The Reservoir model and architecture for open federated cloud computing

B. Rochwerger
D. Breitgand
E. Levy
A. Galis
K. Nagin
I. M. Llorente
R. Montero
Y. Wolfsthal
E. Elmroth
J. Cáceres
M. Ben-Yehuda
W. Emmerich
F. Galán

The emerging cloud-computing paradigm is rapidly gaining momentum as an alternative to traditional IT (information technology). However, contemporary cloud-computing offerings are primarily targeted for Web 2.0-style applications. Only recently have they begun to address the requirements of enterprise solutions, such as support for infrastructure service-level agreements. To address the challenges and deficiencies in the current state of the art, we propose a modular, extensible cloud architecture with intrinsic support for business service management and the federation of clouds. The goal is to facilitate an open, service-based online economy in which resources and services are transparently provisioned and managed across clouds on an on-demand basis at competitive costs with high-quality service. The Reservoir project is motivated by the vision of implementing an architecture that would enable providers of cloud infrastructure to dynamically partner with each other to create a seemingly infinite pool of IT resources while fully preserving their individual autonomy in making technological and business management decisions. To this end, Reservoir could leverage and extend the advantages of virtualization and embed autonomous management in the infrastructure. At the same time, the Reservoir approach aims to achieve a very ambitious goal: creating a foundation for next-generation enterprise-grade cloud computing.

Introduction

In the Web 2.0 era, companies can grow from inception to a massive scale at incredible rates. For example, MySpace acquired 20 million users in two years; Google YouTube** reached the same number of users in just 16 months [1]. However, to leverage this potential rate of growth, companies must properly address critical business decisions related to their service delivery infrastructure.

Current approaches to cloud computing

The emerging cloud-computing paradigm [2], as exemplified by the Amazon Elastic Compute Cloud (EC2) [3], represents a promising conceptual foundation for hosting and deployment of Web-based services while theoretically relieving service providers from the responsibility of provisioning the computational resources needed to support these services. Cloud computing enables individuals or companies with market domain expertise to build and run a software-as-a-service (SaaS) company with minimal effort developing software and without managing any hardware operations. This helps to reduce software complexity and costs, expedite time to market, and enhance the accessibility to services.

With cloud computing, companies can lease infrastructure resources on-demand from a virtually unlimited pool. The pay-as-you-go billing model applies charges for the resources actually used per unit time. This way, a business can optimize its IT (information technology) investment and improve availability and scalability.

0018-8646/09/$5.00 © 2009 IBM

While cloud computing holds huge promise for the future of service computing, the following inherent deficiencies in current offerings can be identified:

Inherently limited scalability of single-provider clouds—Although most infrastructure cloud providers today claim infinite scalability, in reality it is reasonable to assume that even the largest players may start facing scalability problems as the cloud-computing usage rate grows. In the long term, scalability problems may be expected to worsen as cloud providers serve an increasing number of online services, each accessed continuously by massive numbers of global users.

Lack of interoperability among cloud providers—Contemporary cloud technologies have not been designed with interoperability in mind. This results in an inability to scale through business partnerships across cloud-computing providers. In addition, it prevents small and medium-sized cloud-infrastructure providers from entering the cloud-provisioning market. Overall, this stifles competition and locks consumers to a single vendor.

No built-in business service management support—Business service management (BSM) is a management strategy that enables businesses to align their IT management with their high-level business goals. The key aspect of BSM is service-level agreement (SLA) management. Current cloud-computing solutions are not designed to support the BSM practices that are well established in the daily management of enterprise IT departments. As a result, enterprises that want to transform their operations to cloud-based technologies cannot do so incrementally and will likely find such a transformation to be a disruptive step.

We argue that none of these problems, or other major problems such as security and availability, can be remedied by retrofitting existing architectures. On the contrary, these issues must be addressed by designing a cloud-computing architecture from basic principles. In this paper, we propose a reference model and architecture that systematically addresses some of those deficiencies and serves as a potential foundation for delivering IT services as utilities over the Internet.

Reservoir approach

The Reservoir approach is to enable on-demand delivery of IT services at competitive costs without requiring a large capital investment in infrastructure. This approach is inspired by a strong desire to make the delivery of IT services similar to the delivery of common utilities. For example, a common scenario in the electric grid is for one facility to dynamically acquire electricity from a neighboring facility to meet a spike in demand. Just as in other industries in which no single provider serves all customers at all times, the next-generation cloud-computing infrastructure should support a model that enables multiple independent providers to cooperate seamlessly to maximize their mutual benefit.

More specifically, to truly fulfill the promise of cloud computing, there should be technological capabilities to federate disparate data centers, including those owned by separate organizations. Only through federation and interoperability can infrastructure providers take advantage of their aggregated capabilities to provide a seemingly infinite service computing utility. Informally, we refer to the infrastructure that supports this paradigm as a *federated cloud*.

An additional important advantage offered by the federated cloud approach is that it democratizes the supply side of cloud computing by allowing small and medium-sized businesses (SMBs) and new entrants to become cloud providers. This encourages competition and innovation.

One of the aforementioned limiting factors in current cloud-computing offerings is the lack of support for BSM, specifically for business-aligned SLA management. While specific cloud-computing solutions can be enhanced with some aspects of BSM, the provisioning of complex services across a federated network of possibly disparate data centers is a difficult and as yet unsolved problem. A service may be a composition of numerous distributed resources including computing, storage, and network elements. Provisioning such a service consumes physical resources but should not cause an SLA violation of any other running application with a probability larger than some predefined threshold. Because SLAs serve as risk mitigation mechanisms, this threshold represents the risk that a cloud provider and the cloud customer are willing to accept.

With BSM, applications are properly dimensioned, and their nonfunctional characteristics governed by SLAs, such as performance, availability, and security, are ensured with optimal cost through continuous IT optimization. We argue that because of the immense scale envisioned by cloud computing, support for BSM should be maximally automated and embedded into the cloud infrastructure.

