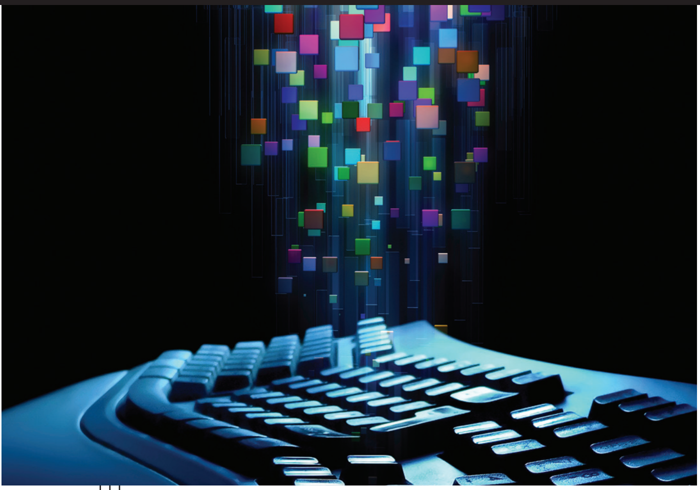

Reservoir— When One Cloud Is Not Enough

Benny Rochwerger, David Breitgand, Amir Epstein, David Hadas, Irit Loy, and Kenneth Nagin, *IBM Haifa Research Lab*; Johan Tordsson, *Umeå University*; Carmelo Ragusa and Massimo Villari, *University of Messina*; Stuart Clayman, *University College London*; Eliezer Levy, *SAP Research*; Alessandro Maraschini, *Telespazio*; Philippe Massonet, *CETIC*; Henar Muñoz, *Telefónica I+D*; Giovanni Toffetti, *University of Lugano*

As demand for cloud services grows, the increases in cost and complexity for the cloud provider could become a major obstacle. Technologies developed under the Reservoir research project help cloud providers deal with complexity and scalability issues.

Cloud computing is the latest incarnation of the utility computing model envisioned in the 1960s.[1] Just as an electric utility hiding beyond the wall plug powers a wide variety of devices, individuals and organizations can now fulfill most of their computing needs from a computing utility hidden in the network.[2]

Cloud computing's analogy to an electrical power grid does not end with the consumption model. Power-generation plants are built to support a certain maximum capacity, which is determined by analyzing the average utilization; and then overprovisioned for predicted spikes. Demand that exceeds this maximum capacity is delegated to neighboring providers. Similarly, cloud computing providers can handle requests that exceed their capacity by delegating them to other cloud computing providers. This is a flexible and cost-efficient alternative to overprovisioning.

Grid computing, an earlier incarnation of the utility computing model, was driven by the need for more compute power.[3] Cloud computing, on the other hand, is driven by the need of companies and individuals to deal with the ever-increasing cost and complexity of IT services. Whether the customer is a company looking to outsource its IT, or a start-up with a great idea but little funding, or a big one-time project in need of resources, cloud computing offers a seemingly infinite pool of resources, without any capital expenses or system administration overhead.

In grid computing, resource sharing is a goal by definition—scientific centers share their infrastructure with one another to achieve additional compute power. The need for resource sharing in cloud computing might not be that obvious at the moment. However, as cloud computing becomes a mainstream technology, providers will likely choose to support a federated model driven purely by business goals—that is, be only as big as needed to be profitable, and rely on others when more resources are necessary.

THE FEDERATED CLOUD MODEL

The primary goal of Reservoir (an acronym derived from resources and services virtualization without boundaries), a European research initiative, is to develop the technologies needed to deal with the scalability problem inherent in the single-provider cloud computing model.[4] Reservoir explores the notion of a federated cloud in which computing infrastructure providers lease excess capacity to others in need of temporary additional resources.

In the Reservoir model, two or more independent cloud computing providers can join together to create a federated cloud. Federation participants who have excess capacity can share their resources, for an agreed-upon price, with participants needing additional resources. This sharing and paying model helps individual providers avoid overprovisioning of resources to deal with spikes in capacity demand.

