MALLA REDDY COLLEGE OF ENGINEERING AND TECHNOLOGY

Compiled By,
Faculty of Cloud Computing
Department of CSE

Unit-1

SYLLABUS

UNIT - I

Systems Modeling ,Clustering and virtualization: Distributed system Models and Enabling Technologies, Computer Clusters for scalable parallel computing, Virtual Machines and Virtualization of clusters and data centers

Unit-II

Foundations: Introduction to cloud computing, Migrating into a Cloud , Enriching the 'Integration as a Service' Paradigm for the cloud era, The Enterprise Cloud computing paradigm

UNIT--III

Infrastructure as a service

Virtual machines provisioning and Migrations services, On the Management of Virtual machines for Cloud Infrastructures, Enhancing cloud computing environments using a cluster as a service, Secure distributed data storage in cloud computing

Platform as a service

Aneka, Comet cloud, T-systems Work flow engine for cloud, Understanding scientific Applications for cloud Environments

Unit IV

Monitoring and Management: An Architecture Federated cloud commuting, SLA Management in cloud computing, Performance Prediction for HPC on Clouds,Best Practices in Architecting cloud applications in the AWS cloud, Building content delivery networks using clouds, Resource cloud Mashups

UNIT-V

Governance and case studies: Organizational Readiness and change Management in the cloud age, Data security in the cloud, Legal Issues in cloud computing, Achieving production readiness for cloud services.

Text Books

1. Cloud Computing: Principles and Paradigms by Rajkumar Buyya, Wiley, 2011.

2. Distributed and cloud computing, Kai Hwang, Geoffrey C. Fox,Jack J.Donnagarra,Elsevier,2012

UNIT 1

INTRODUCTION

- Centralized computing: This is a computing paradigm by which all computer resources are centralized in one physical system. All resources (processors, memory, and storage) are fully shared and tightly coupled within one integrated OS. Many data centers and supercomputers are centralized systems, but they are used in parallel, distributed, and cloud computing applications

- Parallel computing: In parallel computing, all processors are either tightly coupled with centralized shared memory or loosely coupled with distributed memory. Some authors refer to this discipline as parallel processing. Inter processor communication is accomplished through shared memory or via message passing. A computer system capable of parallel computing is commonly known as a parallel computer. Programs running in a parallel computer are called parallel programs. The process of writing parallel programs is often referred to as parallel programming.

- **Distributed computing**: This is a field of computer science/engineering that studies distributed systems. A distributed system consists of multiple autonomous computers, each having its own private memory, communicating through a computer network. Information exchange in a distributed system is accomplished through message passing. A computer program that runs in a distributed system is known as a distributed program. The process of writing distributed programs is referred to as distributed programming.

- **Cloud computing**: An Internet cloud of resources can be either a centralized or a distributed computing system. The cloud applies parallel or distributed computing, or both. Clouds can be built with physical or virtualized resources over large data centers that are centralized or distributed. Some authors consider cloud computing to be a form of utility computing or service computing.

- **Ubiquitous computing** refers to computing with pervasive devices at any place and time using wired or wireless communication.

- **The Internet of Things (IoT)** is a networked connection of everyday objects including computers, sensors, humans, etc. The IoT is supported by Internet clouds to achieve ubiquitous computing with any object at any place and time.

- **High-performance computing(HPC)** emphasize the raw speed performance. The speed of HPC systems has increased from Gflops in the early 1990s to now Pflops in 2010. This improvement was driven mainly by the demands from scientific, engineering, and manufacturing communities.

- **High-throughput computing (HTC)** systems pay more attention to high-flux computing. The main application for high-flux computing is in Internet searches and web services by millions or more users simultaneously. The performance goal thus shifts to measure high throughput or the number of tasks completed per unit of time. HTC technology needs to not only improve in terms of batch processing speed, but also address the acute problems of cost, energy savings, security, and reliability at many data and enterprise computing centers.

Degrees of Parallelism:

Bit-level parallelism (BLP) converts bit-serial processing to word-level processing gradually. Over the years, users graduated from 4-bit microprocessors to 8-,16-, 32-, and 64-bit CPUs.

Instruction-level parallelism (ILP), in which the processor executes multiple instructions simultaneously rather than only one instruction at a time.

Data-level parallelism (DLP) was made popular through SIMD (single instruction, multiple data) and vector machines using vector or array types of instructions. DLP requires even more hardware support and compiler assistance to work properly.

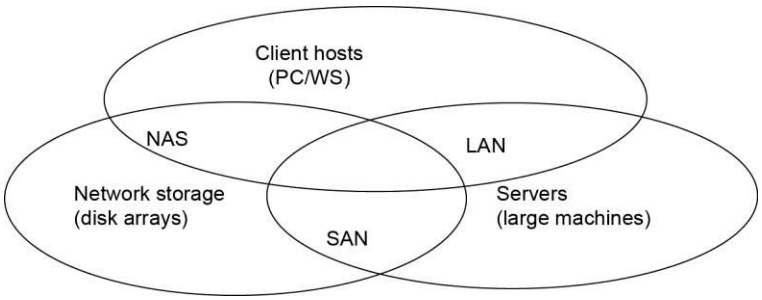
Ever since the introduction of multicore processors and chip multiprocessors (CMPs), we have been exploring **Task-level parallelism (TLP**).

TECHNOLOGIES FOR NETWORK-BASED SYSTEMS

Multicore CPUs and Multithreading Technologies: Today, advanced CPUs or microprocessor chips assume a multicore architecture with dual, quad, six, or more processing cores. These processors exploit parallelism at ILP and TLP levels. Both multi-core CPU and many-core GPU processors can handle multiple instruction threads at different magnitudes today. Multiple cores are housed in the same chip with an L2 cache that is shared by all cores. In the future, multiple CMPs could be built on the same CPU chip with even the L3 cache on the chip. Multicore and multithreaded CPUs are equipped with many high-end processors, including the Intel i7, Xeon, AMD Opteron, Sun Niagara, IBM Power 6, and X cell processors. Each core could be also multithreaded.

Memory, Storage, and Wide-Area Networking: Memory chips have experienced a 4x increase in capacity every three years. For hard drives, capacity increased from 260 MB in 1981 to 250 GB in 2004. Disks or disk arrays have exceeded 3 TB in capacity. The rapid growth of flash memory and solid-state drives (SSDs) also impacts the future of HPC and HTC systems.

System-Area Interconnects: The nodes in small clusters are mostly interconnected by an Ethernet switch or a local area network(LAN).



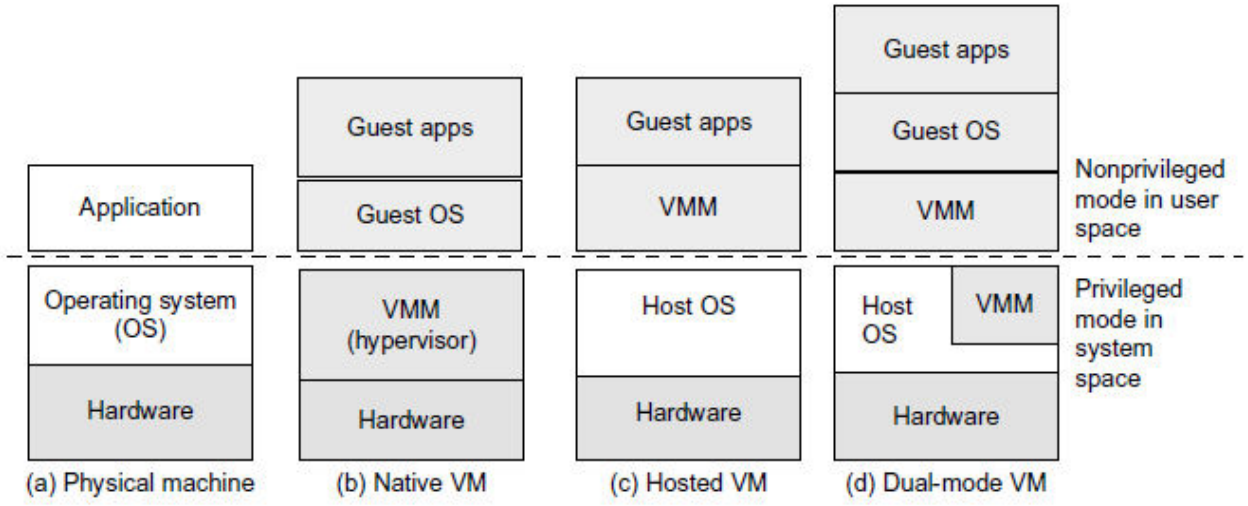
As Figure shows, a LAN typically is used to connect client hosts to big servers.A storage area network (SAN) connects servers to network storage such as disk arrays. Network attached

storage (NAS) connects client hosts directly to the disk arrays. All three types of networks often appear in a large cluster built with commercial network components.

Wide-Area Networking: High-bandwidth networking increases the capability of building massively distributed systems. The rapid growth of Ethernet bandwidth from 10 Mbps in1979 to 1 Gbps in 1999, and 40 ~ 100 GE in 2011. It has been speculated that 1 Tbps network links will become available by 2013.

Virtual Machines and Virtualization Middleware

Virtual machines (VMs) offer novel solutions to underutilized resources, application inflexibility, software manageability, and security concerns in existing physical machines. Today, to build large clusters, grids, and clouds, we need to access large amounts of computing, storage, and networking resources in a virtualized manner. We need to aggregate those resources, and hopefully, offer a single system image. In particular, a cloud of provisioned resources must rely on virtualization of processors, memory, and I/O facilities dynamically.



Three VM architectures in (b), (c), and (d), compared with the traditional physical machine shown in (a).

The host machine is equipped with the physical hardware. The VM is built with virtual resources managed by a guest OS to run a specific application. Between the VMs and the host platform, one needs to deploy a middleware layer called a virtual machine monitor (VMM).

Figure shows a native VM installed with the use of a VMM called a hypervisor in privileged Mode. The guest OS could be a Linux system and the hypervisor is the XEN system developed at Cambridge University. This hypervisor approach is also called bare-metal VM, because the hypervisor handles the bare hardware (CPU, memory, and I/O) directly. Architecture is the host VM shown in Figure(c). Here the VMM runs in non-privileged mode. The host OS need not be modified. The VM can also be implemented with a dual mode, as shown in Figure 1.12(d). Part of the VMM runs at the user level and another part runs at the supervisor level. In this case, the host OS may have to be modified to some extent. Multiple VMs can be ported to a given hardware system to support the virtualization process. The VM approach offers hardware independence of the OS and applications.

VM Primitive Operations: The VMM provides the VM abstraction to the guest OS. With full virtualization, the VMM exports a VM abstraction identical to the physical machine so that a standard OS such as Windows 2000 or Linux can run just as it would on the physical hardware

Low-level VMM operations are

- the VMs can be multiplexed between hardware machines,
- a VM can be suspended and stored in stable storage
- a suspended VM can be resumed or provisioned to a new hardware platform
- a VM can be migrated from one hardware platform to another

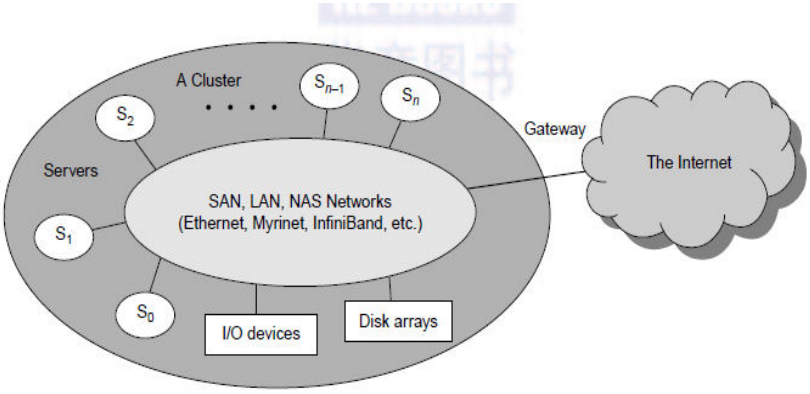
These VM operations enable a VM to be provisioned to any available hardware platform. They also enable flexibility in porting distributed application executions. Furthermore, the VM approach will significantly enhance the utilization of server resources.

SYSTEM MODELS FOR DISTRIBUTED AND CLOUD COMPUTING

Distributed and cloud computing systems are built over a large number of autonomous computer nodes. These node machines are interconnected by SANs, LANs, or WANs in a hierarchical manner. Massive systems are considered highly scalable, and can reach web-scale connectivity, either physically or logically. Massive systems are classified into four groups: clusters, P2Pnetworks, computing grids, and Internet clouds over huge data centers. In terms of node number, these four system classes may involve hundreds, thousands, or even millions of computers as participating nodes. These machines work collectively, cooperatively, or collaboratively at various levels.

1. **Clusters of Cooperative Computers** A computing cluster consists of interconnected stand-alone computers which work cooperatively as a single integrated computing resource.

Cluster Architecture



A cluster of servers interconnected by a high-bandwidth

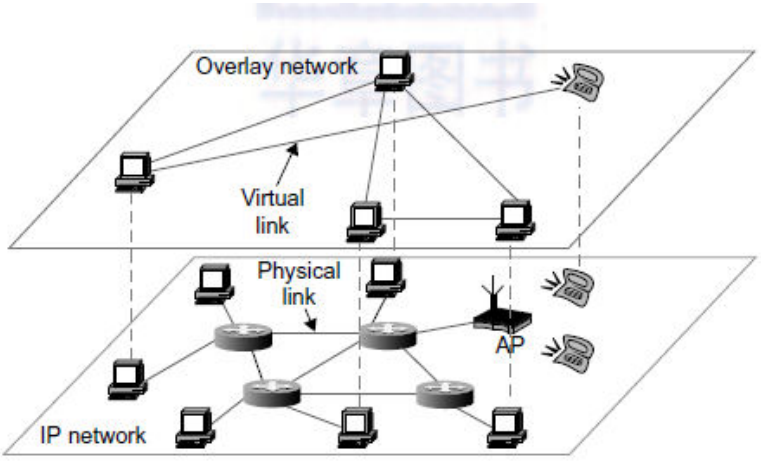
Figure shows the architecture of a typical server cluster built around a low-latency, high bandwidth interconnection network. Through hierarchical construction using a SAN, LAN, or WAN, one can build scalable clusters with an increasing number of nodes. The cluster is connected to the Internet via a virtual private network (VPN) gateway. The gateway IP address locates the cluster. The system image of a computer is decided by the way the OS manages the shared cluster resources

Single-System Image: An ideal cluster should merge multiple system images into a single-system image (SSI). Cluster designers desire a cluster operating system or some middleware to support SSI at various levels, including the sharing of CPUs, memory, and I/O across all cluster nodes. An SSI is an illusion created by software or hardware that presents a collection of resources as one integrated, powerful resource. SSI makes the cluster appear like a single machine to the user.