The basic principle of the Reservoir model, as presented in this paper, is that each infrastructure provider is an autonomous business with its own business goals. A provider federates with other providers (i.e., other Reservoir sites) based on its own local preferences. The IT management at a specific Reservoir site is fully autonomous and governed by policies that are aligned with the business goals of the site. To optimize this alignment once it is initially provisioned, resources composing a service may be moved to other Reservoir sites based on economic, performance, or availability considerations. Our research addresses those issues and

seeks to minimize the barriers to delivering services as utilities with guaranteed levels of service and proper risk mitigation.

Cloud computing is the latest incarnation of a general-purpose public computing utility. In recent years we have seen many efforts toward delivering computing as a utility—from grid computing [4], which made significant progress in the area of high-performance scientific computing, to attempts at building enterprise-level utilities [5]. However, none of these attempts has materialized into a general-purpose compute utility accessible by anyone at any time from anywhere.

What makes cloud computing different is that industry such trends as the ubiquity of broadband networks, the fast penetration of virtualization technology for x86-processor-based servers [6], and the adoption of SaaS [7] are finally creating an opportunity and a need for a global computing utility. The reluctance to use online services as a replacement for traditional software is lessening; the success of companies such as Salesforce.com proves that with the right set of security warranties and a competitive price, companies are willing to trust even their most valuable data—customer relations—to an online service provider. At the same time, virtualization has made it possible to decouple the functionality of a system as it is captured by the software stack (operating system, middleware, application, configuration, and data) from the physical computational resources on which it executes [8]. This, in turn, enables a new model of online computing: Instead of specially crafted online software, we can now think in terms of general-purpose online virtual machines (VMs) that can perform any computational task. Finally, as virtualization enters the mainstream, the era of a general-purpose compute utility is now within reach.

The specific contributions we present in this paper are the following:

- Delineation of motivation and realistic use cases for enterprise-grade federated cloud computing.
- Definition of a model and an open architecture for federation and the interoperability of autonomous clouds to form a global fabric of resources that can be provided on demand with guaranteed service levels.
- Definition of an open, loosely coupled cloud-computing stack in which each level operates autonomously at a different level of abstraction and interacts with the layers above and below via standardized interfaces.

The remainder of this paper is organized as follows: In the next section we discuss specific use cases and derive requirements for Reservoir, and in the section following that we present the Reservoir federation model and architecture and provide definitions of the concepts used. We then describe the Reservoir architecture in greater detail and provide the rationale for the design choices we propose. That is followed by offering insight into the state of the art for virtualization, grid computing, and BSM, and then we conclude with a summary.

Use cases and requirements analysis

In this section, we first review several similar use cases that involve SAP systems [9]. Because of their complexity, these systems serve as a useful conceptual benchmark for deriving requirements and validating the Reservoir model. Next, we present key design principles and a subset of primary requirements. These requirements reflect the distinctions to be made between Reservoir and current cloud and virtualization offerings.

SAP systems

SAP systems are used for a variety of business applications, such as SAP CRM (customer relationship management) and SAP ERP (enterprise resource planning). For simplicity, we assume that the different SAP systems follow the same architecture described below, but each system is installed and operated as an independent system. A given SAP system consists of generic components that are customized by configuration and components that are custom-coded for a specific installation.

An SAP system is typically a three-tier system (**Figure 1**):

- Requests are handled by the SAP Web dispatcher.
- In the middle tier, there are two types of components: multiple stateful dialog instances (DIs) and a single central instance (CI) that performs central services such as application-level locking, messaging, and registration of DIs. The number of DIs can be changed while the system is running to adapt to load.
- A single database management system (DBMS) and a single storage system serve the SAP system.

As shown in **Figure 2**, the components can be arranged in a variety of configurations, from a minimal configuration in which all components run on a single machine, to larger ones in which there are several DIs, each running on a separate machine, and a separate machine with the CI and the DBMS.

Virtualized data center use case

Consider a data center that uses virtualization technology to consolidate the operation of different types of SAP applications and all their respective environments; for example, test and production. The applications are offered as a service to external customers or alternatively,

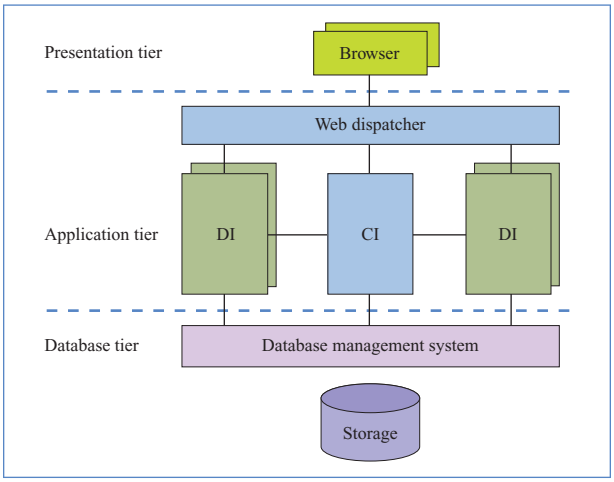

Figure 1

Abstraction of an SAP system (CI: central instance; DI: dialog instance).

