ON THE MANAGEMENT OF VIRTUAL MACHINES FOR CLOUD INFRASTRUCTURES

By IGNACIO M. LLORENTE, RUBE´ N S. MONTERO, BORJA SOTOMAYOR, DAVID BREITGAND, ALESSANDRO MARASCHINI, ELIEZER LEVY, and BENNY ROCHWERGER

Prepared by: Dr. Faramarz Safi
Islamic Azad University, Najafabad Branch,
Esfahan, Iran.

THE ANATOMY OF CLOUD INFRASTRUCTURES

This chapter focuses on the subject of IaaS clouds and, more specifically, on the **efficient management of virtual machines** in this type of cloud. There are many commercial IaaS cloud providers in the market, such as those cited earlier, and all of them share five characteristics:

(i) They provide on-demand provisioning of computational resources;

(ii) they use virtualization technologies to lease these resources;

(iii) they provide public and simple remote interfaces to manage those resources;

(iv) they use a pay-as-you-go cost model, typically charging by the hour;

(v) they operate data centers large enough to provide a seemingly unlimited amount of resources to their clients (usually touted as “infinite capacity” or “unlimited elasticity”).

Private and hybrid clouds share these same characteristics, but instead of selling capacity over publicly accessible interfaces, focus on providing capacity to an organization’s internal users.

THE ANATOMY OF CLOUD INFRASTRUCTURES

Virtual Infrastructure (VI) management—the management of virtual machines distributed across a pool of physical resources—becomes a key concern when building an IaaS cloud and **poses a number of challenges**.

- In traditional physical resources, virtual machines require a fair amount of configuration, including preparation of the machine's software environment and network configuration.
- In a virtual infrastructure, this configuration must be done on-the-fly, with as little time between the time the VMs are requested and the time they are available to the users.
- This is further complicated by the need to configure groups of VMs that will provide a specific service (e.g., an application requiring a Web server and a database server).
- Additionally, a virtual infrastructure manager must be capable of allocating resources efficiently, taking into account an organization's goals (such as minimizing power consumption and other operational costs) and reacting to changes in the physical infrastructure.

THE ANATOMY OF CLOUD INFRASTRUCTURES

- Virtual infrastructure management in **private clouds** has to deal with an additional problem:
- **Unlike** large IaaS cloud providers such as Amazon, private clouds typically do not have enough resources to provide the illusion of “infinite capacity.”
- The immediate provisioning scheme used in public clouds, where resources are provisioned at the moment they are requested, is **ineffective in private clouds**.
- Support for additional provisioning schemes is required for applications that require resources at specific times, such as **best-effort provisioning** and **advance reservations** to guarantee quality of service (QoS).
- Thus, efficient resource allocation algorithms and policies and the ability to combine both private and public cloud resources, resulting in a hybrid approach, become even more important.

THE ANATOMY OF CLOUD INFRASTRUCTURES

Managing virtual **infrastructures in a private/hybrid cloud is a different problem** than **managing a virtualized data center**, and existing tools lack several features that are required for building IaaS clouds.

- **Traditional methods** can only operate with some preconfigured placement policies, which are generally simple (round robin, first fit, etc.) and based only on CPU speed and utilization of a fixed and predetermined number of resources, such as memory and network bandwidth.
- Thus, there are still several gaps in existing **VI solutions**. Filling these gaps will require **addressing a number of research challenges** such as **virtual machine management**, **resource scheduling**, **SLAs**, **federation of resources**, and **security**.
- In this chapter, we focus on three problems addressed by the **Virtual Machine Management Activity** of **RESERVOIR** (Resources and Services Virtualization without Barriers), an European Union FP7-funded project :
  - **Distributed management of virtual machines,**
  - **Reservation-based provisioning of virtualized resources, and**
  - **Provisioning to meet SLA commitments.**

Distributed Management of Virtual Machines

- The problem of efficiently selecting or scheduling computational resources is well known. However, the state of the art in **VM-based resource scheduling** follows a **static approach**, where resources are initially selected using a greedy allocation strategy, with minimal or no support for other placement policies.
- To efficiently schedule resources, **VI managers** must be able to support **flexible and complex scheduling policies** and must *leverage* (use) the ability of VMs to suspend, resume, and migrate.
- This complex task is one of the core problems that the RESERVOIR project tries to solve. The problem of how to manage VMs distributed across a pool of physical resources will be described. OpenNebula, the virtual infrastructure manager developed by the RESERVOIR project will be explained later.

Reservation-Based Provisioning of Virtualized Resources

- A particularly interesting problem when provisioning virtual infrastructures is **how to deal with situations where the demand for resources is known beforehand**—for example, when an experiment depending on some complex piece of equipment is going to run from 2 pm to 4 pm, and computational resources must be available at exactly that time to process the data produced by the equipment.
- Commercial clouds do have **infinite resources** to handle this situation. On the other hand, when dealing with **finite capacity**, a different approach is needed. However, the intuitively simple solution of reserving the resources beforehand is not so simple, because it is known to cause resources to be underutilized, due to the difficulty of scheduling other requests around an inflexible reservation.