Published by the IEEE Computer Society

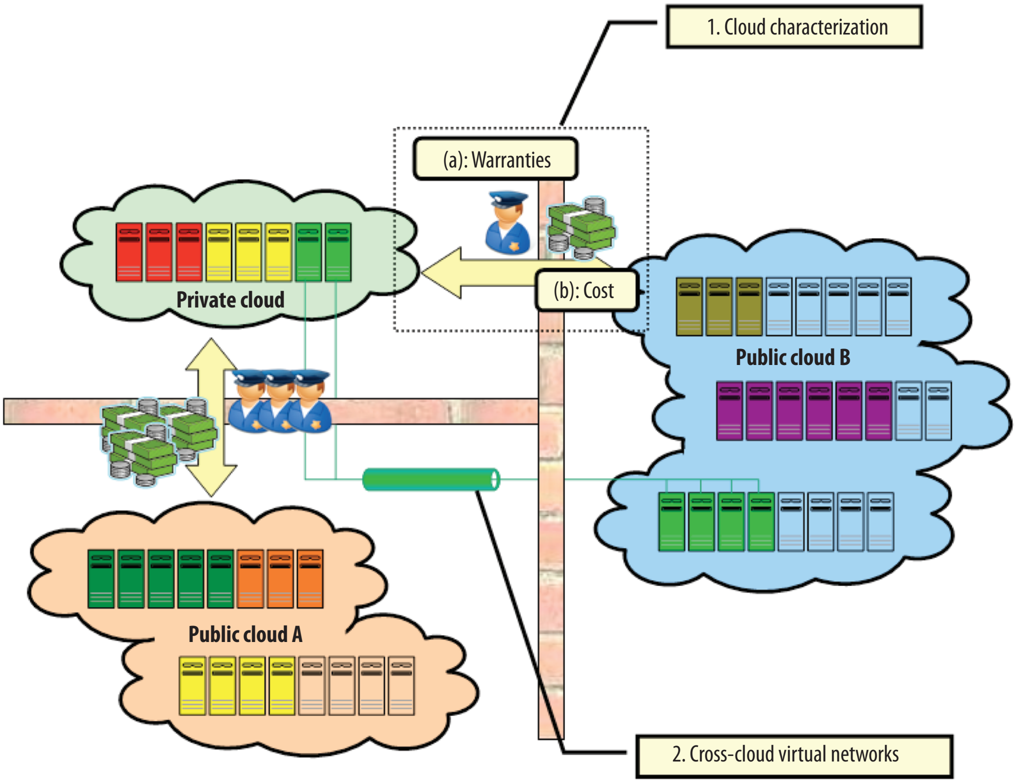

Figure 1. Challenges in the federated cloud computing model. (1) Finding the "best" cloud for any particular workload requires careful balancing among many parameters, such as quality-of-service warranties and cost. (2) The model maintains a consistent logical topology regardless of the physical location of the different application components (virtual machines).

In a multicloud environment, a system that makes automated decisions must address federated placement—that is, the process of determining which cloud to use for a particular workload, given that not all clouds are equal in terms of warranties and prices. For example, a particular cloud might be inexpensive, but might not provide availability warranties, making it inappropriate for mission-critical workloads. Another cloud, however, might guarantee "five nines availability" (that is, 99.999 percent availability) but be more expensive. Figure 1 shows this characterization of each of the clouds in a federation.

From an infrastructure viewpoint, to support maximum optimization, applications must be completely location-free. In other words, the system must be able to deploy the application's components (encapsulated in virtual machines) anywhere in the federated cloud. Moreover, it should be able to migrate these components at any time, even across clouds. This model requires the development of technology that supports location-independent virtual networks (shown in Figure 1) and offers VMs consistent access to data. At the same time, this technology should let customers limit the system's flexibility to ensure application correctness and compliance with government regulations and company policies.

For consumers, a main advantage of cloud computing is the capability to provide, or release, resources on demand. Cloud computing providers should enact these elasticity capabilities automatically to meet demand variations. Clearly, contracts and rules agreed on by cloud computing providers and consumers should drive the behavior and limits of automatic growth and shrinking. Users' ability to grow their applications when facing increased demand must be complemented by the provider's ability to scale and overcommit resources.

Cloud infrastructures are subject to the same threats as other distributed systems. The security requirements for cloud users are related to the ability to select or associate different security policies with cloud service deployments and the ability to monitor these policies. The security requirements for cloud providers necessitate isolating customer deployments at both the virtual and physical infrastructure levels. Isolation of the virtual infrastructure includes isolating not only VMs, but also virtual networks and virtualized storage. The ability to guarantee that services are provisioned only in clouds with the appropriate level of security policies is an important requirement for federated clouds. Other requirements include data location, regulatory compliance, recovery, investigative support, and long-term viability of cloud deployments.

