PART III

PLATFORM AND SOFTWARE AS A SERVICE (PaaS/IaaS)

CHAPTER 9

ANEKA—INTEGRATION OF PRIVATE AND PUBLIC CLOUDS

CHRISTIAN VECCHIOLA, XINGCHEN CHU, MICHAEL MATTESS, and RAJKUMAR BUYYA

9.1 INTRODUCTION

A growing interest in moving software applications, services, and even infrastructure resources from in-house premises to external providers has been witnessed recently. A survey conducted by F5 Networks between June and July 2009[1] showed that such a trend has now reached a critical mass; and an increasing number of IT managers have already adopted, or are considering adopting, this approach to implement IT operations. This model of making IT resources available, known as Cloud Computing [1], opens new opportunities to small, medium-sized, and large companies. It is not necessary anymore to bear considerable costs for maintaining the IT infrastructures or to plan for peak demand. Instead, infrastructure and applications can scale elastically according to the business needs at a reasonable price. The possibility of instantly reacting to the demand of customers without long-term planning is one of the most appealing features of cloud computing, and it has been a key factor in making this trend popular among technology and business practitioners.

As a result of this growing interest, the major players in the IT industry such as Google, Amazon, Microsoft, Sun, and Yahoo have started offering cloud-computing-based solutions that cover the entire IT computing stack, from hardware to applications and services. These offerings have become quickly

[1]The survey, available at http://www.f5.com/pdf/reports/cloud-computing-survey-results-2009.pdf, interviewed 250 IT companies with at least 2500 employees worldwide and targeted the following personnel: managers, directors, vice presidents, and senior vice presidents.

Cloud Computing: Principles and Paradigms, Edited by Rajkumar Buyya, James Broberg and Andrzej Goscinski Copyright © 2011 John Wiley & Sons, Inc.

popular and led to the establishment of the concept of “Public Cloud,” which represents a publicly accessible distributed system hosting the execution of applications and providing services billed on a pay-per-use basis. After an initial enthusiasm for this new trend, it soon became evident that a solution built on outsourcing the entire IT infrastructure to third parties would not be applicable in many cases, especially when there are critical operations to be performed and security concerns to consider. Moreover, with the public cloud distributed anywhere on the planet, legal issues arise and they simply make it difficult to rely on a virtual public infrastructure for any IT operation. As an example, data location and confidentiality are two of the major issues that scare stakeholders to move into the cloud—data that might be secure in one country may not be secure in another. In many cases though, users of cloud services don’t know where their information is held and different jurisdictions can apply. It could be stored in some data center in either Europe, (a) where the European Union favors very strict protection of privacy, or (b) America, where laws such as the U.S. Patriot Act[2] invest government and other agencies with virtually limitless powers to access information including that belonging to companies. In addition, enterprises already have their own IT infrastructures. In spite of this, the distinctive feature of cloud computing still remains appealing, and the possibility of replicating in-house (on their own IT infrastructure) the resource and service provisioning model proposed by cloud computing led to the development of the “Private Cloud” concept.

Private clouds are virtual distributed systems that rely on a private infrastructure and provide internal users with dynamic provisioning of computing resources. Differently from public clouds, instead of a pay-as-you-go model, there could be other schemes in place, which take into account the usage of the cloud and proportionally bill the different departments or sections of the enterprise. Private clouds have the advantage of keeping in-house the core business operations by relying on the existing IT infrastructure and reducing the burden of maintaining it once the cloud has been set up. In this scenario, security concerns are less critical, since sensitive information does not flow out of the private infrastructure. Moreover, existing IT resources can be better utilized since the Private cloud becomes accessible to all the division of the enterprise. Another interesting opportunity that comes with private clouds is the possibility of testing applications and systems at a comparatively lower price rather than public clouds before deploying them on the public virtual infrastructure. In April 2009, a Forrester Report [2] on the benefits of delivering in-house cloud computing solutions for enterprises

[2]The U.S. Patriot Act is a statute enacted by the United States Government that increases the ability of law enforcement agencies to search telephone, e-mail communications, medical, financial, and other records; it eases restrictions on foreign intelligence gathering within the United States. The full text of the act is available at the Web site of the Library of the Congress at the following address: http://thomas.loc.gov/cgi-bin/query/z?c107:H.R.3162.ENR (accessed December 5, 2009).

highlighted some of the key advantages of using a private cloud computing infrastructure:

- *Customer Information Protection*. Despite assurances by the public cloud leaders about security, few provide satisfactory disclosure or have long enough histories with their cloud offerings to provide warranties about the specific level of security put in place in their system. Security in-house is easier to maintain and to rely on.
- *Infrastructure Ensuring Service Level Agreements (SLAs)*. Quality of service implies that specific operations such as appropriate clustering and failover, data replication, system monitoring and maintenance, disaster recovery, and other uptime services can be commensurate to the application needs. While public clouds vendors provide some of these features, not all of them are available as needed.
- *Compliance with Standard Procedures and Operations*. If organizations are subject to third-party compliance standards, specific procedures have to be put in place when deploying and executing applications. This could be not possible in the case of virtual public infrastructure.

In spite of these advantages, private clouds cannot easily scale out in the case of peak demand, and the integration with public clouds could be a solution to the increased load. Hence, hybrid clouds, which are the result of a private cloud growing and provisioning resources from a public cloud, are likely to be best option for the future in many cases. Hybrid clouds allow exploiting existing IT infrastructures, maintaining sensitive information within the premises, and naturally growing and shrinking by provisioning external resources and releasing them when needed. Security concerns are then only limited to the public portion of the cloud, which can be used to perform operations with less stringent constraints but that are still part the system workload.

Platform as a Service (PaaS) solutions offer the right tools to implement and deploy hybrid clouds. They provide enterprises with a platform for creating, deploying, and managing distributed applications on top of existing infrastructures. They are in charge of monitoring and managing the infrastructure and acquiring new nodes, and they rely on virtualization technologies in order to scale applications on demand. There are different implementations of the PaaS model; in this chapter we will introduce Manjrasoft Aneka, and we will discuss how to build and deploy hybrid clouds based on this technology. Aneka [3] is a programming and management platform for building and deploying cloud computing applications. The core value of Aneka is its service-oriented architecture that creates an extensible system able to address different application scenarios and deployments such as public, private, and heterogeneous clouds. On top of these, applications that can be expressed by means of different programming models can transparently execute under the desired service-level agreement.

The remainder of this chapter is organized as follows: In the next section we will briefly review the technologies and tools for Cloud Computing by presenting both the commercial solution and the research projects currently available, we will then introduce Aneka in Section 9.3 and provide an overview of the architecture of the system. In Section 9.4 we will detail the resource provisioning service that represents the core feature for building hybrid clouds. Its architecture and implementation will be described in Section 9.5, together with a discussion about the desired features that a software platform support hybrid clouds should offer. Some thoughts and future directions for practitioners will follow, before the conclusions.