Provisioning to Meet SLA Commitments

- IaaS clouds can be used to deploy services that will be consumed by users other than the one that has deployed the services.
- There is a distinction between the **cloud consumer** (i.e., the **service owner**; for instance, the company that develops and manages the applications) and the **end users** of the resources provisioned on the cloud (i.e., the **service user**; for instance, the users that access the applications).
- Furthermore, **service owners** will enter into **service-level agreements (SLAs)** with their end users, covering guarantees such as the timeliness with which these services will respond.
- Requirements are formalized in infrastructure SLAs between **the service owner** and **cloud provider**, separate from the high-level **SLAs between the service owner** and **its end users**.

Provisioning to Meet SLA Commitments

- In many cases, either the service owner is not resourceful enough to perform an exact service sizing or service workloads are hard to anticipate in advance.
- Therefore, to protect high-level SLAs, the cloud provider should cater for **elasticity on demand**.
- **Scaling and de-scaling** of an application is best managed by the application itself. The reason is that in many cases, resource allocation decisions are **application-specific** and are being driven by the **application level metrics**.

Provisioning to Meet SLA Commitments

RESERVOIR proposes a flexible framework where service owners may register **service-specific elasticity rules** and **monitoring probes**, and these rules are being executed to **match environment conditions**.

The elasticity of the application should be contracted and formalized as part of the SLA between the **cloud provider** and **service owner**.

This poses **interesting research issues** on the IaaS side, which can be grouped around two main topics:

- SLA-oriented capacity planning that guarantees that there is enough capacity to guarantee service elasticity.
- Continuous resource placement and scheduling optimization that lowers operational costs and takes advantage of available capacity transparently to the service while keeping the service SLAs.

DISTRIBUTED MANAGEMENT OF VIRTUAL INFRASTRUCTURES

- Managing VMs in a pool of distributed physical resources is a key concern in IaaS clouds, requiring the use of a virtual infrastructure manager.
- OpenNebula is capable of managing groups of interconnected VMs—with support for the Xen, KVM, and VMWare platforms—within data centers and private clouds that involve a large amount of virtual and physical servers.
- OpenNebula can also be used to build hybrid clouds by interfacing with remote cloud sites.

DISTRIBUTED MANAGEMENT OF VIRTUAL INFRASTRUCTURES
VM Model and Life Cycle

- The primary target of OpenNebula is to manage VMs. Within OpenNebula, a VM is modeled as having the following attributes:
- A capacity in terms of memory and CPU.
- A set of NICs attached to one or more virtual networks.
- A set of disk images. In general it might be necessary to transfer some of these image files to/from the physical machine the VM will be running in.
- A state file (optional) or recovery file that contains the memory image of a running VM plus some hypervisor-specific information.

DISTRIBUTED MANAGEMENT OF VIRTUAL INFRASTRUCTURES
VM Model and Life Cycle in OpenNebula

The life cycle of a VM within OpenNebula follows several stages:

- **Resource Selection.** Once a VM is requested to OpenNebula, a feasible placement plan for the VM must be made. OpenNebula's default scheduler provides an implementation of a **rank scheduling policy**, allowing site administrators to configure the scheduler to prioritize the resources that are more suitable for the VM, using information from the VMs and the physical hosts. In addition, OpenNebula can also use the **Haizea lease manager** to support more complex scheduling policies.
- **Resource Preparation.** The disk images of the VM are transferred to the target physical resource. During the boot process, the VM is contextualized, a process where the disk images are specialized to work in a given environment. For example, if the VM is part of a group of VMs offering a service (a compute cluster, a DB-based application, etc.), contextualization could involve setting up the network and the machine hostname, or registering the new VM with a service (e.g., the head node in a compute cluster). Different techniques are available to contextualize a worker node, including use of an automatic installation system (for instance, Puppet or Quattor), a context server, or access to a disk image with the context data for the worker node (OVF recommendation).
- **VM Termination.** When the VM is going to shut down, OpenNebula can transfer back its disk images to a known location. This way, changes in the VM can be kept for a future use.

DISTRIBUTED MANAGEMENT OF VIRTUAL INFRASTRUCTURES
VM Management

OpenNebula manages a VMs life cycle by orchestrating **three** different management areas:
1) **virtualization** by interfacing with a physical resource's hypervisor, such as Xen, KVM, or VMWare, to control (e.g., boot, stop, or shutdown) the VM;

2) **image management** by transferring the VM images from an image repository to the selected resource and by creating on-the-fly temporary images; and

3) **networking** by creating local area networks (LAN) to interconnect the VMs and tracking the MAC addresses leased in each network.

- **Virtualization:** OpenNebula manages VMs by interfacing with the physical resource virtualization technology **(e.g., Xen or KVM)** using a set of pluggable drivers that decouple the managing process from the underlying technology. Whenever the core needs to manage a VM, it uses high-level commands such as "start VM," "stop VM,".
- **Image Management:** VMs are supported by a set of virtual disks or images, which contains the OS and any other additional software needed by the VM. OpenNebula assumes that there is an image repository that can be any storage medium or service, local or remote, that holds the base image of the VMs. There are a number of different possible configurations depending on the user's needs. For example, users may want all their images placed on a separate repository with only HTTP access. Alternatively, images can be shared through NFS between all the hosts.

