Rethinking the architecture of Cloud Management Platform for trust, control and flexibility

Abstract

Paper submissions should at most 13 typeset pages, excluding bibliography and well-marked appendices. There is no limit on the length of the bibliography and appendices, but reviewers are not required to read them.

1 Introduction

[4]

2 Background

2.1 Cloud Management Platform

The distributed software system used by Infrastructure-as-a-Service (IaaS) Cloud Providers to manage their large pools of processing, storage and networking resources is generally known as Cloud management platform (CMP). The CMP systems allow users to create virtual machines and allocate network and storage resources to these virtual machines. VMware vCloud[**?**] is a commercially available proprietary CMP system whereas Amazon has developed their own CMP system known as Amazon Web Service[**?**] (AWS) over which they provide IaaS cloud services to the end users. There are also open-source CMP systems available i.e. OpenStack[**?**], OpenNebula[**?**], CloudStack[**?**] and Eucalyptus[**?**].

The high level architecture of these CMP systems are quiet similar and the components of these systems can be partitioned into two parts. Components having global view of the provider's infrastructure and manage it can be grouped together as cloud controller whereas those that are instantiated on each physical machine and manages just one particular machine can be group together as node controller. Cloud controller is tasked with infrastructure wide resource management i.e. allocating resources to the users and monitoring the resource usage. It also provides interface to the users for resource allocation requests and controlling resources allocated to them. Logically only one instance of cloud controller runs in the provider's infrastructure but it is usually replicated for scalability and fault tolerance.

Node controller is instantiated on each physical machine and implements functionality to manage resources of a single machine. It interacts with the local virtual machine monitor to allocate, deallocate and monitor the local resources of the machine. It also provides an interface to the cloud controller to manage resources of the local machine and for reporting local resource usage.

Suppose a user wants to create a virtual machine in the cloud. He will interact with the cloud controller along with the configuration of the virtual machine. The cloud controller will run the virtual machine placement algorithm based on the resource usage information provided by node controllers running on each physical machine and also policies configured by provider. After selecting a physical machine, cloud controller will interact with the node controller running on that physical machine to create virtual machine for the user.

Currently cloud management platforms are being deployed in two different settings. In the first setting, also known as private cloud, CMP systems are being used by enterprises to manage their private data centers while in the second setting, known as public cloud, the cloud provider, beside management, uses CMP systems for leasing resources to multiple customers. Public cloud introduces additional challenges to the design of CMP systems. CMP system has to provide isolation and security among multiple tenants leasing the cloud resources. Since customers are sharing the resources, CMP system has to provide fair share of resources, quality of service and correct billing for the resource usage. Existing CMP system do address some of these challenges and research community is also working on resolving these issues.[**?**]

One important challenge introduced by public deployment of CMP systems is that the customers of the cloud provider has to trust the cloud provider with their data and computation but actually customers have to trust the human administrators of the provider as well thus introducing another attack surface. None of the existing CMP systems even try to solve this problem. Secondly, CMP systems lacks the mechanisms to provide flexibility to the customers to deploy new security, network or storage services for their virtual machines. Currently if the customer wants to deploy virtualization or network based security tools or memory or storage deduplication tools for his virtual machines, the provider has to modify their CMP sys-

tem to support customer's request and existing CMP systems lack this flexibility. There is also lack of establishing customer's trust on the CMP and its billing system. Thus customer has no way to audit that the resources usage that they are billed for are really used by them.

2.2 Self-service Cloud

Self-service Cloud Computing (SSC)[4] was proposed to solve security and flexibility issues of the public cloud. Current public cloud infrastructure is powered by virtualization. Each physical machine is running hypervisor along with management virtual machine. The management virtual machine manages resources of the physical machine. It creates virtual machines for the customers and provides services to these virtual machines. It has privileged interface to the customers virtual machines which is necessary for management i.e creation, checkpoint and migration. This privileged interface is also used for providing services to the customers virtual machines like security, storage, network, deduplication and other services.

For the administrative tasks, human administrators of the public cloud provider have access to the management virtual machine. The malicious administrators can misuse the privilege interface of the management virtual machine to compromise the integrity or confidentiality of the customers virtual machines and Thus raises the security concerns for the customers of the public cloud provider. On the other hand customers requiring new customized services for their virtual machines has to rely on the provider as these services can only be implemented in the management virtual machine controlled by provider thus restricting the flexibility.

Self-service Cloud Computing (SSC) solves these two problems by partitioning the management virtual machine into a system-wide management virtual machine and per-client management virtual machine. System-wide management virtual machine still manages physical resources but has no access to the customers virtual machines, thus solving the problem of security. Per-client management virtual machine gives customers privilege interface to their virtual machines, thus allowing them to implement new customized services for their virtual machines and solving flexibility problem. SSC uses trusted platform module (TPM) for establishing customer's trust on the provider's infrastructure.

One side effect of this architecture is that malicious customers can misuse the provider's resources by hosting illegal contents or running botnet and the provider has no interface to monitor customer's activities. SSC solves this problem by introducing mutually trusted services. Mutually trusted service is piece a code that both customer and provider agree upon to run with customer's privileges. Mutually trusted services make sure that the customer is not misusing resources without violating integrity and confidentiality of the customer's virtual machine.

2.3 Limitations of SSC

Self-Service Cloud computing[4] was proposed to provide the customers of the IaaS cloud provider integrity and confidentiality even from human administrators and also the flexibility to deploy their own customized services for their virtual machines. SSC does solve the security and flexibility issues but there are still some open issues remain to be solve.

Self-service Cloud Computing (SSC) architecture is limited to just one physical machine and it is not clear how it can be integrated with cloud management platform (CMP). SSC complicates the management for the customers if the customers virtual machines are allocated on multiple physical machines and they have to interact with multiple instances of their per-client management virtual machine. SSC also complicates the maintenance of the cloud infrastructure. Since the provider does not have access to customers virtual machines, customers have to be involved in each maintenance tasks like migration, checkpointing and thus complicating the maintenance.

One other important problem with SSC is that it exposes provider infrastructure details and management and maintenance policies to the customers. These customers can be competitors or even worse attackes. Thus exposing such information is the main bottleneck in the adoption of Self-service Cloud Computing (SSC) by cloud providers. In SSC, customers are assumed to be technical savvy and it is expected that the source code of mutually trusted services is reviewed and audited by the customers which is an unrealistic assumption. SSC also requires customers to be involved in the trusted virtual machines creation process and thus reducing the useability of SSC.

3 Threat Model

Since Self-Service Cloud Management Platform (SSC) is based on Self-Service Cloud computing (SSC)[**?**], its threat model is quite similar to SSC. SSC assumes that the cloud providers like Amazon EC2 or Microsoft Azure, planning to support SSC are trusted while human employees working as administrators are not trusted. Human administrators at the cloud provider, maliciously or by mistake, can effect the integrity and confidentiality of the cloud provider's customers.

