Chapter 1

State-of-the-Art Technologies for Large-Scale Computing

Florian Feldhaus and Stefan Freitag
Dortmund University of Technology, Dortmund, Germany

Chaker El Amrani
Université Abdelmalek Essaâdi, Tanger, Morocco

1.1 INTRODUCTION

Within the past few years, the number and complexity of computer-aided simulations in science and engineering have seen a considerable increase. This increase is not limited to academia as companies and businesses are adding modeling and simulation to their repertoire of tools and techniques. Computer-based simulations often require considerable computing and storage resources. Initial approaches to address the growing demand for computing power were realized with supercomputers in 60 seconds. Around 1964, the CDC6600 (a mainframe computer from Control Data Corporation) became available and offered a peak performance of approximately 3×10^6 floating point operations per second (flops) (Thornton, 1965). In 2008, the IBM Roadrunner[1] system, which offers a peak performance of more than 10^{15} flops, was commissioned into service. This system was leading the TOP500 list of supercomputers[2] until November 2009.

[1] www.lanl.gov/roadrunner.
[2] www.top500.org.

Large-Scale Computing, First Edition. Edited by Werner Dubitzky, Krzysztof Kurowski, Bernhard Schott.

Supercomputers are still utilized to execute complex simulations in a reasonable amount of time, but can no longer satisfy the fast-growing demand for computational resources in many areas. One reason why the number of available supercomputers does not scale proportional to the demand is the high cost of acquisition (e.g., $133 million for Roadrunner) and maintenance.

As conventional computing hardware is becoming more powerful (processing power and storage capacity) and affordable, researchers and institutions that cannot afford supercomputers are increasingly harnessing *computer clusters* to address their computing needs. Even when a supercomputer is available, the operation of a local cluster is still attractive, as many workloads may be redirected to the local cluster and only jobs with special requirements that outstrip the local resources are scheduled to be executed on the supercomputer.

In addition to current demand, the acquisition of a cluster computer for processing or storage needs to factor in potential increases in future demands over the computer's lifetime. As a result, a cluster typically operates below its maximum capacity for most of the time. E-shops (e.g., Amazon) are normally based on a computing infrastructure that is designed to cope with peak workloads that are rarely reached (e.g., at Christmas time).

Resource providers in academia and commerce have started to offer access to their underutilized resources in an attempt to make better use of spare capacity. To enable this provision of free capacity to third parties, both kinds of provider require technologies to allow remote users restricted access to their local resources. Commonly employed technologies used to address this task are *grid computing* and *cloud computing*. The concept of grid computing originated from academic research in the 1990s (Foster et al., 2001). In a grid, multiple resources from different administrative domains are pooled in a shared infrastructure or computing environment. Cloud computing emerged from commercial providers and is focused on providing easy access to resources owned by a single provider (Vaquero et al., 2009).

Section 1.2 provides an overview of grid computing and the architecture of grid middleware currently in use. After discussing the advantages and drawbacks of grid computing, the concept of virtualization is briefly introduced. Virtualization is a key concept behind cloud computing, which is described in detail in Section 1.4. Section 1.5 discusses the future and emerging synthesis of grid and cloud computing before Section 1.7 summarizes this chapter and provides some concluding remarks.

1.2 GRID COMPUTING

Foster (2002) proposes three characteristics of a grid:

1. Delivery of nontrivial qualities of service
2. Usage of standard, open, general-purpose protocols and interfaces
3. Coordination of resources that are not subject to centralized control

Endeavors to implement solutions addressing the concept of grid computing ended up in the development of grid middleware. This development was and still is driven by communities with very high demands for computing power and storage capacity. In the following, the main grid middleware concepts are introduced and their implementation is illustrated on the basis of the gLite[3] middleware, which is used by many high-energy physics research institutes (e.g., CERN). Other popular grid middleware include Advanced Resource Connector (ARC[4]), Globus Toolkit,[5] National Research Grid Initiative (NAREGI[6]), and Platform LSF MultiCluster.[7]

Virtual Organizations. A central concept of many grid infrastructures is a *virtual organization*. The notion of a virtual organization was first mentioned by Mowshowitz (1997) and was elaborated by Foster et al. (2001) as "a set of the individuals and/or institutions defined by resource sharing rules."

