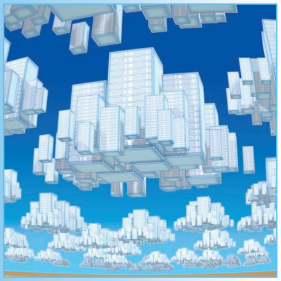

Plug into the Supercloud

Cloud computing is often compared to the power utility model, but today's cloud providers don't simply supply raw computing resources as a commodity; they also act as distributors, dictating cloud services that aren't compatible across providers. A supercloud is a cloud service distribution layer that's completely decoupled from the provider. Leveraging a nested paravirtualization layer called the Xen-Blanket, the supercloud maintains the control necessary to implement hypervisor-level services and management. Using the Xen-Blanket to transform various cloud provider services into a unified offering, the authors have deployed a supercloud across the Amazon Elastic Compute Cloud, an enterprise cloud, and Cornell University.

Dan Williams and Hani Jamjoom
IBM T.J. Watson Research Center

Hakim Weatherspoon
Cornell University

As part of the growing trend toward commoditizing computing resources, cloud computing is often compared to other utility models, such as electricity. Akin to power generators, cloud providers offer massive amounts of computing resources. However, unlike today's electricity utility model, in which consumers are generally agnostic as to where their power is generated, cloud users (consumers) are tightly coupled to providers' infrastructures and must adhere to those providers' varying specifications (such as the virtualization stack and management APIs). As it stands, today's cloud delivery model resembles the "War of the Currents" from the late 1880s,[1] in which direct current (DC) power distribution was tightly coupled to a local generator.

We're interested in building cloud infrastructures that are akin to Westinghouse's "universal system" — the foundation of modern power generation, distribution, and commoditization. In the universal system, Westinghouse showed how power — using Tesla's transformer — could be generated and consumed in many different voltages.[1] In such a model, consumers care only about power's availability because it can readily be transformed into the correct voltage. In this way, consumers can be completely agnostic to where and how the electricity is generated. The universal system thus demonstrates how to decouple power generation from consumption. This decoupling was a corner piece to creating power distribution networks and, ultimately, commoditizing electricity.

To solve the tight coupling of today's clouds, we propose *superclouds*, cloud distribution layers that aren't bound to any provider or physical resource. On the surface, supercloud users see a collection of computing resources, similar to today's clouds. Beneath the surface, a supercloud "transforms" multiple underlying cloud offerings into a universal cloud abstraction. The supercloud specifies and fully controls the entire cloud stack, independent from the provider infrastructure. A supercloud thus decouples cloud providers and users.

At first glance, a supercloud might seem to require radically redesigning today's clouds; it doesn't. Here, we demonstrate how to create superclouds on top of existing clouds by leveraging nested virtualization to perform a similar function to the Tesla transformer in Westinghouse's universal system. Toward this goal, we present the design of a nested paravirtualization hypervisor, called the Xen-Blanket, which can run on different providers, thus enabling superclouds to span clouds. Subsequently, a supercloud can manage guest virtual machines (VMs) across computing resources, irrespective of the provider's virtualization stack. For example, a supercloud can live-migrate VMs between cloud providers. It can also run any cloud management stack, including OpenStack (www.openstack.org), Eucalyptus (www.eucalyptus.com), or a completely custom stack.

We built a supercloud that spans several diverse environments by deploying the Xen-Blanket in each one. In particular, the Xen-Blanket runs on hypervisors based on Xen or the Linux Kernel-based Virtual Machine (KVM), public and private infrastructures within the Amazon Elastic Compute Cloud (EC2), an enterprise cloud, and Cornell University. Within the supercloud, we've migrated VMs to and from Amazon EC2 without needing to modify them; the cloud user need not be aware of which provider is supplying the resources supporting the VM. Furthermore, our supercloud exploits a resource oversubscription model that no cloud providers currently offer. Consequently, it can host 40 CPU-intensive VMs on EC2 for 41 percent of the price per hour of 40 small instances with matching performance.

Supercloud

A supercloud provides a uniform cloud service comprising resources obtained from several diverse infrastructure-as-a-service (IaaS)[2] cloud resource providers (see Figure 1). Here, we examine the role superclouds play in a utility model and the challenges we faced when designing and implementing them.