SSC requires that the physical machines used to run customers virtual machines must be equipped with IOMMU and a Trusted Platform Module (TPM) chip. SSC does not protects against the hardware based attacks like cold boot attacks. Although SSC aims to pro-

tect the integrity and confidentiality of customers virtual machines but it does not ensure availability or protects against denial of service attacks. SSC also assumes a trusted Certifying Authority service provided by the cloud provider which is used to attest the keys of virtual TPMs.

SSC also assumes a trusted external third party code verifier which attests the code of client cloud controller (CCC), client node controller (CNC) and the mutually trusted service domains(MTSD) provided by the cloud provider. The third party code verifier make sure that CCC, CNC and MTSD does not violate the integrity and confidentiality of the customers virtual machines.

4 Design

This section describes the design of Self-service cloud management platform (SSC).

4.1 Overview

In Self-Service Cloud Management Platform (SSC), the cloud controller is partitioned into provider's cloud controller (PCC) and client's cloud controller (CCC). PCC has knowledge of cloud-wide resource distribution and topology of cloud infrastructure. PCC takes resource allocation and maintenance decisions i.e. VM placement or migration but PCC does not implement these decisions. CCC implements the allocation and maintenance decisions made by PCC. Each customer has his own private instance of CCC running in the cloud. Customers forward their virtual machine creation requests to their instances of CCC which forwards these requests to the centralized PCC. PCC makes the VM placement decisions and returns virtual machine to physical machine mapping to the corresponding CCC which will contact directly to the physical machines for virtual machine creation. CCC also provides management interface to the customers to control their virtual machines.

The node controller in SSC is also partitioned into system node controller (SNC) and client node controller (CNC). SNC's main responsibility is to monitor the resource usage of the physical machine and reports back to provider's cloud controller while CNC is used by client cloud controller to create and manage client's virtual machines running on that physical machine. Physical machines in SSC are equipped with SSC-enabled hypervisor[4]. SNC runs as a user process in Sdom0 while CNC runs in Udom0. In SSC, the virtual machine placement and client's node controller are transparent to the customer and CCC provides a consolidated view to the customers of their virtual machines.

SSC allows customers to specify relationships among their virtual machines. In SSC, customers can specify a security virtual machine which might monitor memory or the network traffic of customer's other virtual machines. These relationships are explained in more detail in 5.

4.2 Components

This section describes in detail all the components of SSC and the interactions between them.

4.2.1 Provider's Cloud Controller

Provider's Cloud Controller (PCC) manages cloud resources and does book keeping of the resource usage. It provides administrative interface to the human administrators of the cloud provider. Administrators can perform managerial and maintenance tasks, can specify resource provision policies and can monitor cloud-wide resource usage through PCC. System node controller, running on each physical machine, periodically reports resource usage of the physical machine to PCC and thus PCC has cloud wide knowledge of resource usage.

Provider's cloud controller provides an interface to the cloud customers for the creation of client's cloud controller (CCC) and other than that PCC does not interact with the customers. PCC receives resource allocation requests from CCC. Basically the request consists of number of the virtual machines requested by the customer, configuration of each virtual machine and the relationship between them. PCC makes virtual machine placement plan based on the customer's request, resource availability and the policies specified by the provider. Based on provider's policy, PCC might also include mutually trusted service domains (MTSD) into the list of virtual machine to be created. These mutually trusted services are used for regulatory compliance or trusted resource metering. PCC returns the placement decision to the requesting CCC.

Provider's cloud controller also provides interface for maintenance tasks to the provider's administrators. Maintenance tasks may include upgrading or replacing physical machines or network equipment which might require to migrate virtual machines from effected physical machines. PCC takes virtual machine migration requests from administrators and forwards the requests to the corresponding client's cloud controllers which will perform the actual migration.

4.2.2 Client's Cloud Controller

Client's Cloud Controller (CCC), an entity mutually trusted by the customers and the provider, provides virtual machine management interface to the customers and provides maintenance interface to the provider. CCC is part of trusted computing base for SSC. It virtualizes the cloud for the customers and gives them a consolidated view of their virtual machines running in the cloud. It acts on be-

half of the customer while interacting with other components in the cloud for managing customer's virtual machines without exposing provider's internal mechanisms to the customers.

Customers needs to initiate the creation of their personal client cloud controller in order to use cloud infrastructure. Customers forwards CCC creation request to provider's cloud controller which will create CCC in trusted way. Section 4.3 describes in more detail on the trusted creation of CCC and establishing customer's trust on CCC. Once the trust is established, CCC will be the root of trust for the customer and will perform all TPM related protocols with other components of the cloud transparent to the customer.

Client's cloud controller (CCC) handles virtual machine creation requests from the customer and for each request it gets the virtual machine placement plan from PCC. After having placement plan, CCC directly communicates with the system node controller (SNC) running on the physical machines listed on the placement plan for the creation of client node controller(CNC). CCC makes sure that CNC is created and booted in trusted way by running TPM attestation protocol (more details in 4.3). CCC also establishes secure connection with all the CNCs. Once all the CNCs are created, CCC will forward the virtual machine configurations and relationships to the corresponding CNC for the creation.

CCC also handles maintenance requests from provider's cloud controller (PCC). In SSC, the maintenance requests are just requests for virtual machine migration from one physical machine to another physical machine. So, the migration request consist of the identity of the virtual machine to be migrated along with source and destination physical machines. Once CCC receives migration request from PCC, it will initiate the creation of CNC at the destination machine (if not already created) and then signals the CNC at the source to migrate the identified virtual machine to the destination physical machine. The CNC at the destination machine will receive the migrating virtual machine and resumes it on the completion of migration.

Client cloud controller is packaged as a complete virtual machine. In current SSC prototype, CCC is implemented as a user program running over Linux operating system. The underlying infrastructure running CCC virtual machine is equipped with SSC-enabled hypervisor[4] which provides isolation and protection to the CCC virtual machine. CCC virtual machine is also equipped persistent storage. All the accesses to this storage are encrypted by CCC for confidentiality. This storage is mainly used in the case of CCC failure for recovery. CCC stores secret keys on the encrypted storage, which were used to establish secure connection with client node controllers. During recovery, CCC uses the secret keys stored on the encrypted storage for re-establishing secure connections with the CNCs.

In SSC, CCC can not be migrated. CCC is the root of trust for the customers and is created in a trusted way by direct interaction of the customer with the physical machine running CCC. Since migration can not be done transparently to the customers, allowing it will complicate the management for the customers. Also the recovery process of CCC requires customer involvement. On detecting the failure of CCC, the customers have to reinitiate the creation of CCC. Once CCC creation is completed, it will recover the keys from persistent storage for re-establishing secure connection with CNCs.

