The Cloud Interoperability Challenge

Rajiv Ranjan
Commonwealth Scientific and Industrial Research Organization, Australia

Welcome to the second installment of "Blue Skies." This department, which will appear four times a year, will provide in-depth analyses of the most recent and influential research related to cloud technologies and innovations. In this issue, I'll overview research issues and directions related to cloud interoperability. In the cloud computing landscape, "cloud interoperability" typically refers to the ability to seamlessly deploy, migrate, and manage application workloads across heterogeneous hardware and software resources provided by multiple datacenter cloud providers (such as Amazon and GoGrid).

Cloud Computing Paradigm

In the cloud computing model, users access services according to their requirements, without knowing where the services are hosted or how they're delivered.[1,2] An increasing number of IT vendors (such as Amazon, GoGrid, and Rackspace) promise to offer information and communication technology (ICT) resources such as hardware (CPU, GPUs, storage, and network), software (databases, stream-processing systems, and data-mining frameworks), and applications (email, video on demand, and social networking). These services are referred to as *infrastructure as a service* (IaaS), *platform as a service* (PaaS), and *software as a service* (SaaS). Figure 1 shows the layered architecture of the cloud computing model. Cloud resources are hosted in large datacenters, often referred to as *data farms*, operated by companies such as Amazon, Apple, Google, and Microsoft. Having the flexibility to rent ICT resources on demand to avoid upfront investment has attracted many enterprises that now exploit cloud computing to deliver their application services.

The proliferation of cloud computing has revolutionized hosting and delivery of Internet-based application services. In the standard cloud application deployment approach, an application is architected to be deployed and managed over a single datacenter (for example, Amazon or GoGrid). Such an approach has several shortcomings. Datacenter failure can leave thousands of application users without access to essential (and in some cases paid) services. Moreover, exploiting a single datacenter makes it extremely difficult to exploit location-based placement of data and processing driven by geolocation of application users.

Interconnecting multiple cloud-based datacenters lets every application owner improve overall quality of service (QoS), reliability, and flexibility of their applications. However, with the almost monthly launching of new cloud services and capabilities by both large (such as Amazon Web Services and

 2325-6095/14/$31.00 © 2014 IEEE

FIGURE 1. Layered architecture of cloud computing (adapted from earlier work.[2])

Microsoft Azure) and small (such as Rackspace and Ninefold) companies, decision makers (application developers, CIOs, and so on) will likely be overwhelmed by the available choices. Decision making is further complicated by interoperability challenges that exist across multiple cloud providers[3–6]:

- heterogeneous virtualization technologies;
- nonstandardized service descriptions, pricing, and service-level agreement (SLA) definitions;
- heterogeneous APIs; and
- nonstandardized technologies for authentications and authorizations.

One side effect of the lack of interoperability among cloud providers is vendor lock-in, which also means lack of ability to migrate application components and associated workload from cloud provider A to cloud provider B.

This article investigates the technical challenges from the user-to-multiple-cloud interoperability perspective. In this cloud integration model, application owners are responsible for provisioning their application components over resources belonging to multiple providers. In this scenario, owners typically implement or use an application provisioner software program (such as RightScale [www.rightcale.com] or CloudSwitch [https://home.cloudswitch.com]), which distributes application components across multiple resource providers to meet the SLAs in an optimal way.

This article doesn't cover the challenges inherent in the cloud-to-cloud interoperability perspective—that is, the cloud integration model consisting of multiple cloud providers that cooperatively integrate (via federated middleware software) their datacenter resources to support seamless migration of application workload and components across each other. An interesting discussion on interoperability appears elsewhere.[7]