DISTRIBUTED MANAGEMENT OF VIRTUAL INFRASTRUCTURES
VM Management

OpenNebula uses the following concepts for its image management model:

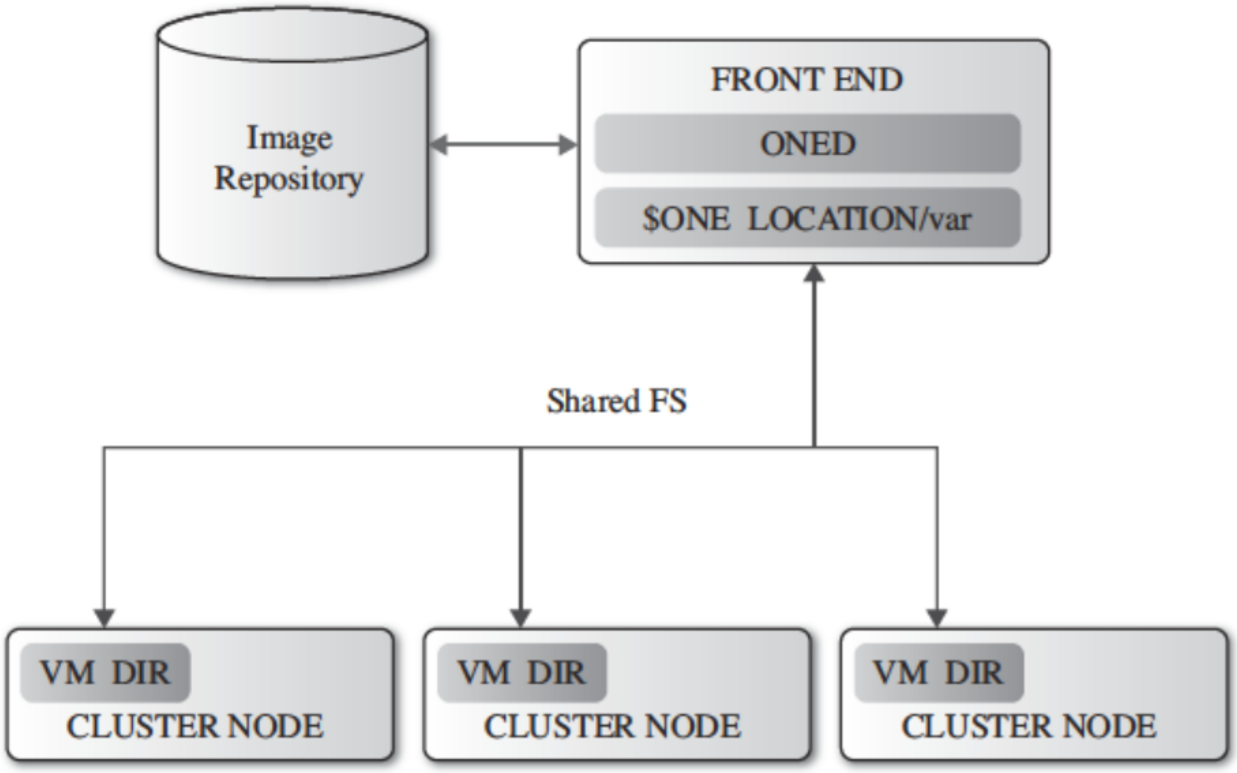


FIGURE 6.1. Image management in OpenNebula.

DISTRIBUTED MANAGEMENT OF VIRTUAL INFRASTRUCTURES
VM Management

- **Image Repositories** refer to any storage medium, local or remote, that hold the base images of the VMs. An image repository can be a dedicated file server or a remote URL from an appliance provider, but they need to be accessible from the OpenNebula front-end.

- **Virtual Machine Directory** is a directory on the cluster node where a VM is running. This directory holds all deployment files for the hypervisor to boot the machine, checkpoints, and images being used or saved—all of them specific to that VM. This directory should be shared for most hypervisors to be able to perform live migrations. Any given VM image goes through the following steps along its life cycle:

DISTRIBUTED MANAGEMENT OF VIRTUAL INFRASTRUCTURES
VM Management

- **Preparation** implies all the necessary changes to be made to the machine's image so it is prepared to offer the service to which it is intended. OpenNebula assumes that the images that conform to a particular VM are prepared and placed in the accessible image repository.
- **Cloning** the image means taking the image from the repository and placing it in the VM's directory in the physical node where it is going to be run before the VM is actually booted. If a VM image is to be cloned, the original image is not going to be used, and thus a copy will be used. There is a qualifier (Clone) for the images that can mark them targeting for cloning or not.
- **Save/Remove** if the save qualifier is disabled, once the VM has been shutdown, the images and all the changes thereof are going to be disposed of. However, if the save qualifier is activated, the image will be saved for later use.

DISTRIBUTED MANAGEMENT OF VIRTUAL INFRASTRUCTURES
VM Management

- **Networking.** Services deployed on a cloud, from a computing cluster to the classical three-tier business application, require several interrelated VMs, with a Virtual Application Network (VAN) being the primary link between them. OpenNebula dynamically creates these VANs and tracks the MAC addresses leased in the network to the service VMs. Other TCP/IP services such as DNS, NIS, or NFS are the responsibility of the service.
- The physical hosts that will co-form the fabric of our virtual infrastructures well need to have some constraints in order to effectively deliver virtual networks to our virtual machines. Therefore, from the point of view of networking, a physical cluster is defined as a set of hosts with one or more network interfaces, each of them connected to a different physical network.