Virtual organization is used to overcome the temporal and spatial limits of conventional organizations. The resources shared by a virtual organization are allowed to change dynamically: Each participating resource/institution is free to enter or leave the virtual organization at any point in time. One or more resource providers can build a grid infrastructure by using grid middleware to offer computing and storage resources to multiple virtual organizations. Resources at the same location (e.g., at an institute or computing center) are forming a (local) grid site. Each grid site offers its resources through grid middleware services to the grid. For the management and monitoring of the grid sites as well as the virtual organizations, central services are required. The main types of service of a grid middleware may be categorized into (Fig. 1.1) (Foster, 2005; Burke et al., 2009) the following:

- Execution management
- Data management
- Information services
- Security

Execution Management. The execution management services deal with monitoring and controlling compute tasks. Users submit their compute tasks together with a description of the task requirements to a central workload management system (WMS). The WMS schedules the tasks according to their requirements to free resources discovered by the information system. As there may be thousands of concurrent tasks to be scheduled by the WMS, sophisticated scheduling mechanisms are needed. The simulation and analysis of the

[3] http://glite.cern.ch/.
[4] www.nordugrid.org/middleware.
[5] www.globus.org/toolkit.
[6] www.naregi.org/link/index_e.html.
[7] www.platform.com.

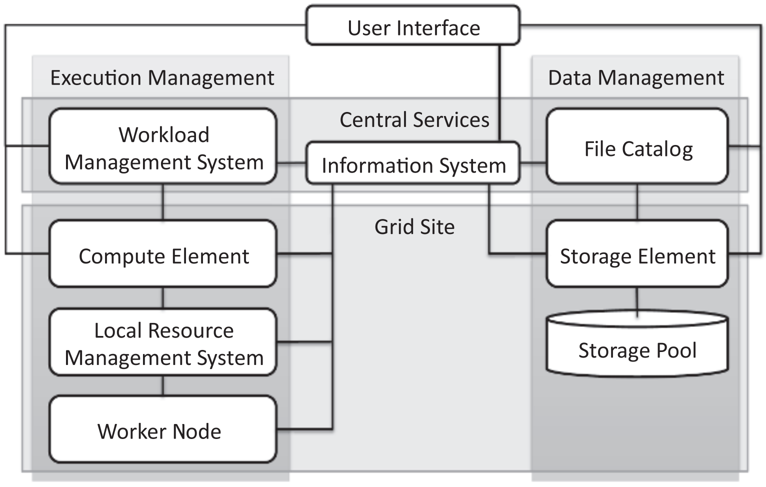

Figure 1.1 *Overview of grid middleware components.*

corresponding scheduling algorithms has become an important research area in its own right (Section 1.6).

Each grid site needs to run a compute element that is responsible for user authentication at the grid site and to act as an interface between local resources and the grid. The compute element receives compute tasks from the WMS and submits them to a local resource management system,[8] which then schedules the tasks to be executed on a free worker node.

With the LCG[9] and the CREAM[10] compute elements (Aiftimiei et al., 2010), the gLite middleware currently offers two choices for this task. Whereas the LCG is the standard compute element of gLite, the CREAM compute element was developed to be more lightweight and also allows direct job submission if mechanisms other than the central gLite WMS should be used for job scheduling and resource matching.

Data Management. Besides offering potentially powerful computing resources, a grid may also provide storage capacity in the form of storage elements.

With the Disk Pool Manager (DPM) (Abadie et al., 2007), the CERN Advanced STORage manager (CASTOR)[11] and dCache,[12] gLite currently supports three storage services. All of these are able to manage petabytes of storage on disk and/or tape. Retrieving and storing data are possible via

[8] Examples for resource management systems include TORQUE or the SUN Grid Engine.
[9] Large Hadron Collider Computing Grid.
[10] Computing Resource Execution and Management.
[11] http://castor.web.cern.ch/.
[12] www.dcache.org.

various protocols, for example, dcap, xrootd,[13] gridFTP, and SRM (Badino et al., 2009). The LCG storage element defines the minimum set of protocols that have to be supported to access the storage services.

For gLite, the central LCG File Catalog enables virtual organizations to create a uniform name space for data and to hide the physical data location. This is achieved by using logical file names that are linked to one or more physical file names (consisting of the full qualified name of the storage element and the absolute data path for the file on this specific storage element). Replicas of the data can be created by copying the physical files to multiple storage elements and by registering them under one unique logical file name in the LCG File Catalog. Thus, the risk of data loss can be reduced.

Information System. The information system discovers and monitors resources in the grid. Often the information system is organized hierarchically. The information about local resources is gathered by a service at each grid site and then sent to a central service. The central service keeps track of the status of all grid services and offers an interface to perform queries. The WMS can query the information to match resources to compute tasks.

For gLite, a system based on the Lightweight Directory Access Protocol, is used. The Berkeley Database Information Index (BDII) service runs at every site (siteBDII) and queries all local gLite services in a given time interval. In the same way, the siteBDII is queried by a TopLevelBDII, which stores the information of multiple grid resources. The list of available resources is kept in the information supermarket and is updated periodically by the TopBDII.

To account for the use of resources, a grid middleware may offer accounting services. Those register the amount of computation or storage used by individual or groups of users or virtual organizations. This allows billing mechanisms to be implemented and also enables the execution management system to schedule resources according to use history or quota.

