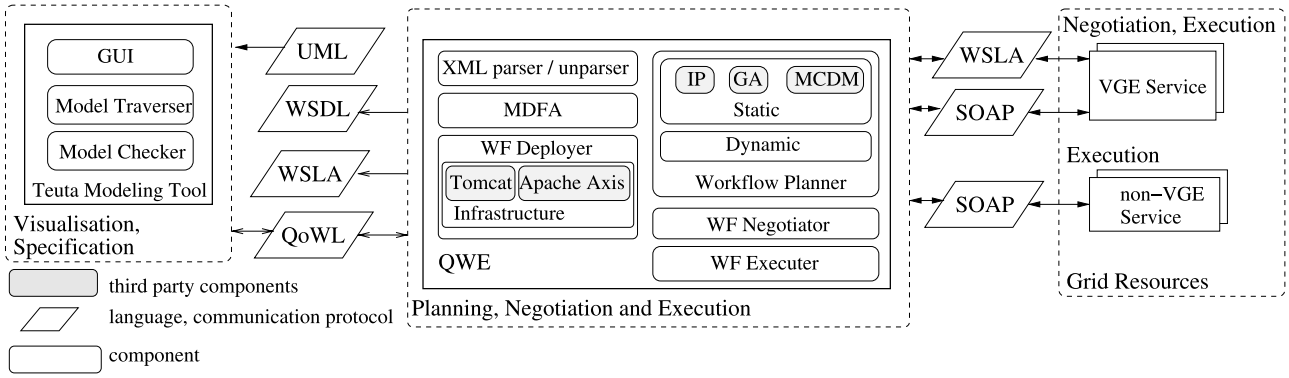

Figure 1 Amadeus architecture.

provision and management of HPC applications as for example maxillo facial surgery application used in medical practice for preparation of medical treatments [5]. The main components include

1. a *Visualization and Specification component*,
2. a *Planning, Negotiation and Execution component* called QoS-aware Grid Workflow Engine (QWE), and
3. a set of *Grid Resources*.

The specification and visualization component comprises a tool for UML-based Grid workflow modeling and visualization. A user may specify the workflow by composing predefined workflow elements. For each workflow element different properties (such as execution time, price and location affinity) may be specified that indicate the user's QoS requirements. After the validation of the specified workflow, a corresponding XML representation is generated following the syntax of *QoWL* [3]. The QWE engine interprets the QoWL workflow, applies the selected optimization strategy, negotiates with services, selects appropriate services and finally executes the workflow. The *requested QoS* and the *negotiated QoS* may be expressed using a language for the specification of electronic contracts, as for example Web Service Level Agreement (WSLA). For the activities annotated with QoS constraints, we use QoS aware services (i. e., Vienna Grid Environment (VGE) services), which are able to provide certain QoS guarantees. For other activities non-VGE services may be used.

CLOUD: FoSII. The Foundations of Self-governing ICT Infrastructures (FoSII) research project is proposing solutions for autonomic management of SLAs in the Cloud. In FoSII we are developing models and concepts for achieving adaptive service provisioning and SLA management via resource monitoring and knowledge management techniques. Figure 2 depicts the components of the FoSII infrastructure. Each FoSII service implements three interfaces:

1. the negotiation interface necessary for the establishment of SLA agreements,
2. the service management interface necessary for starting service, uploading data, and similar management actions, and

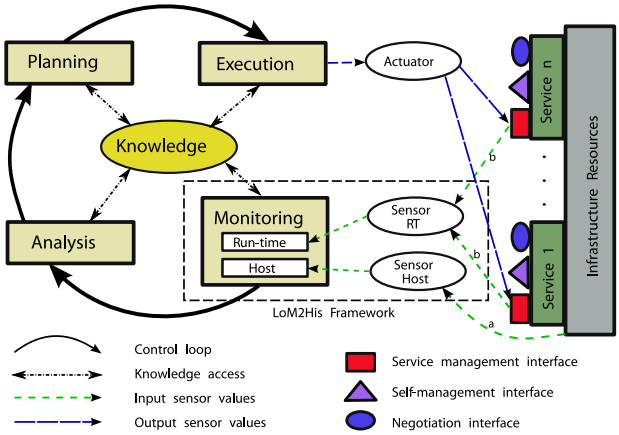

Figure 2 FoSII architecture.

