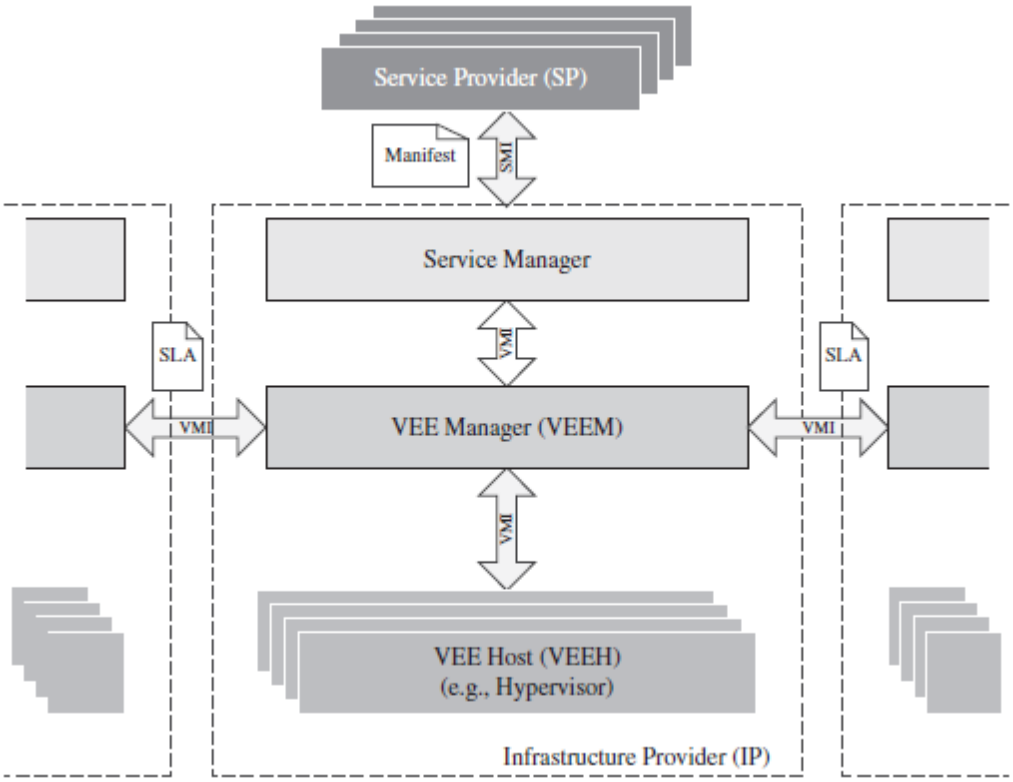
UNIT IV

A MODEL FOR FEDERATED CLOUD COMPUTING

In our model for federated cloud computing we identify two major types of actors: Service Providers (SPs) are the entities that need computational resources to offer some service. However, SPs do not own these resources; instead, they lease them from Infrastructure Providers (IPs), which provide them with a seemingly infinite pool of computational, network, and storage resources.

To create the illusion of an infinite pool of resources, IPs shared their unused capacity with each other to create a federation cloud. A Framework Agreement



is document that defines the contract between two IPs—that is, it states the terms and conditions under which one IP can use resources from another IP. Within each IP, optimal resource utilization is achieved by partitioning physical resources, through a virtualization layer, into Virtual Execution Environments (VEEs)—fully isolated runtime environments that abstract away the physical characteristics of the resource and enable sharing. We refer to the virtualized computational resources, alongside the virtualization layer and all the management enablement components, as the Virtual Execution Enviroment Host (VEEH).

According to above architecture i.e RESERVOIR.The rationale behind this particular layering is to keep a clear separation of concerns and responsibilities and to hide low-level infrastructure details and decisions from high-level management and service providers.

The **Service Manager** is the only component within an IP that interacts with SPs. It receives Service Manifests, negotiates pricing, and handles billing. Its two most complex tasks are (1) deploying and provisioning VEEs based on the Service Manifest and (2) monitoring and enforcing SLA compliance by throttling a service application's capacity.

The Virtual Execution Environment Manager (VEEM) is responsible for the optimal placement of VEEs into VEE Hosts subject to constraints determined by the Service Manager. The continuous optimization process is driven by a site-specific programmable utility function. The VEEM is free to place and move VEEs anywhere, even on the remote sites (subject to overall cross-site agreements), as long as the placement satisfies the constraints. Thus, in addition to serving local requests (from the local ServiceManager), VEEM is responsible for the federation of remote sites.