Hardware, Software, and Middleware Support: Clusters exploring massive parallelism are commonly known as MPPs. Special cluster middleware supports are needed to create SSI or high availability (HA). Both sequential and parallel applications can run on the cluster, and special parallel environments are needed to facilitate use of the cluster resources

2. **Grid Computing Infrastructures: A** computing grid offers an infrastructure that couples computers, software/middleware, special instruments, and people and sensors together. The grid is often constructed across LAN, WAN, or Internet backbone networks at a regional, national, or global scale. Enterprises or organizations present grids as integrated computing resources. They can also be viewed as virtual platforms to support virtual organizations. The computers used in a grid are primarily workstations, servers, clusters, and supercomputers. Personal computers, laptops, and PDAs can be used as access devices to a grid system.

3. **Peer-to-Peer Network Families**



The structure of a P2P system by mapping a physical IP network to an overlay network

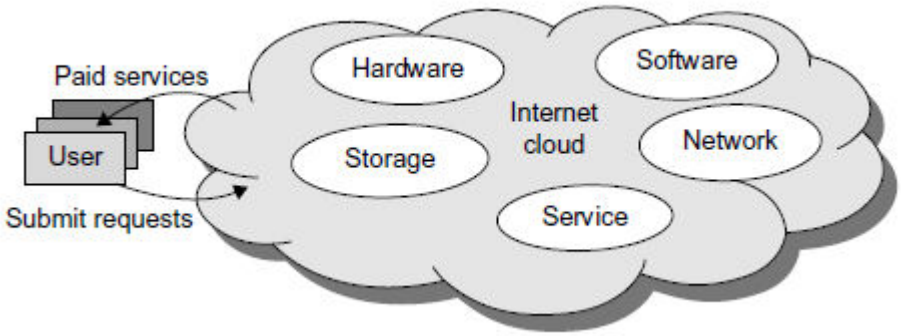
In a P2P system, every node acts as both a client and a server, providing part of the system resources. Peer machines are simply client computers connected to the Internet. All client machines act autonomously to join or leave the system freely. This implies that no master-slave relationship exists among the peers. No central coordination or central database is needed. In other words, no peer machine has a global view of the entire P2P system. The system is self-organizing with distributed control. Unlike the cluster or grid, a P2P network does not use a dedicated interconnection network. The physical network is simply an ad hoc network formed at various Internet domains randomly using the TCP/IP and NAI protocols

Data items or files are distributed in the participating peers. Based on communication or file-sharing needs, the peer IDs form an overlay network at the logical level. This overlay is a virtual

network formed by mapping each physical machine with its ID, logically, through a virtual mapping

P2P performance is affected by routing efficiency and self-organization by participating peers. Fault tolerance, failure management, and load balancing are other important issues in using overlay networks. Lack of trust among peers poses another problem. Peers are strangers to one another. Security, privacy, and copyright violations are major worries

4. **Cloud Computing over the Internet:** A cloud is a pool of virtualized computer resources. A cloud can host a variety of different workloads, including batch-style backend jobs and interactive and user-facing applications.



Virtualized resources from data centers to form an Internet cloud

Cloud computing applies a virtualized platform with elastic resources on demand by provisioning hardware, software, and data sets dynamically (see Figure 1.18). The idea is to move desktop computing to a service-oriented platform using server clusters and huge databases at data centers

THREE CLOUD SERVICE MODELS

- **Infrastructure as a Service (IaaS)** This model puts together infrastructures demanded by users—namely servers, storage, networks, and the data center fabric. The user can deploy and run on multiple VMs running guest OSes on specific applications. The user does not manage or control the underlying cloud infrastructure, but can specify when to request and release the needed resources.
- **Platform as a Service (PaaS)** This model enables the user to deploy user-built applications onto a virtualized cloud platform. PaaS includes middleware, databases, development tools, and some runtime support such as Web 2.0 and Java. The platform includes both hardware and software integrated with specific programming interfaces. The provider supplies the API and software tools (e.g., Java, Python, Web 2.0, .NET). The user is freed from managing the cloud infrastructure.
- **Software as a Service (SaaS)** This refers to browser-initiated application software over thousands of paid cloud customers. The SaaS model applies to business processes, industry applications, consumer relationship management (CRM), enterprise resources planning (ERP),human resources (HR), and collaborative applications. On the customer side, there is no upfront investment in servers or software licensing. On the provider side, costs are rather low, compared with conventional hosting of user applications.

SOFTWARE ENVIRONMENTS FOR DISTRIBUTED SYSTEMS AND CLOUDS

Service Oriented Architecture (SOA)

- A paradigm for organizing and utilizing distributed capabilities that may be under the control of different ownership domains and implemented using various technology stacks
- A set of components which can be invoked and whose interface descriptions can be published and discovered (W3C)
- SOA is an architectural style whose goal is to achieve loose coupling among interacting software agents
- A service is a unit of work done by a service provider to achieve desired end results for a service consumer
- Both provider and consumer are roles played by software agents on behalf of their owners

Distributed operating systems

The computers in most distributed systems are loosely coupled Thus, a distributed system inherently has multiple system images This is mainly due to the fact that all node machines run with an independent operating system To promote resource sharing and fast communication among Node machines, it is best to have a distributed OS that manages all resources coherently and efficiently. Such a system is most likely to be a closed system, and it will likely rely on message passing and RPCs for inter node communications

MOSIX2 for Linux Clusters

Runs with a virtualization layer in the Linux environment This layer provides a partial single-system image to user applications This is mainly due to the fact that all node machines run with an independent operating system Supports both sequential and parallel applications, and discovers resources and migrates software processes among Linux nodes Can manage a Linux cluster or a grid of multiple clusters

PARALLEL AND DISTRIBUTED PROGRAMMING MODELS

Message-Passing Interface (MPI) This is the primary programming standard used to develop parallel and concurrent programs to run on a distributed system MPI is essentially a library of subprograms that can be called from C or FORTRAN to write parallel programs running on a distributed system The idea is to embody clusters, grid systems, and P2P systems with upgraded web services and utility computing applications

MapReduce This is a web programming model developed by Google for scalable data processing on large clusters over large data sets The model is applied mainly in web-scale search and cloud computing Applications. The master node specifies a Map function to divide the input into Sub problems Applies a Reduce function to merge all intermediate values with the same intermediate key Highly scalable to explore high degrees of parallelism at different job levels A typical MapReduce computation process can handle terabytes of data on tens of thousands or more client machines Thousands of MapReduce jobs are executed on Google's clusters every day

Hadoop Library offers a software platform that was originally developed by a Yahoo! Group. The package enables users to write and run applications over vast amounts of distributed data Users can easily scale Hadoop to store and process petabytes of data in the web space.

Economical: Comes with an open source version of MapReduce that minimizes overhead in task spawning and massive data communication
Efficient: Processes data with a high degree of parallelism across a large number of commodity nodes
Reliable: Automatically keeps multiple data copies to facilitate redeployment of computing tasks upon unexpected system failures

Amdahl's Law

Let us suppose a uniprocessor workstation executes a given program in time T minutes

Now the same program is partitioned for parallel execution on a cluster of many nodes

We assume that a fraction α of the code must be executed sequentially.

Therefore, (1 - α) of the code can be compiled for parallel execution by n processors

The total execution time of the program is calculated by:

$\alpha T + (1 - \alpha)T/n$

where, the first term is the sequential execution time on a single processor and the second term is the parallel execution time on n processing nodes

Amdahl's Law: Speedup factor

The speedup factor of using the n-processor system over the use of asingle processor is expressed by:

$\text{Speedup} = S = T/[\alpha T + (1 - \alpha)T/n]$

$= 1/[\alpha + (1 - \alpha)/n]$

The maximum speedup of n is achieved only if the code is fully parallelizable with $\alpha = 0$

As the cluster becomes sufficiently large, that is, $n \to \infty$, S approaches $1/\alpha$, an upper bound on the speedup S

The sequential bottleneck is the portion of the code that cannot be parallelized
ifα = 0:25 -> 1 - α = 0:75, Max. speedup=4

Gustafson's Law

Scaling the problem size to match the cluster capability (scaled-workload speedup)

Let W be the workload in a given program. When using an n-processor system, the user scales the workload to

$$W' = \alpha W + (1 - \alpha) n W$$

The parallel execution time of a scaled workload W' on n processors is defined by scaled-workload speedup

$$S' = W'/W = [\alpha W + (1 - \alpha) n W]/W$$

$$= \alpha + (1 - \alpha)n$$

Thus efficiency is

$$E' = S' / n = \alpha /n + (1 - \alpha)$$

For $\alpha = 0:25$ and $n = 256$, $E = 75\%$

Availability

A system is highly available if it has a long mean time to failure (MTTF) and a short mean time to repair (MTTR)

System Availability = MTTF = (MTTF + MTTR)

Failure may occur in hardware, software or network component. Any failure that will pull down the operation of the entire system is called a single point of failure. A reliable computing system must be designed with no single point of failure.

In general, as a distributed system increases in size, availability decreases due to a higher chance of failure and a difficulty in isolating the failures

ENERGY EFFICIENCY IN DISTRIBUTED COMPUTING

Primary performance goals in conventional parallel and distributed computing systems are high performance and high throughput, considering some form of performance reliability (e.g., fault tolerance and security). However, these systems recently encountered new challenging issues including energy efficiency, and workload and resource outsourcing

Energy Consumption of Unused Servers: To run a server farm (data center) a company has to spend a huge amount of money for hardware, software, operational support, and energy every year. Therefore, companies should thoroughly identify whether their installed server farm (more specifically, the volume of provisioned resources)is at an appropriate level, particularly in terms of utilization.

Reducing Energy in Active Servers: In addition to identifying unused/underutilized servers for energy savings, it is also necessary to apply appropriate techniques to decrease energy consumption in active distributed systems with negligible influence on their performance.

Application Layer: Until now, most user applications in science, business, engineering, and financial areas tend to increase a system's speed or quality. By introducing energy-aware applications, the challenge is to design sophisticated multilevel and multi-domain energy management applications without hurting performance.

Middleware Layer: The middleware layer acts as a bridge between the application layer and the resource layer. This layer provides resource broker, communication service, task analyzer, task

scheduler, security access, reliability control, and information service capabilities. It is also responsible for applying energy-efficient techniques, particularly in task scheduling.

Resource Layer: The resource layer consists of a wide range of resources including computing nodes and storage units. This layer generally interacts with hardware devices and the operating system; therefore, it is responsible for controlling all distributed resources in distributed computing systems. Dynamic power management (DPM) and dynamic voltage-frequency scaling (DVFS) are two popular methods incorporated into recent computer hardware systems. In DPM, hardware devices, such as the CPU, have the capability to switch from idle mode to one or more lower power modes. In DVFS, energy savings are achieved based on the fact that the power consumption in CMOS circuits has a direct relationship with frequency and the square of the voltage supply.

Network Layer: Routing and transferring packets and enabling network services to the resource layer are the main responsibility of the network layer in distributed computing systems. The major challenge to build energy-efficient networks is, again, determining how to measure, predict, and create a balance between energy consumption and performance.

A **computer cluster** is a collection of interconnected stand-alone computers which can work together collectively and cooperatively as a single integrated computing resource pool. Clustering explores massive parallelism at the job level and achieves high availability (HA) through stand-alone operations.
The benefits of computer clusters and massively parallel processors (MPPs) include scalable performance, High Availability, fault tolerance, modular growth, and use of commodity components

Design Objectives of Computer Clusters

Clusters are classified using six orthogonal attributes: scalability, packaging, control, homogeneity, programmability, and security

1. **Scalability:** Clustering of computers is based on the concept of modular growth The scalability could be limited by a number of factors, such as the multicore chip technology, cluster topology, packaging method, power consumption, and cooling scheme applied. The purpose is to achieve scalable performance constrained by the aforementioned factors

2. **Packaging:** Cluster nodes can be packaged in a **compact** or a **slack** fashion.
   In a compact cluster, the nodes are closely packaged in one or more racks sitting in a room, and the nodes are not attached to peripherals (monitors, keyboards, mice, etc.)
   In a slack cluster, the nodes are attached to their usual peripherals and they may be located in different rooms, different buildings, or even remote regions. Packaging directly affects communication wire length, and thus the selection of interconnection technology used. While a compact cluster can utilize a high-bandwidth, low-latency communication network that is often proprietary, nodes of a slack cluster are normally connected through standard LANs or WANs

3. **Control:** A cluster can be either controlled or managed in a centralized or decentralized fashion. A compact cluster normally has centralized control, while a slack cluster can be controlled either way. In a centralized cluster, all the nodes are owned, controlled, managed, and administered by a central operator. In a decentralized cluster, the nodes have individual owners This lack of a single point of control makes system administration

of such a cluster very difficult. It also calls for special techniques for process scheduling, workload migration, checkpointing, accounting, and other similar tasks.

4. **Homogeneity:** A homogeneous cluster uses nodes from the same platform, that is, the same processor architecture and the same operating system; often, the nodes are from the same vendors. A heterogeneous cluster uses nodes of different platforms. Interoperability is an important issue in heterogeneous clusters. For instance, process migration is often needed for load balancing or availability. In a homogeneous cluster, a binary process image can migrate to another node and continue execution. This is not feasible in a heterogeneous cluster, as the binary code will not be executable when the process migrates to a node of a different platform.

5. **Security:** Intra cluster communication can be either exposed or enclosed. In an exposed cluster, the communication paths among the nodes are exposed to the outside world. An outside machine can access the communication paths, and thus individual nodes, using standard protocols (e.g., TCP/IP). Such exposed clusters are easy to implement, but have several disadvantages:

   • Being exposed, intra cluster communication is not secure, unless the communication subsystem performs additional work to ensure privacy and security.
   • Outside communications may disrupt intra cluster communications in an unpredictable fashion.
   • Standard communication protocols tend to have high overhead. A disadvantage is that there is currently no standard for efficient, enclosed intra cluster communication.

6. **Dedicated versus Enterprise Clusters:** A dedicated cluster is typically installed in a desk side rack in a central computer room. It is homogeneously configured with the same type of computer nodes and managed by a single administrator group like a frontend host. Dedicated clusters are used as substitutes for traditional mainframes or supercomputers. A dedicated cluster is installed, used, and administered as a single machine An enterprise cluster is mainly used to utilize idle resources in the nodes. Each node is usually a full-fledged SMP, workstation, or PC, with all the necessary peripherals attached. The nodes are typically geographically distributed, and are not necessarily in the same room or even in the same building. The nodes are individually owned by multiple owners.