3. the self-management interface necessary to devise actions in order to prevent SLA violations.

FoSII is implemented using the principles of *autonomic computing*. Autonomic computing research methodology can be exemplified using QoS, where the management is done through the following steps:

1. *Monitoring*: The QoS managed element is monitored using adequate software sensors;
2. *Analysis*: The monitored and measured metrics (e. g., execution time, reliability, availability, etc.) are analyzed using the knowledge base (condition definition, condition evaluation, etc.);
3. *Planning*: Based on the evaluated rules and the results of the analysis, the planning component delivers necessary changes on the current setup, e. g., renegotiation of services which do not satisfy the established QoS guarantees.
4. *Execution*: Finally, the planned changes are executed using software actuators and other tools (e. g., *VieSLAF* framework [6]), which query for new services.

4 Grid vs Cloud – A Comparison of QoS Management Aspects

In order to evaluate QoS management strategies in Grids and Clouds we analyse different aspects as exemplified in Table 1. We examine service management, QoS manage-

Table 1 QoS aspects in Grids and Clouds.

Aspect	Grid QoS	Cloud QoS
Service management	Pre-installed services [2]	Installation on demand [9]
QoS management	Application based [5]	Application agnostic [6]
SLA attainment strategies	Request/response models [5]	Self-adaptable models [7]

ment, and SLA attainment strategies for sample Grid and Cloud systems, namely Amadeus and FoSII.

4.1 Service Management

Grid: Vienna Grid Environment – VGE. As shown in Fig. 1 Amadeus uses the Vienna Grid Environment (VGE) service for the provision of native HPC applications. Services have to be preinstalled while QoS is negotiated on demand as discussed next. VGE services may be configured in order to offer QoS guarantees with respect to response time, price and location affinity. VGE services support a dynamic QoS negotiation model where clients may negotiate various QoS guarantees with multiple service providers [2] relying on a generic QoS module, which usually comprises an *application-specific performance model*, a *pricing model*, a *compute resource manager* that supports advance reservation, and a *QoS manager*. An application-specific performance model usually takes as input a *request descriptor*, containing input meta data, and a *machine descriptor* which specifies the amount of machine resources (e. g., number of processors, main memory, etc.) that may be provided for an application. Input *meta data* indicates the information that describes the input data of a service. Commonly, *meta data* is used for the performance evaluation of a service during the generation of QoS offers. *Payload data* denotes input and output data of a service operation.

Cloud: Automatic Service Deployer (ASD). In FoSII we use the Automatic Service Deployer (ASD) where services are installed on demand using Virtual Appliances (VA), which store service images [9]. As shown in Fig. 3 to interface with a broker ASD considers a repository (e. g., Application Content Service (ACS), standard proposed by the OGF). All the master copies of all the deployable services, i. e., VAs, are stored in the repository. The VAs are either defined by an external entity or the ASD is capable of acquiring it from a running system. The repository allows the broker to determine which services are available for deployment and which are the static ones. If the deployed services are not available, it checks whether any of the latter resources can deliver the service taking into account the deployment cost. The Workspace Service (WS) offers the virtualization capabilities – virtual machine creation, removal and management – of a given site as a WSRF service. A typical service broker has two main connections with the outside world: the Candidate Set Generators (CSG), and the Information Services. The task of the CSG is to offer a list of sites, which can perform the requested service according to the SLA and other requirements. In most of the cases the candidate set generator is an integral part of the broker thus instead of the candidate set adjustments, the broker queries the candidate site list as it would do without ASD. As a result the service call is executed as a composed service instead of a regular call. The composition contains the deployment task as its starting point and the actual service call as its dependent task. Since both, the CSG and the brokers heavily rely on the information system, the ASD can influence their decision through publishing dynamic data. This data could state service presence on sites where the service is not even deployed.

