Supercloud: Opportunities and Challenges

Qin Jia   Zhiming Shen   Weijia Song   Robbert van Renesse   Hakim Weatherspoon

Cornell University

ABSTRACT

Infrastructure as a Service (IaaS) clouds couple applications tightly with the underlying infrastructures and services. This *vendor lock-in* problem forces users to apply ad-hoc deployment strategies in order to tolerate cloud failures, and limits the ability of doing virtual machine (VM) migration and resource scaling across different clouds.

This paper presents the *Supercloud*, a cloud service comprising resources obtained from several diverse IaaS cloud providers, and discusses opportunities, limitations, and future research directions. Currently, the Supercloud has been deployed using resources from several major cloud providers, including Amazon EC2, Rackspace, HP Cloud, and some private clouds. VMs run in a virtual network and can be migrated seamlessly across different clouds, with different hypervisors and device models. Using case studies we demonstrate that, being able to deploy applications to more regions and granting more control to end-users, the Supercloud can reduce latency and cost compared to the underlying cloud providers.

1. INTRODUCTION

The cloud paradigm, an elastic cluster of computing and storage resources, is wildly successful. Cloud computing promises to catalyze the technology economy, revolutionize health care, military, government, financial systems, scientific research, and society. Unfortunately, current realizations of public and private cloud have limitations. New applications demand properties that today's cloud platforms either struggle to provide efficiently or lack altogether: robustness and availability despite failure or attack, security of data and integrity of computation, energy efficiency, low latency and high bandwidth to clients worldwide, low cost through efficient use of spot markets, and so on. Many of these disadvantages result from clouds generally being under a single administrative domain.

There are various public Infrastructure as a Service (IaaS) cloud offerings such as Amazon elastic compute cloud (EC2), Rackspace, Windows Azure, Google Compute Engine, and the HP Cloud. Each provides on-demand computation and storage to users and charges according to the time the resources are used. Virtualization techniques ensure isolation between tenants. Datacenters in various locations host vir-
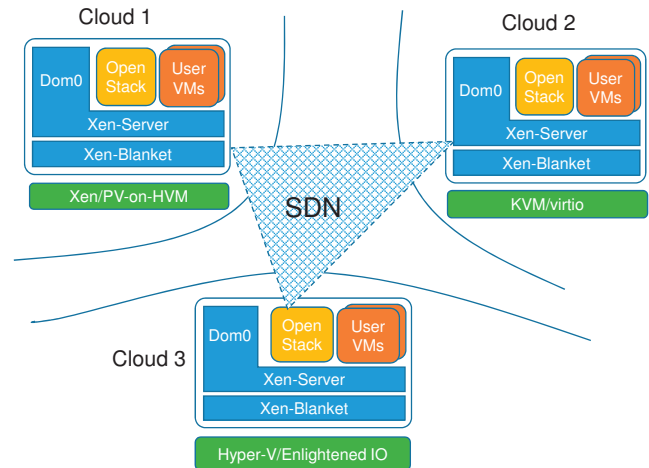
Figure 1: Example Supercloud deployment.

tual machines and storage. If applications could be built out of resources obtained from various providers and even private clouds, various potential benefits would emerge:

- *Proximity*: clients would connect to the nearest data center, independent of the provider, to obtain the lowest latency and highest bandwidth;

- *Cost*: prices at providers vary, even over time, and services could migrate their virtual resources to take advantage of low prices;

- *Availability*: resources at different providers are more likely to fail independently than resources at the same provider;

- *Security*: sensitive data could be placed and processed based on trust, possibly using relatively scarce private resources, while public resources are used for less sensitive operations;

- *Overload handling*: an overloaded private cloud could harness public resources to handle excess load.

Unfortunately, incompatibility among different cloud providers prevents such use. For example, Amazon EC2, HP Cloud, and Windows Azure are built on top of different hypervisors (Xen, KVM, and Hyper-V, respectively). They use different virtual machine (VM) image formats, and it is not possible to live migrate a VM from one type of hypervisor to another. They also provide different interfaces to infrastructure such as storage, and do not support networking between virtual machines in different clouds. As a result, applications are *locked-in* to a specific cloud, which makes using resources from multiple cloud providers very difficult.