A Universal System for the Cloud

Utility models for resources in a universal system, such as electricity, include three distinct roles: generators, distributors, and consumers. At the back end, generators offer resources. A distributor consumes the raw resources as a commodity from various generators and builds a service around them. Then, consumers interact with the distributor service to use the resources.

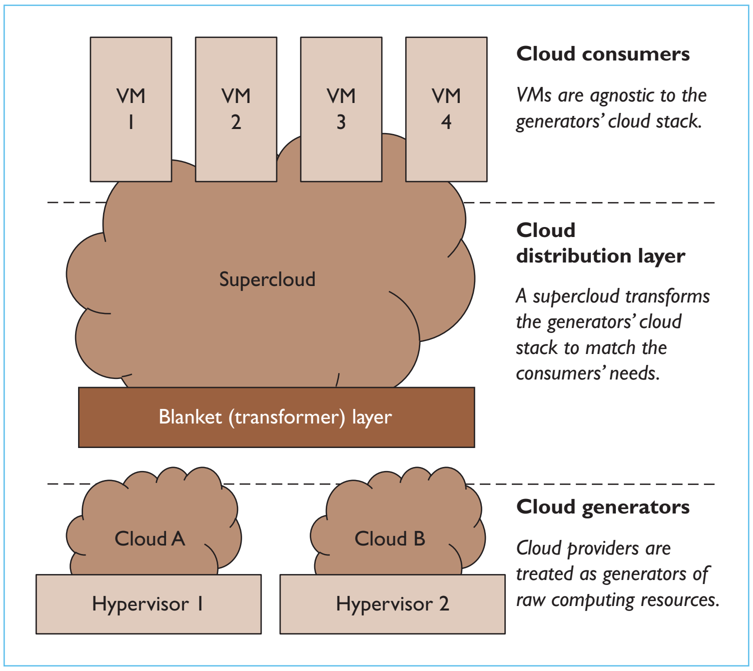
VM 1
VM 2
VM 3
VM 4
Cloud consumers
VMs are agnostic to the generators' cloud stack.
Supercloud
Cloud distribution layer
A supercloud transforms the generators' cloud stack to match the consumers' needs.
Blanket (transformer) layer
Cloud A
Cloud B
Hypervisor 1
Hypervisor 2
Cloud generators
Cloud providers are treated as generators of raw computing resources.

Figure 1. Supercloud overview. A supercloud creates a distribution layer between the cloud generator and consumers.

The common model used for IaaS clouds has only two entities: a cloud provider and a cloud user. Viewing the cloud as a utility, the cloud provider (such as Amazon EC2, Rackspace, or Google Compute Engine) acts as a generator by supplying computing resources as VMs. The cloud user acts as a consumer by instantiating VMs to run application workloads.

In today's clouds, the distributor role is assumed by either the cloud provider or a third party, but it's limited in either case. Some cloud providers act as both the generator and distributor. They fully control the physical resources and can thus implement rich features as services to users. However, these features are intrinsically bound to that single provider, preventing distribution services that span providers, such as tolerance against provider failure.[3] Alternatively, third-party vendors, such as RightScale (www.rightscale.com), might act as distributors. By interacting with multiple cloud providers, they offer cloud service and management features. However, these vendors lack