DISTRIBUTED MANAGEMENT OF VIRTUAL INFRASTRUCTURES
VM Management

Three different VANs can be distinguished as follows:

- One is mapped on top of the public Internet network, and we can see a couple of virtual machines taking advantage of it. Therefore, these two VMs will have access to the Internet.
- The other two are mapped on top of the private physical network: The Red and Blue VANs. Virtual machines connected to the same private VAN will be able to communicate with each other, otherwise they will be isolated and won't be able to communicate.

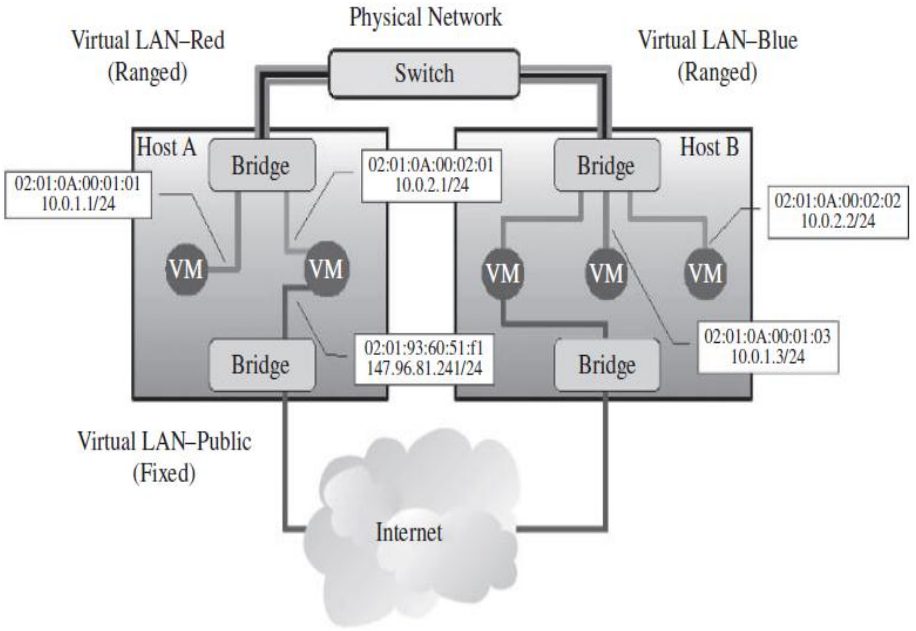


FIGURE 6.2. Networkig model for OpenNebula.

SCHEDULING TECHNIQUES FOR ADVANCERESERVATION OF CAPACITY

- While a VI manager like OpenNebula can handle all the minutiae of managing VMs in a pool of physical resources, scheduling these VMs efficiently is a different and complex matter.
- **Immediate provisioning model** is used by commercial cloud providers, such as Amazon, since their data centers' capacity is assumed to be infinite.
- **Best-effort provisioning** where requests have to be queued and prioritized
- **Advance provisioning** where resources are **pre-reserved** so they will be guaranteed to be available at a given time period.
- However, when managing a private cloud with limited resources, **an immediate provisioning model is insufficient**. A **lease-based resource provisioning model** that can act as a scheduling back-end for OpenNebula, **supporting other provisioning models other than the immediate provisioning models in existing cloud providers**. In particular, Haizea adds support for **both** best-effort provisioning and advance reservations, **when managing a finite number of resources**.

SCHEDULING TECHNIQUES FOR ADVANCE RESERVATION OF CAPACITY
Existing Approaches to Capacity Reservation

- Efficient reservation of resources in resource management systems has been studied considerably, particularly in the context of job scheduling.
- In fact, **most modern job schedulers support advance reservation of resources**, but their implementation falls short in several aspects.
- **First of all,** they are constrained by the job abstraction; when a user makes an advance reservation in a job-based system, the user does not have direct and unfettered access to the resources. Cloud users can access the VMs they requested, and are allowed to submit jobs to them.
- **Example-1: PBS Pro** creates a new queue that will be bound to the reserved resources, guaranteeing that jobs submitted to that queue will be executed on them (assuming they have permission to do so)
- **Example-2: Maui and Moab**, simply allow users to specify that a submitted job should use the reserved resources (if the submitting user has permission to do so).

SCHEDULING TECHNIQUES FOR ADVANCERESERVATION OF CAPACITY
Existing Approaches to Capacity Reservation

- Additionally, advance reservations lead to **utilization problems [10,13]**, caused by the need **to vacate resources before a reservation can begin**.
- Traditional job schedulers are unable to efficiently schedule workloads **combining both best-effort jobs and advance reservations**.
- However, **advance reservations** can be supported more efficiently by using a scheduler capable of **preempting running jobs at the start of the reservation and resuming them at the end of the reservation**.
- Preemption can also be used **to run large parallel jobs (which tend to have long queue times) earlier**, and it is specially relevant in the context of urgent computing, where resources have to be provisioned on very short notice and the likelihood of having jobs already assigned to resources is higher.
- While preemption can be accomplished **by canceling a running job**, the least disruptive form of preemption is **check pointing**, where the preempted **job's entire state is saved to disk**, allowing it to resume its work from the last checkpoint.