Even within a single cloud provider, it can be difficult to take advantage of available diversity. For example, cloud providers maintain multiple geographic *regions* and one or more *availability zones* within a region. Availability zones are designed to be failure independent. Amazon makes underutilized resources in certain availability zones available at a low price, but it does not support migration of virtual machines between availability zones.

The Supercloud [15] is a cloud that uses resources from various private and public clouds. Figure 1 shows an example of a Supercloud deployment. The Supercloud uses nested machine virtualization in the form of the Xen-Blanket [14] hypervisor, offering uniformity in virtual devices and VM images. Nested virtualization paves over heterogeneity such as different virtual machine monitors and image representations as well as small differences in hardware.

Thus far, we have implemented Xen-Blanket on top of Xen and KVM. We have demonstrated live migration of virtual machines, for example, from a local cluster to Amazon EC2 to Rackspace to HP Cloud to Cornell's public Red Cloud and back to the local cluster. The Supercloud uses Open vSwitch devices to provide virtual networking between virtual machines and to support live migration.

In addition, the Supercloud provides virtual storage that spans the various underlying clouds. Currently we do this through a shared NFS server. The Supercloud uses OpenStack to manage compute resources from different providers and to provide the API for managing the second layer VMs.

Table 1 shows the underlying clouds that the Supercloud currently supports or that we hope to support soon. Fractus is a private cloud owned by our group at Cornell University. It runs the Eucalyptus cloud management platform on KVM.

This paper presents various use cases that demonstrate some of the advantages that the Supercloud has over traditional public or private clouds, describes its current status, and discusses some of the challenges that remain.

2. CASE STUDIES

In this section we describe and evaluate two examples of how users can take advantage of the unique features of the Supercloud. The first example reduces the cost of running services in the cloud by exploiting Amazon's spot market pricing and the Supercloud's ability to migrate between availability zones, while the second uses multiple clouds to improve access latency for clients. Both use cases would be hard to realize without support of the Supercloud infrastructure.

2.1 Smart Spot Instances

Most cloud resources are currently priced on an *on-demand* basis: the provider guarantees that the resources will always be available until the user terminates them bar unforeseen outages. These resources have stable prices.

Amazon EC2 also provides a *spot market* that allows users to take advantage of computing resources in underutilized availability zones. Such spot virtual machine instances perform the same as on-demand instances but are often significantly cheaper. However, the price of a spot instance can change rapidly—sometimes every 5 minutes, and occasionally the price can grossly exceed the price of an on-demand

Provider	Hypervisor	Status
Amazon EC2	Xen	Enabled
Rackspace	Xen	Enabled
HP Cloud	KVM	Enabled
Fractus	KVM	Enabled
Google Compute Engine	KVM	In-progress
Windows Azure	Hyper-V	In-progress

Table 1: Current deployment of Supercloud

instance. Users specify a maximum bid price when starting the instance, and Amazon will automatically terminate the instance when the price exceeds the maximum bid. Spot instances thus have to be prepared for termination at any time, which severely limits usability. This is exacerbated by the fact that cloud providers, including Amazon, do not support user-level migration.
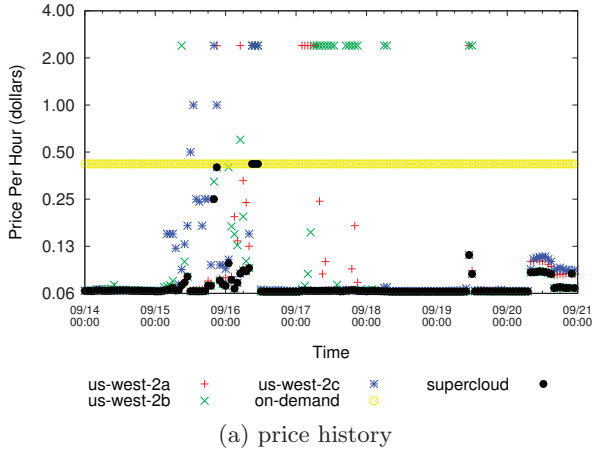
Because the Supercloud does support "user-level" migration, users are able to take advantage of spot market pricing with little possibility of being terminated at unpredictable times. Importantly, the Supercloud can always leverage the cheapest resources provided by Amazon spot instances. As the spot instance is charged according to the price at the beginning of each billing hour, a Supercloud VM can be migrated to the cheapest availability zone.