Fundamental Cluster Design Issues

1. **Scalable Performance: S**caling of resources (cluster nodes, memory capacity, I/O bandwidth, etc.) leads to a proportional increase in performance. Both scale-up and scale-down capabilities are needed, depending on application demand or cost-effectiveness considerations. Clustering is driven by scalability

2. **Single-System Image (SSI):** A set of workstations connected by an Ethernet network is not necessarily a cluster. A cluster is a single system.

3. **Availability Support:** Clusters can provide cost-effective HA capability with lots of redundancy in processors, memory, disks, I/O devices, networks, and operating system images

4. **Cluster Job Management:** Clusters try to achieve high system utilization from traditional workstations or PC nodes that are normally not highly utilized. Job management software is required to provide batching, load balancing, parallel processing, and other functionality

5. **Inter node Communication:** The inter node physical wire lengths are longer in a cluster than in an MPP. A long wire implies greater interconnect network latency. But, longer wires have more problems in terms of reliability, clock skew, and cross talking. These problems call for reliable and secure communication protocols, which increase overhead. Clusters often use commodity networks (e.g., Ethernet) with standard protocols such as TCP/IP.

6. **Fault Tolerance and Recovery:** Clusters of machines can be designed to eliminate all single points of failure. Through redundancy, a cluster can tolerate faulty conditions up to a certain extent. Heartbeat mechanisms can be installed to monitor the running condition of all nodes. In case of a node failure, critical jobs running on the failing nodes can be saved by failing over to the surviving node machines. Rollback recovery schemes restore the computing results through periodic check pointing.

7. **Cluster Family Classification:** computer clusters are divided into three classes

   - **Compute clusters** These are clusters designed mainly for collective computation over a single large job. The compute clusters do not handle many I/O operations, such as database services. When a single compute job requires frequent communication among the cluster nodes, the cluster must share a dedicated network, and thus the nodes are mostly homogeneous and tightly coupled. This type of clusters is also known as a **Beowulf cluster**
   - **High-Availability clusters** HA (high-availability) clusters are designed to be fault-tolerant and achieve HA of services. HA clusters operate with many redundant nodes to sustain faults or failures.
   - **Load-balancing clusters** These clusters shoot for higher resource utilization through load balancing among all participating nodes in the cluster. All nodes share the workload or function as a single virtual machine (VM). Requests initiated from the user are distributed to all node computers to form a cluster. This results in a balanced workload among different machines, and thus higher resource utilization or higher performance. Middleware is needed to achieve dynamic load balancing by job or process migration among all the cluster nodes.

A Basic Cluster Architecture

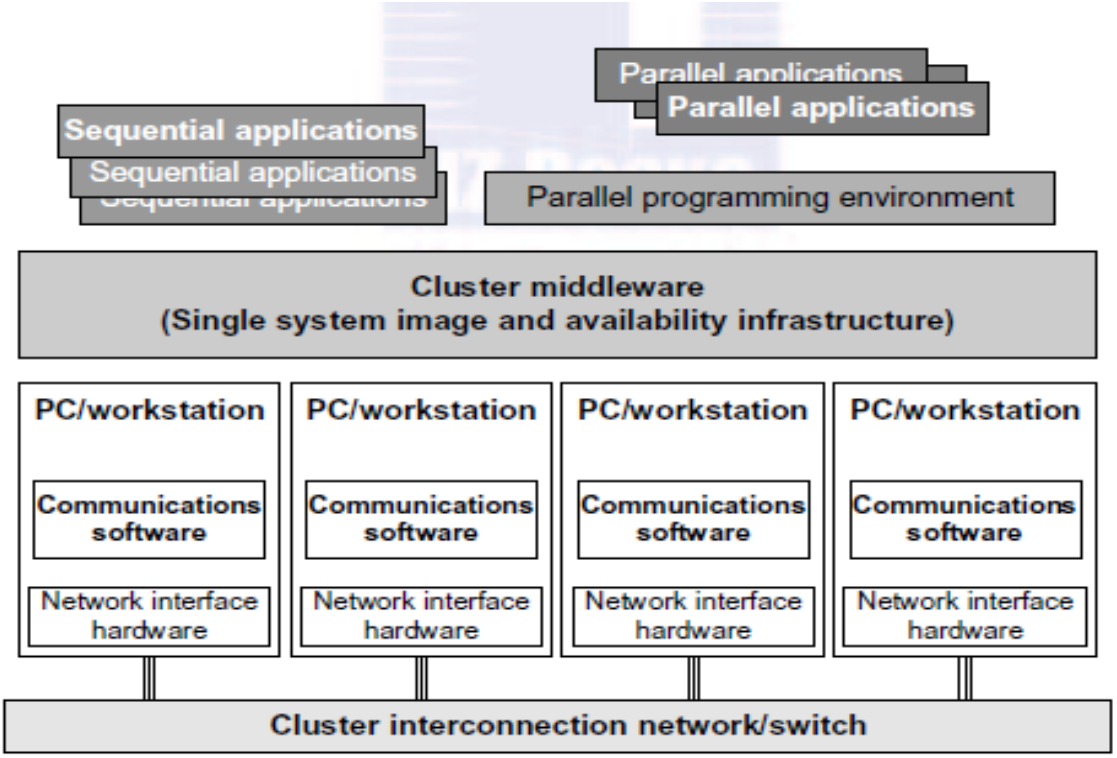
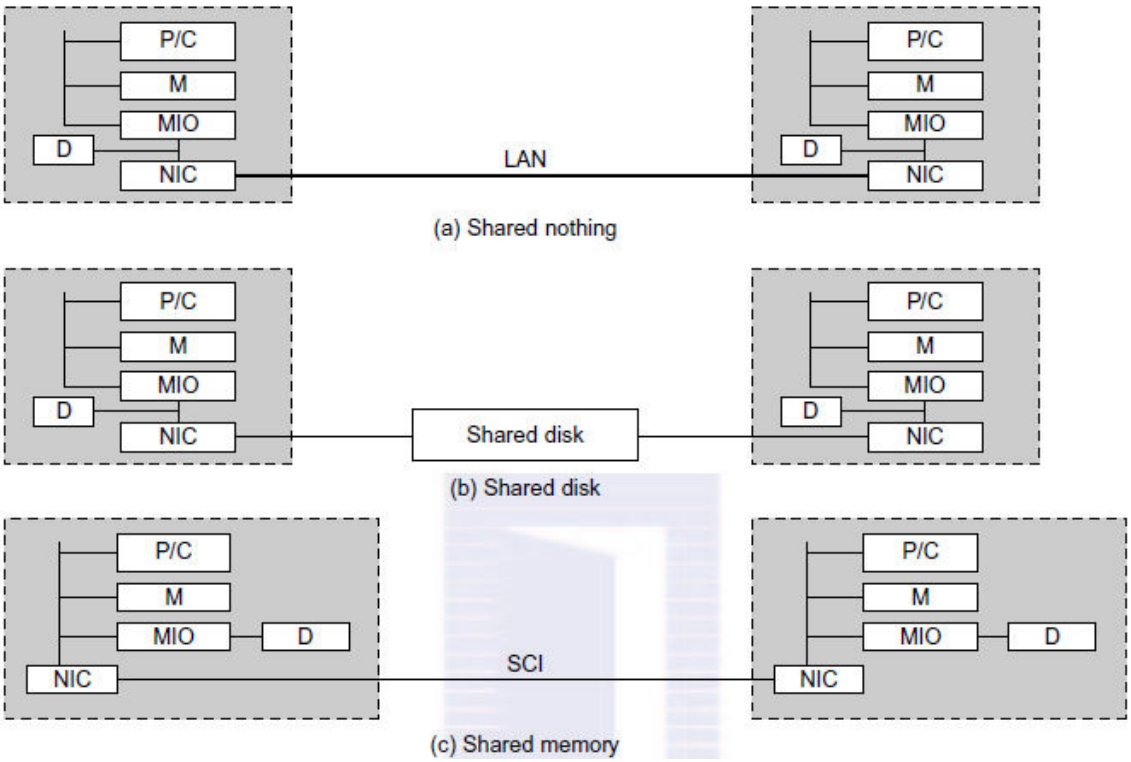


Figure shows simple cluster of computers built with commodity components and fully supported with desired SSI features and HA capability. The processing nodes are commodity workstations, PCs, or servers. The node operating systems should be designed for multiuser, multitasking, and multithreaded applications. The nodes are interconnected by one or more fast commodity networks. These networks use standard communication protocols and operate at a speed that should be two orders of magnitude faster than that of the current TCP/IP speed over Ethernet.
The network interface card is connected to the node's standard I/O bus (e.g., PCI). When the processor or the operating system is changed, only the driver software needs to change

cluster middleware combines together all node platforms at the user space. An availability middleware offers HA services. An SSI layer provides a single entry point, a single file hierarchy, a single point of control, and a single job management system. In addition to running sequential user programs, the cluster supports parallel programming based on standard languages and communication libraries using PVM, MPI, or OpenMP. The programming environment also includes tools for debugging, profiling, monitoring, and so forth. A user interface subsystem is needed to combine the advantages of the web interface and the Windows GUI. It should also provide user-friendly links to various programming environments, job management tools, hypertext, and search support so that users can easily get help in programming the computer cluster.

Resource Sharing in Clusters

Clustering improves both availability and performance
The nodes of a cluster can be connected in one of three ways, as shown in Figure.



The shared-nothing architecture in Part(a) is used in most clusters, where the nodes are connected through the I/O bus. This architecture simply connects two or more autonomous computers via a LAN such as Ethernet

A shared-disk cluster is shown in Part (b) is in favor of small-scale availability clusters in business applications. When one node fails, the other node takes over.. This is what most business clusters desire so that they can enable recovery support in case of node failure. The shared disk can hold checkpoint files or critical system images to enhance cluster availability. Without shared disks, check pointing, rollback recovery, failover, and failback are not possible in a cluster.

The shared-memory cluster in Part (c) is much more difficult to realize. The nodes could be connected by a scalable coherence interface (SCI) ring, which is connected to the memory bus of each node through an NIC module. In the other two architectures, the interconnect is attached to the I/O bus. The memory bus operates at a higher frequency than the I/O bus.

DESIGN PRINCIPLES OF COMPUTER CLUSTERS

General-purpose computers and clusters of cooperative computers should be designed for scalability, availability, Single System Image, High Availability, Fault tolerance, and Rollback recovery

1. **Single System Image**: A single system image is the illusion, created by software or hardware, that presents a collection of resources as an integrated powerful resource. SSI makes the cluster appear like a single machine to the user, applications, and network. A cluster with multiple system images is nothing but a collection of independent computers (Distributed systems in general)

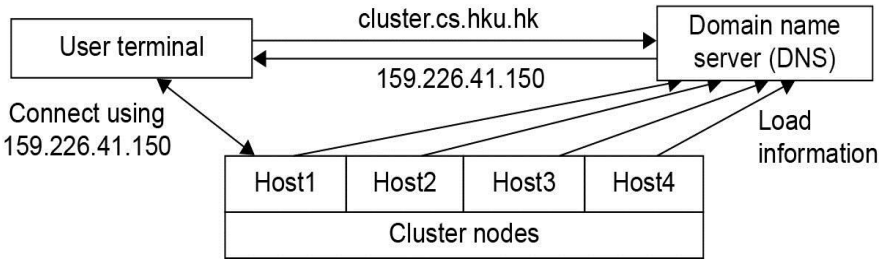
Single-System-Image Features

- **Single System**: The entire cluster is viewed by the users as one system, which has multiple processors.
- **Single Control**: Logically, an end user or system user utilizes services from one place with a single interface.
- **Symmetry**: A user can use a cluster service from any node. All cluster services and functionalities are symmetric to all nodes and all users, except those protected by access rights.
- **Location Transparent**: The user is not aware of the whereabouts of the physical device that eventually provides a service.

Basic SSI Services

A. Single Entry Point

telnet cluster.usc.edu
telnet node1.cluster.usc.edu



1. Four nodes of a cluster are used as host nodes to receive users' login requests.
2. To log into the cluster a standard Unix command such as "telnet cluster.cs.hku.hk", using the symbolic name of the cluster system is issued.
3. The symbolic name is translated by the DNS, which returns with the IP address 159.226.41.150 of the least-loaded node, which happens to be node Host1.
4. The user then logs in using this IP address.
5. The DNS periodically receives load information from the host nodes to make load-balancing translation decisions.

B. **Single File Hierarchy**: xFS, AFS, Solaris MC Proxy

The illusion of a single, huge file system image that transparently integrates local and global disks and other file devices (e.g., tapes). Files can reside on 3 types of locations in a cluster:

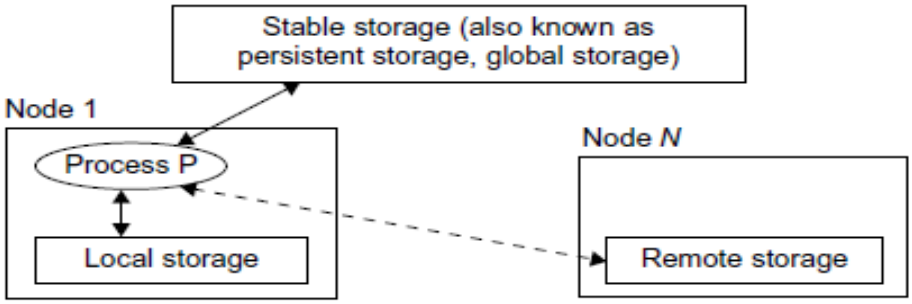
Local storage - disk on the local node.

Remote storage - disks on remote nodes.

Stable storage -

Persistent - data, once written to the stable storage, will stay there at least for a period of time (e.g., a week), even after the cluster shuts down.

Fault tolerant - to some degree, by using redundancy and periodical backup to tapes.



Three types of storage in a single file hierarchy. Solid lines show what process P can access and the dashed line shows what P may be able to access

C. **Single I/O, Networking, and Memory Space:** To achieve SSI, we need a:

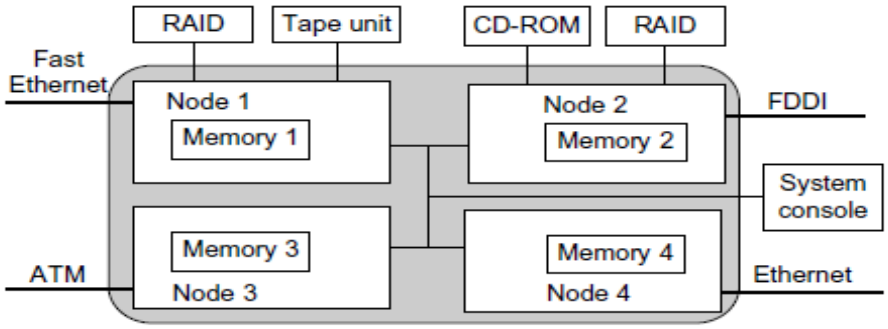
- single control point
- single address space
- single job management system
- single user interface
- single process control

Single Networking: A properly designed cluster should behave as one system. Any process on any node can use any network and I/O device as though it were attached to the local node. Single networking means any node can access any network connection.