THE RESERVOIR TESTBED

To develop, experiment, and gain insight into the possible roadblocks of the federated cloud model, we built a multicloud environment that aggregates resources from three sites:

- University of Messina, Italy—15 biprocessor dual-core machines, each with 8 Gbytes of RAM;
- Thales Group site, France—12 biprocessor quad-core machines, each with 4 Gbytes of RAM; and
- Umeå University, Sweden—three quad-core machines, each with 8 Gbytes of RAM.

All of the machines use the Linux Kernel-based Virtual Machine (KVM) Monitor[5] as the hypervisor and the Reservoir cloud-management middleware, which is packaged in a self-contained VM for maximum flexibility and ease of administration.

> **The ability to dynamically scale a service up and down is critical because it lets the cloud computing user avoid overprovisioning, while still being able to automatically adjust to changing loads.**

Although relatively small compared to production clouds, this setup provides an excellent environment for experimentation with cloud federation issues: the clouds are geographically distant, and are owned and managed by entirely independent organizations with different network and security setups.

FEDERATED CLOUD-MANAGEMENT SOLUTIONS

The Reservoir project aims to research and develop advanced technologies so that cloud infrastructure providers can efficiently run their businesses and provide value to their customers.

Dynamic service elasticity

The ability to dynamically scale a service up and down is critical because it lets the cloud computing user avoid overprovisioning, while still being able to automatically adjust to changing loads. In Reservoir, the scaling process is automated through elasticity rules,[6] a mechanism for specifying an application's dynamic capacity requirements at deployment time. Elasticity rules follow the event-condition-action approach, in which certain conditions trigger automated actions to change the service capacity. Such capacity changes include

- scaling up—resizing a running service component, such as increasing or decreasing a component's allocated memory; or
- scaling out—adding or removing instances of service components.

Dynamic elasticity can be specified explicitly by adding elasticity rules to the service manifest—a descriptor of the service based on the Open Virtualization Format (OVF) standard.[7] Such an approach benefits from the inherent knowledge providers have of their applications. For example, for the SAP application, we defined a key performance indicator (KPI) as the total number of active sessions currently served by the SAP system. We also defined an elasticity rule specifying that when this KPI exceeds a threshold, a new VM is automatically started.

An alternative to using explicit rules is implicit service-level agreement (SLA) protection. In this case, instead of elasticity rules, the service manifest includes a section on performance objectives (for example, response time must be less than 50 milliseconds for 90 percent of the time for a 10-minute window). These objectives are coupled with a control strategy (for example, minimizing the number of VMs within the service-level objectives boundaries).

We developed an engine that constructs an approximate model of the system response for each performance objective as a function of the service configuration in terms of VM instances, input workload, and other relevant KPIs. The engine updates this model continuously to obtain autonomic control.[8]

We construct an approximate model in two steps.

1. At service staging time, we provide artificial workloads to different system configurations and measure the system response.
2. At runtime, the engine controls the system configuration through the approximate models, enriching them with additional information as it investigates new combinations of workload, KPIs, and configurations.

This continuous learning process is a key feature of the autonomic controller.

As part of the Reservoir validation, we ran several experiments to assess the system responses of various services under different working conditions. Figure 2 shows an example of a Kriging model of the SAP use case throughput as a function of its incoming workload and the number of VMs (dialog instances).[8]

Admission control

As the number of deployed services increases, the probability that all elastic services will simultaneously request resources up to the maximal contracted capacity range diminishes. Moreover, as a system grows in size, the variance of total resource demand in the system becomes smaller.

Drawing our inspiration from results in network bandwidth multiplexing, we defined a notion of equivalent virtual capacity required to host the given mix of elastic services while keeping the probability of resource allocation congestion below the acceptable risk level (ARL). The infrastructure provider sets this risk level in accordance with its business goals. A conservative approach would set the ARL at the level of the strictest SLA availability percentile. As long as there is enough physical capacity to place equivalent capacity, a system will honor its SLA for all services, while efficiently multiplexing physical resources.

In Reservoir, we enhanced cloud-management functionality with admission control. Admission control continuously calculates the anonymized equivalent capacity based on the statistics gathered for the service portfolio. When Reservoir accepts a new service into the cloud, the admission control policy calculates its impact on the equivalent capacity. The policy assumes a pessimistic estimation of resource usage for the new service—namely, that it would use its maximal resource allocation as specified in the service manifest. Using placement functions, Reservoir accepts the new service only if it can feasibly place the resulting equivalent capacity on available physical resources.