Furthermore, the Supercloud can reduce the chance of VM instance termination. Users are able to set up a higher maximum bid, migrating the VM when the price exceeds a certain threshold (which can be set to a value lower than the maximum bid). As a result, the probability of unforeseen termination can be significantly reduced because a VM can be migrated before the price exceeds the maximum bid. Furthermore, charged costs are constant since Amazon only charges at the beginning of the billing hour, so migrations would occur once per hour at most. As long as an instance is migrated within the same cloud (Amazon in this case), networking charges would be small compared to migrating across different clouds. Amazon charges $0.01/GB, so, for example, migrating a 2GB VM across availability zones would cost $0.02.
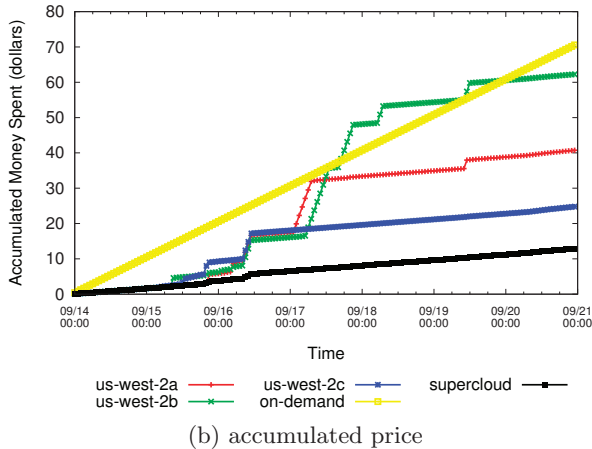
To validate the potential, we did a study of the price history of c3.2xlarge Linux spot instances from Sep. 14 2014 00:00 to Sep. 21 2014 00:00 in the Amazon Oregon region. As shown in Figure 2(a), the price at instances in availability zone `us-west-2a` changed dramatically from $0.065 to $2.40 per hour while the regular on-demand instance always cost $0.41 per hour.

Next, we approximate what the price of a smart spot instance would be. We ignore some overhead, such as being charged twice for an instance during a migration (typically a few minutes). The Supercloud would be able to keep the price down around $0.065, even as some availability zones were extremely expensive. On Sep. 16, the spot instances in all three availability zones were more expensive than normal instances, so the Supercloud can migrate the VMs to regular on-demand instances.

Figure 2(b) shows the total money spent since launching a spot VM instance in the Supercloud versus the cost of spot instances in each of the availability zones or an on-demand instance. In `us-west-2c` and `us-west-2b`, the total cost was $24.78 and $65.26, respectively, which were the lowest and highest cumulative spot instance prices. The Supercloud

(a) price history

(b) accumulated price

Figure 2: Saving money using Amazon spot instances.

spot instance cost only $12.92, which was 1.9× to over 5.1× cheaper than any spot instance. Furthermore, a regular on-demand instance would have cost $70.56 during the same period, 5.5× more than the Supercloud.

2.2 Improving Proximity to Clients

We now demonstrate that the Supercloud can reduce latency to end-users by enabling clients to access the closest datacenter of the closest cloud provider. For this case study, we used PlanetLab nodes to serve as end-user clients. PlanetLab has nodes spread across the world on six of the seven continents. We selected 16 such PlanetLab nodes (see the x-axis of Figure 3). Furthermore, we spawned a VM instance in various cloud provider regions. Next, each of the 16 PlanetLab nodes executed network measurements to a VM spawned in each of the cloud provider regions.

We measured latency using UDP round-trip-times (using small UDP packets) and the number of hops (using `tracer-oute`). To do the latency measurements, we wrote a simple UDP echoing service. We first booted a VM into an ordinary Linux kernel and measured the performance by deploying the service in the *first layer VM* directly. We then rebooted the same VM in a Xen-Blanket environment's *second layer VM* and opened access to the service by mapping a port in Xen-Blanket's Domain-0 to a first layer VM. We adopted

this option because only Domain-0 has a public IP address allocated by the underlying first layer cloud infrastructure VM.[1] See Figure 1 for the architecture showing the first layer VM (in green) and second layer (in blue and orange), and the second layers' (Xen-Blanket) Domain-0.