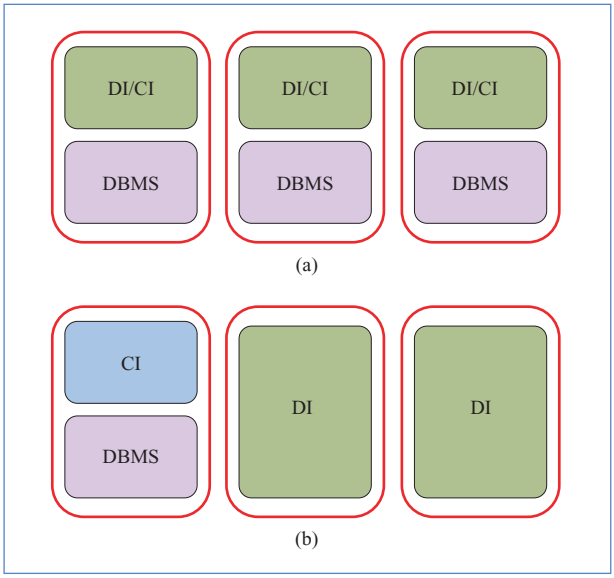

Figure 2

Sample SAP system deployments: (a) multiple SAP systems where each system runs in the same machine (represented as rounded red rectangles); (b) single SAP system where the large components (CI and DBMS) run together on the same machine, and each DI runs on a dedicated machine. For simplicity, it can be assumed that the Web dispatcher is collocated with the CI on both figures.

the data center is operated by the IT department of an enterprise for its internal users, enterprise employees.

A special variation is the case in which the data center serves an on-demand SaaS setup in which customers are external and each customer, or *tenant*, gets the same base version of the application. However, each tenant configures and customizes the application to suit its specific needs. It is reasonable to assume that a tenant in this case is an SMB.

There are several typical aspects of virtualized data centers. The infrastructure provider must manage the life cycle of the application for hundreds or thousands of tenants while keeping a very low total cost of ownership (TCO). This includes setting up new tenants, backing up the databases, managing the customizations and configurations of tenants, and obtaining patches and newer versions of the software from the service provider, SAP. Also, setting up a new tenant in the SaaS-for-SMBs case is completely automated by a Web-based wizard. The new tenant runs through a series of configuration questions and uploads such master data items as product catalog and customer lists. Following these steps, the tenant is up and running, typically using a trial version. The provisioning of the storage, database, and application server resources is part of this automated setup. The customers are billed a fixed monthly subscription fee or a variable fee based on their application usage.

There are several well-known approaches to multi-tenancy of the same database schema [10]. Regardless of the approach taken, multi-tenancy calls for flexible virtualization schemes where, for example, the DBMS component and the storage system are shared between multiple tenants. The main reason for this sharing is to minimize the TCO per tenant.

In summary, the key challenges in these use cases from the point of view of the infrastructure provider are the following:

- Managing thousands of different service components that comprise a variety of service applications executed by thousands of virtual execution environments on a complex infrastructure that also includes network and storage systems.
- Consolidating many applications on the same infrastructure, thereby increasing hardware utilization and optimizing power consumption while keeping the operational cost at a minimum.
- Guaranteeing the individual SLAs of the many customers of the data center, who face different and fluctuating workloads.

Primary requirements

From the use case described in the previous section, we derived the following main requirements for the cloud infrastructure:

Automated and fast deployment—The Reservoir cloud should support automated provisioning of complex

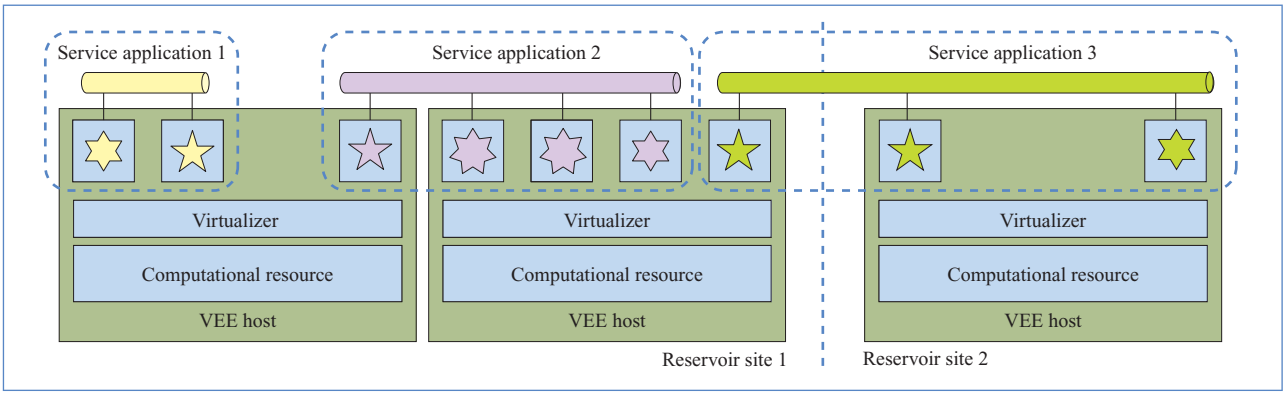

Figure 3

Service applications are executed by a set of VEEs (represented by squares) distributed across the VEE hosts in a Reservoir cloud. VEEs for a particular service application may all be collocated in the same VEE host (as in service application 1), they may be spread across VEE hosts within the same site (as in service application 2), or they may be spread across sites (as in service application 3).

service applications based on a formal contract that specifies the infrastructure SLAs. The same contract should be reused to provision multiple instances of the same application for different tenants with different customizations.

Dynamic elasticity—The Reservoir cloud should dynamically adjust resource allocation parameters (memory, processing, network bandwidth, and storage) of individual virtual execution environments seamlessly. Moreover, the number of virtual execution environments must be dynamically and seamlessly adjusted to adapt to the changing load.

Automated continuous optimization—The Reservoir cloud should continuously optimize alignment of infrastructure resources management with the high-level business goals.

Virtualization technology independence—The Reservoir cloud should support different virtualization technologies transparently.

Reservoir model for federated cloud computing

In the Reservoir model, there is a clear separation between the functional roles of service providers and infrastructure providers. Service providers are the entities that understand the needs of a particular business and offer service applications to address those needs. Service providers do not own the computational resources needed by these service applications; instead, they lease resources from infrastructure providers that provide them with a seemingly infinite pool of computational, network, and storage resources.