9.2 TECHNOLOGIES AND TOOLS FOR CLOUD COMPUTING

Cloud computing covers the entire computing stack from hardware infrastructure to end-user software applications. Hence, there are heterogeneous offerings addressing different niches of the market. In this section we will concentrate mostly on the Infrastructure as a Service (IaaS) and Platform as a Service (PaaS) implementations of the cloud computing model by first presenting a subset of the most representative commercial solutions and then discussing the few research projects and platforms, which attracted considerable attention.

Amazon is probably the major player for what concerns the Infrastructure-as-a-Service solutions in the case of public clouds. Amazon Web Services [4] deliver a set of services that, when composed together, form a reliable, scalable, and economically accessible cloud. Within the wide range of services offered, it is worth noting that Amazon Elastic Compute Cloud (EC2) [5] and Simple Storage Service (S3) [6] allow users to quickly obtain virtual compute resources and storage space, respectively. GoGrid [7] provides customer with a similar offer: it allows users to deploy their own distributed system on top of their virtual infrastructure. By using the GoGrid Web interface users can create their custom virtual images, deploy database and application servers, and mount new storage volumes for their applications. Both GoGrid and Amazon EC2 charge their customers on a pay-as-you-go basis, and resources are priced per hours of usage. 3Tera AppLogic [8] lays at the foundation of many public clouds, it provides a grid operating system that includes workload distribution, metering, and management of applications. These are described in a platform-independent manner, and AppLogic takes care of deploying and scaling them on demand. Together with AppLogic, which can also be used to manage and deploy private clouds, 3Tera also provides cloud hosting solutions and, because of its grid operating system, makes the transition from the private to the public virtual infrastructure simple and completely transparent. Solutions that are completely based on a PaaS approach for public clouds are Microsoft Azure and Google AppEngine. Azure [9] allows developing scalable applications for the cloud. It is a cloud services operating system that serves as the development, runtime, and control environment for the Azure Services Platform. By using the Microsoft Azure SDK, developers can create services that leverage the .NET framework. These services are then uploaded to the Microsoft Azure portal and executed on top of Windows Azure. Additional services such as workflow management and execution, web services orchestration, and SQL data storage are provided to empower the hosted applications. Azure customers are billed on a pay-per-use basis and by taking into account the different services: compute, storage, bandwidth, and storage transactions. Google AppEngine [10] is a development platform and a runtime environment focusing primarily on web applications that will be run on top of Google's server infrastructure. It provides a set of APIs and an application model that allow developers to take advantage of additional services provided by Google such as *Mail*, *Datastore*, *Memcache*, and others. Developers can create applications in Java, Python, and JRuby. These applications will be run within a sandbox, and AppEngine will take care of automatically scaling when needed. Google provides a free limited service and utilizes daily and per minute quotas to meter and price applications requiring professional service.

Different options are available for deploying and managing private clouds. At the lowest level, virtual machine technologies such as Xen [11], KVM [12], and VMware [13] can help building the foundations of a virtual infrastructure. On top of this, virtual machine managers such as VMWare vCloud [14] and Eucalyptus [15] allow the management of a virtual infrastructure and turning a cluster or a desktop grid into a private cloud. Eucalyptus provides a full compatibility with the Amazon Web Services interfaces and supports different virtual machine technologies such as Xen, VMWare, and KVM. By using Eucalyptus, users can test and deploy their cloud applications on the private premises and naturally move to the public virtual infrastructure provided by Amazon EC2 and S3 in a complete transparent manner. VMWare vCloud is the solution proposed by VMWare for deploying virtual infrastructure as either public or private clouds. It is built on top of the VMWare virtual machine technology and provides an easy way to migrate from the private premises to the public infrastructure that leverages VMWare for infrastructure virtualization. For what concerns the Platform-as-a-Service solutions, we can notice DataSynapse, Elastra, Zimory Pools, and the already mentioned AppLogic. DataSynapse [16] is a global provider of application virtualization software. By relying on the VMWare, virtualization technology provides a flexible environment that converts a data center into a private cloud. Elastra [17] cloud server is a platform for easily configuring and deploying distributed application infrastructures on clouds: by using a simple control panel, administrators can visually describe the distributed application in terms of components and connections and then deploying them on one or more cloud providers such Amazon EC2 or VMware ESX. Cloud server can provision resources from either private or public clouds, thus deploying application on hybrid infrastructures. Zimory [18], a spinoff company from Deutsche Telekom, provides a software infrastructure layer that automates the use of resource pools based on Xen, KVM, and VMware virtualization technologies. It allows creating an internal cloud composed by sparse private and public resources that both host the Zimory's software agent and provides facilities for quickly migrating applications from one data center to another and utilizing at best the existing infrastructure.

The wide range of commercial offerings for deploying and managing private and public clouds mostly rely on a few key virtualization technologies, on top of which additional services and features are provided. In this sense, an interesting research project combining public and private clouds and adding advanced services such as resource reservation is represented by the coordinated use of OpenNebula [19] and Haizea [20]. OpenNebula is a virtual infrastructure manager that can be used to deploy and manage virtual machines on local resources or on external public clouds, automating the setup of the virtual machines regardless of the underlying virtualization layer (Xen, KVM, or VMWare are currently supported) or external cloud such as Amazon EC2. A key feature of OpenNebula's architecture is its highly modular design, which facilitates integration with any virtualization platform and third-party component in the cloud ecosystem, such as cloud toolkits, virtual image managers, service managers, and VM schedulers such as Haizea. Haizea is a resource lease manager providing leasing capabilities not found in other cloud systems, such as advance reservations and resource preemption. Integrated together, OpenNebula and Haizea constitute a virtual management infrastructure providing flexible and advanced capabilities for resource management in hybrid clouds. A similar set of capabilities is provided by OpenPEX [21], which allows users to provision resources ahead of time through advance reservations. It also incorporates a bilateral negotiation protocol that allows users and providers to come to an agreement by exchanging offers and counter offers. OpenPEX natively supports Xen as a virtual machine manager (VMM), but additional plug-ins can be integrated into the system to support other VMMs. Nimbus [22], formerly known as Globus Workspaces, is another framework that provides a wide range of extensibility points. It is essentially a framework that allows turning a cluster into an Infrastructure-as-a-Service cloud. What makes it interesting from the perspective of hybrid clouds is an extremely modular architecture that allows the customization of many tasks: resource scheduling, network leases, accounting, propagation (intra VM file transfer), and fine control VM management.

All of the previous research platforms are mostly IaaS implementation of the cloud computing model: They provide a virtual infrastructure management layer that is enriched with advanced features for resource provisioning and scheduling. Aneka, which is both a commercial solution and a research platform, positions itself as a Platform-as-a-Service implementation. Aneka provides not only a software infrastructure for scaling applications, but also a wide range of APIs that help developers to design and implement applications that can transparently run on a distributed infrastructure whether this be the local cluster or the cloud. Aneka, as OpenNebula and Nimbus, is characterized by a modular architecture that allows a high level of customization and integration with existing technologies, especially for what concerns resource provisioning. Like Zimory, the core feature of Aneka is represented by a configurable software agent that can be transparently deployed on both physical and virtual resources and constitutes the runtime environment for the cloud. This feature, together with the resource provisioning infrastructure, is at the heart of Aneka-based hybrid clouds. In the next sections we will introduce the key feature of Aneka and describe in detail the architecture of the resource provisioning service that is responsible of integrating cloud resources into the existing infrastructure.