Figure 3 shows the minimum of the latencies that the PlanetLab nodes measured to the closest availability zones of the different clouds. For example, a client in United Kingdom (`cl.uk`) can receive the best latency from an Amazon datacenter in Ireland. However, Amazon does not have a site in Hong Kong; instead, the Hong Kong (`cuhk.hk`) client obtains better latency from Rackspace than from Amazon. Note that in our current prototype we sometimes experience a penalty for adding a second layer of virtualization (see `cuhk.hk`). Nonetheless, with the ability to deploy a service on all clouds, the Supercloud can closely match the best latency of any provider. Table 2 shows the aggregate latency and number of hops over all clients.

3. STATUS AND CHALLENGES

Broadly speaking, a cloud consists of computing, storage, and networking resources, and the Supercloud is no exception. Below we will discuss each in turn, describing its current status and what needs to be addressed.

3.1 Supercloud computing

The Supercloud requires nested virtualization. We use Xen-Blanket [14], a nested virtualized Xen hypervisor. Xen-Blanket runs on various widely supported hardware virtualized VMs (HVM) and provides a para-virtualized (PV) environment to run (second layer) guest VMs. Xen-Blanket leverages the `PV-on-HVM` drivers on Xen, `virtio` drivers on KVM, and `enlightened I/O` drivers on Hyper-V to optimize performance. It makes the underlying (heterogeneous) infrastructure transparent and appears as a homogeneous Xen environment to a second layer guest VM. The key is that a user can control not only second layer guest VMs, but second layer Xen-Blanket hypervisors, thus enabling a Supercloud to be a "user-controlled" cloud. However, the underlying heterogeneity, extra layer of indirection, management software, and scheduling policies pose challenges to a Supercloud.

One particular case of heterogeneity that is difficult to handle is supporting different versions of underlying processors with different CPU flags, which can prevent migration. The issue is that CPUID faulting is not available to a second-layer hypervisor. To address this issue, we modified the CPUID hook in Xen-Blanket to only expose a subset of CPU flags that are common in modern CPUs. Therefore, applications running in the Supercloud see the same CPU flags everywhere.

Another challenge is to provide good performance in spite of the extra layer of indirection via a second-layer hypervisor. To evaluate performance, we ran the `sysbench` CPU benchmark in the Supercloud and found that the completion time difference between first layer and second layer VMs were within 10%. The second layer hypervisor introduces overhead due to indirection and two levels of scheduling. We believe that this overhead is reasonable, but will continue to work on trying to reduce this overhead.

For second layer guest VM scheduling and placement, we

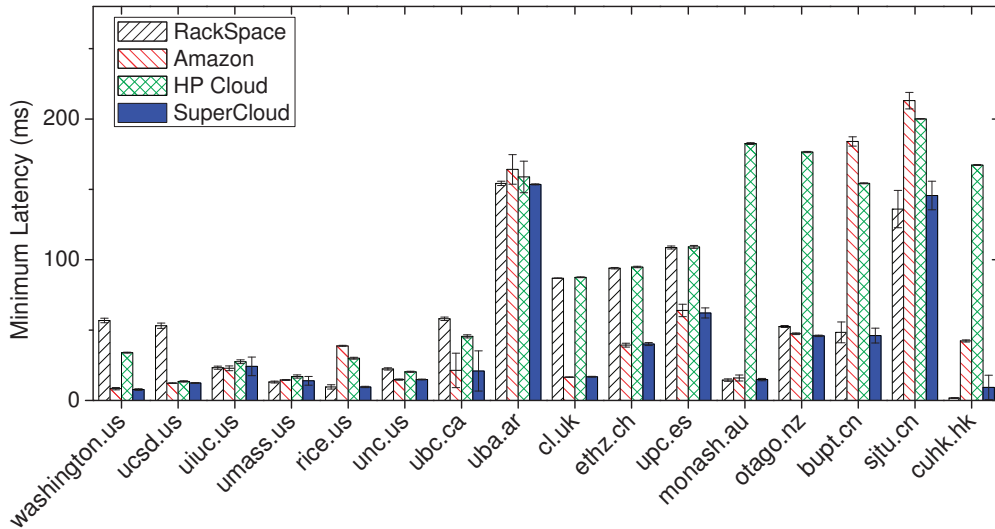[1]Providing our own public IP address pool is future work.