The gLite Monitoring System Collector Server service gathers accounting data on the local resources and publishes these at a central service.

Security. To restrict the access of resources to certain groups of users or virtual organizations, authentication and authorization rules need to be enforced. To facilitate this, each virtual organization issues X.509 certificates to its users (user certificate) and resources (host certificate). Using the certificates, users and resources can be authenticated.

To allow services to operate on behalf of a user, it is possible to create proxy certificates. Those are created by the user and are signed with his user certificate. The proxy certificate usually only has a short lifetime (e.g., 24 hours) for security reasons. A service can the use the proxy certificate to authenticate against some other service on behalf of the user.

Authorization is granted according to membership in a virtual organization, a virtual organization group, or to single users. Access rights are managed centrally by the virtual organization for its users and resources.

[13] http://xrootd.slac.stanford.edu/.

To manage authorization information, gLite offers the Virtual Organization Management Service, which stores information on roles and privileges of users within a virtual organization. With the information from the Virtual Organization Management Service, a grid service can determine the access rights of individual users.

1.2.1 Drawbacks in Grid Computing

A grid infrastructure simplifies the handling of distributed, heterogeneous resources as well as users and virtual organizations. But despite some effort in this direction, the use of inhomogeneous software stacks (e.g., operating system, compiler, libraries) on different resources has been a weakness of grid systems.

Virtualization technology (see Section 1.4) offers solutions for this problem through an abstraction layer above the physical resources. This allows users, for example, to submit customized virtual machines containing their individual software stack. As grid middleware is not designed with virtualization in mind, the adaptation and adoption of virtualization technology into grid technology is progressing slowly. In contrast to grid computing, cloud computing (Section 1.5) was developed based on virtualization technology. Approaches to combine cloud and grid computing are presented in Section 1.6.

1.3 VIRTUALIZATION

As described in Section 1.2, grid computing is concerned mainly with secure sharing of resources in dynamic computing environments. Within the past few years, virtualization emerged and soon became a key technology, giving a new meaning to the concept of resource sharing.

This section describes two different types of technology (resource and platform virtualization) and provides an overview of their respective benefits.

Network Virtualization. At the level of the network layer, virtualization is categorized as *internal* and *external* virtualizations. External virtualization joins network partitions to a single virtual unit. Examples for this type of virtualization are virtual private networks and virtual local area networks (IEEE Computer Society, 2006). Internal virtualization offers network-like functionality to software containers on a resource. This type of virtualization is often used in combination with virtual machines.

Storage Virtualization. In mass storage systems, the capacities of physical storage devices are often pooled into a single virtual storage device, which is accessible by a consumer via the virtualization layer. Figure 1.2 depicts the layered structure of a simple storage model.

For storage systems, virtualization is a means by which multiple physical storage devices are viewed as a single logical unit. Virtualization can be accomplished in two ways at server level, fabric level, storage subsystem level, and file system level: *in-band* and *out-of-band* virtualization (Tate, 2003). In this

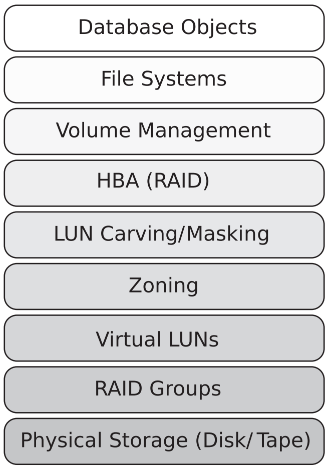

Figure 1.2 *Simple storage model (Bunn et al., 2004). HBA, Host Bus Adapter; RAID, Redundant Array of Independent Disks; LUN, Logical Unit Number.*

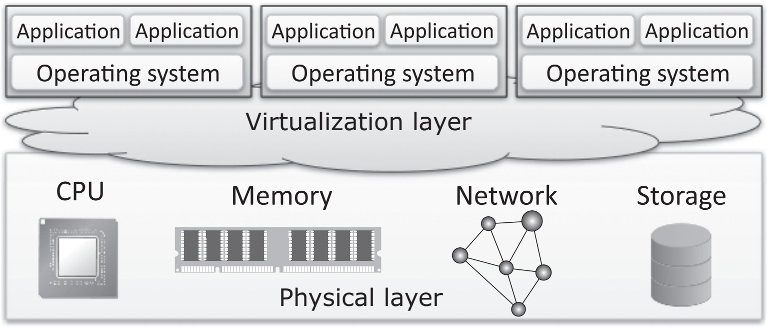

Figure 1.3 *Platform virtualization.*

context, in-band virtualization implies data and control flow over the same channel, whereas these flows are separated in out-of-band storage networks. This usually is achieved by separating data and metadata into different locations. An advantage of the latter approach is the availability of the full storage area network bandwidth to input/output (I/O) requests.