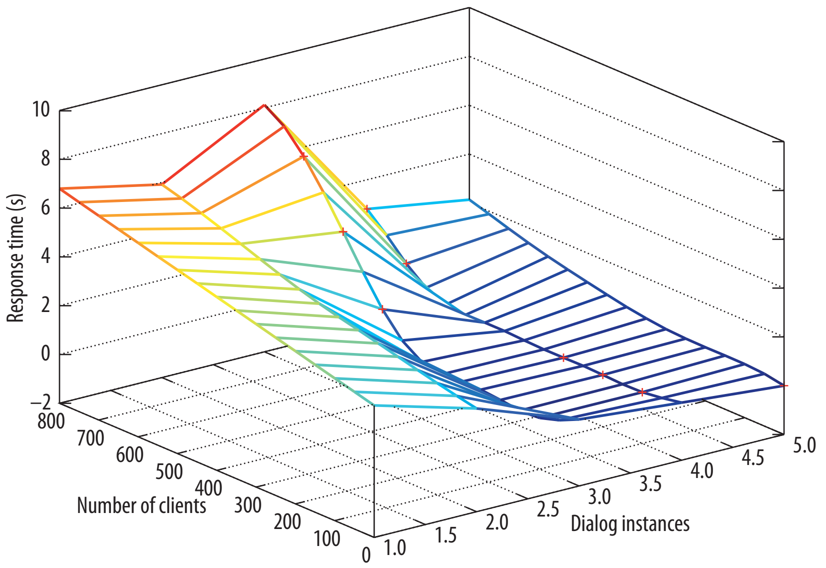

Figure 2. Surrogate model of service throughput as a function of the number of concurrent clients and number of dialog instances. We use this model to find the optimal system configuration for a given workload and strategy (for example, find the cheapest configuration for a throughput threshold).

Table 1. Equivalent capacity in compute units as a function of the number of services and acceptable risk level probability.

Acceptable risk level (ARL)	Equivalent capacity versus maximal demand		
	100 services	200 services	300 services
Maximal demand	2,000	4,000	6,000
0.15	1,058	2,331	3,169
0.10	1,075	2,370	3,220
0.05	1,124	2,489	3,374
0.01	1,523	3,443	4,618

Table 1 presents the results from a simplified simulation study of the theoretical multiplexing gain attainable for different ARL values ranging from 0.15 to 0.01. The simulation comprises three groups of experiments using 100, 200, and 300 simulated services, respectively. Each experiment used 5,760 data points, corresponding to two months' worth of monitoring, in which each data point was collected at 15-minute intervals. Each service specified 20 compute units as its maximal demand. To simulate elasticity, we drew the actual number of resources for each service at any given time from the uniform distribution in the range [1, 20]. We distributed the stability periods between these resource allotment changes exponentially with a mean of 50, which corresponds to about 12.5 hours of stability between conceptual changes in resource allocation due to elasticity.

The equivalent capacity grew as ARL diminished. Moreover, the theoretically attainable multiplexing gain computed as the ratio between the maximal demand and equivalent capacity ranged from 1.89 to 1.29. The actual multiplexing gain depends on the specific placement policy and available physical capacity.

Policy-driven placement optimization

The ability to effectively place VMs is vital for cost-efficient service provisioning. Finding the best mapping or placement of VMs relative to physical machines is one of the most challenging problems for cloud-management systems.

For maximum flexibility in addressing each infrastructure provider's needs and policies, Reservoir supports dynamically pluggable policies to calculate the placement. Each policy defines a different utility function to be optimized. For example, a load-balancing policy attempts to keep VMs equally distributed over physical machines, whereas a power-preservation policy consolidates all VMs in the minimal number of physical machines, thus allowing unused machines to be turned off.

Developers also use the pluggable policy framework to compose multiple policies into policy chains. These chains combine placement on the physical machines in the local cloud with federated VM placement in partnering clouds. The resulting VM placement strategy is a two-step process: the system first tries to place all VMs locally according to the active local placement optimization policy; it then considers federated resources for VMs that cannot be placed locally.

Remote placement constitutes a significant challenge because the clouds are separate management entities.