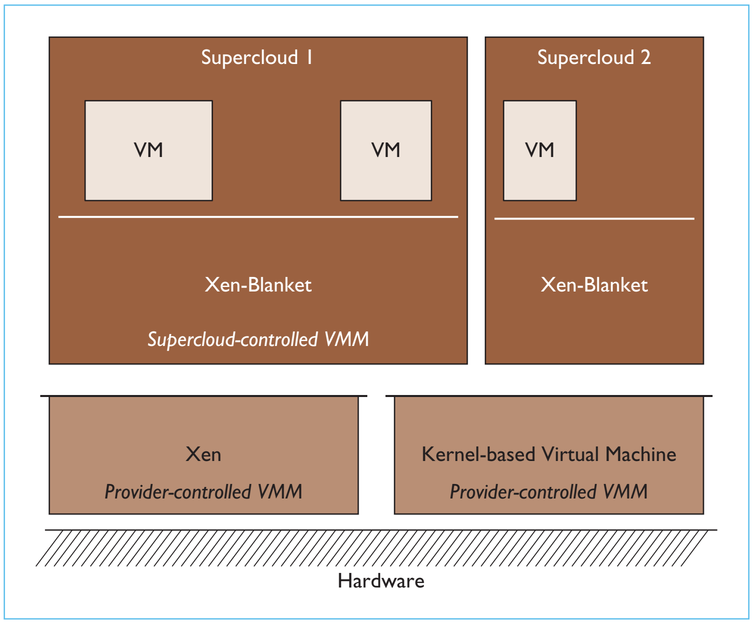

Figure 2. The Xen-Blanket. Controlled by the supercloud, the Xen-Blanket provides a distribution layer across heterogeneous clouds without requiring any additional support from providers.

control over the cloud provider's platform and ultimately can't add features such as live VM migration, CPU bursting, or transparent VM fault tolerance.

Today's cloud model thus lacks a robust way to create cloud distributors. Such distributors should be able to consume resources from multiple cloud providers (generators) while maintaining control over those resources. Put another way, what's missing is a distributor layer that decouples cloud generators from cloud consumers. The supercloud fills this role, treating cloud providers as interchangeable generators from which users can consume raw resources. A supercloud can exploit pricing strategies and spot markets for compute resources between cloud providers, replicate VMs between providers, and simplify management.

Challenges and Requirements

IaaS cloud providers are diverse and heterogeneous, with services tightly coupled to their physical resources. To act as an independent distributor, a supercloud must contain a transformer or a mechanism that completely decouples resources — including computation, network, and storage — from the physical infrastructure. As a first step, we focus on decoupling computation from the physical infrastructure. In IaaS clouds, computation (embodied by VMs) is tied to a cloud provider in two aspects, necessitating a transformer.

First, VM images that run on one cloud provider can't be easily instantiated on different clouds. For example, EC2 and Rackspace use different image formats: Amazon's EC2 Machine Instance (AMI) format and the Open Virtualization Format (OVF),[4] respectively. Paravirtualized device interfaces that VMs use are similarly diverse; VMs on EC2 and Rackspace use Xen and virtio, respectively. A supercloud must use computing resources from various cloud providers as a commodity; thus, it must decouple the VM image format from the provider.

Second, IaaS clouds are diverse in terms of the services they provide to VMs. For example, Amazon EC2 provides CloudWatch (integrated monitoring), AutoScaling, and Elastic Load Balancing, whereas Rackspace supports VM migration to combat server host degradation, and CPU bursting to borrow cycles from other instances. Moreover, resource management opportunities — in particular, tools that operate at the hypervisor level — aren't consistently available between providers. For example, no unified set of tools exists with which users can specify VM colocation on physical machines, page sharing between VMs, or resource oversubscription. A supercloud must enhance cloud provider services with the set of features they lack, such that services can function seamlessly regardless of where the cloud resources are obtained.

Our approach toward superclouds enables a universal system for a cloud utility by creating a distinct, feature-rich distributor that's completely separate from the generator.

Enabling a Supercloud

At its core, a supercloud leverages the Xen-Blanket, a nested virtualization system that can transform any provider-specific VM instance into a unified, distributor-specified one. As Figure 2 shows, nested virtualization consists of a second-layer hypervisor inside a VM instance, which is running on top of a (first-layer) hypervisor. In this model, the provider (the generator) continues to own first-layer hypervisors, while a supercloud (the distributor) owns second-layer ones. Also, in this model, different superclouds (that is, different distributors) can coexist.

A supercloud user (the consumer) runs VM instances and the cloud management stack on top of the second-layer Xen-Blanket hypervisor.

We refer to the second virtualization layer as the Blanket layer. By running the Xen-Blanket on top of different providers, the supercloud can consume resources from multiple cloud generators, offer them to its consumers, and seamlessly switch providers. More details of the Xen-Blanket design and implementation appear elsewhere.[5]

The Xen-Blanket Transformer

The Xen-Blanket encompasses two primary concepts. First, the top half exposes a single, supercloud-controlled VM interface and service suite to supercloud users such that a guest VM image can run on any provider infrastructure without modifications. Second, its bottom half communicates with the underlying hypervisor interface, which could vary depending on the provider. No modifications to the underlying hypervisor are expected or required.