Platform Virtualization. The virtualization technologies discussed so far focus on resource virtualization. In contrast, platform virtualization embraces the complete computing infrastructure. Platform virtualization introduces a new layer between the physical hardware of the computer and the operating system layer (Fig. 1.3). This additional layer allows not only the coexistence but also the parallel execution of multiple operating system instances.

First and foremost, resource providers benefit from platform virtualization by reduced hardware, maintenance, and energy costs. Higher quality of service in terms of redundancy and security can also be achieved. For instance, platform virtualization allows the eviction of applications from physical nodes to prepare for maintenance. This can be achieved transparently and without interruption to the individual applications if they are running in virtual machines (Bhatia and Vetter, 2008).

End users of virtual machines benefit, on the one hand, from a flexible and fast deployment and, on the other hand, from a predictable and "clean" environment. Therefore, virtualization has become very popular for software testing and deployment.

1.4 CLOUD COMPUTING

Since 2007, cloud computing has become a new buzzword in the computer and information technology world. Vaquero et al. (2009) studied various existing definitions and tried, with limited success, to extract a set of attributes shared by all the definitions for cloud computing. There seems to be a reasonable consensus that the following characteristics are usually associated with cloud computing:

- **Scalability:** the ability to handle the growing use or the number of resources in a graceful manner or the ability to add resources in a seamless way
- **Pay per Use:** users pay for the computing resources (CPU, storage, network bandwidth, etc.) they consume
- **Virtualization:** providing an abstracted (or virtual) view of computing resources

Virtualization technologies help to generate the "illusion" of infinite computing and storage resources in a cloud computing environment. Depending on the services offered by a cloud, we can distinguish three types of clouds: Software as a service (SaaS), Platform as a service (PaaS), and Infrastructure as a service (IaaS). Figure 1.4 illustrates these in relation to the hardware layer.

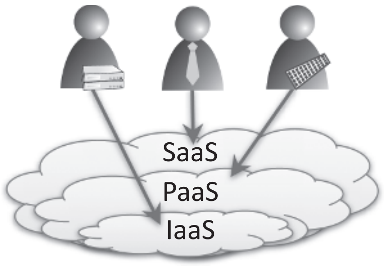

Figure 1.4 *Overview of "as-a-service" types.*

SaaS. This means that software is made available on demand to the end user by a distributed or decentralized cloud computing environment instead of a local computer. Depending on the available infrastructure and technology, one of two methods is applied to provide the software. The conventional method uses a terminal server and one or more clients. The software is installed at the server site. For access, users connect via the client to the server. The contemporary method to provide a SaaS is to deliver the service as part of a Web application; hence, access is normally through a Web browser.

Keeping the software at a central repository reduces the total overhead for maintenance (e.g., installation and updating); instead of several software installations, only one is maintained. The use of SaaS requires a reliable and fast link to the cloud provider. In some cases, a local installation is still preferable to improve performance.

One of the established SaaS providers is Google. Google Docs,[14] for example, offers an online word processing and spreadsheet software.

PaaS. PaaS environments (e.g., Google App Engine[15]) are used for the development and deployment of applications while avoiding the need to buy and manage the physical infrastructure. Compared to SaaS, PaaS provides facilities like database integration, application versioning, persistence, scalability, and security. Together, these services support the complete life cycle of developing and delivering Web applications and services available entirely through the Internet.

IaaS. IaaS defines the highest service level in privacy and reliability that resource providers currently offer their customers. In this case, the term infrastructure includes virtual and physical resources.

With respect to the interfaces offered by a cloud, a distinction is made between compute clouds and storage clouds. Compute cloud providers employ, for example, virtualization and by doing so offer easy access to remote computing power. In contrast to this, storage clouds focus on the persistent storage of data. Most compute clouds also offer interfaces to a storage facility which can be used to upload virtual appliances.

To satisfy the increasing demand for dynamic resource provisioning, the number of cloud providers is increasing steadily. Established providers are Amazon EC2,[16] FlexiScale,[17] and ElasticHosts.[18]

1.4.1 Drawbacks of Cloud Computing

At the peak of its hype, cloud computing was the proclaimed successor of grid computing. Unfortunately, cloud computing suffers from the same problems

[14] http://docs.google.com.
[15] http://code.google.com/intl/de/appengine/.
[16] http://aws.amazon.com/ec2/.
[17] www.flexiant.com/products/flexiscale/.
[18] www.elastichosts.com.

as grid computing—the difference is that in the cloud paradigm, the problems are located closer to the hardware layer.