Infrastructure providers operate Reservoir sites that own and manage the physical infrastructure on which service applications execute. The federation of collaborating sites forms a Reservoir cloud. To optimize resource utilization, the computational resources within a site are partitioned by a virtualization layer into virtual execution environments (VEEs). VEEs are fully isolated runtime environments that abstract the physical characteristics of the resource and enable sharing. The virtualized computational resources along with the virtualization layer and all of the management enablement components are referred to collectively as the *VEE host*.

A *service application* is a set of software components that works collectively to achieve a common goal. Each component of such service applications executes in a dedicated VEE. The VEEs are placed on the same or different VEE hosts within the site or on different sites (**Figure 3**).

A service application is deployed on the Reservoir cloud using a service manifest, described later in this section, that formally defines the contract and SLA between the service provider and the infrastructure provider.

Resource allocation

Within each Reservoir site, the resource utilization is monitored and the placement of VEEs is constantly updated to achieve optimal utilization with minimal cost. Similarly, the execution of the service applications is monitored and the capacity is constantly adjusted to meet the SLA and requirements specified in the manifest. Reservoir supports two modes of capacity provisioning with respect to service providers: explicit and implicit.

In the explicit capacity requirements for sized-service-applications mode, the service provider conducts sizing and capacity planning studies of the service application

prior to deployment. At deployment time, the service provider precisely specifies the capacity needs of the application under specific workload conditions. In particular, the service provider specifies capacity requirements for the minimal service configuration and the elasticity rules—that is, the rules that govern the automated on-demand allocation and deallocation of additional capacity under varying workload conditions. In this mode, the infrastructure provider is not committed to the high-level service-level objectives (SLOs) for the service (e.g., response time and throughput). Instead, the infrastructure provider commits itself to an infrastructure SLA that governs the terms and conditions of capacity provisioning according to the explicit requirements of the service provider. The service provider supplies explicit capacity requirements and is charged for actual capacity usage in line with the pay-as-you-go model.

The implicit mode covers capacity requirements for unsized service applications. In this mode, the service provider may have only initial sizing estimations for its service or may not have any sizing estimates at all. Therefore, the service is sized within the Reservoir service infrastructure prior to deployment. The infrastructure provider commits itself to an SLA that is formulated in terms of high-level SLOs. As in the explicit capacity for sized-services mode, the service provider pays for the actual usage of capacity. However, while the service provider may ask for usage reports at various levels of detail, it does not have control over the sizing of its service. By default, the infrastructure provider will strive to minimize over-provisioning. In addition, the service provider may specify such policies as minimal resource utilization policy and maximal cost policy.

It is important to note that the ongoing optimizations of the resource allocation are done without human intervention by the Reservoir software stack installed on each site.

Service manifest

The service manifest is one of the key elements of the Reservoir model. The manifest specifies the structure of the service application in terms of component types that are to be deployed as VEEs.

For each of these component types, the manifest specifies a reference to a master image, that is, a self-contained software stack (operating system, middleware, applications, data, and configuration) that fully captures the functionality of the component type. In addition, the manifest contains the information and rules necessary to automatically create from a single, parameterized master image unique VEE instances that can run simultaneously without conflicts [8]. The manifest also specifies the grouping of components into virtual networks and tiers that form the service applications. Given that the emerging Open Virtual Format (OVF) industry standard [11] includes most of this information, the service manifest will extend OVF.

The manifest also specifies the capacity requirements for an explicitly sized service application as agreed upon between the infrastructure provider and the service provider. The minimum and maximum resource requirements of a single instance, for example, the number of virtual CPUs (central processing units), memory size, storage pool size, and the number of network interfaces and their bandwidth, are specified for each component. The capacity specification also includes the minimum and maximum number of VEEs of a particular component type. The dynamic and adaptive part of the capacity requirement is specified using a set of elasticity rules. These rules formally correlate monitored key performance indicators (KPIs) and load parameters (e.g., response time, throughput, and number of active sessions) with resource allocations (e.g., memory, CPUs, bandwidth, and number of VEEs of the same component type). These rules express how the resources allocated to the application, that is, the resources allocated for each VEE as well as the number of VEEs of a particular component type, can be dynamically increased or reduced to satisfy the variable demand for the service application.

Finally, the manifest specifies KPIs that should be monitored by Reservoir to verify SLA compliance and trigger the elasticity rules. This specification may include self-contained probes that periodically provide these KPIs.

To illustrate, a simplified service manifest for a SAP system is shown in **Table 1**. This manifest corresponds to the configuration in which a DI and the DBMS each have a separate VEE, and the CI and Web dispatcher are encapsulated on another VEE. Notice how the manifest fixes the CI and the DBMS as single instances and declares the DI as the elastic entity by providing it with a range of instances. To optimize cost-effectiveness, the service manifest specifies resource requirements under normal load.

To enable dynamic matching of the application capacity to the variances in workload, the manifest defines KPIs that are monitored as indications for the load that the SAP system serves. The overall response time of a certain business transaction or the number of concurrent active user sessions can be used for this purpose. An elasticity rule that triggers the addition of a new DI when this KPI exceeds a threshold value would adapt the resources allocated for the system as the workload increases. For example, if measured response time crossed a prespecified threshold, then a new DI instance would be added.

Table 1 A simplified example of a service manifest for a SAP system. In this example, simple labels are used as master image identifiers, but in a real manifest, fully qualified references (such as URLs) are used.

Component	*Web dispatcher, CI*	*DI*	*DBMS*
Master image	ci.img	di.img	db2.img
No. of virtual CPUs (min./max.)	2/4	4/8	8/8
Memory size (min./max.)	4 GB/8 GB	16 GB/32 GB	32 GB/64 GB
No. of virtual network interface cards	2	2	1
Additional disk size	None	100 GB	1,000 GB
Minimum no. of instances	1	4	1
Maximum no. of instances	1	20	1

Reservoir components

The Reservoir architecture shown in **Figure 4** is designed to provide a clean separation of concerns among the layers operating at different levels of abstraction. The rationale behind this particular layering is to keep a clear separation of concerns and responsibilities and to hide low-level infrastructure details and decisions from high-level management and service providers.