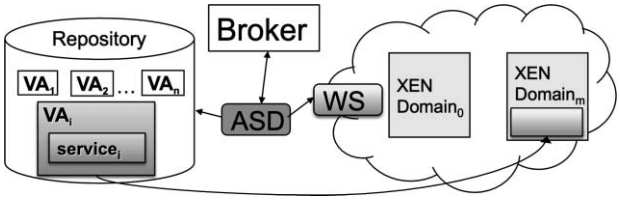

Figure 3 ASD architecture.

4.2 QoS Management

Grid: Application based QoS Management. A distinguishing feature of *Amadeus* is the QoS support during all stages of the workflow lifecycle. At specification time *Amadeus* provides an adequate tool-support for high-level graphical specification of QoS-aware workflows, which allows the association of comprehensive set of QoS constraints to any activity or to the whole workflow. During the planning phase *Amadeus* provides a set of QoS-aware service-oriented components that support automatic constraint-based service negotiation and workflow optimization. During the execution phase, using the information from the planning phase, *Amadeus* attempts to execute the workflow activities in the manner that the specified requirements in terms of QoS constraints are met. QoS-aware services, which are able to give QoS guarantees, serve as resources that are invoked by the QoS-aware workflow. A QoS-aware service can provide QoS guarantees, and enables clients to negotiate about its QoS properties. This kind of support is provided by the VGE services [2]. VGE provides application level QoS support, for example with respect to execution time or price. VGE has been successfully used for the development of a Grid testbed for medical simulation services [3] in the context of the European Commission funded GEMSS project.

Cloud: Application Agnostic QoS Management. The self-management interface as shown in Fig. 2 specifies operations for sensing changes of the desired state and for reacting to those changes. The host monitor sensors continuously monitor the infrastructure resource metrics (input sensor values arrow a in Fig. 2) and provide the knowledge component with the current resource status. The run-time monitor sensors sense future SLA violation threats (input sensor values arrow b in Fig. 2) based on resource usage experiences and predefined thresholds. As shown in Fig. 2, the Low-level Metric to High-level SLA (LoM2HiS) framework is responsible for monitoring and sensing future SLA violation threats. It comprises the host monitor and the run-time monitor. The host monitor monitors low-level resource metrics such as CPU, memory, disk space, incoming bytes, and similar. It extracts the monitored output from the agents, processes them and sends the metric-value pairs through our implemented communication model to the run-time component.

The run-time component receives the metric-value pairs and based on predefined mapping rules maps them into equivalent high-level SLA parameters. An example of an SLA parameter is service availability Av, which is calculated using the resource metrics downtime and uptime as follows:

$$Av = (1 - \text{downtime}/\text{uptime}) * 100\,.$$

During the analysis and planning phases the knowledge component then suggests appropriate actions to solve SLA violation threats. As a conflicting goal, it also tries to reduce energy consumption by removing resources from over-provisioned services. Reactive actions thus include increasing or decreasing memory, storage or CPU usage for each service. After the action has been executed the knowledge component learns the utility of the action in this specific situation via Case Based Reasoning (CBR). CBR contains previously solved cases together with their actions and utilities, and tries to find the most similar case with the highest utility for each new case. Furthermore, it examines the timing and the effectiveness of an action, i. e., whether the action would have helped but was triggered too late, or was unnecessarily triggered too early, and consequently, it updates the threat thresholds from the monitoring component.

4.3 SLA Attainment Strategies

Grid: Request/Response Models. The basic QoS negotiation in VGE is based on a request/offer model where the client requests offers from a set of pre-selected services (e. g., through the VGE registry service). If the client confirms a QoS offer, a QoS contract is established and signed by both parties. Initially the client has to supply a request descriptor containing concrete values for all performance-relevant parameters specified in the application descriptor as well as a QoS request with requested QoS properties (i. e., begin and end time as well as price) to a service. On the service side the QoS management module utilizes heuristics that consider the outcome of the application performance model (i. e., performance descriptor), the availability of resources via the resource manager and the service provider's pricing model for creating appropriate offers to the client's demands. If an appropriate QoS offer can be made and a temporary resource reservation has been made, the QoS offer is returned to the client. Since clients usually negotiate with multiple services, each QoS offer has a short expiration time and eventually it is up to the client to confirm a specific offer before it expires or restart the negotiation with different parameters (i. e., with a new QoS descriptor). If a QoS offer is confirmed by the client, a QoS contract is established and signed by both parties. However, the renegotiation is done on behalf of the users and there are no mechanism for the automatic renegotiation in case of failures.