One consequence of requiring the customer involvement in the recovery process is that the customer is required to have active connection with the CCC and in case of failure of CCC, customer has to initiate the recovery process soon otherwise the maintenance tasks related to the customer's virtual machine will be blocked until CCC is recovered. Usually the maintenance tasks are not that frequent and if there is an urgent task blocked due to unavailability of CCC, the provider cloud controller can just pull the plug from related virtual machines as SSC does not guarantee availability. Although CCC can not be migrated, PCC can force the migration by destroy CCC and let the customer initiate recovery process. The PCC can select the desired physical machine for the CCC virtual machine placement.

4.2.3 System Node Controller

System Node Controller (SNC) runs as a user process in the system management domain, i.e. Sdom0. Each physical machine in the cloud infrastructure has exactly one instance of SNC running on their system management domain. SNC reports the resource availability and usage of the physical machine to the provider's cloud controller (PCC) periodically. SNC also monitors the resource usage by virtual machines of the customers for billing purposes. SNC also provides interface to client's cloud controller for the creation of user management domain i.e. Udom0.

4.2.4 Client Node Controller

Client Node Controller (CNC) is a management interface used by the client's cloud controller (CCC). Through CNC, CCC manages the customer's virtual machines placed on the physical machine where CNC is running. All the physical machines running virtual machines of a particular customer will have an instance of CNC managing those virtual machines and that CNC will be con-

trolled by the CCC of the customer. A physical machine may have more than one instance of CNC running, each for different customers. CNC runs as a user process in the user management domain i.e. Udom0.

In SSC, the client node controllers are transparent to the customers. CCC gives the customers a consolidated view of their virtual machines without exposing their actual placement. For virtual machine creation, CCC provides the list of virtual machines along with their configurations and relationship specifications to the CNC for the creation. CNC creates the virtual machines using underlying SSC-enabled hypervisor infrastructure i.e. Domain Builder (DomB) and also establish relationships between virtual machines. More details on the relationships are in section 5.

On receiving maintenance request from PCC, CCC also initiates the virtual machine migration through CNC. CCC forwards list of virtual machines to be migrated and their destinations to the CNC running on the source and CNC will perform the actual migration.

4.3 Establishing Trust

In SSC, Client's cloud controller (CCC) is the root of trust and is mutually trusted by the customers as well as the provider. There are three requirements to establish the trust of customers over the cloud provider's infrastructure. First, correct version of client cloud controller (CCC) should be instantiated, Second, customers should be able to securely communicate with their instance of CCC and lastly, the integrity and confidentiality of CCC should be protected even from the human administrators of the provider.

In SSC, customers do not have access to the codebase of client's cloud controller. Since CCC interacts with the internal components of the provider's infrastructure, releasing its code to the customers may expose provider to the competitors or even worse attackers. To solve this problem SSC uses third party code verifiers, an entity similar to Certification Authority for PKI. Third party code verifiers make sure that the code of client cloud controller is doing as advertised. This can be done by combination of manual code reviews and automated analysis. In SSC the attestation done by third party code reviewers is mutually trusted by provider and customers. Code verification done by third party verifier is out of the scope of this paper.

In SSC, the physical machines hosting CCC is equipped with SSC-enabled hypervisor [4] which guarantees the integrity and confidentiality of client's cloud controller(CCC). SSC[4] also provides TPM-based protocols for trusted creation of virtual machines and establishing secure connection with them. SSC uses the protocols implemented by SSC for correct and trusted creation of CCC. The attested image of CCC is provided by cloud provider. Customers verifies that the image is attested by third party code verifiers and then executes the SSC protocol for CCC creation. At the end of the protocol, correct and trusted image of CCC will be created and the customer will have secure connection with CCC. Since by design SSC[4] provides integrity and confidentiality, all three requirements are fulfilled for establishing the trust of customer on provider's infrastructure. The protocol is described in the appendix[or here?].

4.4 Rationale

Self-Service Cloud Management Platform (SSC) aims to overcome the shortcomings of Self-Service Cloud computing (SSC) without losing the benefits of SSC. SSC partitions traditional cloud controller into a cloud wide cloud controller (PCC) and per-customer cloud controller (CCC). CCC is packaged as an isolated and independent virtual machine and by using SSC on the underlying system, the integrity and confidentiality of CCC is ensured. Along with third party attestation of CCC enables CCC to be mutually trusted between the customers and the provider.

Once mutual trust on CCC is established, CCC can act for the customer while interacting with provider's infrastructure. Through CCC, customer still have same level of control on their virtual machines as SSC provides but without exposing the provider's infrastructure or management and maintenance policies. CCC also improves the usability of the system by making trust establishment and TPM protocols transparent to the customers. Customers are involved in TPM protocol just once for establishing trust on CCC and then CCC will do all the trust establishment on behalf of the customers.

Beside CCC, the source code of CNC along with its container Udom0 and mutually trusted service domains (MTSDs) also need to be attested by third party. Since CNC is created by CCC and is transparent to the customers, it also becomes a mutually trusted entity. CNC, in terms Udom0, being mutually trusted is

Since both CCC and CNC are mutually trusted, the creation and deployment of MTSDs is simplified as oppose to SSC[4]. For CCC and CNC, MTSD creation is like any other virtual machine creation.

Allowing third party attestation of CCC, CNC and MTSD also improves the usability as customers are not required to be technical experts and review the source code of these services themselves. Another benefit of third party attestation is provider can evolve and upgrade their services more easily by out involving the customers and without exposing the source code to them for mutual trust.

5 VM relationship specifications

Self-Service Cloud computing (SSC) provides customers direct access to the physical machine in the cloud provider's infrastructure. In SSC, customers have their own privileged management domain, named Udom0, through which customers can mange their virtual machines running on that physical machines. Customers can also delegate privileges over their virtual machines to other service virtual machines e.g. giving privileges to rootkit detector service virtual machine over work virtual machines. Customers can also set a virtual machine as a backend device for their other virtual machines. Virtual machine serving as backend provides a virtual device interface to other virtual machines. SSC supports network and storage backend devices. By using backend relationship, customers can implement new personalized network and storage services for their virtual machines e.g. setting firewall virtual machine as network backend or encrypted storage virtual machine as a storage backend.

In SSC, customers do not have direct access to physical machines. SSC hides virtual machine placement from the customers and gives customers a view that all their virtual machines are running on one big system. Basically client cloud controller (CCC) provides this unified view to the customers of their virtual machines without compromising the control customers have on their virtual machines. In SSC customers, through their management domain, can grant privilges or set network or storage backends but in SSC, customers have to communicate these relationship specifications to their CCC along with the virtual machines configurations during the allocation request.