The Virtual Execution Environment Host (VEEH) is responsible for the basic control and monitoring of VEEs and their resources (e.g., creating a VEE, allocating additional resources to a VEE, monitoring a VEE, migrating a VEE, creating a virtual network and storage pool, etc.). Given that VEEs belonging to the same application may be placed on multiple VEEHs and even extend beyond the boundaries of a site, VEEHs must support isolated virtual networks that span VEEHs and sites. Moreover, VEEHs must support transparent VEE migration to any compatible VEEH within the federated cloud, regardless of site location or network and storage configurations. The layered design stresses the use of standard, open, and generic protocols and interfaces to support vertical and horizontal interoperability between layers. Different implementations of each layer will be able to interact with each other.

SECURITY CONSIDERATIONS

Threats of large-scale cross-border virtualization infrastructures are broadly classified into two major categories, namely, external threats and internal threats.

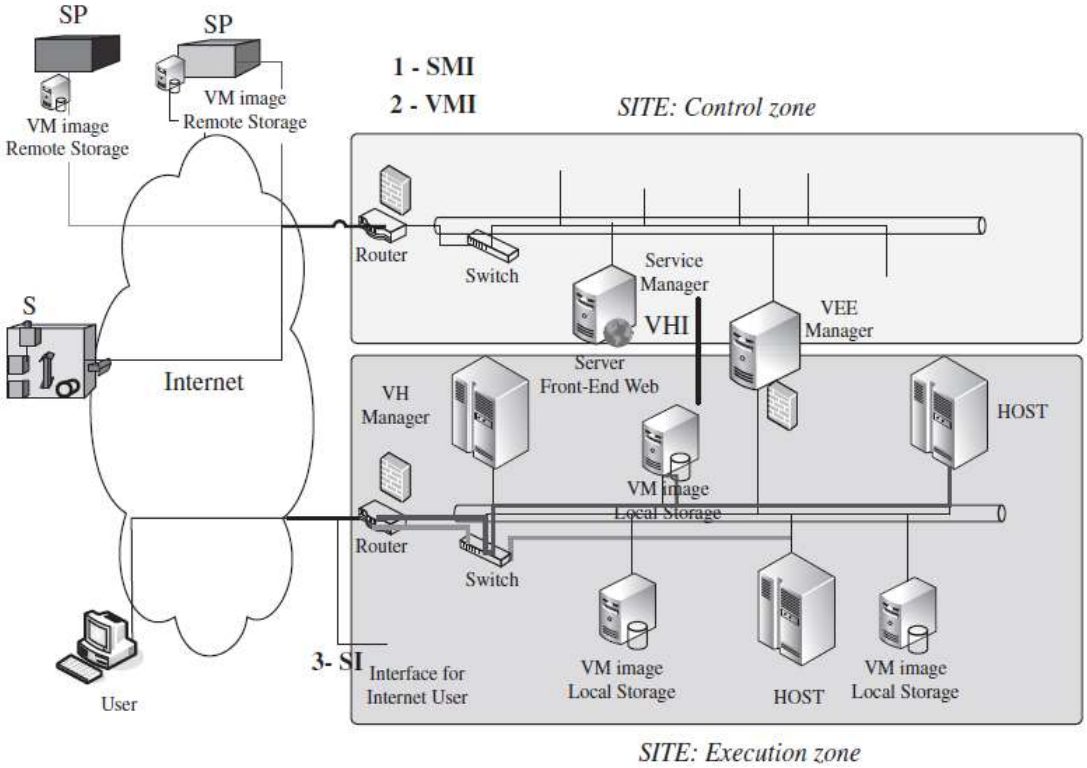
External Threats

The Internet represents the same origin of threats for the communication across the RESERVOIR sites (VMI interfaces) and outside the RESERVOIR sites both for the SMI interface and service interface (SI—interface for service user on Internet). Some threats, related to communication, can be classified as: menin the-middle, TCP hijacking (spoofing), service manifest attacks (malicious manifest/SLA format injection), migration and security policies and identity theft/impersonation (SP or RESERVOIR site pretends to be someone else), and so on. The main goals of these threats are to gain unauthorized access to systems

and to impersonate another entity on the network. These techniques allow the attackers to eavesdrop as well as to change, delete, or divert data. All the interfaces could be instead exposed to the following attacks: denial of service (DoS or distributed DoS), flooding, buffer overflow, p2p-attacks, and so on.

Internal Threats

Each RESERVOIR site has a logical representation with three different layers, but these layers can be compounded by one or more hardware components. Figure gives an overview of these entities and relative mapping with a simplified view of the hardware. It is possible to split the site in two different virtual zones: control and execution zone; in the control zone the components are: Service Manager (SM), VEEM (in bridge configuration between control



and execution zone), network components (router, switch, cable, etc.), SMI/ VMI interfaces, and VHI internal interface.