Service manager

The service manager is the highest level of abstraction, interacting with the service providers to receive their service manifests, negotiate pricing, and handle billing. Its two most complex tasks are deploying and provisioning VEEs based on the service manifest and monitoring and enforcing SLA compliance by throttling the capacity of a service application.

The service manager receives service manifests from the service providers. Based on information in the manifest, it deploys and provisions the service application by interacting with the VEE manager to allocate VEEs and their associated resources. From the service requirements in the manifest, such as SLOs and elasticity rules, the service manager derives a list of required resources and their configuration as well as placement constraints based on such factors as cost, licensing, and confidentiality. For unsized service applications (represented by the box labeled "Service application 3" in Figure 3), the service manager is responsible for generating explicit rules based on site policy. Deployment and provisioning decisions are based on performance and SLA compliance and adjusted according to such business considerations as, for example, costs, security, and offers.

The service manager is also responsible for monitoring the deployed services and adjusting their capacity, that is, the number of VEE instances and their allocation of resources such as memory and CPUs to ensure SLA compliance and alignment with high-level business goals, such as cost-effectiveness.

Finally, the service manager is responsible for accounting and billing. While existing cloud-computing infrastructures tend to be quite inflexible and usually employ fixed-cost, postpaid subscription models, we consider both postpaid and prepaid billing models based on resource usage. Both models are based on the resource utilization information provided by the service manager accounting system and are processed according to the

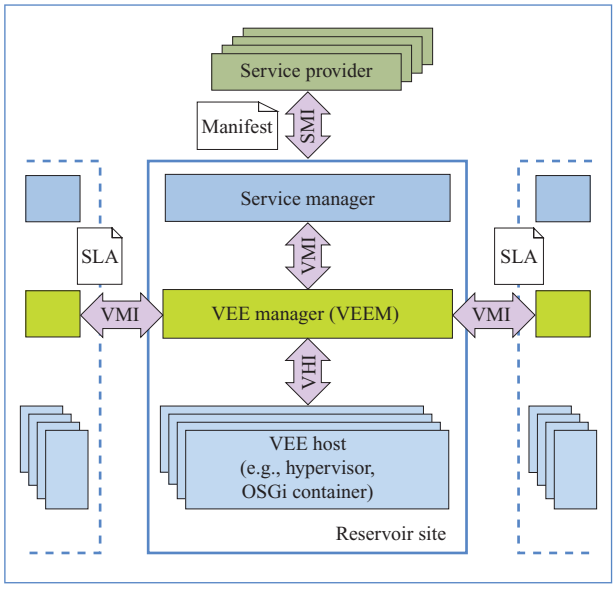

Figure 4

The Reservoir architecture: major components and interfaces (OSGi: Open Services Gateway initiative; SMI: service management interface; VMI: VEE management interface; VHI: virtual host interface).

rules in the business information model to perform cost calculation.

Virtual execution environment manager

The virtual execution environment manager (VEEM) is the next level of abstraction, interacting with the service manager above it, the VEE hosts below, and the VEE managers at other sites in order to enable federation.

The VEEM is responsible for the optimal placement of VEEs into VEE hosts subject to constraints determined by the service manager. The continuous optimization process is driven by a site-specific programmable utility function. The VEEM is free to place and move VEEs anywhere, even on the remote sites (subject to overall cross-site agreements), as long as the placement satisfies the constraints. Thus, in addition to serving local requests from the local service manager, VEEM is responsible for the federation of remote sites.

At the VEEM level, a service is realized as a set of interrelated VEEs, a *VEE group*, and hence it should be managed as a whole. For example, the service manifest may define a specific deployment order, placement constraints in the form of affinity rules, or rollback policies. The VEEM also provides the functionality needed to handle the dynamic nature of the service workload, such as the ability to add and remove VEEs from an existing VEE group or to change the capacity of a single VEE.

Operating in federated environments puts additional requirements on the VEEM for submission, management, and monitoring of VEEs on remote sites. The VEEM at the primary site performs this by assuming the role of a service manager of the remote VEEM in all cross-site interactions. A clear delegation of responsibility and separation of concerns among the service manager, the VEEM at the primary site, and the remote VEEM follows from this distinct role definition. For placement decisions, the primary VEEM takes into account agreements with other sites and detailed information about local resources before deciding on local or remote VEE placement. Notably, the primary VEEM does not get involved in the internal placement decisions on the remote site because this is a concern of the remote VEEM. The interfaces used by the primary VEEM to interact with a remote VEEM are the same as those used by a service manager for interactions with the primary VEEM.

Virtual execution environment host

The virtual execution environment host (VEEH) is the lowest level of abstraction, interacting with the VEE manager to realize its IT management decisions onto a diverse set of virtualization platforms.

The VEEH is responsible for the basic control and monitoring of VEEs and their resources. This includes such activities as creating a VEE, allocating additional resources to a VEE, monitoring a VEE, migrating a VEE, and creating a virtual network and storage pool. Each VEEH type encapsulates a particular type of virtualization technology, and all VEEH types expose a common interface such that VEEM can issue generic commands to manage the life cycle of VEEs. The receiving VEEH is responsible for translating these commands into commands specific to the virtualization platform that it has abstracted.

Given that VEEs belonging to the same application may be placed on multiple VEEHs and even extend beyond the boundaries of a site, VEEHs must support isolated virtual networks that span VEEHs and sites. Moreover, VEEHs must support transparent VEE migration to any compatible VEEH in a Reservoir cloud, regardless of site location or network and storage configurations.