Single Point of Control: The system administrator should be able to configure, monitor, test, and control the entire cluster and each individual node from a single point. Many clusters help with this through a system console that is connected to all nodes of the cluster

Single Memory Space: Single memory space gives users the illusion of a big, centralized main memory, which in reality may be a set of distributed local memory spaces.

Single I/O Address Space: A single I/O space implies that any node can access the RAIDs



A cluster with single networking, single I/O space, single memory, and single point of control

Other Services

Single Job Management: All cluster jobs can be submitted from any node to a single job management system. GlUnix, Codine, LSF, etc.

Single User Interface: The users use the cluster through a single graphical interface. Such an interface is available for workstations and PCs like CDE in Solaris/NT

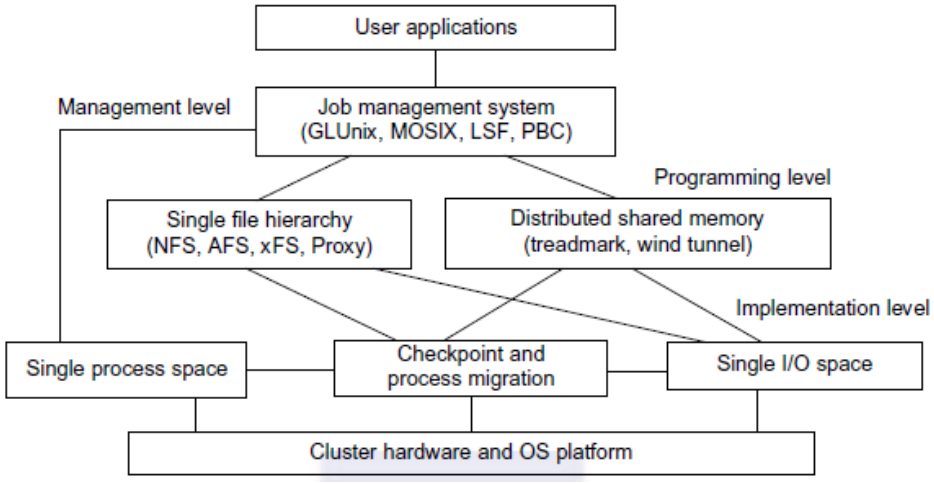
Single process space All user processes created on various nodes form a single process space and share a uniform process identification scheme. A process on any node can create (e.g., through a UNIX fork) or communicate with (e.g., through signals, pipes, etc.) processes on remote nodes.

Middleware support for SSI clustering SSI features are supported by middleware developed at three cluster application levels:

• **Management level** This level handles user applications and provides a job management system such as GLUnix, MOSIX, Load Sharing Facility (LSF), or Codine.

• **Programming level** This level provides single file hierarchy (NFS, xFS, AFS, Proxy) and distributed shared memory (TreadMark, Wind Tunnel).

• **Implementation level** This level supports a single process space, checkpointing, process migration, and a single I/O space. These features must interface with the cluster hardware and OS platform.



Relationship among clustering middleware at the job management, programming, and implementation levels.

2. High Availability through Redundancy:

- **Reliability** measures how long a system can operate without a breakdown.
- **Availability** indicates the percentage of time that a system is available to the user, that is, the percentage of system uptime.
- **Serviceability** refers to how easy it is to service the system, including hardware and

software maintenance, repair, upgrades, and so on.

A system's reliability is measured by the mean time to failure (MTTF), which is the average time of normal operation before the system (or a component of the system) fails. The metric for serviceability is the mean time to repair (MTTR), which is the average time it takes to repair the system and restore it to working condition after it fails.

The availability of a system is defined by:

$$\text{Availability} = \text{MTTF} / (\text{MTTF} + \text{MTTR})$$

Failure is any event that prevents the system from normal operation

- **Unplanned failures** The system breaks, due to an operating system crash, a hardware failure, a network disconnection, human operation errors, a power outage, and so on. All these are simply called failures. The system must be repaired to correct the failure.
- **Planned shutdowns** The system is not broken, but is periodically taken off normal operation for upgrades, reconfiguration, and maintenance.

Transient versus Permanent Failures

A lot of failures are **transient** in that they occur temporarily and then disappear. They can be dealt with without replacing any components. A standard approach is to roll back the system to a known state and start over.

Permanent failures cannot be corrected by rebooting. Some hardware or software component must be repaired or replaced. For instance, rebooting will not work if the system hard disk is broken.

Partial versus Total Failures

A failure that renders the entire system unusable is called a **total** failure. A failure that only affects part of the system is called a **partial** failure if the system is still usable, even at a reduced capacity

Redundancy Techniques

Isolated Redundancy: A key technique to improve availability in any system is to use redundant components. When a component (the primary component) fails, the service it provided is taken over by another component (the backup component). Furthermore, the primary and the backup components should be isolated from each other, meaning they should not be subject to the same cause of failure. Clusters provide HA with redundancy in power supplies, fans, processors, memories, disks, I/O devices, networks, operating system images, and so on. In a carefully designed cluster, redundancy is also isolated.

N-Version Programming to Enhance Software Reliability

A common isolated-redundancy approach to constructing a mission-critical software system is called N-version programming. The software is implemented by N isolated teams who may not even know the others exist. Different teams are asked to implement the software using different algorithms, programming languages, environment tools, and even platforms In a fault-tolerant

system, the N versions all run simultaneously and their results are constantly compared. If the results differ, the system is notified that a fault has occurred.

3. **Fault-Tolerant Cluster Configurations:** The cluster solution was targeted to provide availability support for two server nodes with three ascending levels of availability: hot standby, active takeover, and fault-tolerant. The level of availability increases from standby to active and fault-tolerant cluster configurations. The shorter is the recovery time, the higher is the cluster availability. Failback refers to the ability of a recovered node returning to normal operation after repair or maintenance. Activeness refers to whether the node is used in active work during normal operation.

- **Hot standby server clusters**: In a hot standby cluster, only the primary node is actively doing all the useful work normally. The standby node is powered on (hot) and running some monitoring programs to communicate heartbeat signals to check the status of the primary node, but is not actively running other useful workloads. The primary node must mirror any data to shared disk storage, which is accessible by the standby node. The standby node requires a second copy of data.
- **Active-takeover clusters**: In this case, the architecture is symmetric among multiple server nodes. Both servers are primary, doing useful work normally. Both failover and failback are often supported on both server nodes. When a node fails, the user applications fail over to the available node in the cluster. Depending on the time required to implement the failover, users may experience some delays or may lose some data that was not saved in the last checkpoint.

- **Failover cluster:** When a component fails, this technique allows the remaining system to take over the services originally provided by the failed component. A failover mechanism must provide several functions, such as failure diagnosis, failure notification, and failure recovery. Failure diagnosis refers to the detection of a failure and the location of the failed component that caused the failure. A commonly used technique is heartbeat, whereby the cluster nodes send out a stream of heartbeat messages to one another. If the system does not receive the stream of heartbeat messages from a node, it can conclude that either the node or the network connection has failed.

Recovery Schemes

Failure recovery refers to the actions needed to take over the workload of a failed component. There are two types of recovery techniques. In **backward recovery**, the processes running on a cluster periodically save a consistent state (called a checkpoint) to a stable storage. After a failure, the system is reconfigured to isolate the failed component, restores the previous checkpoint, and resumes normal operation. This is called rollback. Backward recovery is relatively easy to implement in an application-independent, portable fashion

If execution time is crucial, such as in real-time systems where the rollback time cannot be tolerated, a **forward recovery** scheme should be used. With such a scheme, the system is not rolled back to the previous checkpoint upon a failure. Instead, the system utilizes the failure diagnosis information to reconstruct a valid system state and continues execution. Forward recovery is application-dependent and may need extra hardware

Checkpointing and Recovery Techniques

Checkpointing is the process of periodically saving the state of an executing program to stable storage, from which the system can recover after a failure. Each program state saved is called a checkpoint. The disk file that contains the saved state is called the checkpoint file.
Checkpointing techniques are useful not only for availability, but also for program debugging, process migration, and load balancing

Checkpointing can be realized by the operating system at the **kernel level**, where the OS transparently checkpoints and restarts processes
A less transparent approach links the user code with a checkpointing **library in the user space**. Check pointing and restarting are handled by this runtime support. This approach is used widely because it has the advantage that user applications do not have to be modified.

A third approach requires the **user (or the compiler)** to insert checkpointing functions in the application; thus, the application has to be modified, and the transparency is lost. However, it has the advantage that the user can specify where to checkpoint. This is helpful to reduce checkpointing overhead. Checkpointing incurs both time and storage overheads.

Checkpoint Overheads
During a program's execution, its states may be saved many times. This is denoted by the time consumed to save one checkpoint. The storage overhead is the extra memory and disk space required for checkpointing. Both time and storage overheads depend on the size of the checkpoint file.

Choosing an Optimal Checkpoint Interval

The time period between two checkpoints is called the checkpoint interval. Making the interval larger can reduce checkpoint time overhead.
Wong and Franklin derived an expression for optimal checkpoint interval

$$\text{Optimal checkpoint interval} = \text{Square root } (\text{MTTF} \times t_c)/h$$

MTTF is the system's mean time to failure. This MTTF accounts the time consumed to save one checkpoint, and h is the average percentage of normal computation performed in a checkpoint interval before the system fails. The parameter h is always in the range. After a system is restored, it needs to spend h × (checkpoint interval) time to recompute.

Incremental Checkpoint
Instead of saving the full state at each checkpoint, an incremental checkpoint scheme saves only the portion of the state that is changed from the previous checkpoint In full-state checkpointing, only one checkpoint file needs to be kept on disk. Subsequent checkpoints simply overwrite this file. With incremental checkpointing, old files needed to be kept, because a state may span many files. Thus, the total storage requirement is larger

Forked Checkpointing
Most checkpoint schemes are blocking in that the normal computation is stopped while checkpointing is in progress. With enough memory, checkpoint overhead can be reduced by making a copy of the program state in memory and invoking another asynchronous thread to perform the checkpointing concurrently. A simple way to overlap checkpointing with computation is to use the UNIX fork( ) system call. The forked child process duplicates the

parent process's address space and checkpoints it. Meanwhile, the parent process continues execution. Overlapping is achieved since checkpointing is disk-I/O intensive

User-Directed Checkpointing
The checkpoint overheads can sometimes be substantially reduced if the user inserts code (e.g., library or system calls) to tell the system when to save, what to save, and what not to save. What should be the exact contents of a checkpoint? It should contain just enough information to allow a system to recover. The state of a process includes its data state and control state

Checkpointing Parallel Programs The state of a parallel program is usually much larger than that of a sequential program, as it consists of the set of the states of individual processes, plus the state of the communication network. Parallelism also introduces various timing and consistency problems

Consistent Snapshot
A global snapshot is called consistent if there is no message that is received by the checkpoint of one process, but not yet sent by another process. Graphically, this corresponds to the case that no arrow crosses a snapshot line from right to left

Coordinated versus Independent Checkpointing
Checkpointing schemes for parallel programs can be classified into two types. In coordinated checkpointing (also called consistent checkpointing), the parallel program is frozen, and all processes are checkpointed at the same time. In independent checkpointing, the processes are checkpointed independent of one another.

Cluster Job Scheduling and Management

A Job Management System (*JMS*) should have three parts:

- A **user server** lets the user submit jobs to one or more queues, specify resource requirements for each job, delete a job from a queue, inquire about the status of a job or a queue.
- A **job scheduler** that performs job scheduling and queuing according to job types, resource requirements, resource availability, and scheduling policies.
- A **resource manager** that allocates and monitors resources, enforces scheduling policies, and collects accounting information**.**

JMS Administration

- JMS should be able to dynamically reconfigure the cluster with minimal impact on the running jobs.
- The administrator's prologue and epilogue scripts should be able to run before and after each job for security checking, accounting, and cleanup.
- Users should be able to cleanly kill their own jobs.
- The administrator or the JMS should be able to cleanly suspend or kill any job.
    - Clean means that when a job is suspended or killed, all its processes must be included.
    - Otherwise some "orphan" processes are left in the system, wasting cluster resources and may eventually render the system unusable**.**

Several types of jobs execute on a cluster.

- Serial jobs run on a single node.
- Parallel jobs use multiple nodes.
- Interactive jobs are those that require fast turnaround time, and their input/output is directed to a terminal.
  - These jobs do not need large resources, and the users expect them to execute immediately, not made to wait in a queue.
- Batch jobs normally need more resources, such as large memory space and long CPU time.
  - But they do not need immediate response.
  - They are submitted to a job queue to be scheduled to run when the resource becomes available (e.g., during off hours).

Multi-Job Scheduling Schemes

- Cluster jobs may be scheduled to run at a specific time (**calendar scheduling**) or when a particular event happens (**event scheduling**).
- Jobs are scheduled according to priorities based on submission time, resource nodes, execution time, memory, disk, job type, and user identity.
- With **static priority**, jobs are assigned priorities according to a predetermined, fixed scheme.
  - A simple scheme is to schedule jobs in a first-come, first-serve fashion.
  - Another scheme is to assign different priorities to users.

  With **dynamic priority**, the priority of a job may change over time.

Job Scheduling Issues and Schemes for Cluster Nodes

Issue	Scheme	Key Problems
Job priority	**Non-preemptive**	Delay of high-priority jobs
	Preemptive	Overhead, implementation
Resource required	**Static**	Load imbalance
	Dynamic	Overhead, implementation
Resource sharing	**Dedicated**	Poor utilization
	Space sharing	Tiling, large job
Scheduling	**Time sharing**	Process-based job control with context switch overhead
	Independent	Severe slowdown
	Gang scheduling	Implementation difficulty
Competing with foreign (local) jobs	**Stay**	Local job slowdown
	Migrate	Migration threshold, Migration overhead

Scheduling Modes

Dedicated Mode:

- Only one job runs in the cluster at a time, and at most one process of the job is assigned to a node at a time.
- The single job runs until completion before it releases the cluster to run other jobs.

Space Sharing:
Multiple jobs can run on disjoint partitions (groups) of nodes simultaneously.
- At most one process is assigned to a node at a time.
- Although a partition of nodes is dedicated to a job, the interconnect and the I/O subsystem may be shared by all jobs.

Time sharing :
- Multiple user processes are assigned to the same node.
  Time-sharing introduces the following parallel scheduling policies:

    - **Independent Scheduling (Independent):** Uses the operating system of each cluster node to schedule different processes as in a traditional workstation.
    - **Gang Scheduling**: Schedules all processes of a parallel job together. When one process is active, all processes are active.
    - **Competition with Foreign (Local) Jobs**: Scheduling becomes more complicated when both cluster jobs and local jobs are running. The local jobs should have priority over cluster jobs.