Cloud: Self-adaptable Models. In FoSII we developed a knowledge management tool to sense possible SLA violations before they happen. For the decision making we use knowledge data bases proposing the reactive actions by utilizing CBR.

CBR is the process of solving problems based on past experience. It tries to solve a *case* (a formatted instance of a problem) by looking for similar cases from the past and reusing the solutions of those cases to solve the current one. In order to define similarity we use similarity measures as described in [10]. In general, a typical CBR cycle consists of the following phases assuming that a new case was just received: (i) retrieve the most similar case or cases to the new one, (ii) reuse the information and knowledge in the similar case(s) to solve the problem, (iii) revise the proposed solution, (iv) retain the parts of this experience likely to be useful for future problem solving.

```
1. (
2.  (App, 1),
3.   (
4.   ((Incoming Bandwidth, 12.0),
5.    (Outgoing Bandwidth, 20.0),
6.    (Storage, 1200),
7.    (Availability, 99.5),
8.    (Running on PMs, 1)),
9.    (Physical Machines, 20)
10.  ),
11. "Increase Incoming Bandwidth share by 5%",
12.  (
13.   ((Incoming Bandwidth, 12.6),
14.    (Outgoing Bandwidth, 20.1),
15.    (Storage, 1198),
16.    (Availability, 99.5),
17.    (Running on PMs, 1)),
18.    (Physical Machines, 20)
19   ),
20.  0.002
21. )
```

As shown in the code snippet a complete case consists of

1. the id of the program being concerned (line 2),
2. the initial case (measurements by the monitoring component and mapped to the SLAs) consisting of the SLA

parameter values of the program and global Cloud information like number of running virtual machines (lines 4–10),
3. the executed action (line 11),
4. the resulting case (measured some time interval later) (lines 12–18) as in (b), and
5. the resulting utility (line 20).

As shown in [7], FoSII's knowledge management tool has been successfully utilized not only to manage and prevent SLA violations but also to manage energy efficiency in Clouds.

5 Conclusion

In this paper we compared Grid and Cloud systems considering QoS aspects like service provisioning and SLA attainment strategies. In many aspects Cloud systems, like the FoSII infrastructure, represent continuance in development of infrastructure for the provision of computational resources as utilities. Although, there are some differences in usage mode of Grid and Clouds, in both approaches there is significant work done towards implementation of QoS management strategies. While in Grid QoS is usually application based and relies on user driven negotiations, Cloud facilitates application agnostic QoS and matured self-management features.

References

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

[2] S. Benkner, I. Brandic, G. Engelbrecht, and R. Schmidt. *VGE – A Service-Oriented Grid Environment for On-Demand Supercomputing.* In: Proc. of the Fifth IEEE/ACM Int'l Workshop on Grid Computing (Grid 2004), Pittsburgh, PA, USA, Nov 2004.

[3] I. Brandic, S. Benkner, G. Engelbrecht, and R. Schmidt. *QoS Support for Time-Critical Grid Workflow Applications.* In: Proc. of the 1st IEEE Int'l Conf. on eScience and Grid Computing, Melbourne, Australia, Dec 2005.

[4] I. Brandic. Mapping the SLA Landscape for High Performance Clouds. HPC in the Cloud, 7. Feb. 2011. http://www.hpcinthecloud.com/hpccloud/2011-02-07/mapping_the_sla_landscape_for_high_performance_clouds.html.