The resource broker in a grid matches the job requirements (e.g., operating system and applications) with available resources and selects the most adequate resource for job[19] submission. In the context of cloud computing, the understanding of the term *job* must be revised to include virtual appliances. Nevertheless, a cloud customer is interested in deploying his job at the most adequate resource, so a matchmaking process is required. Without the existence of a cloud resource broker, the matchmaking is carried out manually.

A consequence caused by a missing cloud resource broker and by the natural human behavior to prefer favorite service providers often leads to a *vendor lock-in*.[20] The severity of the vendor lock-in problem is increased by proprietary and hence incompatible platform technologies at cloud provider level. For customers with data-intensive applications, it is difficult and expensive to migrate to a different cloud provider.[21] With the vendor lock-in, the customers strongly depend on the provided quality of service in a cloud (e.g., availability, reliability, and security). If a cloud suffers from poor availability, this implies poor availability of the customer's services.

A problem already present in grid computing is the limited network bandwidth between the data location and the computing resource. To bypass this bottleneck, commercial cloud providers started to physically mail portable storage devices to and from customers. For truly massive volumes of data, this "crude" mode of transferring data is relatively fast and cost-effective.[22]

1.4.2 Cloud Interfaces

A few years ago, standardization became one of the key factors in grid computing. Middleware like gLite and UNICORE have adopted open standards such as the Open Grid Services Architecture (OGSA) (Basic Execution Services) (Foster et al., 2008) and the Job Submission Description Language (JSDL) (Anjomshoaa et al., 2005). For cloud computing, it is not apparent if providers are interested in creating a standardized API. This API would ease the migration of data and services among cloud providers and results in a more competitive market.

Analyzing APIs of various cloud providers reveals that a common standard is unattainable in the foreseeable future. Many providers offer an API similar to the one of Amazon EC2 because of its high acceptance among customers.

At the moment, API development goes into two directions. The first direction is very similar to the developments for platform virtualization: The overlay

[19] A grid *job* is a binary executable or command to be submitted and run in a remote resource (machine) called server or "gatekeeper."

[20] A vendor lock-in makes a customer dependent on a vendor for products and services, unable to use another vendor without substantial switching costs.

[21] Data transfers within a cloud are usually free, but in- and outgoing traffic are charged.

[22] http://aws.amazon.com/importexport/.

API `libvirt`[23] was created, which supports various hypervisors (e.g., Xen, KVM, VMware, and VirtualBox). For cloud computing, there is the `libcloud` project[24]; the `libcloud` library hides inhomogeneous APIs of cloud providers (e.g., Slicehost, Rackspace, and Linode) from the user. The second direction tends toward a commonly accepted, standardized API for cloud providers. This API would make the development of `libcloud` needless.

The open cloud computing interface (OCCI) is one of the proposed API standards for cloud providers. It is targeted at providing a common interface for the management of resources hosted in IaaS clouds. OCCI allows resource aggregators to use a single common interface to multiple cloud resources and customers to interact with cloud resources in an ad hoc way (Edmonds et al., 2009).

A client encodes the required resources in a URL (see Fig. 1.5). Basic operations for resource modifications (create, retrieve, update, and delete) are mapped to the corresponding http methods `POST`, `GET`, `PUT`, and `DELETE` (Fielding et al., 1999). For example, a `POST` request for the creation of a computing resource looks similar to

```
POST /compute HTTP/1.1
Host: one.grid.tu-dortmund.de
Content-Length: 36
Content-Type: application/x-www-form-urlencoded
compute.cores=8&compute.memory=16384
```

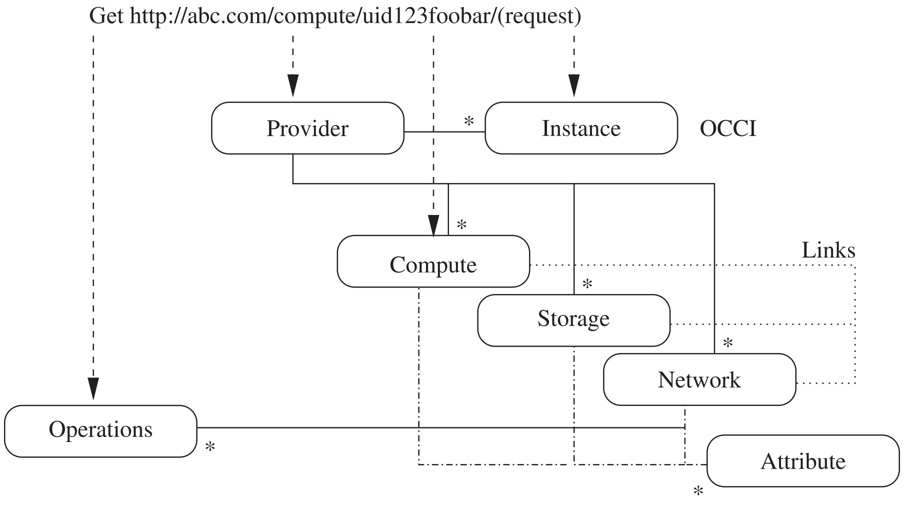

Figure 1.5 *Open cloud computing interface API.*

[23] http://libvirt.org/index.html.
[24] http://libcloud.apache.org.