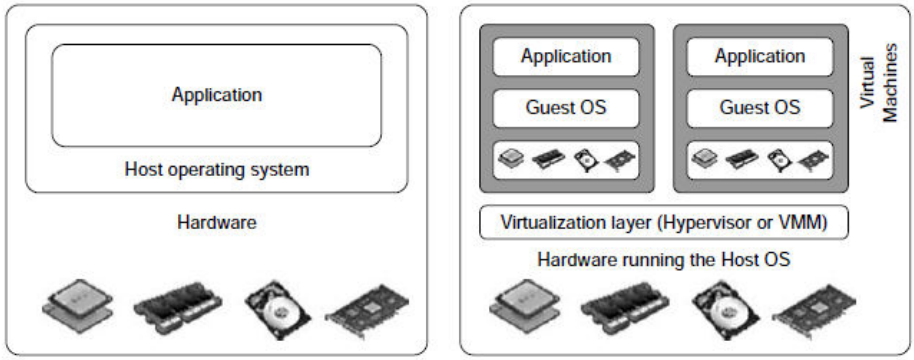
1. **Migration Scheme Issues Node Availability**: Can the job find another available node to migrate to?

    - Berkeley study : Even during peak hours, 60% of workstations in a cluster are available.

2. **Migration Overhead**: What is the effect of the migration overhead? The migration time can significantly slow down a parallel job.
    - Berkeley study : a slowdown as great as 2.4 times.
    - Slowdown is less if a parallel job is run on a cluster of twice the size.
    - e.g. a 32-node job on a 60-node cluster – migration slowdown no more than 20%, even when migration time of 3 minutes.

3. **Recruitment Threshold**: the amount of time a workstation stays unused before the cluster considers it an idle node. What should be the recruitment threshold?

Virtualization is a computer architecture technology by which multiple virtual machines (VMs) are multiplexed in the same hardware machine. The purpose of a VM is to enhance resource sharing by many users and improve computer performance in terms of resource utilization and application flexibility. Hardware resources (CPU, memory, I/O devices, etc.) or software resources (operating system and software libraries) can be virtualized in various functional layers



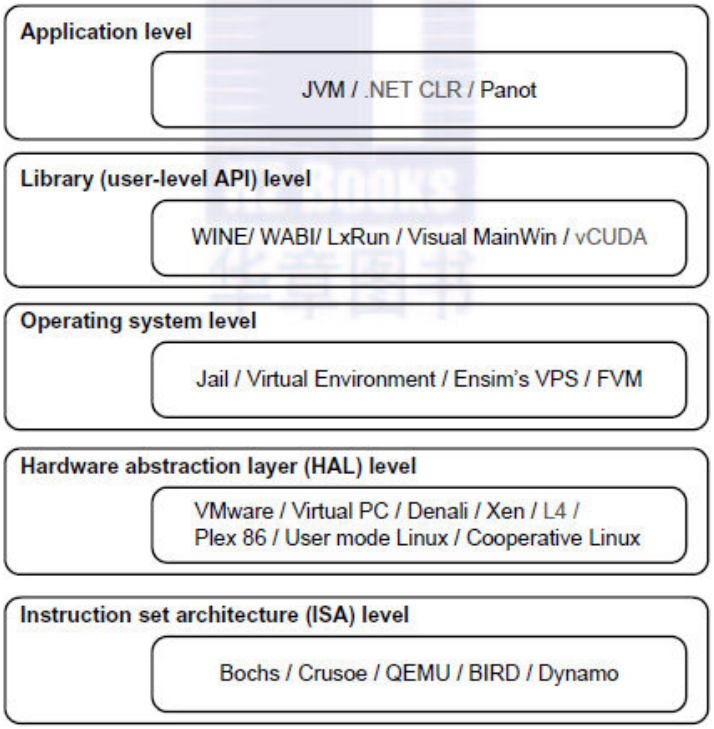
A traditional computer runs with a host operating system specially tailored for its hardware architecture, as shown in Figure(a). After virtualization, different user applications managed by their own operating systems (guest OS) can run on the same hardware, independent of the host OS. This is often done by adding additional software, called a virtualization layer as shown in Figure(b). This virtualization layer is known as hypervisor or virtual machine monitor (VMM). The VMs are in the upper boxes, where applications run with their own guest OS over the virtualized CPU, memory, and I/O resources. The main function of the software layer for virtualization is to virtualize the physical hardware of a host machine into virtual resources to be used by the VMs

Levels of Virtualization Implementation

The virtualization software creates the abstraction of VMs by interposing a virtualization layer at various levels of a computer system.

Common virtualization layers include

- Instruction set architecture (ISA) level
- Hardware level
- Operating system level
- Library support level
- Application level



Instruction Set Architecture Level: Virtualization is performed by emulating a given ISA by the ISA of the host machine.

e.g, MIPS binary code can run on an x-86-based host machine with the help of ISA emulation. Typical systems: Bochs, Crusoe, Quemu, BIRD, Dynamo

Advantage:

- It can run a large amount of legacy binary codes written for various processors on any given new hardware host machines
- best application flexibility

Limitation:

- One source instruction may require tens or hundreds of native target instructions to perform its function, which is relatively slow.
- V-ISA requires adding a processor-specific software translation layer in the complier.

Virtualization at Hardware Abstraction level: Virtualization is performed right on top of the hardware.

- It generates virtual hardware environments for VMs, and manages the underlying hardware through virtualization.
- Typical systems: VMware, Virtual PC, Denali, Xen

Advantage:

- Has higher performance and good application isolation

Limitation:

- Very expensive to implement (complexity)

Virtualization at Operating System (OS) level: It is an abstraction layer between traditional OS and user applications.

- This virtualization creates isolated containers on a single physical server and the OS-instance to utilize the hardware and software in datacenters.
- Typical systems: Jail / Virtual Environment / Ensim's VPS / FVM

Advantage:

- Has minimal starup/shutdown cost, low resource requirement, and high scalability; synchronize VM and host state changes.

Limitation:

- All VMs at the operating system level must have the same kind of guest OS
- Poor application flexibility and isolation.

Library Support level: It creates execution environments for running alien programs on a platform rather than creating VM to run the entire operating system.

- It is done by API call interception and remapping.
- Typical systems: Wine, WAB, LxRun , VisualMainWin

Advantage:

- It has very low implementation effort

Limitation:

- Poor application flexibility and isolation

User-Application level: It virtualizes an application as a virtual machine.

- This layer sits as an application program on top of an operating system and exports an abstraction of a VM that can run programs written and compiled to a particular abstract machine definition.
- Typical systems: JVM , NET CLI , Panot

Advantage:

- has the best application isolation

Limitation:

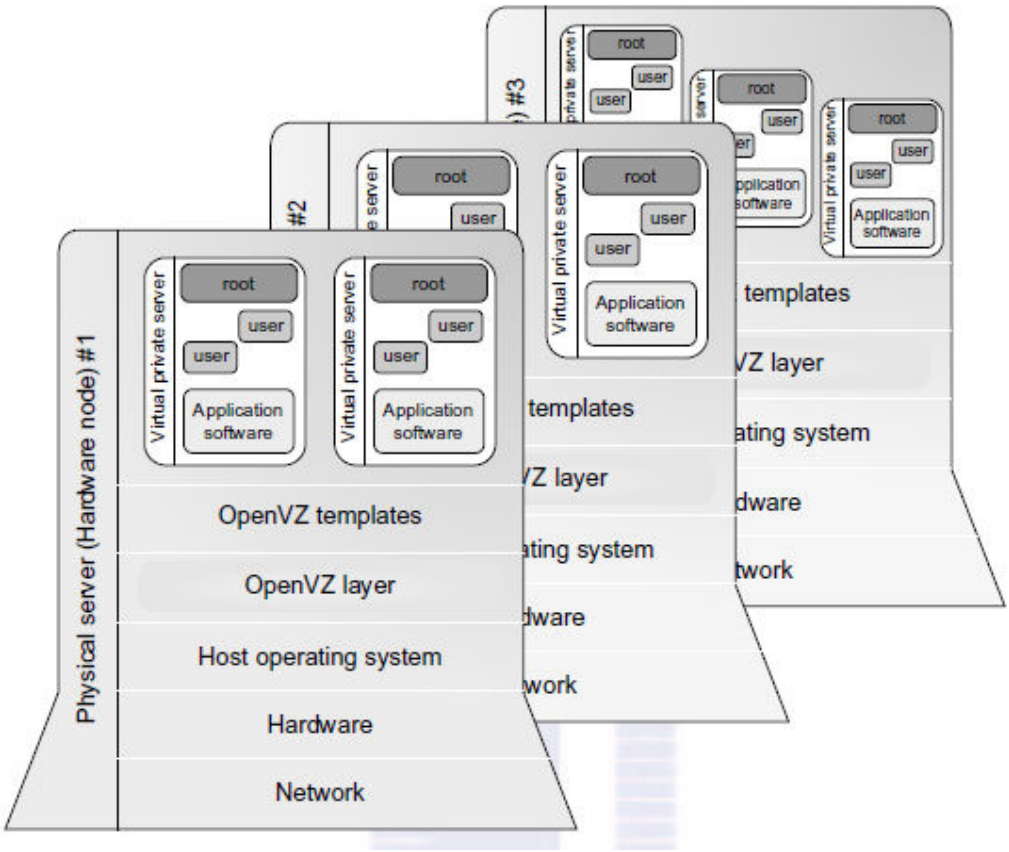
- Low performance, low application flexibility and high implementation complexity.

Table 3.1 Relative Merits of Virtualization at Various Levels (More "X"'s Means Higher Merit, with a Maximum of 5 X's)

Level of Implementation	Higher Performance	Application Flexibility	Implementation Complexity	Application Isolation
ISA	X	XXXXX	XXX	XXX
Hardware-level virtualization	XXXXX	XXX	XXXXX	XXXX
OS-level virtualization	XXXXX	XX	XXX	XX
Runtime library support	XXX	XX	XX	XX
User application level	XX	XX	XXXXX	XXXXX

OS-Level Virtualization

Operating system virtualization inserts a virtualization layer inside an operating system to partition a machine's physical resources. It enables multiple isolated VMs within a single operating system kernel. This kind of VM is often called a virtual execution environment (VE), Virtual Private System (VPS), or simply container. From the user's point of view, VEs look like real servers. This means a VE has its own set of processes, file system, user accounts, network interfaces with IP addresses, routing tables, firewall rules, and other personal settings. Although VEs can be customized for different people, they share the same operating system kernel. Therefore, OS-level virtualization is also called single-OS image virtualization



Operating system virtualization from the point of view of a machine stack

Advantages of OS Extension for Virtualization

1. VMs at OS level has minimum startup/shutdown costs
2. OS-level VM can easily synchronize with its environment

Disadvantage of OS Extension for Virtualization

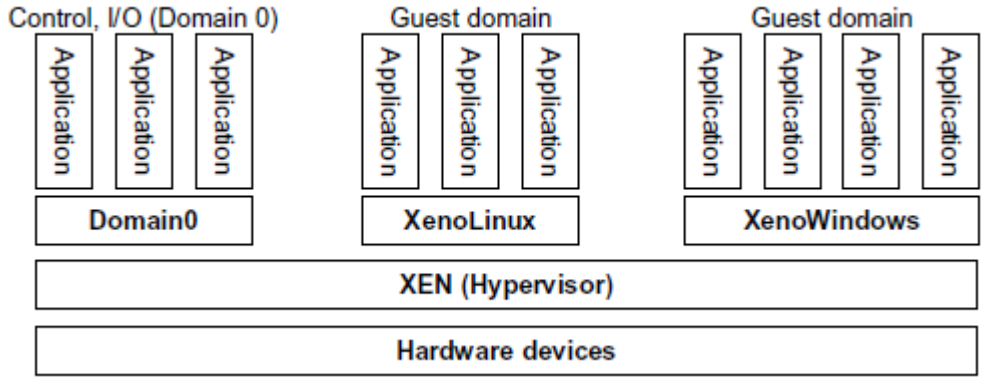
All VMs in the same OS container must have the same or similar guest OS, which restrict application flexibility of different VMs on the same physical machine.

Hypervisor and Xen Architecture

The hypervisor supports hardware-level virtualization (see Figure 3.1(b)) on bare metal devices like CPU, memory, disk and network interfaces. The hypervisor software sits directly between the physical hardware and its OS. This virtualization layer is referred to as either the VMM or the hypervisor. The hypervisor provides hypercalls for the guest OSes and applications. Depending on the functionality, a hypervisor can assume a micro-kernel architecture like the Microsoft Hyper-V. Or it can assume a monolithic hypervisor architecture like the VMware ESX for server virtualization

The Xen Architecture

Xen is an open source hypervisor program developed by Cambridge University. Xen is a microkernel hypervisor, which separates the policy from the mechanism. The Xen hypervisor implements all the mechanisms, leaving the policy to be handled by Domain 0. Xen does not include any device drivers natively. It just provides a mechanism by which a guest OS can have direct access to the physical devices. Xen provides a virtual environment located between the hardware and the OS.



The Xen architecture's special domain 0 for control and I/O, and several guest domains for user applications

The core components of a Xen system are the hypervisor, kernel, and applications The guest OS, which has control ability, is called Domain 0, and the others are called Domain U. Domain 0 is a privileged guest OS of Xen. It is first loaded when Xen boots without any file system drivers being available. Domain 0 is designed to access hardware directly and manage devices.

Therefore, one of the responsibilities of Domain 0 is to allocate and map hardware resources for the guest domains (the Domain U domains).

Binary Translation with Full Virtualization

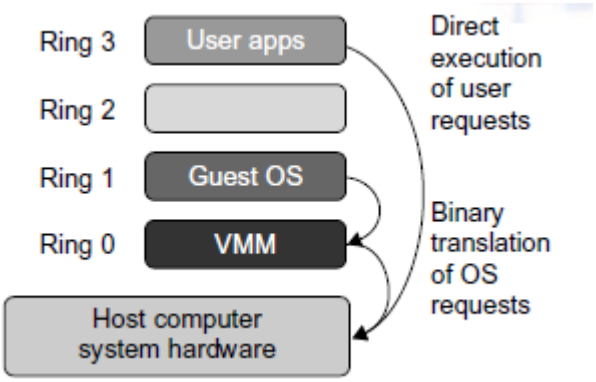
Depending on implementation technologies, hardware virtualization can be classified into two categories: full virtualization and host-based virtualization.