9.3 ANEKA CLOUD PLATFORM

Aneka [3] is a software platform and a framework for developing distributed applications on the cloud. It harnesses the computing resources of a heterogeneous network of workstations and servers or data centers on demand. Aneka provides developers with a rich set of APIs for transparently exploiting these resources by expressing the application logic with a variety of programming abstractions. System administrators can leverage a collection of tools to monitor and control the deployed infrastructure. This can be a public cloud available to anyone through the Internet, a private cloud constituted by a set of nodes with restricted access within an enterprise, or a hybrid cloud where external resources are integrated on demand, thus allowing applications to scale.

Figure 9.1 provides a layered view of the framework. Aneka is essentially an implementation of the PaaS model, and it provides a runtime environment for executing applications by leveraging the underlying infrastructure of the cloud. Developers can express distributed applications by using the API contained in the Software Development Kit (SDK) or by porting existing legacy applications to the cloud. Such applications are executed on the Aneka cloud, represented by a collection of nodes connected through the network hosting the Aneka container. The container is the building block of the middleware and represents the runtime environment for executing applications; it contains the core functionalities of the system and is built up from an extensible collection of services that allow administrators to customize the Aneka cloud. There are three classes of services that characterize the container:

- *Execution Services*. They are responsible for scheduling and executing applications. Each of the programming models supported by Aneka defines specialized implementations of these services for managing the execution of a unit of work defined in the model.
- *Foundation Services*. These are the core management services of the Aneka container. They are in charge of metering applications, allocating

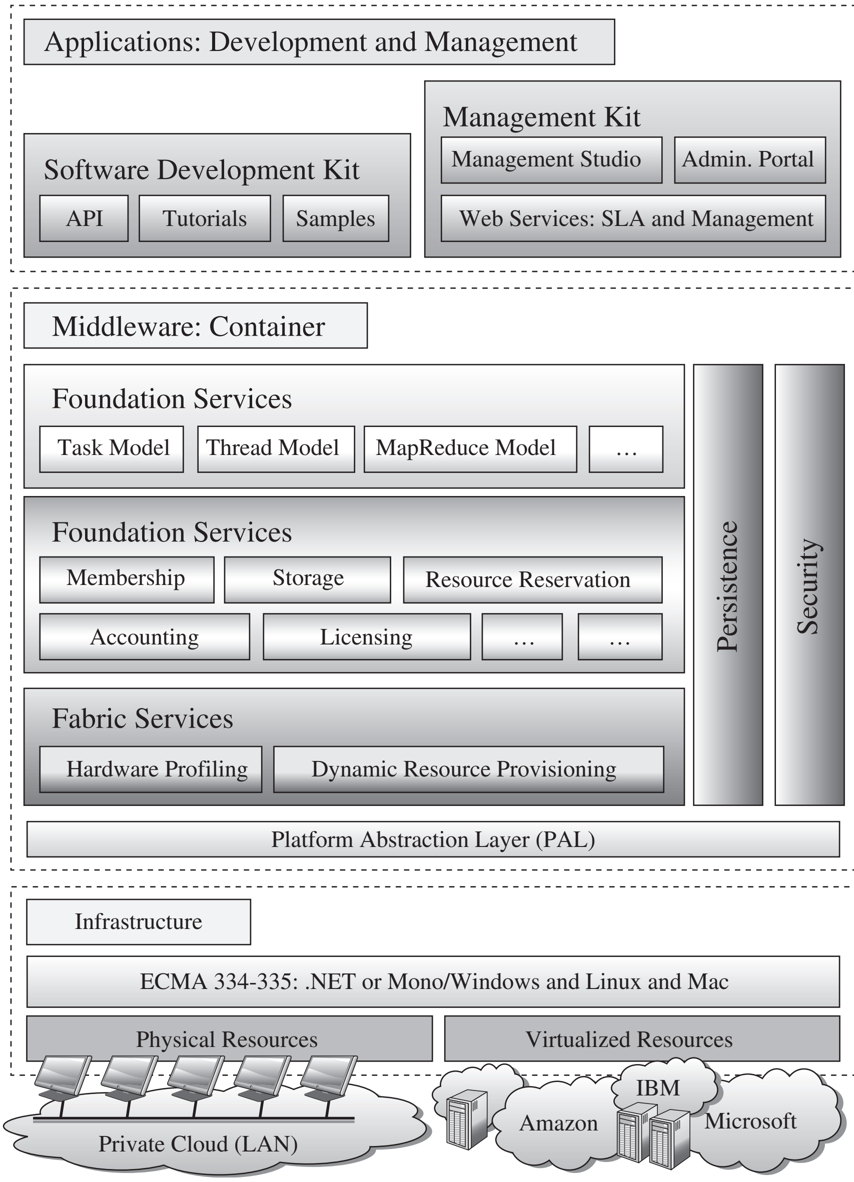

FIGURE 9.1. Aneka framework architecture.

resources for execution, managing the collection of available nodes, and keeping the services registry updated.

- *Fabric Services:* They constitute the lowest level of the services stack of Aneka and provide access to the resources managed by the cloud. An important service in this layer is the *Resource Provisioning Service,* which enables horizontal scaling[3] in the cloud. Resource provisioning makes Aneka elastic and allows it to grow or to shrink dynamically to meet the QoS requirements of applications.

The container relies on a platform abstraction layer that interfaces it with the underlying host, whether this is a physical or a virtualized resource. This makes the container portable over different runtime environments that feature an implementation of the ECMA 334 [23] and ECMA 335 [24] specifications (such as the .NET framework or Mono).

Aneka also provides a tool for managing the cloud, allowing administrators to easily start, stop, and deploy instances of the Aneka container on new resources and then reconfigure them dynamically to alter the behavior of the cloud.

9.4 ANEKA RESOURCE PROVISIONING SERVICE

The most significant benefit of cloud computing is the elasticity of resources, services, and applications, which is the ability to automatically scale out based on demand and users' quality of service requests. Aneka as a PaaS not only features multiple programming models allowing developers to easily build their distributed applications, but also provides resource provisioning facilities in a seamless and dynamic fashion. Applications managed by the Aneka container can be dynamically mapped to heterogeneous resources, which can grow or shrink according to the application's needs. This elasticity is achieved by means of the resource provisioning framework, which is composed primarily of services built into the Aneka fabric layer.

Figure 9.2 provides an overview of Aneka resource provisioning over private and public clouds. This is a typical scenario that a medium or large enterprise may encounter; it combines privately owned resources with public rented resources to dynamically increase the resource capacity to a larger scale.

Private resources identify computing and storage elements kept in the premises that share similar internal security and administrative policies. Aneka identifies two types of private resources: *static* and *dynamic* resources. Static resources are constituted by existing physical workstations and servers that may be idle for a certain period of time. Their membership to the Aneka cloud is manually configured by administrators and does not change over time. Dynamic resources are mostly represented by virtual instances that join and leave the Aneka cloud and are controlled by resource pool managers that provision and release them when needed.