Remote placement constitutes a significant challenge because the clouds are separate management entities. As such, each cloud maintains individual management policies and keeps exact infrastructure details private. To overcome this problem, we use a framework agreement contract that predefines the collaboration between two clouds. This contract specifies the capacity, in terms of the number and size of VMs available to others, together with various nonfunctional constraints, such as cost, quality-of-service and security levels, and contract validity time. Example of remote placement policies include

- revenue maximization, which weights income from service providers against provisioning costs; and
- eventual SLA violation penalties and consolidation of each service's VMs across a minimal number of remote clouds.

Cloud placement optimization problems can be modeled as extended classical combinatorial optimization problems, such as the *generalized assignment problem* (GAP). We use several approaches to handle large-scale placement problems on the order of hundreds of physical hosts and remote sites and a few thousands VMs—from exact solutions based on techniques, such as column generation,[9] to inexact solutions such as approximation algorithms.[10]

Cross-cloud virtual networks

Supporting applications built out of intercommunicating components that can be deployed and migrated across clouds requires a novel approach to networking. Virtual application networks (VANs) are virtual and distributed switching services that connect VMs.[11] VANs revolutionize how networks are organized by using an overlay network between hypervisors. The hypervisors' overlay network decouples the virtual network services offered to VMs from the underlying physical network. As a result, the network created between VMs becomes independent of the physical network's topology. Moreover, the resulting virtual networks can be migrated alongside the VMs connected to them.

In addition, VANs offer high levels of security by isolating virtual networks from one another and from the physical network. Such isolation is crucial for constructing large-scale clouds servicing many independent customers. Unlike virtual LANs (VLANs), which are physical resources, VANs do not introduce virtual network service costs. At the same time, we can minimize their effect on network performance. Providing network isolation of services is vital, not only when they are running in the same cloud, but also in a federated cloud. Furthermore, cloud providers cannot be expected to coordinate their network maintenance, network topologies, and more with one another.

VANs separate clouds using proxies, which act as gateways among clouds. A VAN proxy hides the cloud's internal structure from other clouds in a federation. The VAN proxies of different clouds communicate to ensure that VANs can extend across a cloud boundary while adhering to the limitations discussed previously.

Cross-cloud monitoring

Monitoring in a federated cloud introduces some interesting issues. Although the underlying cloud infrastructure is a distributed system, it is structured in a particular way, with one large set of machines acting as one cloud. Most of the monitoring data stays within the cloud because all of the service providers are within the cloud. Federated VMs are the exception. With many monitoring systems, the sources and consumers are distributed arbitrarily across a network, thus the dataflow paths and the interaction patterns are also arbitrary. Within Reservoir, the pattern is more predictable.

Cross-cloud federation of monitoring requires performing the following tasks:

- address the setup of federated monitoring when the first VM for a service arrives at a cloud;
- create the cloud-to-cloud connections for sending measurements back to the home cloud;
- address the tear-down of remote communication when the last VM for a service is migrated away from a cloud; and

- ensure that remote and connected VMs are kept separate from other services.

Within Reservoir, we built a distributed monitoring system that has all of the necessary monitoring probes and consumers, and supports the closed control loops for the virtual infrastructures, including service clouds and virtual networks. For the monitoring data plane, we use a combination of IP multicast and the Java Message Service publish-subscribe system. We also use JMS for federated monitoring, although cloud-to-cloud monitoring might use different protocols from intracloud monitoring to ensure interoperability.

Cross-cloud live migration

Live migration techniques require a direct communication link between the source and destination hypervisors. However, security and privacy considerations prevent a cloud provider from allowing other clouds direct access to its hypervisors.

To overcome this apparent contradiction, we introduced a novel federated migration channel to transfer VMs from a source host in one cloud to a destination host in another cloud without directly addressing the destination host. The VM passes througjh a secure tunnel connecting proxies in the source and destination clouds. At the destination site, the VM is forwarded to the chosen destination host.

ON-DEMAND ENTERPRISE SYSTEMS

SAP systems are used for various business applications that differ in version and functionality (such as customer relationship management and enterprise resource planning). We chose SAP as the main application to demonstrate the challenges for cloud computing providers because it helps raise the enterprise-grade requirements not captured in the typical Web-based applications prevalent as cloud-based offerings.

A cloud computing provider wishing to host SAP applications faces three main challenges:

- efficiently managing the life cycles of the different SAP applications for hundreds or thousands of tenants while maintaining a low total cost of ownership;
- consolidating many applications on the same infrastructure, thereby increasing hardware utilization and optimizing power consumption, while keeping a site's operational cost to a minimum; and
- guaranteeing the individual SLAs of the infrastructure customers (that is, the service providers).