Figure 3: Minimum latency from various PlanetLab nodes to different clouds.

	Latency (ms)		Number of Hops	
	Mean $\pm$ Std. Dev.	$5^{th}/95^{th}$ Percentile	Mean $\pm$ Std. Dev.	$5^{th}/95^{th}$ Percentile
Amazon	57.50 ± 66.67	$11.33/191.31$	12.35 ± 3.84	$7.00/18.70$
Rackspace	58.34 ± 46.25	$7.66/140.56$	13.41 ± 3.85	$8.50/18.65$
HP Cloud	94.97 ± 69.13	$16.15/186.97$	15.43 ± 4.57	$7.75/20.40$
Supercloud	37.56 ± 45.33	$6.21/147.24$	12.16 ± 3.71	$7.00/18.40$

Table 2: Aggregate Statistics over all PlanetLab nodes.

run XenServer-core [8] within each Xen-Blanket Domain-0 and use OpenStack to manage the Supercloud. XenServer provides the necessary APIs for the cloud platform software to be scheduled and controlled. Unlike the ISO-based XenServer that has a customized kernel, XenServer-core is a series of RPMs that are fully compatible with CentOS. We ported Xen-Blanket to Xen version 4.2.2 and device drivers in Linux kernel 3.4.53 to support XenServer-core.

OpenStack has a *controller node* that can run anywhere. To improve availability, it is possible to deploy multiple hierarchical controllers in different clouds. We have not yet experimented with this.

A good scheduling policy in the Supercloud will be critical. It needs to consider in which cloud to start a new VM, when to migrate the VM, to where the VM should be migrated, and so on. Good scheduling policies can save money while satisfying performance, reliability, and security requirements. We are working on designing a scheduler for the Supercloud. Among other design requirements, the Supercloud should contain a component to learn the workloads of the applications, so that oversubscription can save money when the workload becomes light.

3.2 Supercloud storage

Currently, the Supercloud only offers a centralized NFS server for shared storage. While simple, this storage solution leads to significant latencies and low bandwidth for VMs that run in clouds and regions other than the one where the NFS server is hosted. Also, migration in the Xen-based OpenStack relies on the resource pool feature of the XenServer, which in turn requires a centralized NFS server

to store VM images.

Clearly, for the Supercloud this is not a desirable storage solution. In order to obtain better overall storage performance as well as reduce migration latency, we are designing a distributed storage solution for the Supercloud. We are taking into consideration such issues as resource scaling, power saving, cost saving, fault tolerance, security, and load balancing.

3.3 Supercloud networking

To enable communication between VMs, possibly in different clouds, the Supercloud uses a Virtual Private Network (VPN). Various VPN solutions are available. A VPN based on a centralized forwarding server is not efficient for this environment. We are currently evaluating the *tinc* VPN [13] and solutions using Open vSwitch [11].

The tinc VPN implements an automatic full mesh peer-to-peer routing protocol, minimizing the number of hops traversed between endpoints. Open vSwitch is an OpenFlow-compatible software switch, enabling tunnels across cloud boundaries by installing specific forwarding rules. Using the VXLAN specification, we can form an L2 overlay network running over the Internet. Both the tinc and Open vSwitch solutions are transparent to OpenStack. Even if VMs migrate, they can keep their virtual IP addresses. So far, we have found tinc easier to use, but Open vSwitch provides significantly better performance. We are now planning to construct a virtual network based on Open vSwitch using an OpenFlow controller. One issue to address is how a VM can keep a public IP address after migration.

4. RELATED WORK

Various open source IaaS cloud platforms such as OpenStack [5], Eucalyptus [3], and AppScale [1] provide a unified interface in any environment, but cannot easily federate multiple clouds into a single homogeneous cloud.

The European RESERVOIR project [12] proposes to determine how multiple cloud providers can federate and draw upon each other's resources. It would have to solve many of the same issues that we are already addressing. RESERVOIR proposes that providers work together, for example by using the same hypervisor and so on. Similarly, the IEEE Intercloud Testbed [4] is developing standards that allow clouds to federate and interoperate. The Supercloud does not require cooperation between providers.

RightScale [7] is closely related to the Supercloud, and allows applications to use multiple clouds by providing a portability layer so that application developers do not have to write code specific to the underlying cloud computing, storage, and networking interfaces. However, RightScale does not implement nested virtualization, and thus cannot migrate VMs.