Full virtualization does not need to modify the host OS. It relies on binary translation to trap and to virtualize the execution of certain sensitive, non virtualizable instructions. The guest OS es and their applications consist of noncritical and critical instructions. In a host-based system, both a host OS and a guest OS are used. A virtualization software layer is built between the host OS and guest OS

Full Virtualization

With full virtualization, noncritical instructions run on the hardware directly while critical instructions are discovered and replaced with traps into the VMM to be emulated by software. Both the hypervisor and VMM approaches are considered full virtualization. Why are only critical instructions trapped into the VMM? This is because binary translation can incur a large performance overhead. Noncritical instructions do not control hardware or threaten the security of the system, but critical instructions do. Therefore, running noncritical instructions on hardware not only can promote efficiency, but also can ensure system security.

Binary Translation of Guest OS Requests Using a VMM



This approach was implemented by VMware and many other software companies. VMware puts the VMM at Ring 0 and the guest OS at Ring 1. The VMM scans the instruction stream and identifies the privileged, control- and behavior-sensitive instructions. When these instructions are identified, they are trapped into the VMM, which emulates the behavior of these instructions. The method used in this emulation is called binary translation. Therefore, full virtualization combines binary translation and direct execution. The guest OS is completely decoupled from the underlying hardware. Consequently, the guest OS is unaware that it is being virtualized

Host-Based Virtualization

An alternative VM architecture is to install a virtualization layer on top of the host OS. This host OS is still responsible for managing the hardware. The guest OSes are installed and run on top of

the virtualization layer. Dedicated applications may run on the VMs. Certainly, some other applications can also run with the host OS directly.

This host based architecture has some distinct advantages. First, the user can install this VM architecture without modifying the host OS. The virtualizing software can rely on the host OS to provide device drivers and other low-level services. This will simplify the VM design and ease its deployment. Second, the host-based approach appeals to many host machine configurations.

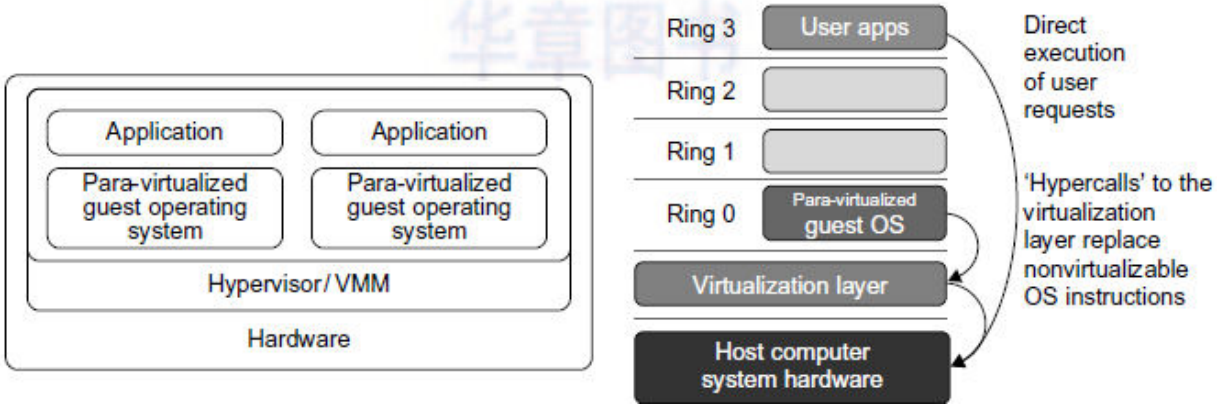
Compared to the hypervisor/VMM architecture, the performance of the host-based architecture may also be low. When an application requests hardware access, it involves four layers of mapping which downgrades performance significantly. When the ISA of a guest OS is different from the ISA of the underlying hardware, binary translation must be adopted. Although the host-based architecture has flexibility, the performance is too low to be useful in practice.

Para-Virtualization with Compiler Support

Para-virtualization needs to modify the guest operating systems. A para-virtualized VM provides special APIs requiring substantial OS modifications in user applications. Para-virtualization attempts to reduce the virtualization overhead, and thus improve performance by modifying only the guest OS kernel.

The guest operating systems are para-virtualized. They are assisted by an intelligent compiler to replace the non virtualizable OS instructions by hypercalls The lower the ring number, the higher the privilege of instruction being executed. The OS is responsible for managing the hardware and the privileged instructions to execute at Ring 0, while user-level applications run at Ring 3.

The best example of para-virtualization is the **KVM**



Para-virtualized VM architecture

VIRTUALIZATION OF CPU, MEMORY, AND I/O DEVICES

To support virtualization, processors such as the x86 employ a special running mode and instructions, known as hardware-assisted virtualization. In this way, the VMM and guest OS run in different modes and all sensitive instructions of the guest OS and its applications are trapped in the VMM. To save processor states, mode switching is completed by hardware.

Modern operating systems and processors permit multiple processes to run simultaneously. If there is no protection mechanism in a processor, all instructions from different processes will access the hardware directly and cause a system crash. Therefore, all processors have at least two modes, user mode and supervisor mode, to ensure controlled access of critical hardware.

Instructions running in supervisor mode are called privileged instructions. Other instructions are unprivileged instructions. In a virtualized environment, it is more difficult to make OSes and applications run correctly because there are more layers in the machine stack. The VMware Workstation is a VM software suite for x86 and x86-64 computers. This software suite allows users to set up multiple x86 and x86-64 virtual computers and to use one or more of these VMs simultaneously with the host operating system. The VMware Workstation assumes the host-based virtualization. Xen is a hypervisor for use in IA-32, x86-64, Itanium, and PowerPC 970 hosts.

CPU Virtualization

A VM is a duplicate of an existing computer system in which a majority of the VM instructions are executed on the host processor in native mode. Thus, unprivileged instructions of VMs run directly on the host machine for higher efficiency. Other critical instructions should be handled carefully for correctness and stability. The critical instructions are divided into three categories: privileged instructions, control sensitive instructions, and behavior-sensitive instructions. Privileged instructions execute in a privileged mode and will be trapped if executed outside this mode. Control-sensitive instructions attempt to change the configuration of resources used. Behavior-sensitive instructions have different behaviors depending on the configuration of resources, including the load and store operations over the virtual memory. A CPU architecture is virtualizable if it supports the ability to run the VM's privileged and unprivileged instructions in the CPU's user mode while the VMM runs in supervisor mode.

Memory Virtualization

Virtual memory virtualization is similar to the virtual memory support provided by modern operating systems. In a traditional execution environment, the operating system maintains mappings of virtual memory to machine memory using page tables, which is a one-stage mapping from virtual memory to machine memory. All modern x86 CPUs include a memory management unit (MMU) and a translation lookaside buffer (TLB) to optimize virtual memory performance. However, in a virtual execution environment, virtual memory virtualization involves sharing the physical system memory in RAM and dynamically allocating it to the physical memory of the VMs. a two-stage mapping process should be maintained by the guest OS and the VMM, respectively: virtual memory to physical memory and physical memory to machine memory. The guest OS continues to control the mapping of virtual addresses to the physical memory addresses of VMs. But the guest OS cannot directly access the actual machine memory.

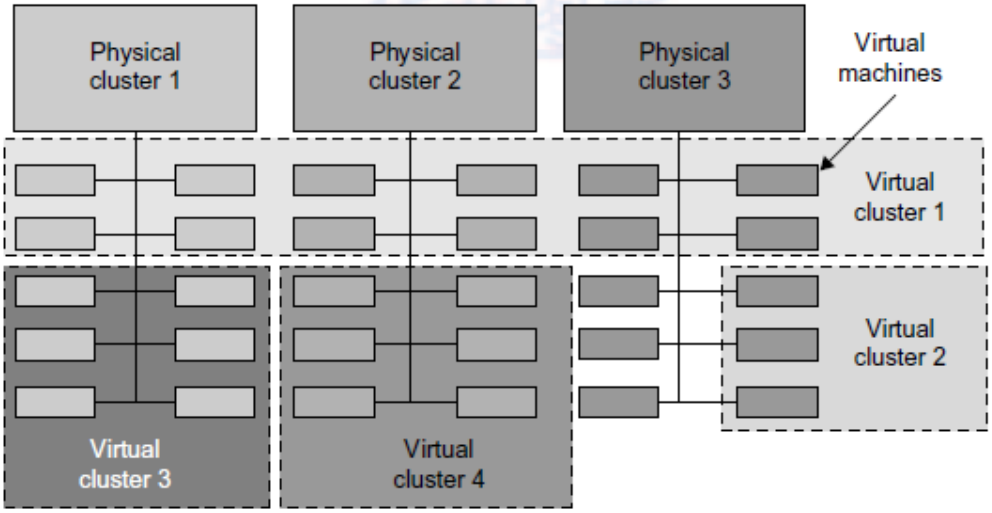
The VMM is responsible for mapping the guest physical memory to the actual machine memory

I/O Virtualization

I/O virtualization involves managing the routing of I/O requests between virtual devices and the shared physical hardwareAll the functions of a device or bus infrastructure, such as device enumeration, identification, interrupts, and DMA, are replicated in software. This software is located in the VMM and acts as a virtual device. The I/O access requests of the guest OS are trapped in the VMM which interacts with the I/O devicesA single hardware device can be shared by multiple VMs that run concurrently

VIRTUAL CLUSTERS AND RESOURCE MANAGEMENT

A physical cluster is a collection of servers (physical machines) interconnected by a physical network such as a LAN.
Virtual clusters are built with VMs installed at distributed servers from one or more physical clusters.
The VMs in a virtual cluster are interconnected logically by a virtual network across several physical networks



A cloud platform with four virtual clusters over three physical clusters

Physical versus Virtual Processor cores

Physical cores	**Virtual cores**
The actual physical cores present in the processor	There can be more virtual cores visible to a single OS than there are physical cores.
More burden on the software to write applications which can run directly on the cores.	Design of software becomes easier as the hardware assists the software in dynamic resource utilization
Hardware provides no assistance to the software and is hence simpler	Hardware provides assistance to the software and is hence more complex.
Poor resource management	Better resource management
The lowest level of system software has to be modified.	The lowest level of system software need not be modified

VIRTUALIZATION FOR DATA-CENTER AUTOMATION

Data-center automation means that huge volumes of hardware, software, and database resources in these data centers can be allocated dynamically to millions of Internet users simultaneously, with guaranteed QoS and cost-effectiveness. This automation process is triggered by the growth of virtualization products and cloud computing services. Virtualization is moving towards

enhancing mobility, reducing planned downtime (for maintenance), and increasing the number of virtual clients.
The latest virtualization development highlights high availability (HA), backup services, workload balancing, and further increases in client bases

Server Consolidation in Data Centers

In data centers, a large number of heterogeneous workloads can run on servers at various times. These heterogeneous workloads can be roughly divided into two categories: chatty workloads and non interactive workloads. Chatty workloads may burst at some point and return to a silent state at some other point. A web video service is an example of this, whereby a lot of people use it at night and few people use it during the day. Non interactive workloads do not require people's efforts to make progress after they are submitted. High-performance computing is a typical example of this. At various stages, the requirements for resources of these workloads are dramatically different. However, to guarantee that a workload will always be able to cope with all demand levels, the workload is statically allocated enough resources so that peak demand is satisfied

Therefore, it is common that most servers in data centers are underutilized. A large amount of hardware, space, power, and management cost of these servers is wasted. Server consolidation is an approach to improve the low utility ratio of hardware resources by reducing the number of physical servers. Among several server consolidation techniques such as centralized and physical consolidation, virtualization-based server consolidation is the most powerful. Data centers need to optimize their resource management the use of VMs increases resource management complexity. This causes a challenge in terms of how to improve resource utilization as well as guarantee QoS in data centers.

Advantages

- Consolidation enhances hardware utilization. Many underutilized servers are consolidated into fewer servers to enhance resource utilization. Consolidation also facilitates backup services and disaster recovery.

- This approach enables more agile provisioning and deployment of resources. In a virtual environment, the images of the guest OSes and their applications are readily cloned and reused.

- The total cost of ownership is reduced. In this sense, server virtualization causes deferred purchases of new servers, a smaller data-center footprint, lower maintenance costs, and lower power, cooling, and cabling requirements.

- This approach improves availability and business continuity. The crash of a guest OS has no effect on the host OS or any other guest OS. It becomes easier to transfer a VM from one server to another, because virtual servers are unaware of the underlying hardware.

To automate data-center operations, one must consider resource scheduling, architectural support, power management, automatic or autonomic resource management, performance of analytical models

UNIT - 2

Introduction to Cloud Computing

"Cloud is a parallel and distributed computing system consisting of a collection of inter-connected and virtualized computers that are dynamically provisioned and presented as one or more unified computing resources based on service-level agreements (SLA) established through negotiation between the service provider and consumers."

"Clouds are a large pool of easily usable and accessible virtualized resources (such as hardware, development platforms and/or services). These resources can be dynamically reconfigured to adjust to a variable load (scale), allowing also for an optimum resource utilization"

"This pool of resources is typically exploited by a pay-per-use model in which guarantees are offered by the Infrastructure Provider by means of customized Service Level Agreements."

"Clouds are hardware based services offering compute, network, and storage capacity where Hardware management is highly abstracted from the buyer, buyers incur infrastructure costs as variable OPEX, and infrastructure capacity is highly elastic."

Key characteristics of cloud computing
(1) the illusion of infinite computing resources;
(2) the elimination of an up-front commitment by cloud users;
(3) the ability to pay for use…as needed

The National Institute of Standards and Technology (NIST) characterizes **cloud computing** as ". . . a pay-per-use model for enabling available, convenient, on-demand network access to a shared pool of configurable computing resources (e.g. networks, servers, storage, applications, services) that can be rapidly provisioned and released with minimal management effort or service provider interaction".

Most common characteristics which a cloud should have:
(i) pay-per-use (no ongoing commitment, utility prices); (ii) elastic capacity and the illusion of infinite resources; (iii) self-service interface; and (iv) resources that are abstracted or virtualized.

1.2 Roots of Cloud Computing

The roots of clouds computing can be tracked by observing the advancement ofseveral technologies, especially in hardware (virtualization, multi-core chips),Internet technologies (Web services, service-oriented architectures, Web 2.0),distributed computing (clusters, grids), and systems management (autonomiccomputing, data center automation).

Figure 1.1 shows the convergence oftechnology fields that significantly advanced and contributed to the adventof cloud computing.