Layers of interoperability

The layered design stresses the use of standard, open, and generic protocols and interfaces to support vertical and horizontal interoperability between layers. Different implementations of each layer will be able to interact with each other. The service management interface with its service manifest exposes a standard interface into the Reservoir cloud for service providers. The service provider may then choose among Reservoir cloud providers, knowing that they share a common language to express their business requirements. The VEE management interface simplifies the introduction of different and independent IT optimization strategies without disrupting other layers or peer VEEMs. Further, the fact that the VEE management interface supports VEEM-to-VEEM communication simplifies cloud federation by limiting the horizontal interoperability to one layer of the stack. The VEEH interface will support plugging in new virtualization platforms such as hypervisors without requiring VEEM recompilation or restart. The Reservoir loosely coupled stack reference architecture should promote a variety of innovative approaches to support cloud computing.

Related work

In this section, we briefly review the state of the art in areas related to the Reservoir model and architecture.

Virtualization technologies

Virtualization has reemerged in recent years as a compelling approach to increasing resource utilization and reducing the cost of IT services. The common theme of all virtualization technologies is hiding the underlying infrastructure by introducing a logical layer between the physical infrastructure and the computational processes.

Virtualization takes many forms. System virtualization [12], also commonly referred to as server virtualization, is the ability to run multiple heterogeneous operating systems on the same physical server [6]. With server virtualization, a control program, commonly known as a hypervisor or VM monitor, is run on a given hardware platform simulating one or more other VMs. Each of these VMs runs its respective operating system just as if it were installed on a standalone hardware platform. Storage virtualization, network virtualization, and logical representations of physical storage and network resources are other examples of virtualization.

Distributed management of virtualization

Given the growing popularity of virtualization, many commercial products and research projects, such as OpenNebula [13], Platform VM Orchestrator [14], IBM Virtualization Manager [15], and VMware DRS [16] are being developed to dynamically overlay physical resources with virtual machines. In general, these efforts are intended to extend the benefits of virtualization from a single resource to a pool of resources, decoupling the VM not only from the physical infrastructure but also from the physical location.

There are also some commercial and scientific infrastructure cloud-computing initiatives including Nimbus [17], Eucalyptus [18], and Amazon EC2. These provide remote interfaces for control and monitoring of virtual resources. Although these solutions simplify the management of VMs on a distributed pool of resources, none of these initiatives for distributed VM management evaluate its extension to a grid-like environment in which different infrastructure sites could collaborate to satisfy peak or fluctuating demands.

In the context of Reservoir, grid interfaces and protocols [19] may enable the required interoperability between the clouds or infrastructure providers. However, Reservoir will strive to overcome interoperability challenges often posed in traditional grids by a very strict separation of concerns and by minimal interfaces, reducing cross-site concerns to a minimum. Reservoir also needs to expand substantially on the current state of the art for grid-wide accounting [20] and increase flexibility to support different billing schemes and accounting for services with indefinite lifetimes (as opposed to finite jobs) with support to account for utilization metrics relevant for VMs.

Business service management

BSM is a management strategy that links IT infrastructure management objectives to the goals of the business. In contrast to a technologically oriented management approach, in BSM IT shares the same overall targets with the business, such as growing revenue, reducing costs, lowering business risk, increasing customer satisfaction, guaranteeing a given return on investment, and complying with regulations.

A key aspect of BSM is SLA management. In Reservoir, new SLA management challenges arise as a result of the dynamic federation of infrastructure providers. Some emerging approaches to SLA management that can be leveraged in Reservoir include data-driven methods (e.g., dynamic setting of SLOs based on statistical analysis [21]), model-driven methods (e.g., developing performance models that govern the design of information and communication technology infrastructure [22]), and language-based approaches (i.e., using formal languages to specify SLAs to assure that they are consistent and unambiguous [23]). Skene et al. [24] have studied SLA monitoring and the formal definition for application-service provisioning. Some earlier work considered SLA management in federated environments; however, that line of research [25] did not address the special considerations of supporting migration and virtualization.

Summary

In this paper, we have presented a new open federated cloud-computing model, architecture, and functionality as being developed in the Reservoir project. The Reservoir model explicitly addresses the limited scalability of a single-provider cloud, the lack of interoperability among cloud providers, and the lack of built-in BSM support in current cloud offerings.

Cloud computing has the potential of becoming a revolutionary technology that changes the way service computing is performed. We believe that the principles introduced in this work are essential for the cloud-computing vision to materialize to its full extent.

The Reservoir project is in its early stages. Work on the implementation and evaluation of the concepts outlined in this paper continues as this paper was prepared for publication. Further discussion of these results is deferred for future work.

Acknowledgments

The research leading to these results is partially supported by the European Community Seventh Framework Programme (FP7/2001-2013) under grant agreement #215605. We thank Eliot Salant from IBM and Juan Hierro from Telefónica without whose work this project would not have been possible. In addition, we thank Shimon Agassi, Katherine Barabash, David Hadas, Irit Loy, and Edi Shmueli of IBM; Manuel Díaz, David Perales, and Emilio Torres of Telefónica; Daniel Henriksson, Francisco Hernandez, and Johan Tordsson of Umeå; Mark Wusthoff of SAP; Fritz Ferstl of Sun Microsystems; and Javier Fontan, Luis Gonzalez, Eduardo Huedo, Rafael Moreno, and Tino Vazquez from

Universidad Complutense de Madrid for their help materializing the ideas expressed in this document.

*Trademark, service mark, or registered trademark of International Business Machines Corporation in the United States, other countries, or both.

**Trademark, service mark, or registered trademark of Google, Inc., Linus Torvalds, or Citrix Systems, Inc., in the United States, other countries, or both.