[5] I. Brandic, S. Pllana, and S. Benkner. Specification, Planning, and Execution of QoS-aware Grid Workflows within the Amadeus Environment. In: Concurrency and Computation: Practice and Experience 20(4):331–345, Mar 2008.

[6] I. Brandic, D. Music, and S. Dustdar. VieSLAF Framework: Facilitating Negotiations in Clouds by Applying Service Mediation and Negotiation Bootstrapping. In: Scalable Computing: Practice and Experiences (SCPE), Special Issue of Scalable Computing on Grid Applications and Middleware & Large Scale Computations in Grids 11(2):189–204, June 2010.

[7] V. Emeakaroha, M. Maurer, I. Brandic, and S. Dustdar. FoSII – Foundations of Self-Governing ICT Infrastructures. In: ERCIM News 83, Special Theme: Cloud Computing Platforms, Software, and Applications, Oct 2010.

[8] 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, Nov 2008.

[9] A. Kertész, G. Kecskeméti, and I. Brandic. Autonomic SLA-aware Service Virtualization for Distributed Systems. In: Proc. of the 19th Euromicro Int'l Conf. on Parallel, Distributed and Network-Based Computing, Ayia Napa, Cyprus, Feb 2011.

[10] M. Maurer, I. Brandic, and R. Sakellariou. Enacting SLAs in Clouds Using Rules. In: Proc. of Euro-Par 2011, Bordeaux, France, Aug–Sep 2011.

[11] D. Nurmi, R. Wolski, Ch. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, and D. Zagorodnov. The Eucalyptus Open-source Cloud-Computing System. In: Proc. of Cloud Computing and Its Applications 2008, Chicago, Illinois, Oct 2008.

[12] B. Rochwerger, D. Breitgand, E. Levy, A. Galis, K. Nagin, I. M. Llorente, R. Montero, Y. Wolfsthal, E. Elmroth, J. Caceres, M. Ben-Yehuda, W. Emmerich, and F. Galan. The RESERVOIR Model and Architecture for Open Federated Cloud Computing. In: IBM System Journal Special Edition on Internet Scale Data Centers 53(4):535–545, 2009.

[13] The WS-Resource Framework, http://www.globus.org/wsrf/.

[14] Oracle Grid Engine, http://www.oracle.com/technetwork/oem/grid-engine-166852.html, 2011.

[15] Enabling Grids for eScience, http://www.eu-egee.org, 2011.

[16] MyGrid, http://www.mygrid.org.uk, 2011.

[17] A. Orgerie and L. Lefevre. When Clouds become Green: the Green Open Cloud Architecture. In: Proc. of the Int'l Conf. on Parallel Computing (Parco2009), pages 228–237, Lyon, France, 2009.

Received: November 22, 2010

Dr. Ivona Brandic is Assistant Professor at the Distributed Systems Group, Information Systems Institute, Vienna University of Technology (TU Wien). Prior to that, she was Assistant Professor at the Department of Scientific Computing, Vienna University. She received her PhD degree from Vienna University of Technology in 2007. She is leading the Austrian national FoSII (Foundations of Self-governing ICT Infrastructures) project funded by the WWTF.

Address: Vienna University of Technology, Information Systems Institute, Argentinierstrasse 8, 1040 Vienna, Austria, Tel.: + 43 1 58801 58417, Fax: + 43 1 58801 18491,
e-mail: ivona@infosys.tuwien.ac.at

Prof. Dr. Schahram Dustdar is Full Professor of Computer Science (Informatics) with a focus on Internet Technologies heading the Distributed Systems Group, Institute of Information Systems, Vienna University of Technology (TU Wien) where he is director of the Vita Lab. In April 2003 he received his Habilitation degree (Venia Docendi) for his work on Process-aware Collaboration Systems – Architectures and Coordination Models for Virtual Teams. More information can be found at: http://www.infosys.tuwien.ac.at/Staff/sd

Address: Vienna University of Technology, Information Systems Institute, Argentinierstrasse 8, 1040 Vienna, Austria, Tel.: +43-1-58801-18414, Fax: + 43 1 58801 18491,
e-mail: sd@infosys.tuwien.ac.at