SCHEDULING TECHNIQUES FOR ADVANCERESERVATION OF CAPACITY
Existing Approaches to Capacity Reservation

- Additionally, some schedulers also support job migration, allowing check-pointed jobs to restart on other available resources, instead of having to wait until the preempting job or reservation has completed.
- Check-pointing-based preemption, requires the job's executable itself to be checkpointable. An application can be made checkpointable by explicitly adding that functionality to an application (application-level and library-level checkpointing) **OR** transparently by using OS-level checkpointing, where the operating system (such as Cray, IRIX, and patched versions of Linux using BLCR [17]) checkpoints a process, without rewriting the program or relinking it with checkpointing libraries.
- Thus, a job scheduler capable of checkpointing-based preemption and migration could be used **to checkpoint jobs before the start of an advance reservation, minimizing their impact on the schedule**.
- However, the **application and library-level checkpointing approaches burden the user with having to modify their applications to make them checkpointable**, imposing a restriction on the software environment. On the other hand, **OS-level checkpointing** is a more appealing option, but still **imposes certain software restrictions on resource consumers**.

SCHEDULING TECHNIQUES FOR ADVANCERESERVATION OF CAPACITY
Existing Approaches to Capacity Reservation

- An alternative approach to supporting advance reservations was proposed by Nurmi et al. [18], which introduced "**virtual advance reservations for queues**" **(VARQ)**.
- This approach **overlays advance reservations over traditional job schedulers** by **first** predicting the time a job would spend waiting in a scheduler's queue and **then** submitting a job (representing the advance reservation) at a time such that, based on the wait time prediction, the probability that it will be running at the start of the reservation is maximized.
- Since **no actual reservations** can be done, **VARQ jobs can run on traditional job schedulers**, which will not distinguish between the regular best-effort jobs and the VARQ jobs.
- Although this is an interesting approach that can be realistically implemented in practice (since it does not require modifications to existing scheduler), **it still depends on the job abstraction**.

SCHEDULING TECHNIQUES FOR ADVANCERESERVATION OF CAPACITY
Existing Approaches to Capacity Reservation

- **Hovestadt et al. [19, 20]** proposed a **planning-based** (**as opposed to queuing based**) approach to job scheduling, where job requests are immediately planned by making a reservation (now or in the future), instead of waiting in a queue.
- Thus, **advance reservations** are implicitly supported by a **planning-based system**. Additionally, each time a new request is received, the **entire schedule is reevaluated** to optimize resource usage.
- For example, a request for an advance reservation can be accepted without using preemption, since the jobs that were originally assigned to those resources can be assigned to different resources.

SCHEDULING TECHNIQUES FOR ADVANCERESERVATION OF CAPACITY
Reservations with VMs

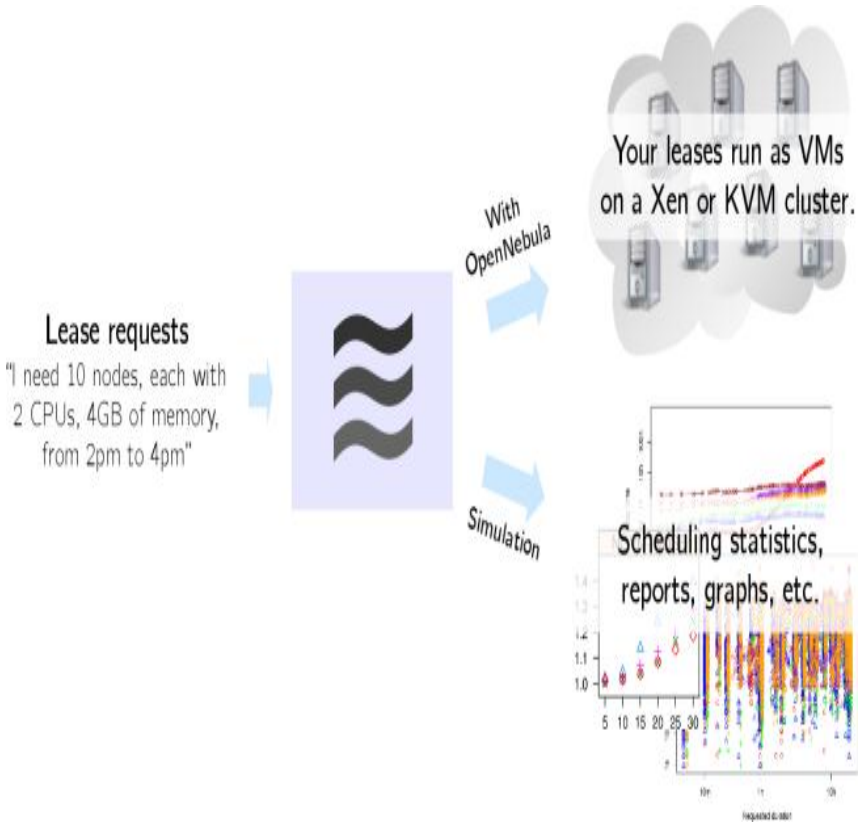
- Virtualization technologies are a key enabler of many features found in IaaS clouds. Virtual machines are also an appealing vehicle for implementing efficient reservation of resources due to:
  - Ability to be suspended,
  - Potentially migrated,
  - Resumed without modifying any of the applications running inside the VM.
- However, virtual machines also raise additional challenges related to the overhead of using VMs:
  - **Preparation Overhead.** When using VMs to implement reservations, a VM disk image must be either prepared on-the-fly or transferred to the physical node where it is needed. Since a VM disk image can have a size in the order of gigabytes, this preparation overhead can significantly delay the starting time of leases. This delay may, in some cases, be unacceptable for advance reservations that must start at a specific time.
  - **Runtime Overhead**. Once a VM is running, scheduling primitives such as **checkpointing** and **resuming** can incur in significant overhead since a VM's entire memory space must be saved to disk, and then read from disk. Migration involves transferring this saved memory along with the VM disk image. Similar to deployment overhead, this overhead can result in noticeable delays.