References

1. J. Meattle, "YouTube vs. MySpace Growth: No Contest!" *Compete, Inc.*, October 18, 2006; see *http://blog.compete.com/2006/10/18/youtube-vs-myspace-growth-google-charts-metrics/*.
2. N. Carr, *The Big Switch: Rewiring the World, from Edison to Google*, W. W. Norton & Company, New York, 2008.
3. Amazon Elastic Compute Cloud (Amazon EC2), Amazon Web Services LLC; see *http://aws.amazon.com/ec2/*.
4. I. Foster, C. Kesselman, and S. Tuecke, "The Anatomy of the Grid: Enabling Scalable Virtual Organizations," *Intl. J. Supercomputer Appl.* **15**, No. 3, 200–222 (2001).
5. K. Appleby, S. Fakhouri, L. Fong, G. Goldszmidt, M. Kalantar, S. Krishnakumar, D. P. Pazel, J. Pershing, and B. Rochwerger, "Océano—SLA Based Management of a Computing Utility," *Proceedings of the 6th IEEE/IFIP International Symposium on Integrated Network Management, Seattle*; Wiley-IEEE Press, Hoboken, NJ, 2001, pp. 855–868.
6. P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield, "Xen and the Art of Virtualization," *Proceedings of the 19th ACM Symposium on Operating Systems Principles*, ACM, New York, 2003, pp. 164–177.
7. M. Turner, D. Budgen, and P. Brereton, "Turning Software into a Service," *Computer* **36**, No. 10, 38–44 (2003).
8. O. Goldshmidt, B. Rochwerger, A. Glikson, I. Shapira, and T. Domany, "Encompass: Managing Functionality," *Proceedings of the 21st IEEE International Parallel and Distributed Processing Symposium*, Wiley-IEEE Press, Hoboken, NJ, 2007, pp. 1–5.
9. SAP AG, SAP Help Portal; see *http://help.sap.com/*.
10. S. Aulbach, T. Grust, D. Jacobs, A. Kemper, and J. Rittinger, "Multi-tenant Databases for Software as a Service: Schema-Mapping Techniques," *Proceedings of the ACM SIGMOD International Conference on Management of Data*; ACM, New York, 2008, pp. 1195–1206.
11. Distributed Management Task Force, Inc., Open Virtualization Format Specification, Version 1.0.0, document no. DAP0243; see *http://www.dmtf.org/standards/published_documents/DSP0243_1.0.0*.
12. G. J. Popek and R. P. Goldberg, "Formal Requirements for Virtualizable Third Generation Architectures," *Communic. ACM* **17**, No. 7, 412–421 (1974).
13. OpenNebula.org, Distributed Systems Architecture Group at Universidad Complutense de Madrid; see *http://www.opennebula.org/doku.php*.
14. Platform Computing Corporation, Platform VM Orchestrator; see *http://www.platform.com/Products/platform-vm-orchestrator*.
15. IBM Corporation, *Extensions: Virtualization Manager*; see *http://www-03.ibm.com/systems/management/director/about/director52/extensions/vm.html*.
16. VMware, Inc., VMware DRS; see *http://www.vmware.com/products/vi/vc/drs.html*.
17. The Globus Alliance, Nimbus; see *http://workspace.globus.org/*.
18. Eucalyptus; see *http://eucalyptus.cs.ucsb.edu/wiki/Presentations*.
19. I. Foster and C. Kesselman, "Globus: A Metacomputing Infrastructure Toolkit," *Intl. J. Supercomputer Appl. High Perf. Computing* **11**, No. 2, 115–128 (1997).
20. P. Gardfjäll, E. Elmroth, L. Johnsson, O. Mulmo, and T. Sandholm, "Scalable Grid-Wide Capacity Allocation with the SweGrid Accounting System (SGAS)," *Concurrency Computation Practice Exper.* **20**, No. 18, 2089–2122 (2008).
21. D. Breitgand, E. A. Henis, O. Shehory, and J. M. Lake, "Derivation of Response Time Service Level Objectives for Business Services," *Proceedings of the 2nd IEEE/IFIP International Workshop on Business-Driven IT Management*, Wiley-IEEE Press, Hoboken, NJ, 2007, pp. 29–38.
22. J. Sauvé, F. Marques, A. Moura, M. Sampaio, J. Jornada, and E. Radziuk, "SLA Design from a Business Perspective," *Proceedings of the 16th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management*, Springer Berlin/Heidelberg, 2005, pp. 72–83.
23. D. D. Lamanna, J. Skene, and W. Emmerich, "SLAng: A Language for Service Level Agreements," *Proceedings of the 9th IEEE Workshop on Future Trends in Distributed Computing Systems*, IEEE Computer Society Press, Washington, DC, 2003, pp. 100–106.
24. J. Skene, A. Skene, J. Crampton, and W. Emmerich, "The Monitorability of Service-Level Agreements for Application-Service Provision," *Proceedings of the 6th International Workshop on Software and Performance*, ACM, New York, 2007, pp. 3–14.
25. P. Bhoj, S. Singhal, and S. Chutani, "SLA Management in Federated Environments," *Proceedings of the 6th IFIP/IEEE International Symposium on Integrated Network Management*, Boston; Wiley-IEEE Press, Hoboken, NJ, 1999, pp. 293–308.

Received July 29, 2008; accepted for publication October 9, 2008

Benny Rochwerger *IBM Research Division, Haifa Research Labs, Haifa University Campus, Mount Carmel, Haifa 31905, Israel (rochwer@il.ibm.com)*. Mr. Rochwerger has an M.S. degree in computer science from the University of Massachusetts Amherst, and a B.Sc. degree in computer engineering from the Technion–Israel Institute of Technology. Since joining IBM in 1995, he has worked in virtualization management, autonomic computing, event processing, grid computing, distributed graphics, and networking. He is currently the lead architect for the Reservoir project.

David Breitgand *IBM Research Division, Haifa Research Labs, Haifa University Campus, Mount Carmel, Haifa 31905, Israel (davidbr@il.ibm.com)*. Dr. Breitgand received his B.Sc., M.Sc., and Ph.D. degrees, all in computer science, from the Hebrew University of Jerusalem. In 2003, he joined IBM, where he is a member of the Storage and Performance Management group, leading the group's research efforts in end-to-end performance analysis and management of networked storage and systems. For the Reservoir project, he focuses on algorithms for cost-effective capacity provisioning for Reservoir services.