[3]Horizontal scaling is the process of adding more computing nodes to a system. It is counterposed to vertical scaling, which is the process of increasing the computing capability of a single computer resource.

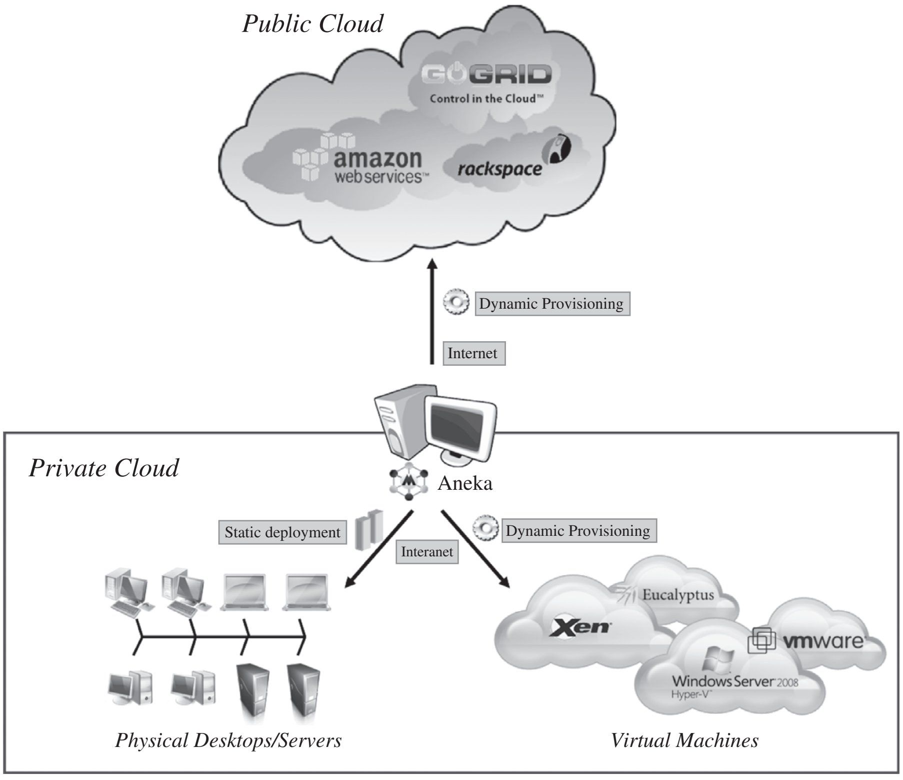

FIGURE 9.2. Aneka resource provisioning over private and public clouds.

Public resources reside outside the boundaries of the enterprise and are provisioned by establishing a service-level agreement with the external provider. Even in this case we can identify two classes: *on-demand* and *reserved* resources. On-demand resources are dynamically provisioned by resource pools for a fixed amount of time (for example, an hour) with no long-term commitments and on a pay-as-you-go basis. Reserved resources are provisioned in advance by paying a low, one-time fee and mostly suited for long-term usage. These resources are actually the same as static resources, and no automation is needed in the resource provisioning service to manage them.

Despite the specific classification previously introduced, resources are managed uniformly once they have joined the Aneka cloud and all the standard operations that are performed on statically configured nodes can be transparently applied to dynamic virtual instances. Moreover, specific operations pertaining to dynamic resources, such as join and leave, are seen as connection and disconnection of nodes and transparently handled. This is mostly due to the indirection layer provided by the Aneka container that abstracts the specific nature of the hosting machine.

9.4.1 Resource Provisioning Scenario

Figure 9.3 illustrates a possible scenario in which the resource provisioning service becomes important. A private enterprise maintains a private cloud, which consists of (a) five physical dedicated desktops from its engineering department and (b) a small data center managed by Xen Hypervisor providing virtual machines with the maximum capacity of 12 VMs. In most of the cases, this setting is able to address the computing needs of the enterprise. In the case of peak computing demand, additional resources can be provisioned by leveraging the virtual public infrastructure. For example, a mission critical application could require at least 30 resources to complete within an hour, and the customer is willing to spend a maximum of 5 dollars to achieve this goal. In this case, the Aneka Resource Provisioning service becomes a fundamental infrastructure component to address this scenario.

In this case, once the client has submitted the application, the Aneka scheduling engine detects that the current capacity in terms of resources (5 dedicated nodes) is not enough to satisfy the user's QoS requirement and to complete the application on time. An additional 25 resources must be provisioned. It is the responsibility of the Aneka Resource Provisioning service to acquire these resources from both the private data center managed by Xen Hypervisor and the Amazon public cloud. The provisioning service is configured by default with a cost-effective strategy, which privileges the use of local resources instead of the dynamically provisioned and chargeable ones. The computing needs of the application require the full utilization of the local data center that provides the Aneka cloud with 12 virtual machines. Such capacity is still not enough to complete the mission critical application in time; and the

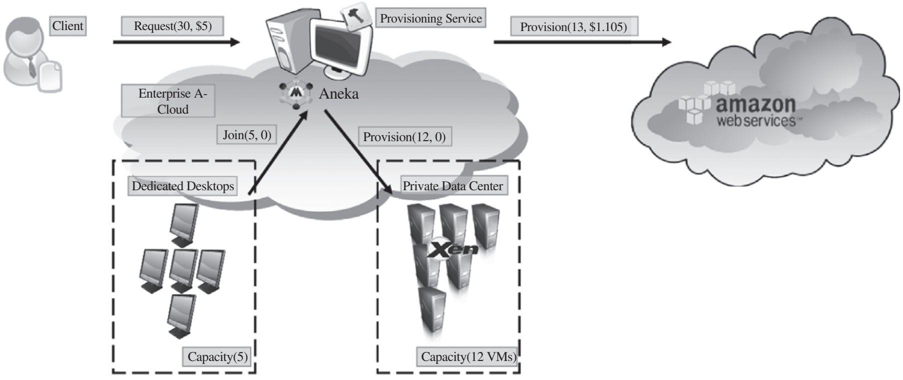

FIGURE 9.3. Use case of resource provisioning under Aneka.

remaining 13 resources are rented from Amazon for a minimum of one hour, which incurs a few dollars' cost.[4]

This is not the only scenario that Aneka can support, and different provisioning patterns can be implemented. Another simple strategy for provisioning resources could be minimizing the execution time to let the application finish as early as possible; this requires Aneka to request more powerful resources from the Amazon public cloud. For example, in the previous case instead of provisioning 13 small instances from Amazon, a major number of resources, or more powerful resources, can be rented by spending the entire budget available for the application. The resource provisioning infrastructure can also serve broader purposes such as keeping the length of the system queue, or the average waiting time of a job in the queue, under a specified value. In these cases, specific policies can be implemented to ensure that the throughput of the system is kept at a reasonable level.