Customers specify relationship between their virtual machines using following two statements:

(**1**) GRANT_PRIVILEGE (VM_1, VM_2)

(**2**) SET_BACKEND (VM_1, VM_2, DEVICE)

GRANT_PRIVILEGE specifies that VM_1 has privileged interface over VM_2 while SET_BACKEND tells that VM_1 should be set as a backend device for VM_2 and DEVICE can either be storage or network.

5.1 Examples

This section gives some examples for specifying virtual machines relationship specifications.

Security consious customers can deploy security services for their virtual machines. Customers can configure a virtual machine with memory introspection service like rootkit detection or malware detection, for monitoring their work virtual machines. Let say customer configured a rootkit detector virtual machine, named ROOTKIT-VM, to monitor his virtual machine, named WORK-VM, during the creation request, customer has to provide following relationship specification along with configurations of the two virtual machines.

GRANT_PRIVILEGE (ROOTKIT-VM, WORK-VM)

Customers interested in storage deduplication services for their virtual machines has to specify following specifications.

SET_BACKEND (DEDUP-VM, VM_1, STORAGE)
SET_BACKEND (DEDUP-VM, VM_2, STORAGE)

Similarly customers interested in network security services like firwall and network intrusion detection system can use following specifications.

SET_BACKEND (FIREWALL-VM, WORK-VM, NETWORK)
SET_BACKEND (FIREWALL-VM, IDS-VM, NETWORK)

Above specification corresponds to the configuration shown in figure**??**. All the network traffic to the Work-VM has to pass through Firewall-VM and once cleared by Firwall-VM, the traffic has to be mirrored to IDS-VM as well. One important point to note here is that SSC just implements the storage or network plumming i.e. setting backend but configuring and setting up services inside the service virtual machines is still the customer's job.

5.2 Consequences of the specifications

Once the virtual machine relationship specifications are received, important information is extracted and used by both client's cloud controller (CCC) and provider's cloud controller (PCC). PCC extracts co-location dependencies between the virtual machines from the relationship specifications. PCC is responsible for virtual machine placement and it requires co-location dependencies between virtual machines before making placement decisions.

The relationship GRANT_PRIVILEGE (VM_1, VM_2) requires that both virtual machines VM_1 and VM_2 should be on the same physical machine because only then VM_1 can perform privilege operations on VM_2. Thus PCC will make sure that both virtual machines should be placed on the same machine. In SSC prototype, the storage based backend relation also requires the virtual machines in the relationship should be on the same physical machines. However virtual machines in network based backend relationship can be placed on different machines. Section 7 contains more details on how this is achieved.

The other important information extracted from the relationship specification is the partial ordering of the virtual machines creation. A virtual machine serving as a

storage backend and is used as root partition for another virtual machine, needs to be created and setup before the booting of the second virtual machine. Similarly for the virtual machine serving as network backend. Also the virtual machines having privileges over other virtual machines for monitoring purposes have to be started and setup before the virtual machines being monitored. Otherwise there will be a loophole in the security of the system as some important security related events might be missed by the monitor virtual machine.

In SSC, virtual machines are created in paused state and customers, through their client cloud controller (CCC), can unpause their virtual machine one by one and do the necessary setup in the backend or monitoring virtual machines. The customers are responsible for configuring backend and monitoring virtual machines while SSC allows them to give privilges to their virtual machines or set them as network or storage backends.

The co-location dependency and the partial ordering has more subtle effect on the maintenance task like virtual machine migration. Co-location dependency between the virtual machine requires concurrent migration of multiple virtual machines while partial ordering requires that the migrating virtual machines should be paused in an order determined by the relationship specification and unpaused at the destination in the reverse order. Section 6 contains more detail on virtual machine migration in SSC.

SSC does impose few restrictions on the virtual machine relationship specifications. First, there can not be any cycle in the relationship meaning the virtual machine relationship graph is directed acyclic graph and all the virtual machines can be topologically sorted. The reason for this restriction is to simplify the virtual machine allocation and migration tasks.

Second, the virtual machine relationship specifications are static. Once the virtual machines are created based on the configurations and specifications provided by the customer, the relationship between the virtual machines can not be modified. Basically the relationship specifications are used in virtual machine placement algorithm and currently SSC has very primitive placement algorithm. Allowing mutable specifications require dynamic and complex virtual machine placement algorithm and load balancer which is out of the scope of this paper.

6 VM Migration in SSC

6.1 Background

Live migration refers to moving a virtual machine from a source physical machine to a destination physical machine without disrupting the applications or services running in the virtual machine. Live migration is useful for multiple scenarios. It allows administrators of the to perform maintenance tasks on the physical machine like upgrading or replacing failed components without disrupting the services of the virtual machine. Live migration is also used for load balancing of the physical host. During off-peak hours, more virtual machines can be consolidated on less number of physical machines using live migration. Remus[7] uses live migration for taking backup of the live service to improve the availability of the service and in case of failure of primary server the backup server will automatically activates.

SSC is based on XEN hypervisor and in XEN, live migration is a two phase process:

(**1**) Iterative push phase

(**2**) Stop-and-copy phase

In the *iterative push phase*, the virtual machine at the source physical machine keep running while its pages are being pushed to the destination physical machine. The first iteration sends all the pages of the virtual machine to the destination while the following iterations only send the modified or dirty pages. Either the number of iterations exceed certain threshold or the number of dirty pages in an iteration falls below certain threshold, the process of live migration jumps to the *stop-and-copy phase*. In this phase the virtual machine at the source physical machine is suspended, final dirty pages are copied to the destination and the virtual machine at the destination is resumed. This suspend at source and resume at destination do introduce a small downtime for the migrating virtual machine.

6.2 Concurrent VM Migration

The co-locations dependency and partial ordering between the virtual machine due to relationship specifications require some modifications to the live migration in SSC. The co-location dependency requires that the all the virtual machines that have co-location dependency between them have to be migrated together while partial ordering requires that these virtual machines suspend/resume operation should be ordered.

The co-location dependency effects the start of *iterative push phase* of the migrating virtual machines. In SSC, we have explored two design choices for the migration of co-location dependent virtual machines. First, the *iterative push phase* of all these virtual machines starts in parallel i.e. at the same time and in the second design choice the phase starts sequentially. These design choices are evaluated in section 8.2 but intuitively the first design choice may suffocate the network bandwidth of the physical machine while the second choice might have larger migration completion time.

The partial ordering mainly effect the *stop-and-copy phase* of the migrating virtual machines. Basically the suspend and resume operations of all the virtual machines need to be synchronized. In SSC, *stop-and-copy phase* is broken into two phases:

(**1**) Suspend phase at source

(**2**) Resume phase at destination

The partial ordering of the virtual machines is extracted from their relationship specifications. The relationship GRANT_PRIVILEGE (VM_1, VM_2) tells that VM_1 is ordered before VM_2 *i.e.,*$VM_1 \prec VM_2$. During the concurrent migration of these virtual machines, VM_1 will be suspended after VM_2 at the source physical machine while at the detination VM_1 will be resumed before VM_2. Similarly for the storage backend relationship. The network backend relationship alone does not have co-location dependency and will not require concurrent virtual machine migration.