Issuing this request triggers the creation of a virtual machine consisting of 8 cores and 16 GB of memory. Other attributes that can be requested are the CPU architecture (e.g., x86), a valid DNS name for the host, and the CPU speed in gigahertz.

The provision of storage (e.g., a virtual hard disk drive) via OCCI requires the specification of the storage size in gigabyte. Users are able to query the status (online, off-line, or "degraded") of the virtual storage, to back up, to resize, or for snapshot creation.

1.5 GRID AND CLOUD: TWO COMPLEMENTARY TECHNOLOGIES

Both grid and cloud computing are state-of-the-art technologies for large-scale computing. Grid and cloud computing could be viewed as mutually complementary technologies. Grid middleware is not likely to be replaced by cloud middleware because a cloud typically encompasses resources of only a single provider, while a grid spans resources of multiple providers. Nevertheless, compared with a grid resource, a cloud resource is more flexible because it adapts dynamically to the user requirements. Not surprisingly, one of the first "scientific" tasks clouds were used for was the deployment of virtual appliances containing grid middleware services. In the first days of cloud computing, only public, pay-per-use clouds (e.g., Amazon EC2) were available. Therefore, this endeavor was carried out only (1) to show its feasibility and (2) for benchmarking cloud capabilities of commercial providers. The added value of cloud computing was enough stimulus to develop open source computing and storage cloud solutions (e.g., OpenNebula,[25] Eucalyptus[26]).

With grid *and* cloud computing to their disposal, national and international e-science initiatives (e.g., D-Grid[27]) are currently reviewing their activities, which are mainly focused on grid computing. One aspect of D-Grid is to provide discipline-specific environments for collaboration and resource sharing to researchers. Independent from the discipline, a common core technology should be utilized. With the advent of cloud computing and its success in recent years, it is planned to be added to the already existing core technology, namely, grid computing.

In this context, two trends toward interoperation of grids and clouds are emerging. The first is implied by the lessons learned so far from grid computing and refers to the creation of a grid of clouds. Similar to single computing and storage resources gathered in a grid, computing and storage resources of a cloud will become part of a larger structure. As briefly shown in Section 1.4, additional value-added services like a cloud resource broker can be part of such a structure. In a possible grid of cloud setup, an information system peri-

[25] http://opennebula.org/.
[26] http://open.eucalyptus.com/.
[27] www.d-grid.de.

odically queries cloud resources, for example, health status, pricing information, and free capacities. This information is used by a cloud broker to find the most adequate resource to satisfy the user's needs. The specification of a common cloud API standard would ease the task of creating such a cloud broker.

The second identified trend started with a gap analysis. A set of components required for the seamless integration of cloud resources into existing grid infrastructures will be the result of this analysis. Using D-Grid as an example for a national grid initiative, work in the fields of user management and authentication, information systems, and accounting have been identified. Usually, authentication in a grid is based on a security infrastructure using X.509 certificates and/or short-lived credentials. Some commercial cloud providers offer a similar interface accepting certificates, but often, a username/password combination (e.g., `EC2_ACCESS_KEY` and `EC2_SECRET_KEY` for Amazon EC2) is employed for authentication.

Concerning the information systems, in D-Grid, each of the supported grid middlewares runs a separate one. The gLite middleware uses a combination of site and top-level BDII; (Web-)MDS is used in Globus Toolkit; and Common Information Service (CIS) is used in UNICORE6. The information provided by these systems is collected and aggregated by D-MON, a D-Grid monitoring service. D-MON uses an adapter for each type of supported grid middleware that converts the data received to an independent data format for further processing. To integrate information provided by a cloud middleware, a new adapter and the definition of measurement categories need to be developed.

After the creation of the missing components, grid and clouds are likely to coexist as part of e-science infrastructures.

1.6 MODELING AND SIMULATION OF GRID AND CLOUD COMPUTING

Today, modeling and simulation of large-scale computing systems are considered as a fruitful R&D area for algorithms and applications. With the increasing complexity of such environments, simulation technology has experienced great changes and has dealt with multiplatform distributed simulation, joining performance, and structure. A high-performance simulator is necessary to investigate various system configurations and to create different application scenarios before putting them into practice in real large-scale distributed computing systems. Consequently, simulation tools should enable researchers to study and evaluate developed methods and policies in a repeatable and controllable environment and to tune algorithm performance before deployment on operational systems. A grid or a cloud simulator has to be able to model heterogeneous computational components; to create resources, a network topology, and users; to enable the implementation of various job scheduling algorithms; to manage service pricing; and to output statistical results.