9.5 HYBRID CLOUD IMPLEMENTATION

Currently, there is no widely accepted standard for provisioning virtual infrastructure from Infrastructure as a Service (IaaS) providers, but each provider exposes its own interfaces and protocols. Hence, it is not possible to seamlessly integrate different providers into one single infrastructure. The resource provisioning service implemented in Aneka addresses these issues and abstracts away the differences of providers' implementation. In this section we will briefly review what the desired features of a hybrid cloud implementation are and then we will give a closer a look at the solution implemented in Aneka together with a practical application of the infrastructure developed.

9.5.1 Design and Implementation Guidelines

The particular nature of hybrid clouds demands additional and specific functionalities that software engineers have to consider while designing software systems supporting the execution of applications in hybrid and dynamic environments. These features, together with some guidelines on how to implement them, are presented in the following:

- *Support for Heterogeneity.* Hybrid clouds are produced by heterogeneous resources such as clusters, public or private virtual infrastructures, and workstations. In particular, for what concerns a virtual machine manager, it must be possible to integrate additional cloud service providers (mostly

[4]At the time of writing (October 2010), the cost for a small Linux-based instance in Amazon EC2 is 0.085 cent/hour and the total cost bore by the customer will be in this case 1.105 UD. We expect this price to decrease even more in the next years.

IaaS providers) without major changes to the entire system design and codebase. Hence, the specific code related to a particular cloud resource provider should be kept isolated behind interfaces and within pluggable components.

- *Support for Dynamic and Open Systems*. Hybrid clouds change their composition and topology over time. They form as a result of dynamic conditions such as peak demands or specific Service Level Agreements attached to the applications currently in execution. An open and extensible architecture that allows easily plugging new components and rapidly integrating new features is of a great value in this case. Specific enterprise architectural patterns can be considered while designing such software systems. In particular, *inversion of control* and, more precisely, *dependency injection*[5] in component-based systems is really helpful.
- *Support for Basic VM Operation Management*. Hybrid clouds integrate virtual infrastructures with existing physical systems. Virtual infrastructures are produced by virtual instances. Hence, software frameworks that support hypervisor-based execution should implement a minimum set of operations. They include requesting a virtual instance, controlling its status, terminating its execution, and keeping track of all the instances that have been requested.
- *Support for Flexible Scheduling Policies*. The heterogeneity of resources that constitute a hybrid infrastructure naturally demands for flexible scheduling policies. Public and private resources can be differently utilized, and the workload should be dynamically partitioned into different streams according to their security and quality of service (QoS) requirements. There is then the need of being able to transparently change scheduling policies over time with a minimum impact on the existing infrastructure and almost now downtimes. Configurable scheduling policies are then an important feature.
- *Support for Workload Monitoring*. Workload monitoring becomes even more important in the case of hybrid clouds where a subset of resources is leased and resources can be dismissed if they are no longer necessary. Workload monitoring is an important feature for any distributed middleware, in the case of hybrid clouds, it is necessary to integrate this feature with scheduling policies that either directly or indirectly govern the management of virtual instances and their leases.

[5]Dependency injection is a technique that allows configuring and connecting components within a software container (such as a Web or an application server) without hard coding their relation but for example by providing an abstract specification—for example, a configuration file that specifies which component to instantiate and to connect them together. A detailed description of this programming pattern can be found at the following link: http://martinfowler.com/articles/injection.html (accessed December 2009).

Those presented are, according to the authors, the most relevant features for successfully supporting the deployment and the management of hybrid clouds. In this list we did not extensively mention security that is transversal to all features listed. A basic recommendation for implementing a security infrastructure for any runtime environment is to use a *Defense in Depth*[6] security model whenever it is possible. This principle is even more important in heterogeneous systems such as hybrid clouds, where both applications and resources can represent treats to each other.

9.5.2 Aneka Hybrid Cloud Architecture

The Resource Provisioning Framework represents the foundation on top of which Aneka-based hybrid clouds are implemented. In this section we will introduce the components that compose this framework and briefly describe their interactions.

The basic idea behind the Resource Provisioning Framework is depicted in Figure 9.4. The resource provisioning infrastructure is represented by a collection of resource pools that provide access to resource providers, whether they are external or internal, and managed uniformly through a specific component called a resource pool manager. A detailed description of the components follows:

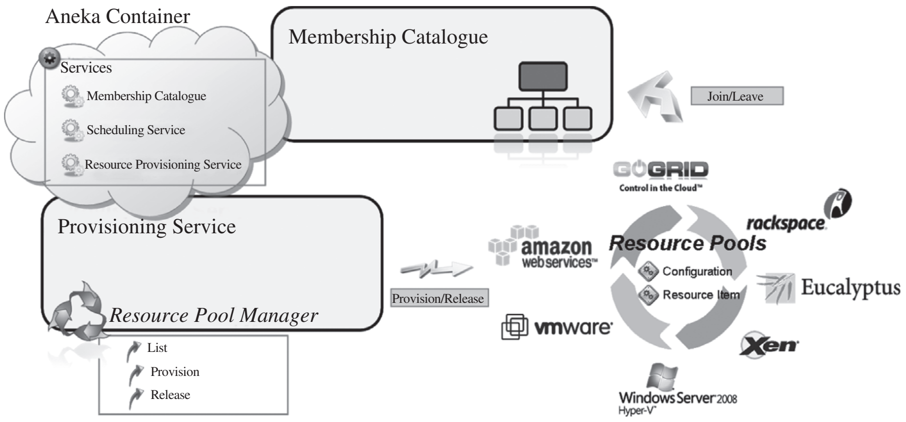

FIGURE 9.4. System architecture of the Aneka Resource Provisioning Framework.

[6]Defense in depth is an information assurance (IA) strategy in which multiple layers of defense are placed throughout an information technology (IT) system. More information is available at the following link: http://www.nsa.gov/ia/_files/support/defenseindepth.pdf (accessed December 2009).

- *Resource Provisioning Service*. This is an Aneka-specific service that implements the service interface and wraps the resource pool manager, thus allowing its integration within the Aneka container.
- *Resource Pool Manager*. This manages all the registered resource pools and decides how to allocate resources from those pools. The resource pool manager provides a uniform interface for requesting additional resources from any private or public provider and hides the complexity of managing multiple pools to the Resource Provisioning Service.
- *Resource Pool*. This is a container of virtual resources that mostly come from the same resource provider. A resource pool is in charge of managing the virtual resources it contains and eventually releasing them when they are no longer in use. Since each vendor exposes its own specific interfaces, the resource pool (a) encapsulates the specific implementation of the communication protocol required to interact with it and (b) provides the pool manager with a unified interface for acquiring, terminating, and monitoring virtual resources.

The request for additional resources is generally triggered by a scheduler that detects that the current capacity is not sufficient to satisfy the expected quality of services ensured for specific applications. In this case a provisioning request is made to the Resource Provisioning Service. According to specific policies, the pool manager determines the pool instance(s) that will be used to provision resources and will forward the request to the selected pools. Each resource pool will translate the forwarded request by using the specific protocols required by the external provider and provision the resources. Once the requests are successfully processed, the requested number of virtual resources will join the Aneka cloud by registering themselves with the Membership Catalogue Service, which keeps track of all the nodes currently connected to the cloud. Once joined the cloud the provisioned resources are managed like any other node.