Overcoming these interoperability challenges will allow us to establish a resource-sharing environment consisting of multiple cloud datacenters (multiclouds) that could belong to different providers. In a federated organization, every application owner will be able to deploy its application components across multiple datacenters, thereby improving their ability to handle disasters (such as a datacenter power failure); optimize the cost of using cloud resources via dynamic migration of application components to cheaper datacenters without worrying about low-level details such as the target datacenter's

virtualization technologies or programming interface; and avoid vendor lock-in because they'll be able to seamlessly migrate their applications. Recent research projects that address some of the technical challenges related to cloud interoperability for establishing multicloud environments include Optimis,[8] Contrail,[9] Mosaic,[10] and Reservoir.[11]

User-to-Multiple-Cloud Interoperability Challenges

Many of these issues lack easy solutions, mainly because of the continued lack of agreement among major cloud providers on standardized approaches to architecting and managing their datacenter resources. Hence, before considering migrating applications to a combination of public and private clouds, application architects and CIOs should seriously consider the following challenges.

Heterogeneous Virtualization Technologies

Virtualization technology is at the heart of any public or private cloud datacenter.[12] It allows providers to get more out of physical resource by allowing multiple instances of virtual cloud resources to run concurrently. Each virtual resource believes it has its own share of hardware resource. Virtualization isolates the hardware resources, thereby enabling fault-tolerant and isolated security context behavior. It allows more efficient utilization by providing the flexibility, agility, and scalability needed for a physical resource to support multitenancy.

The private cloud datacenter area is dominated by vendors such as VMware, which manages datacenter resources using ESX virtualization technology and vCloud API access to the hypervisor. Public cloud providers, such as Amazon and Microsoft, have adopted KVM, Hyper-V, and Xen virtualization technologies for managing their datacenters. In the public cloud case, an application owner who wants to migrate a software resource (such as a webserver) at this layer from a VMWare-based private datacenter to a Xen-based Amazon EC2 datacenter has to customize its configurations (the operating system, management tools, virtualization format, virtual machine [VM] configuration, storage system, and networking environment) to fit the target environment.

Standardization efforts in this area include the Open Virtualization Forum (www.dmtf.org/standards/ovf). OVF describes an open, secure, portable, efficient, and generic format for packaging and distributing software resources at the PaaS and IaaS layers (see Figure 1). Although many cloud vendors (Microsoft, IBM, Dell, HP, VMware, and Xen, among others) have supported the OVF initiative, its popularity and adoption rate are still uncertain. To overcome this heterogeneity in virtualization format and technology, providers such as Amazon, RightScale, and CloudSwitch offer custom scripts that can be used to manually port a software resource from one virtualization format to another. Some datacenter vendors, such as HP and Rackspace, strongly recommend OpenStack as the virtualization technology for solving public-private cloud interoperability challenges. However, it seems unlikely that other leading vendors, such as Amazon, Microsoft, and VMware, will adopt OpenStack in the near future.

Nonstandardized Service Description, Pricing, and SLA Definition

The cloud computing landscape offers many diverse options for hardware and software resources at the IaaS and PaaS layers. Hence, application owners face a daunting task when trying to select cloud resources that meet their QoS constraints (minimized storage costs, minimized data processing costs, and the like). According to Burstorm, there are more than 426 IaaS resource providers with deployments in more than 11,072 locations.[13] Even within a particular provider, there are different variations of cloud services.[4] For example, Amazon alone has 674 offerings differentiated by price, QoS features, and locations. Moreover, each quarter they add about four new resources, new business models (price and terms), and sometimes even new locations. To select the best mix of resource offerings from an abundance of possibilities, application owners must simultaneously consider and optimize complex dependencies and heterogeneous sets of QoS criteria (price, features, location, and so on).

When comparing infrastructure services in cloud computing, an application owner should read the provider's documentation to determine which services are most suitable for hosting an application. However, cloud providers' use of nonstandardized naming terminologies makes it difficult to make accurate comparisons. For example, Amazon refers to its compute services as EC2 Compute Unit, whereas GoGrid refers to its same unit as cloud servers. Furthermore, cloud providers typically publish their service description, pricing policies, and SLA rules on their websites. Because providers might update this information without notifying users, it can be difficult to manually obtain service configurations from cloud providers' websites and documentation (the only sources of information).