The existing discrete-event simulation studies cover a small number of grid sites, with only a few hundreds of hosts, as well as large-scale use case grids, including thousands or millions of hosts. Common simulation studies deal with job scheduling and data replication algorithms to achieve better performance and high availability of data.

A list of grid simulation tools that implement one or more of the above-mentioned functionalities include the following:

1. *OptorSim* (Bell et al., 2002) is being developed as part of the EU DataGrid project. It mainly emphasizes grid replication strategies and optimization.
2. *SimGrid* toolkit (Casanova, 2001), developed at the University of California at San Diego, is a C-based simulator for scheduling algorithms.
3. *MicroGrid* emulator (Song et al., 2000), developed at the University of California at San Diego, can be used for building grid infrastructures. It allows to execute applications, created using the Globus Toolkit, in a virtual grid environment.
4. *GangSim* (Dumitrescu and Foster, 2005), developed at the University of Chicago, aims to study the usage and scheduling policies in a multivirtual organization environment and is able to model real grids of considerable size.

1.6.1 GridSim and CloudSim Toolkits

GridSim (Sulistio et al., 2008) is a Java-based discrete-event grid simulation toolkit, developed at the University of Melbourne. It enables modeling and simulation of heterogeneous grid resources, users, and applications. Users can customize algorithms and workload.

At the lowest layer of the GridSim architecture operates the SimJava (Howell and McNab, 1998) discrete-event simulation engine. It provides GridSim with the core functionalities needed for higher-level simulation scenarios, such as queuing and processing of events, creation of system components, communication between components, and management of the simulation clock.

The GridSim toolkit enables to run reproducible scenarios that are not possible in a real grid infrastructure. It supports high-level software components for modeling multiple grid infrastructures and basic grid components such as the resources, workload traces, and information services. GridSim enables the modeling of different resource characteristics. Therefore, it allocates incoming jobs based on space or time-shared mode. It is able to schedule compute- and data-intensive jobs; it allows easy implementation of different resource allocation algorithms; it supports reservation-based or auction mechanisms for resource allocation; it enables simulation of virtual organization

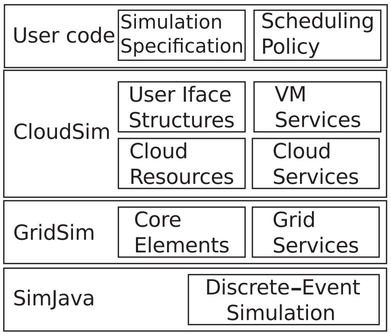

Figure 1.6 *CloudSim architecture layers. Iface, interface; VM, virtual machine.*

scenarios; it is able to visualize tracing sequences of simulation execution; and it bases background network traffic characteristics on a probabilistic distribution (Buyya and Murshed, 2002).

CloudSim (Buyya et al., 2009) is a framework, also developed at the University of Melbourne, that enables modeling, simulation, and experimenting on cloud computing infrastructures. It is built on top of GridSim (Fig. 1.6).

CloudSim is a platform that can be used to model data centers, service brokers, scheduling algorithms, and allocation policies of a large-scale cloud computing platform. It supports the creation of virtual machines on a simulated node, cloudlets, and allows assigning jobs to appropriate virtual machines. It also enables simulation of various data centers to allow investigations on federation and related policies for the migration of virtual machines.

1.7 SUMMARY AND OUTLOOK

The demand for computing and storage resources has been increasing steadily over many years. In the past, static resources of multiple providers were subsumed into a grid. The concept of resource sharing within a grid and its drawbacks were described in Section 1.2. Section 1.3 introduced two different virtualization types: resource and platform virtualization. Especially, platform virtualization is one of the key enabling technologies for the concept of cloud computing. Cloud providers act as service providers and offer to customers software (SaaS), development platforms (PaaS), and infrastructure (IaaS). Section 1.4 provided a brief overview of cloud computing and the three main service types (SaaS, PaaS, and IaaS).

Section 1.5 provided an outlook on efforts attempting to integrate cloud and grid middleware. e-Science initiatives need to provide a uniform and simple interface to both types of resources to facilitate future developments.

Section 1.6 discussed the need to model and simulate grid and cloud environments and important grid simulation toolkits (e.g., the Java-based simulators GridSim and CloudSim).

REFERENCES

L. Abadie, P. Badino, J. P. Baud, et al. Grid-enabled standards-based data management. In *24th IEEE Conference on Mass Storage Systems and Technologies*, pp. 60–71, Los Alamitos, California, IEEE Computer Society. 2007.

C. Aiftimiei, M. Sgaravatto, L. Zangrando, et al. Design and implementation of the gLite CREAM job management service. *Future Generation Computer Systems*, 26(4):654–667, 2010.