A release request is triggered by the scheduling service when provisioned resources are no longer in use. Such a request is then forwarded to the interested resources pool (with a process similar to the one described in the previous paragraph) that will take care of terminating the resources when more appropriate. A general guideline for pool implementation is to keep provisioned resources active in a local pool until their lease time expires. By doing this, if a new request arrives within this interval, it can be served without leasing additional resources from the public infrastructure. Once a virtual instance is terminated, the Membership Catalogue Service will detect a disconnection of the corresponding node and update its registry accordingly.

It can be noticed that the interaction flow previously described is completely independent from the specific resource provider that will be integrated into the system. In order to satisfy such a requirement, modularity and well-designed interfaces between components are very important. The current design, implemented in Aneka, maintains the specific implementation details within the *ResourcePool* implementation, and resource pools can be dynamically configured and added by using the dependency injection techniques, which are already implemented for configuring the services hosted in the container. The current implementation of Aneka allows customizing the Resource Provisioning Infrastructure by specifying the following elements:

- *Resource Provisioning Service*. The default implementation provides a lightweight component that generally forwards the requests to the resource Pool Manager. A possible extension of the system can be the implementation of a distributed resource provisioning service that can operate at this level or at the Resource Pool Manager level.
- *Resource Pool Manager*. The default implementation provides the basic management features required for resource and provisioning request forwarding.
- *Resource Pools*. The Resource Pool Manager exposes a collection of resource pools that can be used. It is possible to add any implementation that is compliant to the interface contract exposed by the Aneka provisioning API, thus adding a heterogeneous open-ended set of external providers to the cloud.
- *Provisioning Policy*. Scheduling services can be customized with resource provisioning aware algorithms that can perform scheduling of applications by taking into account the required QoS.

The architecture of the Resource Provisioning Framework shares some features with other IaaS implementations featuring configurable software containers, such as OpenNebula [19] and Nimbus [22]. OpenNebula uses the concept of *cloud drivers* in order to abstract the external resource providers and provides a pluggable scheduling engine that supports the integration with advanced schedulers such Haizea [20] and others. Nimbus provides a plethora of extension points into its programming API, and among these there are hooks for scheduling and resource management and the remote management (RM) API. The first ones control when and where a virtual machine will run, while the RM API act as unified interface to Infrastructure as a Service (IaaS) implementations such as Amazon EC2 and OpenNebula. By providing a specific implementation of RM API, it is possible to integrate other cloud providers.

In the next paragraph, we will detail the implementation of the Amazon EC2 resource pool to provide a practical example of a resource pool implementation.

9.5.3 Use Case—The Amazon EC2 Resource Pool

Amazon EC2 is one of the most popular cloud resource providers. At the time of writing it is listed among the top 10 companies providing cloud computing services.[7] It provides a Web service interface for accessing, managing, and controlling virtual machine instances. The Web-service-based interface simplifies the integration of Amazon EC2 with any application. This is the case of Aneka, for which a simple Web service client has been developed to allow the interaction with EC2. In order to interact with Amazon EC2, several parameters are required:

- *User Identity*. This represents the account information used to authenticate with Amazon EC2. The identity is constituted by a pair of encrypted keys that are the access key and the secret key. These keys can be obtained from the Amazon Web services portal once the user has signed in, and they are required to perform any operation that involves Web service access.
- *Resource Identity*. The resource identity is the identifier of a public or a private Amazon Machine Image (AMI) that is used as template from which to create virtual machine instances.
- *Resource Capacity*. This specifies the different type of instance that will be deployed by Amazon EC2. Instance types vary according to the number of cores, the amount of memory, and other settings that affect the performance of the virtual machine instance. Several types of images are available, those commonly used are: *small*, *medium*, and *large*. The capacity of each type of resource has been predefined by Amazon and is charged differently.

This information is maintained in the *EC2ResourcePoolConfiguration* class and need to be provided by the administrator in order to configure the pool. Hence, the implementation of *EC2ResourcePool* is forwarding the request of the pool manager to EC2 by using the Web service client and the configuration information previously described. It then stores the metadata of each active virtual instance for further use.

In order to utilize at best the virtual machine instances provisioned from EC2, the pool implements a cost-effective optimization strategy. According to the current business model of Amazon, a virtual machine instance is charged by using one-hour time blocks. This means that if a virtual machine instance is used for 30 minutes, the customer is still charged for one hour of usage. In order to provide a good service to applications with a smaller granularity in terms of execution times, the *EC2ResourcePool* class implements a local cache that keeps track of the released instances whose time block is not expired yet. These instances will be reused instead of activating new instances from Amazon.

With the cost-effective optimization strategy, the pool is able to minimize the cost of provisioning resources from Amazon cloud and, at the same time, achieve high utilization of each provisioned resource.

[7]Source: http://www.networkworld.com/supp/2009/ndc3/051809-cloud-companies-to-watch.html (accessed December 2009). A more recent review ranked Amazon still in the top ten (Source: http://searchcloudcomputing.techtarget.com/generic/0,295582,sid201_gci1381115,00.html#slideshow)

9.5.4 Implementation Steps for Aneka Resource Provisioning Service

The resource provisioning service is a customized service which will be used to enable cloud bursting by Aneka at runtime. Figure 9.5 demonstrates one of the application scenarios that utilize resource provisioning to dynamically provision virtual machines from Amazon EC2 cloud.

The general steps of resource provisioning on demand in Aneka are the following:

- The application submits its tasks to the scheduling service, which, in turns, adds the tasks into the scheduling queue.
- The scheduling algorithm finds an appropriate match between a task and a resource. If the algorithm could not find enough resources for serving all the tasks, it requests extra resources from the scheduling service.
- The scheduling service will send a ResourceProvisionMessage to provision service and will ask provision service to get X number of resources as determined by the scheduling algorithm.