Transformer top half. The top half of the Xen-Blanket exposes a consistent VM interface to supercloud users. A supercloud can thus place VMs on any provider that can run the Blanket layer without modifications. To maximize the number of clouds on which the Blanket layer can run, this layer doesn't need the provider to expose state-of-the-art nested virtualization interfaces (as with the Turtles Project[6]). Instead, it relies on other x86 virtualization techniques, such as paravirtualization or binary translation. For our prototype implementation, we adopted the popular open source Xen hypervisor, which uses paravirtualization techniques when virtualization hardware isn't available. The Xen-Blanket subsequently inherits paravirtualization's limits, most notably its inability to run unmodified operating systems such as Microsoft Windows. However, this limitation isn't fundamental. We can construct a Blanket layer using binary translation (for example, a VMWare-Blanket[7]), on which unmodified operating systems would be able to run. We can also create Blanket layers with other interfaces or even customized hypervisors developed from scratch.

Transformer bottom half. The Xen-Blanket's bottom half ensures that a supercloud can span several different clouds without requiring changes to the underlying cloud system or hypervisor. We assume that hardware-assisted full virtualization for the x86 (called a hardware virtual machine, or HVM, in Xen terminology) is available from cloud providers. However, we don't assume that device I/O is emulated, so the Blanket hypervisor must be aware of the underlying hypervisor's paravirtualized I/O interfaces. The Xen-Blanket interfaces with various underlying paravirtualized device I/O implementations. Paravirtualized device I/O has proven essential for performance, and some clouds require it, such as Amazon EC2. However, no standard paravirtualized device I/O interface currently exists. For example, older Xen-based clouds, including Amazon EC2, require device drivers to communicate with Xen-specific subsystems, such as the XenBus and XenStore, whereas KVM-based systems expect device drivers to interact with the hypervisor via virtio interfaces. The Xen-Blanket supports such nonstandard interfaces by modifying the bottom half to contain cloud-specific Blanket drivers.

Discussion

The Xen-Blanket transforms provider-specific VM image formats and hypervisor environments into a common distributor-defined format. However, for a supercloud to completely decouple cloud generators from consumers, it must also decouple the supporting provider infrastructure. In particular, a supercloud must provide a uniform mechanism for locating VMs on the network and interacting with storage.

Cloud providers offer virtual network abstractions that decouple network addresses from the cloud infrastructure. For example, Amazon's Virtual Private Cloud (VPC) lets users extend their private networks into the cloud. After eliminating technical addressing challenges, a supercloud might need to implement a policy to ensure that, even as VMs migrate between providers, heavily communicating VMs remain colocated on the same cloud provider.[8]

Solving network addressing issues lets VMs access storage across the network from anywhere within the supercloud. However, accessing storage that resides on another cloud provider will result in poor performance. A supercloud can employ caching and replication techniques and balance performance at the cost of storing and transferring data between providers. It could even use RAID-like techniques across providers.[3]

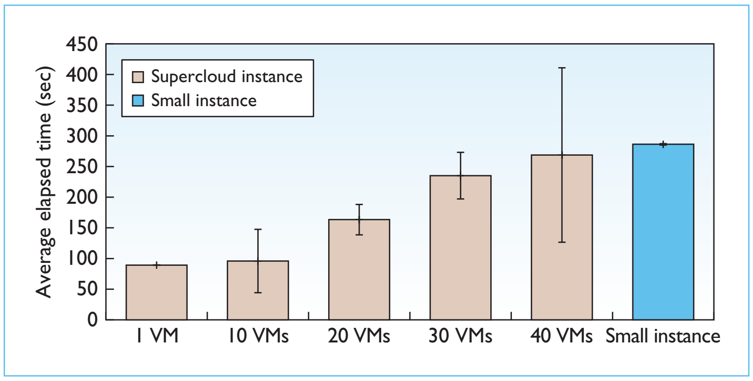

Figure 3. Average elapsed time to run simultaneous kernbench benchmarks. The Xen-Blanket lets a supercloud oversubscribe cloud resources such that each of the 40 virtual machines (VMs) on a quadruple extra-large cluster compute (4XL) instance can simultaneously complete compilation tasks in the same amount of time as a small instance.