In the execution zone instead there are: VEEH, VEEM (in-bridge configuration between control and execution zone), VHI internal interface, network components (router, switch, cable, etc.), network storage (NAS, databases, etc.), and SI (user access interfaces).

The control zone can be considered a trusted area. Some threats can appear through the SMI and VEEM interfaces, since they fall into the same cases of external threats. The firewall located next to the router increases the trustworthiness.

In this zone the weak ring of the system is represented by the VEEM. It is the bridge between two areas, and it allows the exchange of data among the zones. Figure shows a firewall close

to the VEEM, added to prevent any attacks from the execution area. The zone with a high level of risk is represented by the execution zone. This area shares all the hardware components. The hypervisor (VEEH) uses the network, storage, CPU, and ram (host) to load and execute all the VEEs. To better explain the role of each component, it can be useful to evaluate chronologically all the phases necessary to execute a virtual execution environment (VEEH); once all the requirements from the VEEM are received, it downloads the VM image from the SP, stores the image into the NAS, performs the setup configuration, and executes the VM. The internal threats related to these phases can be classified as follows: (1) threats linked to authentication/communication of SPs and other RESERVOIR site; (2) threats related to misbehavior of service resource allocation—to alter the agreement (manifest) during the translation between service manager and VEEM malicious component on SM; (3) data export control legislation—on an international cloud or between two clouds; (4) threats linked to fake command for placement of VEEs and compromising the data integrity of the distributed file system (NFS, SAMBA, CIFS); (5) storage data compromising (fake VEE image); (6) threats linked to compromise data privacy; (7) threats linked to the underlying hypervisor and OS (VEE could break hypervisor/ underlying OS security and access other VEE); and (8) data partitioning between VEE.

To avoid any fraudulent access, the VEEH has to verify authentication/ communication of SPs and other RESERVOIR sites. Thus, the same behavior is analyzed for all the communications in external threats. Relatively to the latter group of threats, the RESERVOIR site has to guarantee different types of isolation—that is, runtime isolation, network isolation, and storage isolation. Runtime isolation resolves all the security problems with the underlying OS. The hypervisor security mechanisms need to be used to provide the isolation.

Network isolation is addressed via the dynamic configuration of network policies and via virtual circuits that involve routers and switches. To avoid fake VEE image loading and do not compromise data privacy, storage isolation has to be performed and secure protocols has to be used. Protocols like NFS, SAMBA, and CIFS are not secure.

TRADITIONAL APPROACHES TO SLA MANAGEMENT

Traditionally, load balancing techniques and admission control mechanisms have been used to provide guaranteed quality of service (QoS) for hosted web applications. These mechanisms can be viewed as the first attempt towards managing the SLAs.

Types of SLA

Service-level agreement provides a framework within which both seller and buyer of a service can pursue a profitable service business relationship.

Service-Level Parameter	Describes an observable property of a service whose value is measurable.
Metrics	These are definitions of values of service properties that are measured from a service-providing system or computed from other metrics and constants. Metrics are the key instrument to describe exactly what SLA parameters mean by specifying how to measure or compute the parameter values.
Function	A function specifies how to compute a metric's value from the values of other metrics and constants. Functions are central to describing exactly how SLA parameters are computed from resource metrics.
Measurement directives	These specify how to measure a metric.

Infrastructure SLA. The infrastructure provider manages and offers guarantees on availability of the infrastructure, namely, server machine, power, network connectivity, and so on. Enterprises manage themselves, their applications that are deployed on these server machines. The machines are leased to the customers and are isolated from machines of other customers.

Application SLA. In the application co-location hosting model, the server capacity is available to the applications based solely on their resource demands. Hence, the service providers are flexible in allocating and de-allocating computing resources among the co-located applications.

It is also possible for a customer and the service provider to mutually agree upon a set of SLAs with different performance and cost structure rather than a single SLA.

LIFE CYCLE OF SLA

Each SLA goes through a sequence of steps starting from identification of terms and conditions, activation and monitoring of the stated terms and conditions, and eventual termination of contract once the hosting relationship ceases to exist. Such a sequence of steps is called SLA life cycle and consists of the following five phases:

1. Contract definition
2. Publishing and discovery
3. Negotiation
4. Operationalization
5. De-commissioning

Contract Definition. Generally, service providers define a set of service offerings and corresponding SLAs using standard templates. These service offerings form a catalog. Individual SLAs for enterprises can be derived by customizing these base SLA templates.

Publication and Discovery. Service provider advertises these base service offerings through standard publication media, and the customers should be able to locate the service provider by searching the catalog. The customers can search different competitive offerings and shortlist a few that fulfill their requirements for further negotiation.