The SAP use case applied Reservoir's federation capabilities for operational flexibility. We successfully experimented with scenarios of initial deployment of a multi-VM application across multiple datacenters. We also successfully exercised automated elasticity to respond to changes in application load.

Our experiments showed that deploying enterprise-grade complex applications in a federated cloud is feasible. However, doing so is not without problems and limitations.

For example, when dealing with a complex multi-VM application, we encountered certain technical problems that stretched the naïve mechanisms of rapid provisioning and elasticity to the limit. For example, the SAP system requires a special start-up sequence, and SAP licensing is coupled to the identity of a real machine. In a cloud infrastructure, the application runs in a VM.

We also found that some of the biggest obstacles in our experiments were the images' size, the time it takes to create them, and the time it takes to start an application from an image. For example, an image for the database management system component can use more than 100 Gbytes and take a few minutes to start.

> **Because the potential flexibility and cost savings are limited in private clouds, we are now seeing the rise of the hybrid cloud computing model.**

Finally, SAP applications use stateful and sticky sessions. That is, once a session is opened with a specific user, a state is maintained for that user in a specific server. As a consequence, even when the number of active sessions decreases, the SAP sessions can be spread across servers, making it difficult to scale down the number of servers.

Cloud computing is not a passing phenomenon. Although companies might still be reluctant to fully embrace the hosted model the technology presents, they are adopting cloud computing methodologies to organize their own datacenters into private clouds. Because the potential flexibility and cost savings are limited in private clouds, we are now seeing the rise of the hybrid cloud computing model. In this model, companies have their own private clouds, but transfer some of their computing needs to a hosted public cloud as needed.[12] This is essentially a partial realization of the Reservoir federated cloud computing model. All market indications show that this trend will continue.

Although we believe that it is only natural that cloud computing providers will eventually reach their optimal capacity and adopt the federated cloud model, we are still a long way from instituting this model, particularly regarding standardization. Contemporary cloud technologies were not designed with interoperability in mind. But, just as with other utilities we get service with standard equipment, not specific to any provider (such as telephones). Without knowing the internal workings of the utility provider, for cloud computing services to fulfill the computing as a utility vision, providers will need to offer standardized services. This, in turn, will accelerate the federated model's adoption. The Reservoir project shows that, even within the limitations of today's technologies (such as the lack of interoperability between hypervisors), a federated cloud has huge potential.

Finally, we have also shown that deploying and running existing enterprise-grade applications (which were not originally designed for the cloud) is possible, although this is not a straightforward process. A new generation of cloud-native business applications will likely emerge. Such applications might further utilize the unique capabilities of clouds—for example, live migration across clouds. In the meantime, enterprises should adopt a model that is a hybrid of on-premises and public on-demand models to fully leverage the cloud computing paradigm's benefits while maintaining their current investments.

Acknowledgments

We thank all members of the Reservoir consortium, whose contributions made this article possible. The research leading to these results is partially supported by the European Community's Seventh Framework Programme (FP7/2001-2013) under grant agreement no. 215605.