Evaluation and Experience

We built a supercloud using the Xen-Blanket across three clouds; our extensive evaluation appears elsewhere.[5] In short, the Xen-Blanket introduces some overhead due to the second layer of virtualization. Microbenchmarks show that it introduces 3 percent overhead for simple operations and up to 12.5 percent for context switch microbenchmarks. In the worst case, overheads can rise to 68 percent due to advanced programmable interrupt controller (APIC) emulation for guest VMs with many virtual CPUs. However, Blanket drivers ensure that I/O performance is good: network drivers can receive packets at line speed on a 1-Gbps link, while disk I/O throughput is within 12 percent of single-level paravirtualized disk performance. Furthermore, the Xen-Blanket's performance is sufficient for common applications such as Web servers (demonstrated with the SPECweb2009 benchmark). Given Xen-Blanket's performance, our evaluation here focuses on two supercloud features: resource oversubscription and cross-provider live VM migration.

Oversubscription

We evaluated oversubscription in a Xen-Blanket-based supercloud on Amazon EC2 and showed that the supercloud can host CPU-intensive VMs at 41 percent of the cost of native EC2. The supercloud exploits the pricing per hour on Amazon EC2 to rent a small instance versus a quadruple extra-large cluster compute instance (cluster 4XL). In particular, as of July 2012, although the cluster 4XL instance is almost a factor of 16.5 times more expensive than a small instance ($0.08 versus $1.30 per hour), some resources are greater than 16.5 times more abundant (for example, 33.5 times more for CPU); others are less than 16.5 times more abundant (10.5 times more for disk). This suggests that if a cloud user has several CPU-intensive VMs normally serviced as small instances, it might be more cost-effective to rent a cluster 4XL instance and oversubscribe the memory and disk. This isn't an option that Amazon provides. However, a supercloud (using the Xen-Blanket) can implement such a configuration.

To illustrate this point, we ran a CPU-intensive macrobenchmark, kernbench, simultaneously in varying numbers of VMs running on a supercloud comprising a single cluster 4XL instance running the Xen-Blanket. We also ran the benchmark inside a small EC2 instance for a comparison point. Figure 3 shows the elapsed time to run the benchmark in each scenario. Each number of VMs on the supercloud corresponds to a different monetary cost. For example, to run a single VM, the cost is $1.30 per hour, while running 40 VMs reduces the cost per VM to $0.0325 per hour. Running a single VM, the benchmark completes in 89 seconds on the supercloud, compared to 286 seconds for a small instance. This is expected, because the cluster 4XL instance is significantly more powerful than a small instance. Furthermore, the average benchmark completion time for even 40 VMs remains 33 seconds faster than for a small instance, although the variance of the benchmark performance significantly increases for large numbers of VMs on the same instance. Because a small instance costs $.08 per VM per hour, this translates to roughly 41 percent of the price per VM per hour.

Cross-Provider Live Migration

Although migrating VMs between multiple clouds is possible, the process is cloud-specific and fundamentally limited. To the best of our knowledge, our supercloud, using the Xen-Blanket, is the first implementation that can perform live VM migration between arbitrary cloud providers (including Amazon EC2). As such, we don't have a comparison point to present.

Live migration typically relies on memory tracing: a hypervisor-level technique that the Xen-Blanket inherits from Xen. Additionally, the supercloud must provide a uniform network

Related Work toward Decoupling Cloud Services

Several techniques exist for deploying applications on multiple clouds. Although they're positive steps toward creating superclouds, none afford the user the flexibility or level of decoupling of the Xen-Blanket on today's public clouds.

Using tools from RightScale (www.rightscale.com), users can create ServerTemplates, deploy them on various clouds, and use clouds' unique features without sacrificing portability. However, RightScale can't perform hypervisor-level services across providers, such as live virtual machine (VM) migration. The Reservoir project is a multicloud agenda to federate two or more independent cloud providers.[1] However, standardization is necessary before federation can extend beyond the testbed. Like a supercloud, fos aims to expose a coherent environment that spans cloud resources and is deployed on the Amazon Elastic Compute Cloud (EC2).[2] However, fos exposes a single system image, forgoing the familiar VM interface and legacy applications contained within.