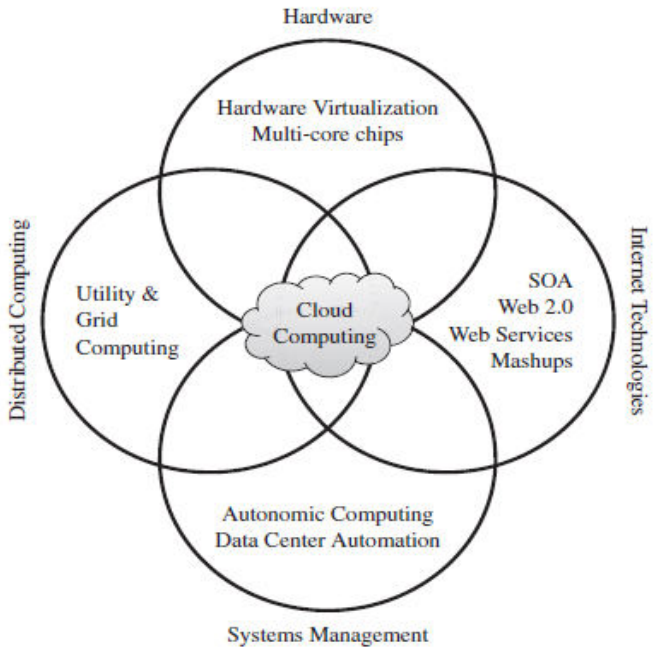


FIGURE 1.1. Convergence of various advances leading to the advent of cloud computing.

The IT world is currently experiencing a switch from in-housegenerated computing power into utility-supplied computing resources deliveredover the Internet as Web services.

Computing delivered as a utility can be defined as "on demand delivery of infrastructure, applications, and business processes in a security-rich, shared,scalable, and based computer environment over the Internet for a fee".

This model brings benefits to both consumers and providers of IT services.Consumers can attain reduction on IT-related costs by choosing to obtaincheaper services from external providers as opposed to heavily investing on ITinfrastructure and personnel hiring. The "on-demand" component of thismodel allows consumers to adapt their IT usage to rapidly increasing or unpredictable computing needs.

Providers of IT services achieve better operational costs; hardware andsoftware infrastructures are built to provide multiple solutions and serve manyusers, thus increasing efficiency and ultimately leading to faster return oninvestment (ROI) as well as lower total cost of ownership (TCO).

In the 1970s, companies who offered common dataprocessing tasks, such as payroll automation, operated time-shared mainframesas utilities, which could serve dozens of applications and often operated closeto 100% of their capacity.
The mainframe era collapsed with the advent of fast and inexpensivemicroprocessors and IT data centers moved to collections of commodityservers. Apart from its clear advantages, this new model inevitably led toisolation of workload into dedicated servers, mainly due to incompatibilitiesbetween software stacks and operating systems.

In addition, the unavailabilityof efficient computer networks meant that IT infrastructure should behosted in proximity to where it would be consumed. Altogether, these factshave prevented the utility computing reality of taking place on moderncomputer systems. These facts reveal the potential of delivering computing services withthe speed and reliability that businesses enjoy with their local machines. Thebenefits of economies of scale and high utilization allow providers to offercomputing services for a fraction of what it costs for a typical company that generates its own computing power.

SOA, Web Services, Web 2.0, and Mashups

The emergence of Web services (WS) open standards has significantly contributedto advances in the domain of software integration. Web servicescan combine together applications running on different messaging product platforms,enabling information from one application to be made available toothers, and enabling internal applications to be made available over theInternet.

WS standards have been created on top of existing ubiquitous technologiessuch as HTTP and XML, thus providing a common mechanism for deliveringservices, making them ideal for implementing a service-oriented architecture(SOA). The purpose of a SOA is to address requirements of loosely coupled,standards-based, and protocol-independent distributed computing. In a SOA,software resources are packaged as "services," which are well-defined, self-containedmodules that provide standard business functionality and areindependent of the state or context of other services.

Services are describedin a standard definition language and have a published interface. The maturity of WS has enabled the creation of powerful services that can beaccessed on-demand, in a uniform way. An enterprise application that follows theSOA paradigm is a collection of services that together perform complexbusiness logic.

In the consumer Web, information and services may be programmaticallyaggregated, acting as building blocks of complex compositions, called servicemashups. Many service providers, such as Amazon, del.icio.us, Facebook, andGoogle, make their service APIs publicly accessible using standard protocols such as SOAP and REST. Consequently, one can put an idea of a fully functional Web application into practice just by gluing pieces with few linesof code.

In the Software as a Service (SaaS) domain, cloud applications can be builtas compositions of other services from the same or different providers. Servicessuch as user authentication, e-mail, payroll management, and calendars areexamples of building blocks that can be reused and combined in a businesssolution in case a single, ready-made system does not provide all those features.

Grid Computing

Grid computing enables aggregation of distributed resources and transparently access to them. Most production grids such as TeraGrid and EGEE seek to share compute and storage resources distributed across differentadministrative domains, with their main focus being speeding up a broadrange of scientific applications, such as climate modeling, drug design, and protein analysis.

A key aspect of the grid vision realization has been building standard Webservices-based protocols that allow distributed resources to be “discovered,accessed, allocated, monitored, accounted for, and billed for, etc., and ingeneral managed as a single virtual system.” The Open Grid Services Architecture(OGSA) addresses this need for standardization by defining a set of corecapabilities and behaviors that address key concerns in grid systems.

Globus Toolkit is a middleware that implements several standard Gridservices and over the years has aided the deployment of several service-orientedGrid infrastructures and applications.

The development of standardized protocols for several grid computingactivities has contributed—theoretically—to allow delivery of on-demandcomputing services over the Internet. However, ensuring QoS in grids hasbeen perceived as a difficult endeavor. Lack of performance isolationhas prevented grids adoption in a variety of scenarios, especially on environmentswhere resources are oversubscribed or users are uncooperative.

Another issue that has lead to frustration when using grids is the availabilityof resources with diverse software configurations, including disparate operatingsystems, libraries, compilers, runtime environments, and so forth. At the sametime, user applications would often run only on specially customized environments.Consequently, a portability barrier has often been present on mostgrid infrastructures, inhibiting users of adopting grids as utility computingenvironments

Utility Computing

In utility computing environments, users assign a “utility” value to theirjobs, where utility is a fixed or time-varying valuation that captures variousQoS constraints (deadline, importance, satisfaction). The valuation is theamount they are willing to pay a service provider to satisfy their demands.The service providers then attempt to maximize their own utility, where said
utility may directly correlate with their profit. Providers can choose to prioritize high yield (i.e., profit per unit of resource) user jobs, leading to a scenario whereshared systems are viewed as a marketplace, where users compete for resourcesbased on the perceived utility or value of their jobs.

Hardware Virtualization

Cloud computing services are usually backed by large-scale data centerscomposed of thousands of computers. Such data centers are built to servemany users and host many disparate applications.
The idea of virtualizing a computer system’s resources, including processors,memory, and I/O devices, has been well established for decades, aiming atimproving sharing and utilization of computer systems. Hardware virtualizationallows running multiple operating systems and software stacks on asingle physical platform.

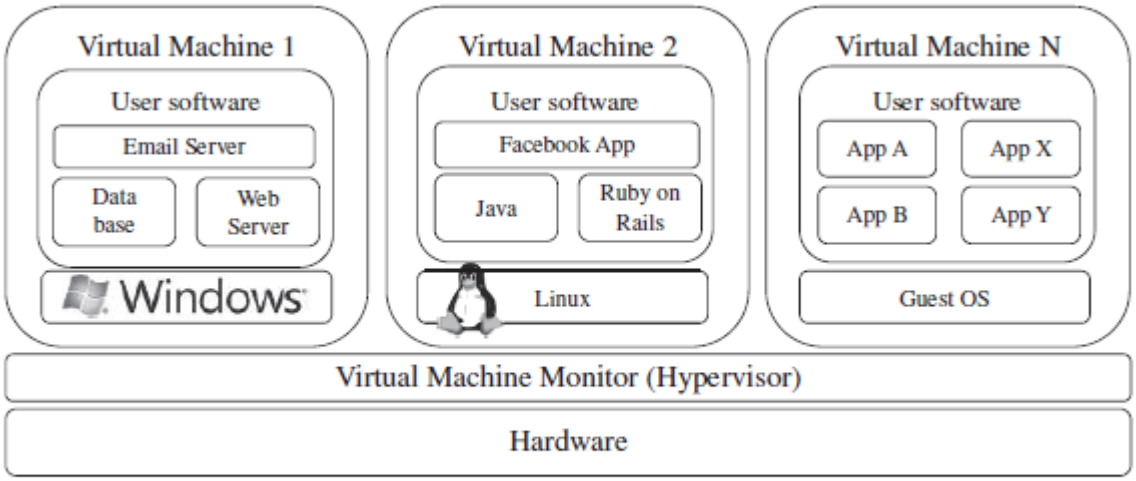


FIGURE 1.2. A hardware virtualized server hosting three virtual machines, each one running distinct operating system and user level software stack.

As depicted in Figure 1.2, a software layer, the virtualmachine monitor (VMM), also called a hypervisor, mediates access to thephysical hardware presenting to each guest operating system a virtual machine(VM), which is a set of virtual platform interfaces

The advent of several innovative technologies—multi-core chips, para-virtualization, hardware-assisted virtualization, and live migration of VMs—has contributed to an increasing adoption of virtualization on server systems.

Perceived benefits were improvements on sharing and utilization, better manageability, and higher reliability.

There are three basic capabilities regarding management of workload in a virtualized system, namely isolation, consolidation, and migration

Workload isolation is achieved since all program instructions are fullyconfined inside a VM, which leads to improvements in security. Betterreliability is also achieved because software failures inside one VM do notaffect others

The consolidation of several individual and heterogeneous workloads onto asingle physical platform leads to better system utilization. This practice is alsoemployed for overcoming potential software and hardware incompatibilities incase of upgrades, given that it is possible to run legacy and new operationsystems concurrently

Workload migration, also referred to as application mobility, targets atfacilitating hardware maintenance, load balancing, and disaster recovery. It isdone by encapsulating a guest OS state within a VM and allowing it to besuspended, fully serialized, migrated to a different platform, and resumedimmediately or preserved to be restored at a later date. A VM's stateincludes a full disk or partition image, configuration files, and an image of itsRAM.

A number of VMM platforms exist that are the basis of many utility orcloud computing environments. The most notable ones are VMWare, Xen, andKVM.

VMWARE ESXi: is a VMM from VMWare. It is a bare-metal hypervisor, meaning that it installs directly on the physical server, whereas others may require a host operating system. It provides advanced virtualization techniques of processor, memory, and I/O. Especially, through page sharing, it can overcommit memory, thus increasing the density of VMs inside a single physical server.

Xen : The Xen hypervisor started as an open-source project and has served as a base to other virtualization products, both commercial and open-source. It has pioneered the para-virtualization concept, on which the guest operating system, by means of a specialized kernel, can interact with the hypervisor, thus significantly improving performance.

KVM: The kernel-based virtual machine (KVM) is a Linux virtualization subsystem. It has been part of the mainline Linux kernel since version 2.6.20, thus being natively supported by several distributions. In addition, activities such as memory management and scheduling are carried out by existing kernel features, thus making KVM simpler and smaller than hypervisors that take control of the entire machine

Virtual Appliances and the Open Virtualization Format

An application combined with the environment needed to run it (operatingsystem, libraries, compilers, databases, application containers, and so forth) isreferred to as a "virtual appliance."

Packaging application environments in theshape of virtual appliances eases software customization, configuration, andpatching and improves portability. Most commonly, an appliance is shaped asa VM disk image associated with hardware requirements, and it can be readilydeployed in a hypervisor. The VMWare virtualappliance marketplace allows users to deploy appliances on VMWare hypervisors or on partners public clouds, and Amazon allows developers to sharespecialized Amazon Machine Images (AMI) and monetize their usage onAmazon EC2.

In a multitude of hypervisors, where each one supports a different VM imageformat and the formats are incompatible with one another, a great deal ofinteroperability issues arises. In order to facilitate packing and distribution of software to be run on VMsseveral vendors, including VMware, IBM, Citrix, Cisco, Microsoft, Dell, andHP, have devised the Open Virtualization Format (OVF). It aims at being"open, secure, portable, efficient and extensible" [32]. An OVF package consistsof a file, or set of files, describing the VM hardware characteristics (e.g., memory, network cards, and disks), operating system details, startup, andshutdown actions, the virtual disks themselves, and other metadata containingproduct and licensing information. OVF also supports complex packagescomposed of multiple VMs.

Autonomic Computing

The increasing complexity of computing systems has motivated research onautonomic computing, which seeks to improve systems by decreasing humaninvolvement in their operation. In other words, systems should managethemselves, with high-level guidance from humans

Autonomic, or self-managing, systems rely on monitoring probes andgauges (sensors), on an adaptation engine (autonomic manager) for computingoptimizations based on monitoring data, and on effectors to carry out changeson the system. IBM's Autonomic Computing Initiative has

contributed todefine the four properties of autonomic systems: self-configuration, self-optimization,self-healing, and self-protection. IBM has also suggested areference model for autonomic control loops of autonomic managers, calledMAPE-K (Monitor Analyze Plan Execute—Knowledge)

The large data centers of cloud computing providers must be managed in anefficient way. In this sense, the concepts of autonomic computing inspiresoftware technologies for data center automation, which may perform taskssuch as: management of service levels of running applications; management ofdata center capacity; proactive disaster recovery; and automation of VMprovisioning

1.3 LAYERS AND TYPES OF CLOUDS

Cloud computing services are divided into three classes
(1) Infrastructure as a Service, (2) Platform as a Service, and (3) Softwareas a Service

Figure 1.3 depicts the layered organization of the cloud stackfrom physical infrastructure to applications.

These abstraction levels can also be viewed as a layered architecture whereservices of a higher layer can be composed from services of the underlying layerA core middleware manages physical resources andthe VMs deployed on top of them; in addition, it provides the required features(e.g., accounting and billing) to offer multi-tenant pay-as-you-go services.