Eliezer Levy *SAP Labs Israel Ltd., 15 Hatidhar Street, Ra'anana, Israel 43665 (levy@sap.com)*. Dr. Levy has a B.Sc. degree in computer science from the Technion–Israel Institute of Technology and a Ph.D. degree in computer sciences from the University of Texas at Austin. He is currently a researcher in SAP Research. He was previously the chief architect of the small businesses business unit in SAP.

Alex Galis *University College London, Torrington Place, London WC1E 7 JE, United Kingdom (a.galis@ee.ucl.ac.uk)*. Dr. Galis is a visiting professor at the Department of Electronic

and Electrical Engineering at University College London. He has coauthored and published more than 125 refereed journal or conference papers, more than 100 reports, and six books on network and service management, intelligent services, programmable networks and services, virtualization of resources, and grid systems.

Kenneth Nagin *IBM Research Division, Haifa Research Labs, Haifa University Campus, Mount Carmel, Haifa 31905, Israel (nagin@il.ibm.com)*. Mr. Nagin received a B.A. degree from the University of Madison and a B.S. degree from the University of Pittsburgh. Since joining IBM in 1985, he has worked on copy services, disaster recovery solutions and storage management tools for direct-access storage devices, model-based software test generation and test consultation, and BladeCenter* management tools. He holds more than 20 patents. His current research is concerned with cloud-computing server virtualization.

Ignacio M. Llorente *Universidad Complutense de Madrid, C/ Profesor José García Santesmases, 28040 Madrid, Spain (llorente@dacya.ucm.es)*. Dr. Llorente has more than 15 years of research experience in the field of high-performance parallel and distributed computing. He is currently a full professor in computer architecture and technology at Universidad Complutense de Madrid, where he leads the Distributed Systems Architecture group.

Ruben Montero *Universidad Complutense de Madrid, C/ Profesor José García Santesmases, 28040 Madrid, Spain (rubensm@dacya.ucm.es)*. Dr. Montero is an associate professor in computer architecture and technology in the Department of Computer Architecture and System Engineering at Universidad Complutense de Madrid, where he is part of the Distributed Systems Architecture group. He has held several research appointments at the Institute for Computer Applications in Science and Engineering (ICASE), at NASA Langley Research Center.

Yaron Wolfsthal *IBM Research Division, Haifa Research Labs, Haifa University Campus, Mount Carmel, Haifa 31905, Israel (wolfsthal@il.ibm.com)*. Dr. Wolfsthal heads the system and services area at the IBM Haifa Research Laboratory. This area is one of the IBM European Union innovation centers, and its mission is to develop leading-edge technologies for IBM advanced information and communication technology products and services, including virtualization infrastructures and systems management. He has 16 years of experience in various research and development and management roles. He holds B.Sc., M.S., and Ph.D. degrees in computer science, all from the Technion–Israel Institute of Technology.

Erik Elmroth *Umeå University, SE-901 87 Umeå Sweden (elmroth@cs.umu.se)*. Dr. Elmroth is the scientific leader of the Grid Computing Research group at Umeå University. He is also an associate professor and serves as deputy head of the Department of Computing Science and deputy director of the High Performance Computing Center North. He has held positions at National Energy Research Scientific Computing Center, Lawrence Berkeley National Laboratory, the University of California Berkeley, and the Massachusetts Institute of Technology. He is a member of the Swedish Research Council's Committee for Research Infrastructures, vice chair of its expert panel on e-science, and scientific secretary of the former e-science group of the Nordic Council of Ministers.

Juan Cáceres *Telefónica I+D, C/ Emilio Vargas, 6. 28043 Madrid, Spain (caceres@tid.es)*. Mr. Cáceres has a Research M.Sc. (Bologna Process) and an M.Sc. degree in computer science from the Technical University of Madrid and is currently working toward his Ph.D. degree in cloud computing at the Universidad Complutense de Madrid. He has more than eight years of experience in middleware and distributed systems development at Telefónica I+D. He has also experience in open source software from setting up the MORFEO Open-Source Software Community, where he leads the Middleware Platform group.

Muli Ben-Yehuda *IBM Research Division, Haifa Research Labs, Haifa University Campus, Mount Carmel, Haifa 31905, Israel (muli@il.ibm.com)*. Mr. Ben-Yehuda is a systems researcher at IBM Haifa Research Laboratory, where he was recently named an IBM Master Inventor. His research interests include I/O for virtualized systems, I/O memory management units, smart I/O devices, and innovative uses of virtual machines. He has contributed to numerous operating systems and hypervisors, including the Linux** kernel, the Xen** virtual machine monitor, and the Linux kernel-based virtual machine project (KVM).

Wolfgang Emmerich *University College London, Torrington Place, London WC1E 7 JE, United Kingdom (W.Emmerich@cs.ucl.ac.uk)*. Dr. Emmerich graduated from the Universität Dortmund and obtained his doctorate from the Universität of Paderborn. He is a professor of Distributed Computing at the Department of Computer Science, where he is director of research and head of the Software Systems Engineering group. He is a Chartered Engineer, a member of the editorial board of *IEEE Transactions of Software Engineering*, and a member of the Institution of Engineering and Technology. Aside from a wide range of research publications, he is author of *Engineering Distributed Objects*, published by Wiley in 2000.

Fermín Galán *Telefónica I+D, C/ Emilio Vargas, 6. 28043 Madrid, Spain (fermin@tid.es)*. Mr. Galán is an R&D engineer at Telefónica I+D in the Business Oriented Infrastructure group, and has been involved in several R&D European collaborative projects. He has an M.Sc. degree in telecommunications engineering from the Technical University of Madrid. He is currently working toward his Ph.D. degree in telematics at the Technical University of Madrid.