Through the Xen-Blanket, superclouds leverage nested virtualization. The Turtles Project enables nested virtualization with one or more levels of full virtualization on Intel hardware.[3] Olivier Berghmans describes the performance of several nested virtualization environments.[4] VMworld Hands-On Labs (HOLs) use VMware ESX in nested mode to produce a private lab environment for each participant.[5] The Xen-Blanket sacrifices full nested virtualization for immediate deployment of a supercloud on a variety of existing clouds.

References

1. B. Rochwerger et al., "Reservoir — When One Cloud Is Not Enough," *Computer*, vol. 44, no. 3, 2011, pp. 44–51.
2. D. Wentzlaff et al., "An Operating System for Multicore and Clouds: Mechanisms and Implementation," *Proc. ACM Symp. Cloud Computing*, ACM, 2010, pp. 3–14.
3. M. Ben-Yehuda et al., "The Turtles Project: Design and Implementation of Nested Virtualization," *Proc. Usenix Operating Systems Design and Implementation Conf.*, Usenix Assoc., 2010, article no. 1–6.
4. O. Berghmans, "Nesting Virtual Machines in Virtualization Test Frameworks," master's thesis, Dept. of Mathematics and Computer Science, Univ. of Antwerp, May 2010.
5. A. Zimman, C. Roberts, and M.V.D. Walt, "VMworld 2011 Hands-On Labs: Implementation and Workflow," *VMware Technical J.*, vol. 1, no. 1, 2012, pp. 70–76.

and storage environment (as we discussed earlier) to perform live multicloud migration.

Within the Xen-Blanket, we implement a proof-of-concept virtual network and shared storage abstraction, shown in Figure 4. Each Xen-Blanket instance within the supercloud runs a virtual switch in Domain 0 to which we attach the virtual network interfaces belonging to Blanket guest VMs. A layer-2 tunnel connects the virtual switches across the Internet. The result is that VMs on either of the two Xen-Blanket instances appear to be sharing a private LAN. We introduce a few basic network services onto the virtual network. We run a gateway server VM with two virtual network interfaces: one attached to the virtual switch and the virtual network, and the other attached to the Xen-Blanket instance's externally visible interface. The gateway server VM runs dnsmasq as a lightweight Dynamic Host Configuration Protocol and DNS server. It also runs an NFS server that exports files onto the virtual network and is mounted by each Xen-Blanket instance's Domain 0. Both Xen-Blanket instances mount the NFS share at the same location. So, during VM migration, the VM root file-system image can always be located at the same file-system location, regardless of the physical machine.

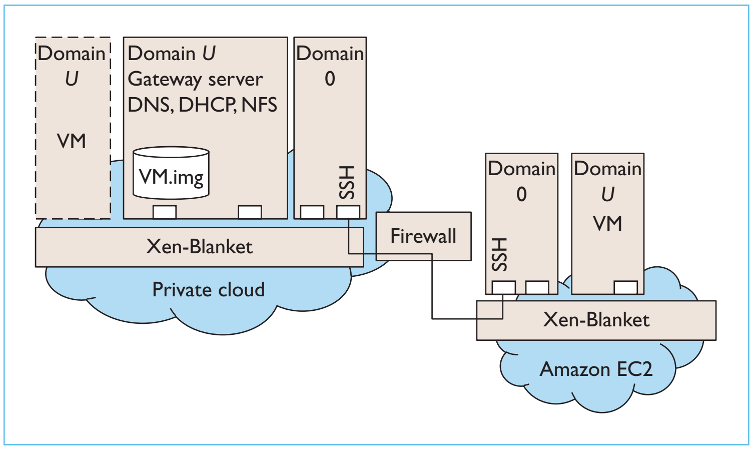

Figure 4. Live migration. Xen-Blanket instances are connected with a layer-2 tunnel, while a gateway server virtual machine provides DNS, Dynamic Host Configuration Protocol, and NFS servers to the virtual network, eliminating the communication and storage barriers to multicloud live migration.

With VMs anywhere in the supercloud able to communicate, maintain their network addresses, and access storage within either cloud, live VM migration proceeds by following the typical procedure in the Blanket hypervisor. However, although we've successfully live-migrated a VM from an enterprise cloud to Amazon EC2 and back, relying on an NFS disk image potentially residing on another cloud instead of a local disk is clearly inefficient. Moreover, the layer-2 tunnel only connects two machines. As future work, we can augment the supercloud with more sophisticated network virtualization, storage, and wide-area live migration techniques.