A. Anjomshoaa, F. Brisard, M. Drescher, et al. Job Submission Description Language (JSDL) Specification: Version 1.0, 2005. http://www.gridforum.org/documents/GFD.56.pdf.

P. Badino, O. Barring, J. P. Baud, et al. The storage resource manager interface specification version 2.2, 2009. http://sdm.lbl.gov/srm-wg/doc/SRM.v2.2.html.

W. Bell, D. Cameron, L. Capozza, et al. Simulation of dynamic grid replication strategies in OptorSim. In M. Parashar, editor, *Grid Computing, Volume 2536 of Lecture Notes in Computer Science*, pp. 46–57, Berlin: Springer, 2002.

N. Bhatia and J. Vetter. Virtual cluster management with Xen. In L. Bougé, M. Alexander, S. Childs, et al., editors, *Euro-Par 2007 Workshops: Parallel Processing, volume 4854 of Lecture Notes in Computer Science*, pp. 185–194, Berlin and Heidelberg: Springer-Verlag, 2008.

F. Bunn, N. Simpson, R. Peglar, et al. Storage virtualization: SNIA technical tutorial, 2004. http://www.snia.org/sites/default/files/sniavirt.pdf.

S. Burke, S. Campana, E. Lanciotti, et al. gLite 3.1 user guide: Version 1.2, 2009. https://edms.cern.ch/file/722398/1.2/gLite-3-UserGuide.html.

R. Buyya and M. M. Murshed. GridSim: A toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing. *Concurrency and Computation: Practice and Experience*, 14(13–15):1175–1220, 2002.

R. Buyya, R. Ranjan, and R. N. Calheiros. Modeling and simulation of scalable cloud computing environments and the CloudSim Toolkit: Challenges and opportunities. In Waleed W. Smari and John P. McIntire, editor, *Proceedings of the 2009 International Conference on High Performance Computing and Simulation*, pp. 1–11, Piscataway, NJ: IEEE, 2009.

H. Casanova. Simgrid: A toolkit for the simulation of application scheduling. In R. Buyya, editor, *CCGRID'01: Proc. of the 1st Int'l Symposium on Cluster Computing and the Grid*, pp. 430–441, Los Alamitos, CA: IEEE Computer Society, 2001.

C. L. Dumitrescu and I. Foster. GangSim: A simulator for grid scheduling studies. In *CCGRID'05: Proc. of the 5th IEEE Int'l Symposium on Cluster Computing and the Grid*, pp. 1151–1158, Washington, DC: IEEE Computer Society, 2005.

A. Edmonds, S. Johnston, G. Mazzaferro, et al. Open Cloud Computing Interface Specification version 5, 2009. http://forge.ogf.org/sf/go/doc15731.

R. Fielding, J. Gettys, J. Mogul, et al. RFC 2616: Hypertext transfer protocol—HTTP/1.1. *Status: Standards Track*, 1999. http://www.ietf.org/rfc/rfc2616.txt.

I. Foster. What is the grid?—A three point checklist. *GRIDToday*, 1(6):22–25, 2002.

I. Foster. Globus Toolkit version 4: Software for service-oriented systems. In *Network and parallel computing*, pp. 2–13, 2005.

I. Foster, A. Grimshaw, P. Lane, et al. OGSA basic execution service: Version 1.0, 2008. http://www.ogf.org/documents/GFD.108.pdf.

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

F. Howell and R. McNab. SimJava: A discrete event simulation library for Java. In *Int'l Conference on Web-Based Modeling and Simulation*, pp. 51–56, 1998.

IEEE Computer Society. IEEE standard for local and metropolitan area networks virtual bridged local area networks, 2006. http://ieeexplore.ieee.org.

A Mowshowitz. Virtual organization. *Communications of the ACM*, 40(9):30–37, 1997.

H. J. Song, X. Liu, D. Jakobsen, et al. The MicroGrid: A scientific tool for modeling computational grids. *Scientific Programming*, 8(3):127–141, 2000.

A. Sulistio, R. Buyya, U. Cibej, et al. A toolkit for modelling and simulating data grids: An extension to GridSim. *Concurrency and Computation: Practice and Experience*, 20(13):1591–1609, 2008.

J. Tate. Virtualization in a SAN, 2003. http://www.redbooks.ibm.com/redpapers/pdfs/redp3633.pdf.

J. E. Thornton. Parallel operation in the control data 6600. In *AFIPS 1964 Fall Joint Computer Conference*, pp. 33–41, Spartan Books, 1965.

L. M. Vaquero, L. Rodero-Merino, J. Caceres, et al. A break in the clouds: Towards a cloud definition. *ACM SIGCOMM Computer Communication Review*, 39(1):50–55, 2009.