7 Implementation

In Self-Service Cloud Management Platform (SSC), all the physical machines used for hosting customers virtual machines are equipped with SSC enabled hypervisor. We have ported SSC to XEN 4.3 and in the prototype of SSC all the components i.e. PCC, CCC, SNC and CNC are implemented in C language and these components interact with each other through SSL connection.

7.1 VM placement algorithm

Provider's cloud controller (PCC) implements virtual machine placement algorithm. The current prototype of PCC implements a very primitive and static VM placement algorithm. The algorithm takes virtual machine configurations and relationship specifications and available resources as input and outputs the VM placement mapping. Once the mapping is created it does not change unless either the customer requests for deallocation of resources or the maintenance request from the provider. PCC tries to allocate virtual machine in network backend relationship to same physical machine and if it is not possible it will optimize the placement to keep such crossing to minimum.

7.2 Deployment of VM specification

Client cloud controller (CCC) is responsible for creating virtual machines for customers but the actual creation on the physical machine is done by client node controller (CNC). CCC interprets the relationship specifications of the virtual machine and deploy them based on the placement mapping received from PCC. For GRANT_PRIVILEGE (VM_1, VM_2) relationship CCC extends the configuration of the VM_1 that it has privilege over VM_2. Similarly for the device backends. CNC upon receiving the configurations of all the virtual machine will delegate privileges or set the backends correctly. More subtle case is when two virtual machines having network based backend relationship are placed on different physical machine.

More concretely GRE tunnel over open vSwitch is used to implement network relationship across physical machines. If two virtual machines have relationship SET_BACKEND (VM_1, VM_2, NETWORK) and are placed on different physical machines, CCC will add one transparent virtual machine equipped with open vSwitch on both hosts, named openvSwitch-VM. On the physical machine hosting VM_2, CCC will set openvSwitch-VM as a network backend for VM_2 and similarly on the other machine hosting VM_1, the local openvSwitch-VM will be backend of VM_1. CCC will setup GRE tunnel across these two openvSwitch-VM to implement such relationship.

7.3 Use-cases

This section contain some example virtual machine scenarios to further explain the details of SSC deployment.

7.3.1 Rootkit-VM

Suppose a customer is deploying a webserver virtual machine, Web-VM, and wants to protect it with a Rootkit-VM. Beside providing configurations for both virtual machines, the customer has to provide GRANT_PRIVILEGE (ROOTKIT-VM, WEB-VM) relationship to his CCC.

As Rootkit-VM and Web-VM have co-location dependency, PCC will allocate them on same physical machine. CCC will initiate the creation of CNC on that particular physical machine and then it will pass the configurations of both virtual machine to CNC. While Web-VM's configuration will be same but CCC will extends the configuration of Rootkit-VM with additional information that it has privileges over Web-VM. After creation, CNC will delegate privileges to Rootkit-VM over Web-VM.

Now suppose CCC receives request from PCC to migrate Rootkit-VM and Web-VM to a different physical machine. Since these VMs have co-location dependency, concurrent migration is required. CCC will send the migration request to CNC at the source along with the destination and the ordering of the virtual machines. CNC will do the actual migration with the ordering provided by CCC.

7.3.2 Network Security services

A customers wants to deploy a firewall, named firewall-VM, and an intrusion detection system, named IDS-VM, for his virtual machine, named Work-VM. Customer can configure the virtual machine backends such that all the

traffic to and from Work-VM passes through firewall-VM as well as IDS-VM. Usually IDS doesn't really need to interpose on the network traffic. Customer can configure these three virtual machines in a way that the traffic passes through firewall-VM and then mirrored to the IDS-VM as well. The relationship specifications for such configuration will be the following:

SET_BACKEND (FIREWALL-VM, WORK-VM, NETWORK)
SET_BACKEND (FIREWALL-VM, IDS-VM, NETWORK)

One important point to note here is that the customer has to configure the mirroring in firewall-VM himself, SSC does not do mirror automatically. Now there are three different placement scenarios. First all three VMs are allocated on same physical machine, second, two on same while third on different machine and for third, all are allocated on different machine. One important point to note here is that in case of second scenario, Firewall-VM will always be one of two allocated on same physical machine due to the VM placement algorithm used by PCC.

For allocation for all three scenarios, CCC will update the configurations with backend information and pass it to CNC for virtual machine creation. If the virtual machines in network backend relationship are allocated on different machines, a pair of openvSwitch-VM will created for each such relationship.

8 Evaluation

SSC allows customers to deploy personalized and customized services for their virtual machines. Customers specify relationships between their virtual machines in terms of GRANT_PRIVILEGE or SET_BACKEND. PCC, based on the relationships provided by the customer and current cloud infrastructure resource usage, makes virtual machine placement plan. CCC does the actual deployment of the virtual machines according to the placement plan given by PCC.

For each network backend relation in the resource allocation request, there are two possible deployment scenarios. The virtual machines in the network backend relationship can either be deployed on the same physical machine or on different physical machines. In evaluating SSC our main goal is to measure the effect of these two deployment scenarios on different network services. We evaluated several network services for both deployment scenarios and compare with the scenario where the corresponding service is provided by the cloud provider. Even for the network service provided by the cloud provider the same two deployment scenarios are possible, either the service is co-located with the target virtual machines or the service is on the different machine than the target virtual machines.

Configuration	Throughput(Mbps)	RTT(ms)
$Base_S$	925.4±0.5	0.38±0
SSC_S	924.0±1.2 (0%)	0.62±0 (1.6x)
$Base_D$	848.4±11.2	0.69±0
SSC_D	425.8±5.5 (49.8%)	1.6±0 (2.3x)

Table 1: **Baseline overhead of network services in SSC.**

For the same machine deployment scenario, we compare the performance of network services deployed under SSC infrastructure against Xen Legacy. Similarly we compare for the different machine deployment scenario. To differentiate between the two deployment scenarios, we subscripted the configurations with 'S' if service and target virtual machine are on same machine otherwise we the configurations are subscripted with 'D'.

All the experiments were performed on a Dell Poweredge R610 system equipped with 24GB RAM, eight 2.3GHz Xeon cores with dual threads (16 concurrent executions), Fusion-MPT SAS drives, and a Broadcom NetXtreme II gigabit NIC. All virtual machines in the experiments (dom0, domU, Sdom0, Udom0, UdomU, SD, domB and MTSD) were configured to have 2GB RAM and 2 virtual CPUs.