Middleware-based solutions such as IBM Altocumulus [10] combines various IaaS and PaaS (platform as a service) clouds into a single PaaS platform. The Supercloud provides a lower-level IaaS interface, providing more flexibility to its users.

Docker [2] can deploy applications to multiple clouds by leveraging Linux Containers. Linux Container is a light weight mechanism that achieves resource isolation and separated namespaces. Docker cannot host different operating systems nor can it support live migration.

Ravello [6] leverages a nested hypervisor to deploy a "multi-VM application" to different clouds. A multi-VM is a distributed system involving multiple VMs working together. Ravello's hypervisor, HVX, uses binary translation, which has poor performance compared to para-virtualization technologies adopted by Xen-Blanket. Moreover, unlike the Supercloud, Ravello does not support deploying an application in multiple clouds and optimizing cost by using VM live migration.

There are several solutions that span *storage* resources on multiple cloud systems, such as SPANStore [16] and RACS [9]. Like the Supercloud, these systems benefit from diversity in cloud storage to lower cost or improve availability. However, they don't provide the flexibility of running compute resources in multiple clouds.

5. CONCLUSION

We are building a Supercloud: a cloud from resources obtained from various cloud providers. While most cloud providers have multiple availability zones, by combining various clouds, the Supercloud has significantly more availability zones than any other cloud. Moreover, we can migrate VMs live between availability zones, a feature not available to other cloud providers.

A Supercloud allows companies, organizations, and individuals to move to a cloud computing environment while retaining control over placement and scheduling. In particular, a cloud user controls the location and live migration of their computation, networking, and storage without owning all of the underlying infrastructure—a level of control that is not available today.

In this paper, we showed that the Supercloud can take advantage of spot market pricing. It can migrate spot market instances across availability zones and providers, significantly reducing the cost for services and applications operated in the cloud. Further, we showed that the Supercloud provides significantly decreased and more predictable latencies than any of the underlying cloud providers individually. We also described the current status of the Supercloud implementation and discussed ongoing challenges.

Acknowledgments

6. REFERENCES

[1] AppScale: The Open Source App Engine. http://www.appscale.com/.

[2] Docker. https://www.docker.com/.

[3] Eucalyptus. http://www.eucalyptus.com/.

[4] IEEE Intercloud Testbed. http://cloudcomputing.ieee.org/intercloud.

[5] OpenStack. http://www.openstack.org.

[6] Ravello Systems. http://www.ravellosystems.com/.

[7] Rightscale. http://www.rightscale.com.

[8] xenserver-core. https://github.com/xenserver/buildroot.

[9] H. Abu-Libdeh, L. Princehouse, and H. Weatherspoon. RACS: a case for cloud storage diversity. In *Proceedings of the 1st ACM symposium on Cloud computing*, pages 229–240. ACM, 2010.

[10] E. M. Maximilien, A. Ranabahu, R. Engehausen, and L. Anderson. IBM Altocumulus: a cross-cloud middleware and platform. In *Proceedings of the 24th ACM SIGPLAN*, OOPSLA '09, pages 805–806, New York, NY, USA, 2009. ACM.

[11] openvswitch. http://openvswitch.org/.

[12] B. Rochwerger, D. Breitgand, A. Epstein, D. Hadas, I. Loy, K. Nagin, J. Tordsson, C. Ragusa, M. Villari, S. Clayman, E. Levy, A. Maraschini, P. Massonet, H. Muñoz, and G. Tofetti. Reservoir - When One Cloud Is Not Enough. *Computer*, 44(3):44–51, 2011.

[13] tinc VPN. http://www.tinc-vpn.org/.

[14] D. Williams, H. Jamjoom, and H. Weatherspoon. The Xen-Blanket: virtualize once, run everywhere. In *Proceedings of EuroSys '12*, 2012.

[15] D. Williams, H. Jamjoom, and H. Weatherspoon. Plug into the SuperCloud. *IEEE Internet Computing*, 17(2):28–34, 2013.

[16] Z. Wu, M. Butkiewicz, D. Perkins, E. Katz-Bassett, and H. V. Madhyastha. Spanstore: Cost-effective geo-replicated storage spanning multiple cloud services. In *Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles*, pages 292–308. ACM, 2013.