The Haizea project

http://haizea.cs.uchicago.edu/

- **The Haizea project** (http://haizea.cs.uchicago.edu/) was created to develop a scheduler that can efficiently support advance reservations efficiently by using the suspend/resume/migrate capability of VMs, but minimizing the overhead of using VMs.

- The fundamental resource provisioning abstraction in Haizea is **the lease**, with three types of lease currently supported:
- **Advanced reservation leases**, where the resources must be available ***at a specific time.***
- **Best-effort leases**, where resources are provisioned ***as soon as possible*** and requests are placed on a queue if necessary.
- **Immediate leases**, where resources are provisioned ***when requested*** or not at all.



Haizea's leasing model and the algorithms
Leasing Model

- when managing a private cloud with limited resources, **an immediate provisioning model is insufficient**. A **lease-based resource provisioning model** that can act as a scheduling back-end for OpenNebula, **supporting other provisioning models other than the immediate provisioning models in existing cloud providers**. In particular, Haizea adds support for both best-effort provisioning and advance reservations, **when managing a finite number of resources**. The remainder of this section describes Haizea's leasing model and the algorithms Haizea uses to schedule these leases.

- We define a **lease** as "**a negotiated and renegotiable agreement between a resource provider and a resource consumer, where the former agrees to make a set of resources available to the latter, based on a set of lease terms presented by the resource consumer**."

- The terms must encompass the following:
  - the hardware resources required by the resource consumer, such as CPUs, memory, and network bandwidth;
  - a software environment required on the leased resources;
  - and an availability period during which a user requests that the hardware and software resources be available.

Haizea's leasing model and the algorithms
Leasing Model

- We focus on the availability dimension of a lease and, in particular, on how to efficiently support advance reservations. Thus, we consider the following availability terms:
  - Start time may be unspecified (a best-effort lease) or specified (an advance reservation lease). In the latter case, the user may specify either a specific start time or a time period during which the lease start may occur.
  - Maximum duration refers to the total maximum amount of time that the leased resources will be available.
  - Leases can be preemptable. A preemptable lease can be safely paused without disrupting the computation that takes place inside the lease.

Haizea's leasing model and the algorithms
Leasing Model

- Haizea's resource model considers that it manages ***W*** physical nodes capable of running virtual machines. Each node ***i*** has CPUs, megabytes (MB) of memory, and MB of local disk storage.
- We assume that all disk images required to run virtual machines are available in a repository from which they can be transferred to nodes as needed and that all are connected at a bandwidth of ***B*** MB/sec by a switched network.
- A lease is implemented as a set of ***N*** VMs, each allocated resources described by a ***tuple (p, m, d, b)***, where ***p*** is number of CPUs, ***m*** is memory in MB, ***d*** is disk space in MB, and ***b*** is network bandwidth in MB/sec.
- A disk image ***I*** with a size of size ***(I)*** MB must be transferred from the repository to a node before the VM can start. When transferring a disk image to multiple nodes, we use multicasting and model the transfer time as size(I)/B.

Haizea's leasing model and the algorithms
Leasing Model

- If a lease is preempted, it is suspended by suspending its VMs, which may then be either resumed on the same node or migrated to another node and resumed there.

  - **Suspending a VM** results in a memory state image file (of size m that can be saved to either a local file system or a global file system (f ε {local, global}).
  - **Resumption** requires reading that image back into memory and then discarding the file.
  - **Suspension** of a single VM is done at a rate of ***s*** megabytes of VM memory per second, and we define ***r*** similarly for VM resumption.

Haizea's leasing model and the algorithms
Lease Scheduling

- **Haizea is designed to process lease requests and determine how those requests can be mapped to virtual machines, leveraging their suspend/resume/migrate capability, in such a way that the leases' requirements are satisfied.**
- **The scheduling component** of Haizea **allow best-effort leases to be preempted if resources have to be freed up for advance reservation requests**.
- Additionally, **to address the preparation and runtime overheads mentioned earlier**, **the scheduler allocates resources explicitly for the overhead activities (such as transferring disk images or suspending VMs)** instead of assuming they should be deducted from the lease's allocation.
- **Besides guaranteeing that certain operations complete on time** (e.g., an image transfer before the start of a lease), **the scheduler also attempts to minimize this overhead whenever possible, most notably by reusing disk image transfers and caching disk images on the physical nodes**.

Haizea's leasing model and the algorithms
Lease Scheduling –Best effort

- **Best-effort leases** are scheduled using a **queue**. When a best-effort lease is requested, the lease request is placed at the end of the queue, which is periodically evaluated using a backfilling algorithm to determine if any leases can be scheduled.

- The scheduler does this by **first checking the earliest possible starting time** for the lease on each physical node, which will depend on the required disk images. **For example**, if some physical nodes have cached the required disk image, it will be possible to start the lease earlier on those nodes.