In all the experiments, we used **iperf3**[8] to measure the network throughput of the virtual machines and used **ping** to measure the round trip time (RTT) of network traffic. We dedicated a separate physical machine in the local network of the machines running SSC infrastructure which acts as a server for measuring network throughput and RTT of the target virtual machine. The experimental numbers reported in this section are averaged over five executions; we also report standard deviations.

Before evaluating services deployed under SSC infrastructure, we first performed experiments to measure the basic overhead of two deployment scenarios *i.e.,*same machine and different machine, on the network throughput and RTT. We configured two virtual machines, named Ser-VM and Work-VM, with the relation SET_BACKEND(SER-VM, WORK-VM, NETWORK). Ser-VM just switches the network traffic for Work-VM. We measured the network throughput and RTT for Work-VM which is presented in Table 1. These numbers are the upper bound on the network performance for all the network services evaluated in this section for the given two deployment scenarios for both SSC infrastructure and legacy cloud infrastructure.

Table 1 shows that the effect of same machine deployment in SSC infrastructure on the network throughput is negligible. So, deploying a customized network service in a separate specialized virtual machine is comparable to deploying it in the management domain *i.e.,*dom0. One extra level of indirection in the network path does not effect the network throughput of the virtual machines. Although this indirection does effect the RTT of network traffic *i.e.,*1.6x overhead. This overhead seems high but still the absolute value of RTT is low and will not be an issue for virtual machines serving requests from the Internet like webservers.

For the second deployment scenario *i.e.,*service and target virtual machines are on different machines, we see a high overhead of 49.8%. On further investigation we found out that the virtual network driver of Ser-VM is the bottleneck. All the network traffic of Work-VM is routed through the service and the network traffic is concurrently coming in and out from the service. In case of service deployed by cloud provider, the service runs in Dom0, the domain accessing directly the physical network device. The network hardware used in our experiments supports concurrent bidirectional network bandwidth of 1Gbps. In Xen, the virtual network driver does not support concurrent bidirectional network bandwidth and thus we see that the throughput is reduced by nearly half.

8.1 Network Services

We implemented multiple network security services and also used some off the shelf network security tools. This section presents the evaluation of these security services deployed under SSC infrastructure.

8.1.1 Network Access Control service

Network access control service is used to control the network access of a virtual machine. We configured a virtual machine with open vSwitch, named NAC-VM, and used openflow rules to control the network access of target virtual machines. We enfored access control rules like allowing accesses to whitelist of IP addresses and restricting the visible ports of the target virtual machine. The following relationship specification is used to deploy NAC-VM in SSC infrastructure.

SET_BACKEND (NAC-VM, WORK-VM, NETWORK)

We performed experiments to evaluate the effect of deploying network access control service under SSC on the network performance of the target virtual machine. The experiment results, in Table 2, show that NAC-VM does not have extra overhead over the base overhead of different deployment scenarios presented in Table 1.

Configuration	Throughput(Mbps)
XEN Legacy$_S$	925.1±0.7
SSC$_S$	923.2±1.6 (0%)
XEN Legacy$_D$	846.7±17.2
SSC$_D$	425.2±7.2 (49.7%)

Table 2: **Network Access Control service. In XEN Legacy, the NAC service is deployed in dom0.**

Configuration	Throughput(Mbps)
XEN Legacy$_S$	924.8±1.1
SSC$_S$	924.1±0.4 (0%)
XEN Legacy$_D$	845.4±11.1
SSC$_D$	424.3±3.1 (49.8%)

Table 3: **Network Metering service.**

8.1.2 Trusted metering service

One the major benefit of SSC infrastructure is the support for deploying services mutually trusted by the cloud provider and the clients. Mutually trusted services can be used for regulatory compliance and as well as for mutually trusted billing of the resource usage in the cloud. In the existing cloud infrastructure, there is no way for a customer to audit the resource usage for which he was billed for. In SSC infrastructure, resource metering done by a mutually trusted service will improve customer satisfaction on the resource usage billing.

We implemented the network metering service using libpcap, a packet capturing library developed by the tcpdump team[16]. The service keeps track of the network usage of the target virtual machine. The following specification is required to deploy the metering service on SSC infrastructure.

SET_BACKEND (METER-VM, WORK-VM, NETWORK)

Table 3 shows the impact of network metering service on the network throughput of the target virtual machine. The results show that there is not any additional overhead for both deployments scenarios over baseline.

8.1.3 Intrusion detection and prevention services

To further show the utility of SSC we deployed and evaluated intrusion detection servive (IDS) and intrusion prevention service (IPS). IDS monitors network traffic for hostile activities and known threats. IDS maintains a database of known attack signatures and compares the monitored network traffic against those signatures to detect possible attacks. When there is a match between network flow and the attack signature, IDS raises alarm to

the system adminitrators for appropriate intervention. IPS also works similar to IDS except upon findind a match between attack signature and network flow, IPS blocks that particular network flow.

We used Snort[14] for monitoring network traffic. Snort is an open source realtime network traffic monitoring and analysis system. It can be configured to be an IDS system as well as IPS system. Snort can monitor and analyze multiple network protocols and can used to detect a variety of attacks. In IDS mode, Snort uses libpcap[16] to capture network traffic while in IPS mode, Snort uses Linux's netfilter framework to interpose the network flow. In IPS mode, the kernel forwards each packet to Snort which makes the decision to allow that packet to pass through or not. We configured Snort in a specialized virtual machines named Snort-VM and used following specification to deploy Snort-VM in SSC infrastructure.

SET_BACKEND (SNORT-VM, WORK-VM, NETWORK)

Since Snort-VM is the network backend of Work-VM, the network traffic of Work-VM is routed through Snort-VM. Snort can be configured to monitor particular network protocol by configuring the corresponding Snort's preprocessor modules. For the evaluation of deployment of Snort-VM in SSC we configured Snort to monitor TCP traffic. We used Stream5 preprocessor which is a target-based TCP reassembly module for snort and is capable of tracking TCP and UDP sessions. It can be used to detect TCP protocol anomalies, such as data on SYN packets, data received outside the TCP window *etc.*. In our experiments, we used Snort signatures developed by Snort community and freely available from Snort website[14]. We configured Snort with 242 signatures and perform experiments, using **iperf3**, to measure the network throughtput of Work-VM. **Iperf3** generates TCP traffic and all the packets generated will be tracked by the Snort.

Table 4 shows the results of the experiments to measure the overhead of deploying Snort in SSC infrastructure. The results show that when Snort is configured as IDS, the network throughput is not effected as the numbers for IDS in Table 4 are similar to the base overhead numbers in Table 1. The overhead deploying Snort as IPS in SSC infrastructure is low as shown in Table 4.