Negotiation. Once the customer has discovered a service provider who can meet their application hosting need, the SLA terms and conditions needs to be mutually agreed upon before signing the agreement for hosting the application. For a standard packaged application which is offered as service, this phase could be automated. For customized applications that are hosted on cloud platforms, this phase is manual. The service provider needs to analyze the application's behavior with respect to scalability and performance before agreeing on the specification of SLA. At the end of this phase, the SLA is mutually agreed by both customer and provider and is eventually signed off. SLA negotiation can utilize the WS-negotiation specification.

Operationalization. SLA operation consists of SLA monitoring, SLA accounting, and SLA enforcement. SLA monitoring involves measuring parameter values and calculating the metrics defined as a part of SLA and determining the deviations. On identifying the deviations, the concerned parties are notified. SLA accounting involves capturing and archiving the SLA adherence for compliance.

De-commissioning. SLA decommissioning involves termination of all activities performed under a particular SLA when the hosting relationship between the service provider and the service consumer has ended. SLA specifies the terms and conditions of contract termination and specifies situations under which the relationship between a service provider and a service consumer can be considered to be legally ended.

GRID AND CLOUD

"Grid vs Cloud" is the title of an incredible number of recent Web blogs and articles in on-line forums and magazines, where many HPC users express their own opinion on the relationship between the two paradigms.

Cloud is simply presented, by its supporters, as an evolution of the grid. Some consider grids and clouds as alternative options to do the same thing in a different way. However, there are very few clouds on which one can build, test, or run compute-intensive applications. In fact it still necessary to deal with some open issues. One is when, in term of performance, a cloud is better than a grid to run a specific application. Another problem to be addressed concerns the effort to port a grid application to a cloud. In the following it will be discussed how these and other arguments suggest that we investigate the integration of grids and clouds to improve the exploitation of computing resources in HPC.

Grid and Cloud as Alternatives

Both grid and cloud are technologies that have been conceived to provide users with handy computing resources according to their specific requirements. Grid was designed with a bottom-up approach. Its goal is to share a hardware or a software among different organizations by means of common protocols and policies. The idea is to deploy interoperable services in order to allow the access to physical resources (CPU, memory, mass storage, etc.) and to available software utilities. Users get access to a real machine. Grid resources are administrated by their owners. Authorized users can invoke grid services on remote machines without paying and without service level guarantees.

A grid middleware provides a set of API (actually services) to program a heterogeneous, geographically distributed system.

On the other hand, cloud technology was designed using a top-down approach. It aims at providing its users with a specific high-level functionality: a storage, a computing platform, a specialized service. They get virtual resources from the cloud. The underlying hardware/software infrastructure is not exposed. The only information the user needs to know is the quality of service (QoS) of the services he is paying for. Bandwidth, computing power, and storage represent parameters that are used for specifying the QoS and for billing. Cloud users ask for a high-level functionality (service, platform, infrastructure), pay for it, and become owners of a virtual machine. From a technological point of view, virtualization is exploited to build an insulated environment, which is configured to meet users' requirements and is exploited for easy reconfiguration and backup. A single enterprise is the owner of the cloud platform (software and underlying hardware), whereas customers become owners of the virtual resources they pay for.

Grid & Cloud Integration:

The integration of the two environments is

Grid on Cloud. A cloud IaaS (Infrastructure as a Service) approach is adopted to build up and to manage a flexible grid system. Doing so, the grid middleware runs on a virtual machine. Hence the main drawback of this approach is performance. Virtualization inevitably entails performance losses as compared to the direct use of physical resources.

Cloud on Grid: The stable grid infrastructure is exploited to build up a cloud environment. This solution is usually preferred because the cloud approach mitigates the inherent complexity of the grid. In this case, a set of grid services is offered to manage (create, migrate, etc.) virtual machines. The use of Globus workspaces , along with a set of grid services for the Globus Toolkit 4, is the prominent solution, as in the Nimbus project .

HPC IN THE CLOUD

Three different issues for HPC in the Cloud

- The difference between typical HPC paradigms and those of current cloud environments, especially in terms of performance evaluation.
- A comparison of the two approaches in order to point out their advantages and drawbacks, as far as performance is concerned.
- New performance evaluation techniques and tools to support HPC in cloud systems.

The first and well-known difference between HPC and cloud environments is the different economic approach:
(a) buy-and-maintain for HPC and
(b) pay-per-use in cloud systems.

The performance and cost can be calculated as shown in below

$$\langle \text{cost per hour per instance} \rangle * \langle \text{numberofinstances} \rangle * \langle \text{hours} \rangle$$