- Once these earliest starting times have been determined, the scheduler chooses the nodes that allow the lease to start the soonest.

- The use of **VM suspension/resumption** allows the best-effort leases to be scheduled **even if there are not enough resources available for their full requested duration**.

Haizea's leasing model and the algorithms
Lease Scheduling-Advanced reservation

- **Advance reservations**, on the other hand, **do not go through a queue, since they must start at either the requested time or not at all**.
- Thus, scheduling this type of lease is relatively simple, because it mostly involves checking if there are enough resources available during the requested interval.
- However, the scheduler must also check **if any associated overheads can be scheduled** in such a way that the lease can still start **on time**.
- **For preparation overhead**, the scheduler determines if the required images can be transferred on time.
- These transfers are scheduled using an **Earliest Deadline First (EDF) algorithm**, where **the deadline for the image transfer is the start time of the advance reservation lease**.
- **For runtime overhead**, the scheduler will attempt to schedule the lease without having to preempt other leases; if preemption is unavoidable. The necessary suspension operations are scheduled; if they can be performed on time.

CAPACITY MANAGEMENT TO MEET SLA COMMITMENTS

- If temporal behavior of services with respect to resource demands is highly predictable, then capacity can be efficiently scheduled using reservations.

- In this section we focus on **less predictable elastic workloads**. For these workloads, **exact scheduling of capacity may not be possible**. Rather than that, **capacity planning and optimizations are required**.

- IaaS providers perform two complementary management tasks:

  (1) **Capacity planning** to make sure that SLA obligations are met as contracted with the service providers and;

  (2) **Continuous optimization** of resource utilization in specific workload to make the most efficient use of the existing capacity.

CAPACITY MANAGEMENT TO MEET SLA COMMITMENTS
Infrastructure SLAs

- IaaS can be regarded as a giant virtual hardware store, where computational resources such as virtual machines (VM), virtual application networks (VAN) and virtual disks (VD) can be **ordered on demand in the matter of minutes or even seconds**.
- Chandra et al. [29] quantitatively study advantages of **fine-grain resource allocation** in a shared hosting platform. As this research suggests, **fine-grain temporal and spatial resource allocation** may lead to substantial improvements in capacity utilization.
- Amazon EC2 [1] offers small, large, and extra large general-purpose VM instances and **high-CPU**, **medium** and **extra large** instances. It is possible that more instance types (**e.g., I/O high, memory high, storage high, etc.**) will be added in the future should a demand for them arise. Other IaaS providers—for example, GoGrid [3] and FlexiScale [4]—follow similar strategy.

CAPACITY MANAGEMENT TO MEET SLA COMMITMENTS
Infrastructure SLAs

- Thus, to deploy a service on a cloud, **a service provider orders suitable virtual hardware** and installs its application software on it.
- From the IaaS provider, a given service configuration is a virtual resource array of black box resources, which correspond to the number of instances of resource type.
- For example, a typical three-tier application may contain **ten general-purpose small instances** to run Web front-ends, **three large instances** to run an application server cluster with load balancing and redundancy, and **two large instances** to run a replicated database.
- A risk mitigation mechanism to protect user experience in the IaaS model is offered by infrastructure SLAs (i.e., the SLAs formalizing capacity availability) signed between service provider and IaaS provider.

CAPACITY MANAGEMENT TO MEET SLA COMMITMENTS
Infrastructure SLAs

- There is **no universal approach to infrastructure SLAs**. As the IaaS field matures and more experience is being gained, some methodologies may become more popular than others. Also some methods may be more suitable for specific workloads than other. There are three main approaches as follows.
- **No SLAs.** This approach is based on two premises: **(a)** Cloud always has spare capacity to provide on demand, and **(b)** services are not QoS sensitive and can withstand moderate performance degradation. This methodology is best suited for the best effort workloads.
- **Probabilistic SLAs.** These SLAs allow **us to trade capacity availability for cost of consumption**. Probabilistic SLAs specify clauses that determine availability percentile for contracted resources computed over the SLA evaluation period. **The lower the availability percentile, the cheaper the cost of resource consumption.** This type of SLA is **suitable for small and medium businesses** and **for many enterprise grade applications**.
- **Deterministic SLAs.** These are, in fact, probabilistic SLAs where **resource availability percentile is 100%**. These SLAs are most stringent and difficult to guarantee. From the provider's point of view, they do not admit capacity multiplexing. Therefore this is the most costly option for service providers, which may be applied for critical services.

CAPACITY MANAGEMENT TO MEET SLA COMMITMENTS
Infrastructure SLAs

- We will focus on probabilistic SLAs, however, because they represent the more interesting and flexible option and lay the foundation for the rest of discussion on **statistical multiplexing of capacity**.
- Before we can proceed, we need to define the concept, **elasticity rules**, which are are scaling and de-scaling policies that guide transition of the service from one configuration to another to match changes in the environment. The main motivation for defining these policies stems from the pay-as-you-go billing model of IaaS clouds. The service owner is interested in paying only for what is really required to satisfy workload demands minimizing the over-provisioning overhead. There are **three types of elasticity rules**:
  - **Time-driven:** These rules **change the virtual resources array in response to a timer event**. These rules are useful for predictable workloads—for example, for services with well-known business cycles.
  - **OS Level Metrics-Driven:** These rules react on predicates defined in terms of the OS parameters (see Amazon Auto-scaling Service). These **auto-scaling policies are useful for transparently scaling and de-scaling services**. The problem is, however, that in many cases **this mechanism is not precise enough**.
  - **Application Metrics-Driven:** This is a unique RESERVOIR offering that **allows an application to supply application-specific policies** that will be transparently executed by IaaS middleware in reacting on the monitoring information supplied by the service-specific monitoring probes running inside VMs.