Configuration	IDS	IPS
XEN Legacy$_S$	922.8±1.1	370.6±1.8
SSC$_S$	920.9±1.9 (0%)	363.8±1.8 (1.8%)
XEN Legacy$_D$	841.2±14.2	240.6±12.7
SSC$_D$	422.6±7.1(49.7%)	235.8±2.2 (1.9%)

Table 4: **Network throughput(Mbps) for the virtual machine whose traffic is monitored by snort in IDS and IPS mode.**

Platform	Time (μsec)
XEN Legacy	1014±6
SSC	1688±31 (66%)

Table 5: **VMWall service: time to establish a TCP connection.**

8.1.4 VMWall service

We have also deployed an application level firewall named VMWall [15]. Application level firwall monitors the network traffic of a machine and correlate it to the application generating or receiving that traffic and based on the authenticity of the application, the traffic might be blocked. Usually the application level firwall run as a user process inside the monitored machine and therefore is prone to attacks on operating system or the users having administrative rights. VMWall isolates itself from the target virtual machine using the virtualization and uses virtual machine introspection for correlating the network traffic with the processes running in the virtual machine.

We have implemented VMWall from scratch. We used libvmi[11, 18] for the virtual machine introspection and open vSwitch for network monitoring. Our implementation of VMWall interacts with open vSwitch using openflow protocol. VMWall registers itself with open vSwitch as an openflow controller and whenever open vSwitch observes a new network flow, it invokes VMWall with the header of the packet. VMWall, using libvmi, finds the source or destination applicaiton for that network flow in the target virtual machine. If the application is not in the whitelist, the flow is dropped otherwise VMWall allows the packet to pass and also add a new rule in the open vSwitch to allow all the successive packets of that network flow to pass. Following relationship specification is used to deploy VMWall-VM in SSC.

GRANT_PRIVILEGE (VMWALL-VM, WORK-VM)
SET_BACKEND (VMWALL-VM, WORK-VM, NETWORK)

VMWall-VM requires privileges over the target virtual machine to perform virtual memory introspection while it has to be the network backend of target virtual machine to monitor all the network traffic. To measure the overhead of deploying VMWall in SSC, we perform experiments to measure the TCP connection setup time. We implemented a simple TCP client-server program to measure TCP connection setup time. Table 5 shows the results of

Platform	Time (seconds)
Xen Legacy	23.27±0.11
SSC	23.81±0.03 (2%)

Table 6: **Total migration time for one virtual machine.**

this experiment. In XEN Legacy, VMWall is deployed in the management domain while for Tunnel$_S$, VMWall is deployed as a separate virtual machine, VMWall-VM. The overhead of 66% for Tunnel$_S$ looks high but it is still less than 1 millisecond. Also this is a one-time cost at the setup phase and will be amortized across the duration of connection.

8.2 Migration evaluation

This section contains the evaluation of virtual machine migration in SSC. We performed three set of experiments. First we performed experiments to measure the base overhead of migrating a virtual machine in SSC over Legacy XEN. Second we performed experiments to compare the tradeoffs between parallel and sequential design choices of iterative push phase in SSC. Third set of experiments are done to measure the downtime of virtual machines due to migration in SSC. In all the migration experiments, the virtual machines being migrated have 1GB RAM and 1 vcpu.

Table 6 shows the comparison Xen Legacy and SSC for migrating a single virtual machine and SSC's overhead is negligible. Regarding the comparison between parallel and sequential design choices, we first performed network utilization experiments. We used **iftop** command to find the peak of network utilization during virtual machine migration. Note that this is not average network utilization which will be less than the peak. For migrating multiple virtual machines sequentially, we found out that the peak of network utilization during migration never exceeded 40% while for parallel virtual machine migration, the peak network utilization never exceeded 70% even when four virtual machines were being migrated in parallel. Note that the control domain performing the migration is configured to have 2 vcpus[1]. Next we performed same experiments with control domain configure with 4 vcpus. We observed that when Udom0 is configured with 4 vcpus, the migration of four virtual machines in parallel saturates the network *i.e.,*100% network utilization.

Table 7 contains the experiment results for measuring migration completion time. We compare total migration time for sequential and parallel designs by varying number of concurrent virtual machines migrating and the configuration of the control domain doing migration *i.e.,*Udom0. Obviously the total migration time for parallel design choice is less than sequential design. As shown in Table 7, the Udom0's configuration has negligible effect on the concurrent migration of 2 virtual machines but for 4 concurrent virtual machines migration, Udom0 with 2 vcpus got saturated. Parallel concurrent virtual machine migration design is better choice if low migration completion time is required. Also if network saturation is not the issue, Udom0 should be configured with number of vcpus equal to the number of virtual machine being migrated concurrently. This can be done by hotplugging vcpus into Udom0 and removing them after migration completion. The network utilization can also be controlled by limiting the vcpus of Udom0.

[1]In all the experiments we use standard configuration for virtual machines *i.e.,*2 vcpus and 2GB RAM except in the migration experiments where the virtual machines being migrated have 1 vcpu and 1 GB RAM.

Length Partial Order	Down time(ms)
1	97±4
2	308±3
3	528±8
4	778±7

Table 8: **Down time for migrating VMs.**

In SSC, there can be co-location dependencies between multiple virtual machines. Therefore SSC supports concurrent VM migration. Beside co-location dependencies, the VM relationship specifications in SSC also introduced the partial ordering between the virtual machines. This partial ordering between the virtual machines effects the stop-and-copy phase of migration and thus the downtime of the virtual machines migrating concurrently. We performed experiments to measure the effect of the length of partial order on the downtime of migrating virtual machines.

We observed that the parallel or sequential design choice for iterative push has no effect on the downtime of concurrently migrating virtual machines. Table 8 shows the results of the experiments performed to measure the downtime of virtual machines during migration. Here partial order of length n means that there are n virtual machines being migrated concurrently and all of them are ordered *i.e.,*$\text{VM}_1 \prec \text{VM}_2 \prec \text{VM}_3 \prec \ldots \prec \text{VM}_n$. The results show that the downtime increases linearly with the length of partial order.

9 Related Work

In this section we discuss prior work in cloud computing which are related to different aspects of Self-Service Cloud Computing Platform (SSC).

Number of VMs	Time for Sequential migration(seconds)		Time for Parallel migration(seconds)	
	Udom0 with 2 vcpus	Udom0 with 4 vcpus	Udom0 with 2 vcpus	Udom0 with 4 vcpus
2	47.29±0.18	47.41±0.29	27.91±0.16	28.01±0.26
4	128.89±0.76	103.96±0.20	57.78±0.49	39.21±0.50

Table 7: **Total migration time for one VM.**

Security of Customer's VMs. In SSC, the virtual machines of the customers are provided integrity and confidentiality even against the administrator domain. In general, a lot of work has been done on the security of virtual machines. Among them, one line of work mainly focuses on reducing the TCB of the system for improving the security of the virtual machines running on the system. Among them are Murray *et al.* [10] and Xoar [6] which disaggregates the administrative domain to reduce the attack surface and noHype [9] which completely eliminates the hypervisor and administrative domain. SSC is based on SSC [4] which also uses diaggregation for providing security but it also gives customers the flexibility to implement new services unlike these works.