References

1. R.M. Fano, "The MAC System: The Computer Utility Approach," *IEEE Spectrum*, Jan. 1965, pp. 56-64.
2. M. Armbrust et al., *Above the Clouds: A Berkeley View of Cloud Computing*, tech. rep., EECS Dept., Univ. of Calif., Berkeley, 2009.
3. I. Foster, C. Kesselman, and S. Tuecke, "The Anatomy of the Grid: Enabling Scalable Virtual Organizations," *Int'l J. High-Performance Computing Applications*, vol. 15, no. 3, 2001, pp. 200-222.
4. B. Rochwerger et al., "The Reservoir Model and Architecture for Open Federated Cloud Computing," *IBM J. Research and Development*, vol. 53, no. 4, 2009; http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=5429058.
5. A. Kivity et al., "KVM: The Linux Virtual Machine Monitor," *Proc. Linux Symp.*, 2007, pp. 225-230; www.linuxsymposium.org/archives/OLS/Reprints-2007/nakajima-Reprint.pdf.
6. C. Chapman et al., "Software Architecture Definition for On-Demand Cloud Provisioning," *Proc. 19th ACM Int'l Symp. High-Performance Distributed Computing*, ACM Press, 2010, pp. 61-72.
7. Distributed Management Task Force, "Open Virtualization Format Specification," DSP0243 v1.0.0, 2008; www.dmtf.org/standards/ovf
8. G. Toffetti et al., "Engineering Autonomic Controllers for Virtualized Web Applications," *Proc. Web Eng.*, LNCS 6189, Springer, 2010, pp. 66-80.
9. D. Breitgand and A. Epstein, "SLA-Aware Placement of Multi-Virtual Machine Elastic Services in Compute Clouds", *Proc. IFIP/IEEE M 2011—Special Track on Management of Cloud Services and Infrastructures*, 2011 (to appear).
10. D. Breitgand et al., *Policy Driven Service Placement Optimization in Federated Clouds, Israel,* tech. report H-0299, IBM*,* 2011.
11. D. Hadas, S. Guenender, and B. Rochwerger, *Virtual Network Services for Federated Cloud Computing*, tech. report H-0269, IBM, 2009.
12. D. Linthicum, "The Case for the Hybrid Cloud," *InfoWorld*, 30 Mar. 2010; www.infoworld.com/d/cloud-computing/the-case-the-hybrid-cloud-196.

***Benny Rochwerger** is a senior technical staff member at the IBM Haifa Research Lab, Israel. His research interests include cloud computing, distributed systems, and networking virtualization. Rochwerger received an MS in computer science from the University of Massachusetts Amherst. Contact him at rochwer@il.ibm.com.*

***David Breitgand** is a research staff member at the IBM Haifa Research Lab. His research interests include end-to-end performance analysis and management of networked storage and systems. Breitgand received a PhD in computer science from the Hebrew University of Jerusalem.*

***Amir Epstein** is a research staff member at the IBM Haifa Research Lab. His research interests include approximation methods, algorithmic game theory, and cloud computing. Epstein received a PhD in computer science from Tel Aviv University.*

***David Hadas** is a research staff member at the IBM Haifa Research Lab. His research interests include distributed networking, machine learning, and biological learning processes. Hadas is pursuing an MSc in computer science from Tel Aviv University.*

***Irit Loy** is a research staff member at the IBM Haifa Research Lab. Her research interests include cloud computing, parallel and distributed systems, and file systems. Loy received an MSc in electrical engineering from the Israel Institute of Technology (Technion), Haifa.*

Kenneth Nagin *is a research staff member at the IBM Haifa Research Lab. His research interests include software-based test-generation consultation, cloud computing, and virtualization. Nagin received a BS in computer science from the University of Pittsburgh.*

Johan Tordsson *is a researcher at Umeå University, Sweden. His research interests include automatic resource management for grid and cloud computing environments. Tordsson received a PhD in computer science from Umeå University.*

Carmelo Ragusa *is a researcher at the University of Messina, Italy. His research interests include distributed computing, cloud computing, grids, and SOA. Ragusa received a PhD in communication systems from the University of Surrey, UK.*

Massimo Villari *is an assistant professor at the University of Messina. His research interests include cloud computing and security in distributed systems. Villari received a PhD in electronic engineering from the University of Messina.*

Stuart Clayman *is a senior research fellow in the Department of Electronic Engineering at University College London. His research interests include distributed systems, management systems, and time-ordered data processing. Clayman received a PhD in computer science from University College London.*

Eliezer Levy *is the director of SAP Research in Israel. His research interests include infrastructures for business applications, databases, and distributed computing. Levy received a PhD in computer sciences from the University of Texas at Austin.*

Alessandro Maraschini *is a team leader and software engineer at Telespazio. His research interests include grid and cloud computing technology-based projects. Maraschini received an MSc in electronic engineering from the University La Sapienza, Rome.*

Philippe Massonet *is the scientific coordinator at CETIC, a Belgian ICT applied research center. His research interests include software/security engineering and distributed systems such as clouds and grids. Massonet received an industrial engineering degree from the Université Catholique de Louvain, Belgium.*

Henar Muñoz *is a research and development engineer at Telefónica I+D. Her research interests include cloud computing and semantics technologies. Muñoz received an MSc in telecommunication from the University of Valladolid, Spain.*

Giovanni Toffetti *is a postdoctoral researcher at the University of Lugano, Italy. His research interests include distributed systems, Web applications, and content-based routing. Toffetti received a PhD in computer science from Politecnico di Milano.*