CAPACITY MANAGEMENT TO MEET SLA COMMITMENTS
Infrastructure SLAs

- For a single service, elasticity rules of all three types can be defined, resulting in a complex dynamic behavior of a service during runtime.
- Assuming that **a business day is divided into a number of *usage windows***, the generic template for **probabilistic infrastructure SLAs** is as follows:
- For each $\mathbf{W_i}$, and each resource type $\mathbf{r_j}$ from the **virtual resource array**, **capacity range** $C = (r_j^{min}, r_j^{max})$ is available for the service with probability $\mathbf{p_i}$.
- Probabilistically guaranteeing capacity ranges allows service providers to define its needs flexibly. For example, for business critical usage window, availability percentile may be higher than the regular or off-peak hours.

CAPACITY MANAGEMENT TO MEET SLA COMMITMENTS
Policy-Driven Probabilistic Admission Control

- The purpose of statistical admission control is to guarantee that **there is enough capacity to find a feasible placement with given probability**. This mean over-subscription with probabilistically guaranteed risk of violating SLAs.
- Benefits of statistical multiplexing are well known. This is an extensively studied field in **networking** [30-32] and in the context of **CPU and bandwidth** allocation in shared hosting platforms Urgaonkar et al. [33].
- In this work [33], the resources were treated as contiguous, allowing infinitesimal capacity allocation. We generalize this approach by means of treating each (number of instances of resource ***i*** in the **virtual resources array**) as a **random variable**.
- The **virtual resources array** is, therefore, **a vector of random variables**. Since we assume that each **capacity range for each resource type is finite**, both **the average resource consumption rate and variance are computable** in resource consumption for each service in terms of the capacity units for each resource type.

CAPACITY MANAGEMENT TO MEET SLA COMMITMENTS
Policy-Driven Placement Optimization

- Policy-driven placement optimization complements capacity planning and management by improving a given mapping of physical to virtual resources (e.g., VMs).
- **Efficient capacity planning** with guaranteed minimal over-provisioning is still an open research problem.
- Policy-driven management is a management approach based on **“if(condition) then(action)”** rules defined to deal with the situations that are likely to arise [40].
- These policies serve as a basic building blocks for autonomic computing.
- Partially the difficulties lie in hardness of solving **multiple knapsacks** or its more general version, the **generalized assignment problem**. Both problems are NP-hard in the strong sense.

CAPACITY MANAGEMENT TO MEET SLA COMMITMENTS
Management Policies and Management Goals.

- **Management Policies and Management Goals**. Policy-based management is an overused term. Therefore, it is, beneficial to define and differentiate our approach to policy-driven admission control and placement optimization in the more precise terms.
- The overall optimality criteria of placement, however, are controlled by the management policies, which are defined at a higher level of abstraction than “if (condition)then(action)” rules. To avoid ambiguity, we term these policies **management goals**.
- Management goals, such as “**conserve power**,” “**prefer local resources over remote resources**,” “**balance workload**,” “**minimize VM migrations**,” “**minimize SLA non-compliance**,” and so forth, have complex logical structures.

CAPACITY MANAGEMENT TO MEET SLA COMMITMENTS
Business-Level Goals and Policies.

- Since business goals are defined at such a high level of abstraction, a semantic gap exists between them and the ICT level management goals and policies. Bridging this gap is notoriously difficult. In this work we aim at narrowing this gap and aligning between the high-level business management goals and ICT-level management policies by introducing the notion of **Acceptable Risk Level (ARL)** of capacity allocation congestion.
- Intuitively, we are interested in minimizing the costs of capacity over-provisioning.
- From minimizing the cost of capacity over-provisioning, we are interested in **maximizing yield (usage)** of the existing capacity.
- However, at some point, the conflicts (congestions) in capacity allocation may cause SLA penalties that is opposite to the advantages of yield maximization.

CAPACITY MANAGEMENT TO MEET SLA COMMITMENTS
Infrastructure-Level Management Goals and Policies

- In general, infrastructure-level management policies are derived from the business-level management goals. For example, consider our sample business level management goal to “reduce energy expenses by 30% in the next quarter.”
- This broadly defined goal may imply, among other means for achieving it, that we systematically improve consolidation of VMs on physical hosts by putting excessive capacity into a low-power consumption mode. Thus, a site-wide ICT power conservation-level management policy may be formulated as: “**minimize number of physical machines while protecting capacity availability SLAs of the application services.**”

- Another example, consider the business-level management goal: “**Improve customer satisfaction by achieving more aggressive performance SLAs.**” One possible policy toward satisfying this business-level goal may be formulated as:

  “**Balance load within the site in order to achieve specific average load per physical host.**”
- Another infrastructure-level management policy to improve performance is: “**Minimize the number of VM migrations.**” The rationale for this policy is that performance degradation necessarily occurs during VM migration.