The CloudVisor project [20] aims to protect the customer's virtual machines from administrative, similar to SSC [4], by using nested virtualization. In CloudVisor, a small trusted hypervisor executes on bare hardware and a commodity hypervisor runs over it. The small trusted hypervisor intercepts all the accesses to the customer's virtual machine from commodity hypervisor or administrative domain and encrypts the memory pages to protect the confidentiality of the virtual machines.

The Excalibur system [13] introduces a new abstraction called policy-sealed data. Using this abstraction, customers can encrypt their data which can only be decrypted on physical machines satisfying customer's policy. In policy the customers can specify the software stack or the location of the machine.

Customized service in Cloud. Existing cloud provider does not allow or provide a way to their customers to implement or deploy customized services for their virtual machines. Turtle Project[2], using nested virtualization, allows the customers to deploy their own hypervisor over provider's hypervisor and using it customers can deploy customized services for their virtual machines. One down side of Turtle Project[2] is that it requires the provider to use hypervisor which nesting. Xen-Blanket [19] exactly addresses this issue and use a paravirtualized nested hypervisor for the customers which does not need nesting support from the provider's hypervisor.

SSCalso allows customers to implement and deploy customized services for their virtual machines and requires support from the provider. SSC also provides security to the customers which Xen-Blanket and Turtle project doesn't provide.

Cloud accountability. SSCallows for trusted resource metering for the customers. So customers trust the billing for resource usage done by the provider. There are systems having similar goals. Alibi [5] aims to provide verifiable resource accounting for the customers. Underlying it uses nested virtualization and places a trusted hypervisor below provider's commodity hypervisor to monitor the cpu and memory usage of the customer's virtual machines.

Several researchers have worked on system which just targets some particular service provided by cloud provider to make sure that the provider is providing that service correctly. CloudProof[12] provides cloud storage to the customer in which they can detect violation of integrity, write-serializability and freshness and can also prove the violation if occured. The authors of Hourglass Schemes [17] aims to detect if the cloud provider has advertised encrypted storage service but no really encrypting it. Provable Data Possession [1] allows customers to verify that the data is actually stored on the provider's infrastructure without retrieving the whole data.

Network Middleboxes in Cloud. SSC allows customers to set a virtual machine as network backend of other virtual machines by using network backend relationship specification. This is similar to setting up a network middlebox. We have implemented many network services8 using this network backend relationship and those services can be presented as network middleboxes.

CloudNaaS [3] provides abstractions to the customers to specify virtual machines as network middleboxes and uses openflow enabled switches to make sure the traffic passes through middleboxes. CloudNaaS does not aim to protect against the network administrative attacks while SSC makes sure that if a virtual machine is set as a backend, traffic will pass throught it.

10 Conclusion

References

[1] Giuseppe Ateniese, Randal C. Burns, Reza Curtmola, Joseph Herring, Lea Kissner, Zachary N. J. Peterson, and Dawn Xiaodong Song. Provable data possession at un-

trusted stores. In *ACM Conference on Computer and Communications Security*, pages 598–609, 2007.

[2] M. Ben-Yahuda, M. D. Day, Z. Dubitsky, M. Factor, N. Har'El, A. Gordon, A. Liguori, O. Wasserman, and B. Yassour. The Turtles project: Design and implementation of nested virtualization. In *USENIX/ACM OSDI*, 2010.

[3] Theophilus Benson, Aditya Akella, Anees Shaikh, and Sambit Sahu. Cloudnaas: a cloud networking platform for enterprise applications. In *SoCC*, page 8, 2011.

[4] Shakeel Butt, H. Andrés Lagar-Cavilla, Abhinav Srivastava, and Vinod Ganapathy. Self-service cloud computing. In *ACM Conference on Computer and Communications Security*, pages 253–264, 2012.

[5] Chen Chen, Petros Maniatis, Adrian Perrig, Amit Vasudevan, and Vyas Sekar. Towards verifiable resource accounting for outsourced computation. In *VEE*, pages 167–178, 2013.

[6] P. Colp, M. Nanavati, J. Zhu, W. Aiello, G. Coker, T. Deegan, P. Loscocco, and A. Warfield. Breaking Up is Hard to Do: Security and Functionality in a Commodity Hypervisor . In *ACM SOSP*, 2011.

[7] Brendan Cully, Geoffrey Lefebvre, Dutch T. Meyer, Mike Feeley, Norman C. Hutchinson, and Andrew Warfield. Remus: High availability via asynchronous virtual machine replication. (best paper). In *NSDI*, 2008.

[8] iperf3. http://code.google.com/p/iperf/.

[9] E. Keller, J. Szefer, J. Rexford, and R. Lee. Eliminating the hypervisor attack surface for a more secure cloud. In *ACM CCS*, 2011.

[10] D. Murray, G. Milos, and S. Hand. Improving Xen Security Through Disaggregation. In *ACM VEE*, 2008.

[11] B. Payne, M. Carbone, and W. Lee. Secure and Flexible Monitoring of Virtual Machines. In *ACSAC*, 2007.

[12] Raluca Ada Popa, Jacob R. Lorch, David Molnar, Helen J. Wang, and Li Zhuang. Enabling security in cloud storage slas with cloudproof. In *Proceedings of the 2011 USENIX conference on USENIX annual technical conference*, pages 31–31, 2011.

[13] N. Santos, R. Rodrigues, K. Gummadi, and S. Saroiu. Policy-sealed data: A new abstraction for building trusted cloud services. In *USENIX Security*, 2012.

[14] Snort. http://www.snort.org/.

[15] Abhinav Srivastava and Jonathon T. Giffin. Tamper-resistant, application-aware blocking of malicious network connections. In *RAID*, pages 39–58, 2008.

[16] TCPDump and libpcap. http://www.tcpdump.org.

[17] Marten van Dijk, Ari Juels, Alina Oprea, Ronald L. Rivest, Emil Stefanov, and Nikos Triandopoulos. Hourglass schemes: how to prove that cloud files are encrypted. In *ACM Conference on Computer and Communications Security*, pages 265–280, 2012.

[18] vmitools. http://code.google.com/p/vmitools/.

[19] D. Williams, H. Jamjoom, and H. Weatherspoon. The Xen-Blanket: Virtualize Once, Run Everywhere. In *ACM EuroSys*, 2012.

[20] F. Zhang, J. Chen, H. Chen, and B. Zang. CloudVisor: Retrofitting Protection of Virtual Machines in Multi-tenant Cloud with Nested Virtualization. In *ACM SOSP*, 2011.