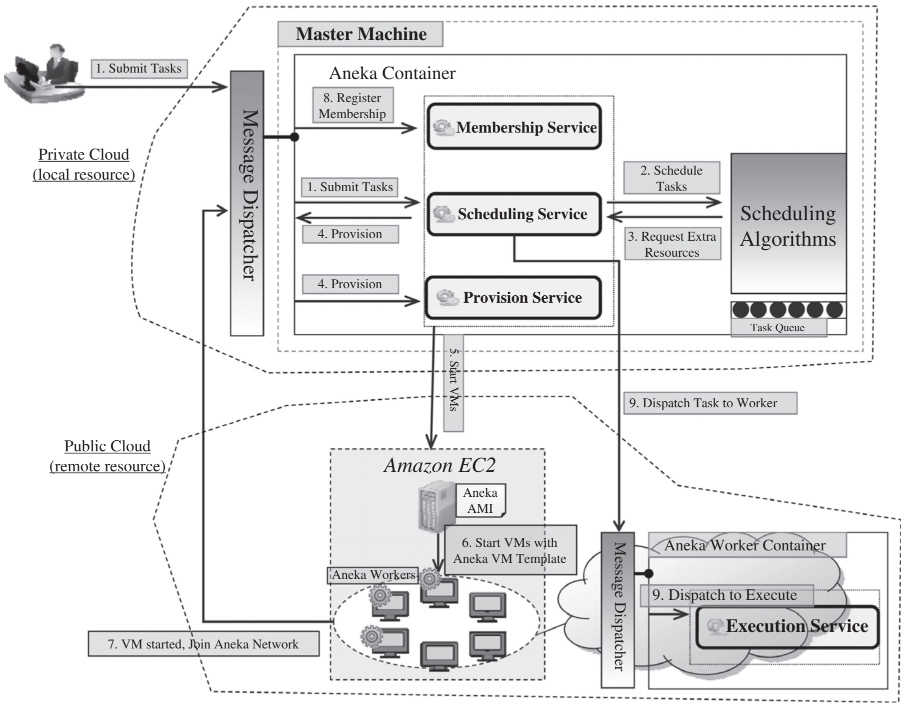

FIGURE 9.5. Aneka resource provisioning (cloud bursting) over Amazon EC2.

- Upon receiving the provision message, the provision service will delegate the provision request to a component called resource pool manager, which is responsible for managing various resource pools. A resource pool is a logical view of a cloud resource provider, where the virtual machines can be provisioned at runtime. Aneka resource provisioning supports multiple resource pools such as Amazon EC2 pool and Citrix Xen server pool.
- The resource pool manager knows how to communicate with each pool and will provision the requested resources on demand. Based on the requests from the provision service, the pool manager starts X virtual machines by utilizing the predefined virtual machine template already configured to run Aneka containers.
- A worker instance of Aneka will be configured and running once a virtual resource is started. All the work instances will then connect to the Aneka master machine and will register themselves with Aneka membership service.
- The scheduling algorithm will be notified by the membership service once those work instances join the network, and it will start allocating pending tasks to them immediately.
- Once the application is completed, all the provisioned resources will be released by the provision service to reduce the cost of renting the virtual machine.

9.6 VISIONARY THOUGHTS FOR PRACTITIONERS

The research on the integration of public and private clouds is still at its early stage. Even though the adoption of cloud computing technologies is still growing, delivering IT services via the cloud will be the norm in future. The key areas of interest that need to be explored include security standardization; pricing models; and management and scheduling policies for heterogeneous environments. At the time of writing, only limited research has been carried out in these fields.

As briefly addressed in the introduction, security is one of the major concerns in hybrid clouds. While private clouds significantly reduce the security risks concerned by retaining sensitive information within corporate boundaries, in the case of hybrid clouds the workload that is delegated to the public portion of the infrastructure is subject to the same security risks that are prevalent in public clouds. In this sense, workload partitioning and classification can help in reducing the security risks for sensitive data. Keeping sensitive operations within the boundaries of the private part of the infrastructure and ensuring that the information flow in the cloud is kept under control is a naïve and probably often limited solution. The major issues that need to be addressed are the following: security of virtual execution environments (either hypervisors or managed runtime environments for PaaS implementations), data retention, possibility of massive outages, provider trust, and also jurisdiction issues that can break the confidentiality of data. These issues become even more crucial in the case of hybrid clouds because of the dynamic nature of the way in which public resources are integrated into the system. Currently, the security measures and tools adopted for traditional distributed systems are used. Cloud computing brings not only challenges for security, but also advantages. Cloud service providers can make sensible investments on the security infrastructure and provide more secured environments than those provided by small enterprises. Moreover, a cloud's virtual dynamic infrastructure makes it possible to achieve better fault tolerance and reliability, greater resiliency to failure, rapid reconstruction of services, and a low-cost approach to disaster recovery.

The lack of standardization is another important area that has to be covered. Currently, each vendor publishes their own interfaces, and there is no common agreement on a standard for exposing such services. This condition limits the adoption of inter-cloud services on a global scale. As discussed in this chapter, in order to integrate IaaS solutions from different vendors it is necessary to implement ad hoc connectors. The lack of standardization covers not only the programming and management interface, but also the use of abstract representations for virtual images and active instances. An effort in this direction is the Open Virtualization Format (OVF) [25], an open standard for packaging and distributing virtual appliances or more generally software to be run in virtual machines. However, even if endorsed by the major representative companies in the field (Microsoft, IBM, Dell, HP, VMWare, and XenSource) and released as a preliminary standard by the Distributed Management Task Force, the OVF specification only captures the static representation of a virtual instance; it is mostly used as a canonical way of distributing virtual machine images. Many vendors and implementations simply use OVF as an import format and convert it into their specific runtime format when running the image. Additional effort has to be spent on defining a common method to represent live instances of applications and in providing a standard approach to customizing these instances during startup. Research in this area will be necessary to completely eliminate vendor lock-in.[8] In addition, when building a hybrid cloud based on legacy hardware and virtual public infrastructure, additional compatibility issues arise due to the heterogeneity of the runtime environments: almost all the hypervisors support the x86 machine model, which could constitute a technology barrier in the seamless transition from private environments to public ones. Finally, as discussed by Keahey et al. [26], there is a need for providing (a) a standardized way for describing and comparing the quality of service (QoS) offerings of different cloud services providers and (b) a standardized approach to benchmark those services. These

[8]In cloud computing, vendor lock-in relates to the condition in which a large installed base of a customer is maintained within the virtual infrastructure of one vendor who does not disclose the internals of their system, thus preventing the possibility of the customer moving their installed base to another provider without considerable costs.

are all areas that have to be explored in order to take advantage of heterogeneous clouds, which, due to their dynamic nature, require automatic methods for optimizing and monitoring the publicly provisioned services. An important step in providing a standardization path and to foster the adoption of cloud computing is the Open Cloud Manifesto,[9] which provides a starting point for the promotion of open clouds characterized by interoperability between providers and true scalability for applications.

Since the integration of external resources comes with a price, it is interesting to study how to optimize the usage of such resources. Currently, resources are priced in time blocks, and often their granularity does not meet the needs of enterprises. Virtual resource pooling, as provided by Aneka, is an initial step in closing this gap, but new strategies for optimizing the usage of external provisioned resources can be devised. For example, intelligent policies that can predict when to release a resource by relying on the statistics of the workload can be investigated. Other policies could identify the optimal number of resources to provision according to the application needs, the budget allocated for the execution of the application, and the workload. Research in this direction will become even more consistent when different pricing models will be introduced by cloud providers. In this future scenario, the introduction of a market place for brokering cloud resources and services will definitely give more opportunities to fully realize the vision of cloud computing. Each vendor will be able to advertise their services and customers will have more options to choose from, eventually by relying on meta-brokering services. Once realized, these opportunities will make the accessibility of cloud computing technology more natural and at a fairer price, thus simplifying the integration of existing computing infrastructure owned within the premises.