Through the Xen-Blanket, superclouds maintain the hypervisor-level control necessary to implement rich cloud services. Moreover, the Xen-Blanket requires no modifications to existing cloud provider infrastructures, enabling superclouds to be deployed today.

Using the Xen-Blanket, we've experimented with developing supercloud services that oversubscribe resources to provide low-cost VMs for CPU-intensive jobs and migrate VMs between cloud providers without requiring downtime or modifications. However, we've just scratched the surface in terms of the applications and functionality that we can implement in a supercloud. In the future, superclouds will offer complete cloud management stacks, innovative or experimental cloud services, or custom features for a specific class of application, all spanning multiple providers.

The Xen-Blanket project website is located at http://xcloud.cs.cornell.edu, and the code is publicly available at http://code.google.com/p/xen-blanket.

Acknowledgments

This work was partially funded and supported by an IBM Faculty Award received by Hakim Weatherspoon, DARPA (number D11AP00266), US National Science Foundation CAREER (number 1053757), NSF TRUST (number 0424422), NSF FIA (number 1040689), NSF EAGER (number. 1151268), and NSF CiC (number. 1047540). We thank the anonymous reviewers for their comments.

References

1. J. Jones, *Empires of Light: Edison, Tesla, Westinghouse, and the Race to Electrify the World*, Random House, 2004.
2. P. Mell and T. Grance, *The NIST Definition of Cloud Computing*, Special Publication 800-145, US Nat'l Inst. Standards and Technology, Sept. 2011.
3. H. Abu-Libdeh, L. Princehouse, and H. Weatherspoon, "RACS: A Case for Cloud Storage Diversity," *Proc. ACM Symp. Cloud Computing*, ACM, 2010, pp. 229–240.
4. "Open Virtualization Format," white paper version 1.00, Distributed Management Task Force, Feb. 2009; www.dmtf.org/sites/default/files/standards/documents/DSP2017_1.0.0.pdf.
5. D. Williams, H. Jamjoom, and H. Weatherspoon, "The Xen-Blanket: Virtualize Once, Run Everywhere," *Proc. ACM EuroSys*, ACM, 2012, pp. 113–126.
6. M. Ben-Yehuda et al., "The Turtles Project: Design and Implementation of Nested Virtualization," *Proc. Usenix Operating Systems Design and Implementation Conf.*, Usenix Assoc., 2010, article no. 1–6.
7. J. Sugerman, G. Venkitachalam, and B.-H. Lim, "Virtualizing I/O Devices on VMware Workstation's Hosted Virtual Machine Monitor," *Proc. Usenix Ann. Technical Conf.*, Usenix Assoc., 2001, pp. 1–14.
8. V. Shrivastava et al., "Application-Aware Virtual Machine Migration in Data Centers," *Proc. IEEE Computer Communications Mini-Conf.* (INFOCOM), IEEE, 2011, pp. 66–70.

Dan Williams is a research staff member in the Cloud Platforms and Transformation group at the IBM T.J. Watson Research Center. His research interests span cloud computing, virtualization, networking, and operating systems. Williams has a PhD in computer science from Cornell University. Contact him at djwillia@us.ibm.com.

Hani Jamjoom is a research manager for the Cloud Platforms and Transformation group at the IBM T.J. Watson Research Center. His research interests include hybrid clouds, network and nested virtualization, and workload migration across cloud platforms, looking at various levels of the application and system stack. Jamjoom has a PhD in computer science from the University of Michigan. He's received two Outstanding Technical Achievement awards, an Outstanding Innovation award, and two Research Division awards among others for his technical contributions to IBM's products and services. Contact him at jamjoom@us.ibm.com.

Hakim Weatherspoon is an assistant professor in the Department of Computer Science at Cornell University. His research interests cover various aspects of fault-tolerance, reliability, security, and performance of large Internet-scale systems such as cloud computing and distributed systems. Weatherspoon has a PhD in computer science from the University of California, Berkeley. He's an Alfred P. Sloan Fellow and recipient of a US National Science Foundation (NSF) CAREER award, DARPA Computer Science Study Panel (CSSP), IBM Faculty Award, the NetApp Faculty Fellowship, Intel Early Career Faculty Honor, and NSF Future Internet Architecture award. Contact him at hweather@cs.cornell.edu.