SLA definitions vary considerably among the major public cloud providers. Amazon promises that its EC2 service will be available with an annual uptime percentage of at least 99.95 percent during the service year, compared with the 99.99 percent typical of private

enterprise datacenters. A clause in the Rackspace SLA definition states that its datacenter infrastructure will be available 100 percent of the time in a month (www.rackspace.com/managed_hosting/support/servicelevels/managedsla), excluding scheduled maintenance. Similarly, public providers handle SLA violations differently. Providers such as 3Tera automatically detect SLA violations and credit the cost directly to the application owner's account. In contrast, Amazon and Rackspace expect the application owner to prove the SLA violations before awarding outage credits.

Heterogeneous APIs

To improve resilience, an intuitive solution is to deploy applications across multiple public and private IaaS providers. Unfortunately, few of the existing providers are compatible. They tend to have proprietary APIs, which are not explicitly designed for cross-cloud interoperability. Tackling such heterogeneities in API implementations requires standardization across layers of the cloud resource stack. Recent developments, including Simple Cloud (www.ibm.com/developerworks/opensource/library/os-simplecloud), Delta Cloud (http://deltacloud.apache.org), jclouds (http://jclouds.apache.org), and Dasein Cloud, simplify this task by implementing a single API that abstracts APIs related to multiple clouds such as AWS EC2 and GoGrid. The research challenge is to develop extensible and interoperable orchestration program modules for resource selection, deployment, monitoring, and control that can operate with multiple cloud resources. Fundamental cloud resources such as CPU, appliances, and storage can be orchestrated via SOAP/RESTful APIs. However, dynamically orchestrating monitoring, data replication, load-balancing, and autoscaling software resources to handle surges in application traffic via an API still isn't viable within a public or private datacenter. Making these APIs operate across multiple private and public datacenters remains an open and much harder research problem.

Nonstandardized Technologies for Authentication and Authorization

There is subtle difference between how private and public datacenter providers verify that users of a datacenter resource both are who they claim to be (authentication) and are allowed to be where they want to go, or to have the information they want to have (authorization). For example, VMware's vSphere supports user account and credential-based authentications. The current credential consists of a password, but vSphere can support certificates, such as X.509 certificates. Authenticated users can then access the datacenter resources they're authorized to use as per access control rules.

> Public cloud providers implement complex, multilayered user authentication and authorization technologies.

Public cloud providers implement complex, multilayered user authentication and authorization technologies. For example, Amazon's AWS Identity and Access Management (http://aws.amazon.com/iam) includes:

- email and password for authenticating via a Web portal service;
- access key and secret access key, which is required when sending query and REST-based requests to specific Amazon resources (such as S3 and EC2);
- a Secure Socket Shell (SSH) key pair for accessing live EC2 CPU (VM) resource instances using Windows' remote desktop utilities;
- an account ID—that is, a unique ID assigned to each AWS account and is also used to share resources across other AWS accounts;
- an X.509 certificate and private key, which is used by the Amazon's command line tools and SOAP API for sending requests to AWS services (for example AutoScaler and LoadBalancer); and
- security groups—that is, rules pertaining to opening communication ports and IP addresses that are allowed to send messages to a CPU resource instance.

Notably, GoGrid supports two types of security mechanisms based on mode of interaction with the infrastructure. In physical interaction mode (through customer portals, remote SSH, and remote desktop connections), cloud users must go through a role-based access-control mechanism that authorizes users based on their name and designated passwords. For programmatically accessing the GoGrid resources and services,

one has to generate an API key. With every API key, a shared secret password and a role are assigned. All calls to GoGrid APIs should be transmitted as encrypted HTTPS messages.