We believe that one of the major areas of interest in the next few years for what concerns the implementation and the deployment of hybrid clouds will be the scheduling of applications and the provisioning of resources for these applications. In particular, due to the heterogeneous nature of hybrid clouds, additional coordination between the private and the public service management becomes fundamental. Hence, cloud schedulers will necessarily be integrated with different aspects such as federate policy management tools, seamless hybrid integration, federated security, information asset management, coordinated provisioning control, and unified monitoring.

9.7 SUMMARY AND CONCLUSIONS

In this chapter we have presented the characteristics of hybrid clouds and discussed their implementation and deployment by using Aneka. Hybrid clouds emerge when an existing private infrastructure grows into a virtual public

[9]The Open Cloud Manifesto is available at: http://www.opencloudmanifesto.org (Accessed, December, 2009).

infrastructure in order to handle its workload. We envision that this specific scenario will be the most common in the future because hybrid clouds can overcome specific disadvantages of both public and private clouds. They can scale on demand and leverage the horse power of third-party data centers and maintain the elaboration of sensitive information within the premises of the enterprise. Different solutions are available for implementing hybrid clouds; the most relevant to the discussed scenario are IaaS and PaaS implementaions. Among these we presented the solution proposed by Aneka, which is an implementation of the PaaS model for cloud computing. The Aneka container—the basic building block of Aneka clouds—can be easily deployed on different hardware: a desktop PC, a workstation, a server, a cluster, and even a virtual machine. This flexibility allows the quick setup of heterogeneous execution environments on top of which distributed applications can run transparently. Such a feature constitutes a key element for integrating computing resources from external providers into the private infrastructure.

In order to support this scenario, we first highlighted which are the desired features of a reference model for hybrid cloud and then presented how these characteristics are reflected into Aneka. Three major components compose the provisioning framework: Resource Provisioning Service, Resource Pool Manager, and Resource Pools. A fundamental role is played by resource pools, which represent collections of computing nodes belonging to the same domain. The abstraction provided by resource pools makes it possible to integrate and leverage either public or private clouds uniformly. As a proof of concept of the presented solution, we discussed a use case scenario that involves the creation of a hybrid cloud composed by a set of workstations that has been augmented by initially provisioning resources from a cluster managed by a Xen Hypervisor and then by leveraging a public virtual infrastructure such as Amazon EC2.

As a conclusion to the chapter, we introduced and discussed the future directions of the research in hybrid clouds, we highlighted the major challenges that have to be faced in order to promote a wider adoption of hybrid clouds, and provided some insights into the initial efforts taken toward this direction.

In the future we aim to extend the resource provisioning framework by providing more advanced scheduling techniques for heterogeneous environments. Additionally, we would like to extend the number of resource provider actually supported. Currently, we are developing support for VMWare, Eucalyptus, and InterGrid [27].

ACKNOWLEDGMENTS

We would like to thank Dileban Karunamoorthy and Mukaddim Pathan for their useful insights on structuring the content of the chapter and their suggestions for improving the presentation of the content.

REFERENCES

1. M. Armbrust, A. Fox, R. Griffith, A.D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoika, and M. Zaharia, Above the clouds: A Berkeley view of cloud computing, Technical Report, UC Berkeley Reliable Adaptive Distributed Systems Laboratory, February 2009.
2. J. Staten, S. Yates, J. Ryme, F. Gillett, and L. E. Nelson, Deliver Cloud Benefits Inside Your Walls: Economic and Self-Service Gains Are Within Reach, *Forrester Research Inc.*, April 2009.
3. C. Vecchiola, X. Chu, and R. Buyya, *Aneka: A software platform for .NET cloud computing*, in *High Performance & Large Scale Computing*, W. Gentzsch, L. Grandinetti, and G. Joubert (eds.), IOS Press, Amsterdam, 2009.
4. Amazon Web Services, http://aws.amazon.com (accessed October 21, 2010).
5. Amazon Elastic Compute Cloud, http://aws.amazon.com/ec2/ (accessed October 21, 2010).
6. Amazon Simple Storage Service, http://aws.amazon.com/s3/ (accessed October 21, 2010).
7. GoGrid, http://www.gogrid.com (accessed October 21, 2010).
8. 3tera AppLogic, http://www.3tera.com/AppLogic/ (accessed October 21, 2010).
9. Microsoft Azure, http://www.microsoft.com/windowsazure/ (accessed October 21, 2010).
10. Google AppEngine, http://code.google.com/appengine/ (accessed October 21, 2010).
11. Xen Hypervisor, http://xen.org/ (accessed October 21, 2010).
12. Kernel-based Virtual Machine, http://www.linux-kvm.org/page/Main_Page (accessed October 21, 2010).
13. VMWare Business Infrastructure Virtualization, http://www.vmware.com/ (accessed October 21, 2010).
14. VMWare vCloud, http://www.vmware.com/solutions/cloud-computing/ (accessed October 21, 2010).
15. D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, and D. Zagorodnov, The Eucalyptus open-source cloud computing system, in *Proceedings of 9th IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid 2009)*, Shanghai, China, May 2009.
16. Datasynapse, http://www.datasynapse.com (accessed October 21, 2010).
17. Elastra−Manage IT complexity, http://www.elastra.com (accessed October 21, 2010).
18. Zimory, http://www.zimory.com (accessed October 21, 2010).
19. I. Llorente, R. Moreno-Vozmediano, and R. Montero, Cloud computing for on-demand grid resource provisioning, in *Advances in Parallel Computing*, IOS Press, Amsterdam, 2009, p. 18.
20. B. Sotomayor, K. Keahey, and I. Foster, Combining batch execution and leasing using virtual machines, in *HPDC '08: Proceedings of the 17th International Symposium on High Performance Distributed Computing*, ACM, 2008, pp. 87−96.

21. S. Venugopal, J. Broberg, and R. Buyya, OpenPEX: An open provisioning and execution systems for virtual machines, in *Proceedings of the 17th International Conference on Advanced Computing and Communications (ADCOM 2009)*, Bengaluru, India, December 14–18, 2009.

22. K. Keahey, I. Foster, T. Freeman, and X. Zhang, Virtual workspaces: Achieving quality of service and quality of life on the grid, *Scientific Programming*, **13**(4): 265–276, 2005.

23. J. Jagger, N. Perry, and P. Sestoft, *C# Annotated Standard*, Morgan Kaufmann, San Francisco, 2007.

24. J.S. Miller, S. Ragsdale, *The Common Language Infrastructure Annotated Standard*, Addison-Wesley, Reading, MA, 2004.

25. Open Virtualization Format Specification, Distributed Management Task Force, http://www.dmtf.org/sites/default/files/standards/documents/DSP0243_1.1.0.pdf (accessed October 21, 2010)

26. K. Keahey, M. Tsugawa, A. Matsunaga, and J. A. B. Fortés, Sky computing, *IEEE Internet Computing*, **13**(5):43–51, 2009.

27. A. Di Costanzo, M. Dias de Assunção, and R. Buyya, Harnessing cloud technologies for a virtualized distributed computing infrastructure, *IEEE Internet Computing*, **13**(5):14–22, 2009.