To solve these interoperability issues, several efforts to develop standardized technology are underway. The OpenID standard enables servers to authenticate users in a decentralized manner. A user who creates an account with an OpenID identity provider can use that account to authenticate with any Web resource that has implemented OpenID authentication. OpenID has gained some acceptance by public (Google App Engine and Azure) and private (OpenStack) datacenter providers. Similarly, providers such as Amazon and Azure support Web Services Security (WS-Security) for message authentication. The most recent release in the OpenID family is OpenID Connect (http://openid.net/connect), an interoperable authentication protocol based on the OAuth 2.0 family of specifications (http://oauth.net). OpenID Connect lets users authenticate across websites and apps without having to manage and own password files. However, cloud providers have yet to truly deploy and test them. Although some providers have adopted technologies such as OpenID and WS-Security, many efforts are still needed to develop consensus among public and private datacenter providers regarding the adoption of standardized authentication and authorization technologies. This, in turn, continues to worsen the cloud interoperability problem.

The critical challenge is to develop standardization solutions for each of the technical issues I've noted, which to a large extent also require horizontal and vertical interoperability in the cloud stack. However, one of the main problems in the push for cloud standardization is that too many efforts are underway. Hence, coordinating different standardization efforts will pose a challenge for the future.

References

1. L. Wang et al., eds., *Cloud Computing: Methodology, Systems, and Applications*, CRC Press, 2011.
2. R. Ranjan et al., "Cloud Resource Orchestration Programming: Overview, Issues, and Directions," *IEEE Internet Computing*, to appear.
3. M. Menzel et al., "A Configuration Crawler for Virtual Appliances in Compute Clouds," *Proc. 2013 IEEE Int'l Conf. Cloud Eng.* (IC2E 13), 2013, pp. 201–209.
4. M. Zhang et al., "A Declarative Recommender System for Cloud Infrastructure Services Selection," *Proc. 9th Int'l Conf. Economics of Grids, Clouds, Systems, and Services* (GECON 2012), 2012, pp. 102–113.
5. G.A. Lewis, "The Role of Standards in Cloud Computing Interoperability," tech. note CMU/SEI-2012-TN-012, Software Eng. Inst., Carnegie Mellon Univ., 2013.
6. M. Laverick, *Chapter 4: Hybrid Clouds and the Interoperability Challenge*, 2nd ed., TechTarget, 2014; http://searchcloudcomputing.bitpipe.com/detail/RES/1369944316_856.html.
7. D. Bernstein, "Types of Intercloud: Federation and Multi-Cloud," blog, 12 Apr. 2013; http://cloudstrategypartners.blogspot.com.au/2013/04/intercloud-not-all-same-federation.html.
8. A. Juan Ferrer, et al., "OPTIMIS: A Holistic Approach to Cloud Service Provisioning," *Future Generation Computer Systems*, vol. 28, No. 1, 2012, pp. 66–77.
9. R. Carlini et al., "Cloud Federations in Contrail," *Proc. Euro-Par 2011: Parallel Processing Workshops*, vol. 7155, 2012, pp. 159–168.
10. D. Petcu et al., "Architecturing a Sky Computing Platform," *Proc. 2010 Int'l Conf. Towards a Service-based Internet* (ServiceWave 10), 2010, pp. 1–13.
11. L. Rodero-Merino et al., "From Infrastructure Delivery to Service Management in Clouds," *Future Generation Computer Systems*, vol. 26, no. 8, 2010, pp. 1226–1240.
12. P. Barham et al., "Xen and the Art of Virtualization," *Proc. 19th ACM Symp. Operating Systems Principles*, 2003, pp. 164–177.
13. T.C.K. Chou, "Think Vertical," blog, 5 Sept. 2013, www.layeredtech.com/blog/think-vertical.

RAJIV RANJAN *is a senior research scientist, Julius Fellow, and project leader at the Commonwealth Scientific and Industrial Research Organisation (CSIRO), Australia. His research interests include cloud computing, content-delivery networks, and big data analytics for Internet of Things (IoT) and multimedia applications. Ranjan has a PhD in computer science and software engineering from the University of Melbourne. A full biography is available on page XX.*