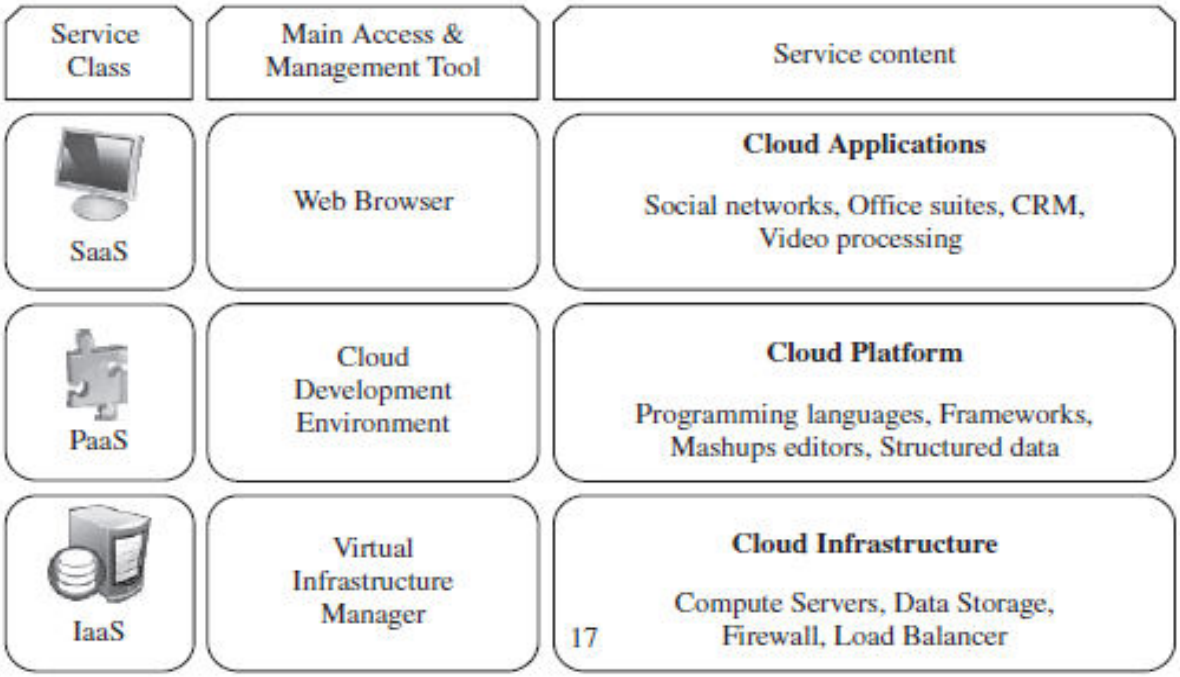


FIGURE 1.3 The cloud computing stack

Infrastructure as a Service

Offering virtualized resources (computation, storage, and communication) ondemand is known as Infrastructure as a Service (IaaS). A cloud infrastructureenables on-demand provisioning of servers running several choices of operatingsystems and a customized software stack. Infrastructure services are consideredto be the bottom layer of cloud computing systems.

Amazon Web Services mainly offers IaaS, which in the case of its EC2service means offering VMs with a software stack that can be customizedsimilar to how an ordinary physical server would be customized. Users aregiven privileges to perform numerous activities to the server, such as: startingand stopping it, customizing it by installing software packages, attachingvirtual disks to it, and configuring access permissions and firewalls rules.

Platform as a Service

Acloud platform offers an environment on which developers create and deployapplications and do not necessarily need to know how many processors or howmuch memory that applications will be using. In addition, multiple programmingmodels and specialized services (e.g., data access, authentication, andpayments) are offered as building blocks to new applications.

Google AppEngine, an example of Platform as a Service, offers a scalableenvironment for developing and hosting Web applications, which shouldbe written in specific programming languages such as Python or Java, and usethe services' own proprietary structured object data store.

Software as a Service

Applications reside on the top of the cloud stack. Services provided by thislayer can be accessed by end users through Web portals. Therefore, consumersare increasingly shifting from locally installed computer programs to on-linesoftware services that offer the same functionally. Traditional desktop applicationssuch as word processing and spreadsheet can now be accessed as a servicein the Web. This model of delivering applications, known as Software as aService (SaaS), alleviates the burden of software maintenance for customersand simplifies development and testing for providers.

Salesforce.com, which relies on the SaaS model, offers business productivityapplications (CRM) that reside completely on their servers, allowing costumersto customize and access applications on demand.

Deployment Models

A cloud can be classified as public, private, community, or hybrid based on model of deployment as shown in Figure 1.4.

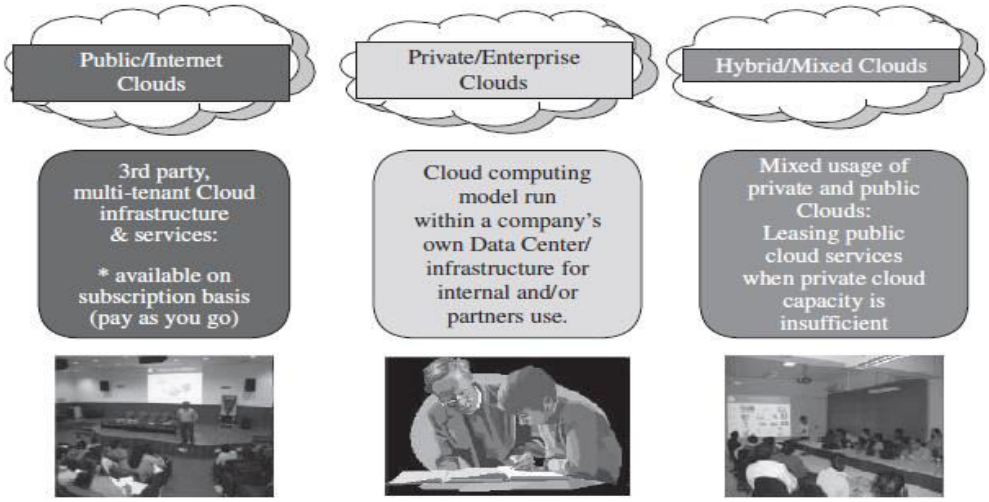


FIGURE 1.4. Types of clouds based on deployment models.

Public cloud: "cloud made available in a pay-as-you-go manner to the general public"
Private cloud: "internal data center of a business or other organization, not made available to the general public."
Community cloud:"shared by several organizations and supports a specific community that has shared concerns (e.g., mission, security requirements, policy, and compliance considerations)
Hybrid cloudtakes shape when a private cloud is supplemented with computing capacity from public clouds.
The approach of temporarily renting capacity to handle spikes in load is known as **"cloud-bursting"**
Desired Features of a Cloud

Certain features of a cloud are essential to enable services that truly represent the cloud computing model and satisfy expectations of consumers, and cloud offerings must be
(i) self-service, (ii) per-usage metered and billed, (iii) elastic, and (iv) Customizable

Self-Service: clouds must allow self-service access so that customers can request, customize, pay, and use services without intervention of human operators

Per-Usage Metering and Billing: Cloud computing eliminates up-front commitment by users, allowing them to request and use only the necessary amount. Services must be priced on a short term basis (e.g., by the hour), allowing users to release (and not pay for) resources as soon as they are not needed

Elasticity: Cloud computing gives the illusion of infinite computing resources available on demand. Therefore users expect clouds to rapidly provide resources in any quantity at any time. In particular, it is expected that the additional resources can be (a) provisioned, possibly automatically, when an application load increases and (b) released when load decreases (scale up and down)

Customization: resources rented from the cloud must be highly customizable. customization means allowing users to deploy specialized virtual appliances and to be given privileged (root) access to the virtual servers.

CLOUD INFRASTRUCTURE MANAGEMENT

A key challenge IaaS providers face when building a cloud infrastructure is managing physical and virtual resources, namely servers, storage, and networks, in a holistic fashion. The orchestration of resources must be performed in a way to rapidly and dynamically provision resources to applications.
The software toolkit responsible for this orchestration is called a virtual infrastructure manager (VIM). This type of software resembles a traditional operating system—but instead of dealing with a single computer, it aggregates resources from multiple computers, presenting a uniform view to user and applications.

Features

Virtualization Support: The multi-tenancy aspect of clouds requires multiple customers with disparate requirements to be served by a single hardware infrastructure. Virtualized resources (CPUs, memory, etc.) can be sized and resized with certain flexibility. These features make hardware virtualization, the ideal technology to create a virtual infrastructure that partitions a data center among multiple tenants.
Self-Service, On-Demand Resource Provisioning: Self-service access to resources has been perceived as one the most attractive features of clouds. This feature enables users to directly obtain services from clouds, such as spawning the creation of a server and tailoring its software, configurations, and security policies, without interacting with a human system administrator. This capability "eliminates the need for more time-consuming, labor-intensive, human driven procurement processes familiar to many in IT".

Multiple Backend Hypervisors: Different virtualization models and tools offer different benefits, drawbacks, and limitations. Thus, some VI managers provide a uniform management layer regardless of the virtualization technology used. This characteristic is more visible in open-source VI managers, which usually provide pluggable drivers to interact with multiple hypervisors. In this direction, the aim of libvirt is to provide a uniform API that VI managers can use to manage domains (a VM or container running an instance of an operating system) in virtualized nodes using standard operations that abstract hypervisor specific calls.

Storage Virtualization: Virtualizing storage means abstracting logical storage from physical storage. By consolidating all available storage devices in a data center, it allows creating virtual disks independent from device and location.

Interface to Public Clouds: Extending the capacity of a local in-house computing infrastructure by borrowing resources from public clouds is advantageous. In this fashion, institutions can make good use of their available resources and, in case of spikes in demand, extra load can be offloaded to rented resources. A VI manager can be used in a hybrid cloud setup if it offers a driver to manage the life cycle of virtualized resources obtained from external cloud providers. To the applications, the use of leased resources must ideally be transparent.

Virtual Networking: Virtual networks allow creating an isolated network on top of a physical infrastructure independently from physical topology and locations. A virtual LAN (VLAN) allows isolating traffic that shares a switched network, allowing VMs to be grouped into the same broadcast domain. Additionally, a VLAN can be configured to block traffic originated from VMs from other networks.

Dynamic Resource Allocation: Increased awareness of energy consumption in data centers has encouraged the practice of dynamic consolidating VMs in a fewer number of servers. In cloud infrastructures, where applications have variable and dynamic needs, capacity management and demand prediction are especially complicated. This fact triggers the need for dynamic resource allocation aiming at obtaining a timely match of supply and demand. Energy consumption reduction and better management of SLAs can be achieved by dynamically remapping VMs to physical machines at regular intervals. Machines that are not assigned any VM can be turned off or put on a low power state. In the same fashion, overheating can be avoided by moving load away from hotspots.

Virtual Clusters: Several VI managers can holistically manage groups of VMs. This feature is useful for provisioning computing virtual clusters on demand, and interconnected VMs for multi-tier Internet applications.

Reservation and Negotiation Mechanism: When users request computational resources to available at a specific time, requests are termed advance reservations (AR), in contrast to best-effort requests, when users request resources whenever available. To support complex requests, such as AR, a VI manager must allow users to "lease" resources expressing more complex terms (e.g., the period of time of a reservation). This is especially useful in clouds on which resources are scarce; since not all requests may be satisfied immediately, they can benefit of VM placement strategies that support queues, priorities, and advance reservations. Additionally, leases may be negotiated and renegotiated, allowing provider and consumer to modify a lease or present counter proposals until an agreement is reached. This feature is illustrated by the case in which an AR request for a given slot cannot be satisfied, but the provider can offer a distinct slot that is still satisfactory to the user. This problem has been addressed in OpenPEX, which incorporates a bilateral negotiation protocol that allows users and providers to come to an alternative agreement by exchanging offers and counter offers.

High Availability and Data Recovery: The high availability (HA) feature of VI managers aims at minimizing application downtime and preventing business disruption. A few VI managers accomplish this by providing a failover mechanism, which detects failure of both physical and virtual servers and restarts VMs on healthy physical servers. This style of HA protects from host, but not VM, failures. For mission critical applications, when a failover solution involving restarting VMs does not suffice, additional levels of fault tolerance that rely on redundancy of VMs are implemented. In this style, redundant and synchronized VMs (running or in standby) are kept in a secondary physical server. The HA solution monitors failures of system components such as servers, VMs, disks, and network and ensures that a duplicate VM serves the application in case of failures. Data backup in clouds should take into account the high data volume involved in VM management. Frequent backup of a large number of VMs, each one with multiple virtual disks attached, should be done with minimal interference in the systems performance. In this sense, some VI managers offer data protection mechanisms that perform incremental backups of VM images. The backup workload is often assigned to proxies, thus offloading production server and reducing network overhead

Case Studies

Apache VCL: The Virtual Computing Lab project has been incepted in 2004 by researchers at the North Carolina State University as a way to provide customized environments to computer lab users. Apache VCL provides the following features: (i) multi-platform controller, based on Apache/PHP; (ii) Web portal and XML-RPC interfaces; (iii) support for VMware hypervisors

(ESX, ESXi, and Server); (iv) virtual networks; (v) virtual clusters; and (vi) advance reservation of capacity.

AppLogic. AppLogic: is a commercial VI manager, the flagship product of 3tera Inc. from California, USA. The company has labeled this product as a Grid Operating System.
AppLogic provides the following features: Linux-based controller; CLI and GUI interfaces; Xen backend; Global Volume Store (GVS) storage virtualization; virtual networks; virtual clusters; dynamic resource allocation; high availability; and data protection.

Citrix Essentials: The Citrix Essentials suite is one the most feature complete VI management software available, focusing on management and automation of data centers. It is essentially a hypervisor-agnostic solution, currently supporting Citrix XenServer and Microsoft Hyper-V. Citrix Essentials provides the following features: Windowsbased controller; GUI, CLI, Web portal, and XML-RPC interfaces; support for XenServer and Hyper-V hypervisors; Citrix Storage Link storage virtualization;virtual networks; dynamic resource allocation; three-level high availability (i.e., recovery by VM restart, recovery by activating paused duplicate VM, and running duplicate VM continuously); data protection with Citrix Consolidated Backup.

Enomaly ECP: The Enomaly Elastic Computing Platform, in its most complete edition, offers most features a service provider needs to build an IaaS cloud. Enomaly ECP provides the following features: Linux-based controller; Web portal and Web services (REST) interfaces; Xen back-end; interface to the Amazon EC2 public cloud; virtual networks; virtual clusters (ElasticValet) Eucalyptus.

The Eucalyptus: framework was one of the first open-source projects to focus on building IaaS clouds. It has been developed with the intent of providing an open-source implementation nearly identical in functionality to Amazon Web Services APIs. Eucalyptus provides the following features: Linux-based controller with administration Web portal; EC2-compatible (SOAP, Query) and S3-compatible (SOAP, REST) CLI and Web portal interfaces; Xen, KVM, and VMWare backends; Amazon EBS-compatible virtual storage devices; interface to the Amazon EC2 public cloud; virtual networks.

Nimbus3: The Nimbus toolkit is built on top of the Globus framework. Nimbus provides most features in common with other open-source VI managers, such as an EC2-compatible front-end API, support to Xen, and a backend interface to Amazon EC2. However, it distinguishes from others by providing a Globus Web Services Resource Framework (WSRF) interface. It also provides a backend service, named Pilot, which spawns VMs on clusters managed by a local resource manager (LRM) such as PBS and SGE.

OpenNebula: OpenNebula is one of the most feature-rich open-source VI managers. It was initially conceived to manage local virtual infrastructure, but has also included remote interfaces that make it viable to build public clouds. Altogether, four programming APIs are available: XML-RPC and libvirt for local interaction; a subset of EC2 (Query) APIs and the OpenNebula Cloud API (OCA) for public access. OpenNebula provides the following features: Linux-based controller; CLI, XML-RPC, EC2-compatible Query and OCA interfaces; Xen, KVM, and VMware backend; interface to public clouds (Amazon EC2, ElasticHosts); virtual networks; dynamic resource allocation; advance reservation of capacity.

OpenPEX: OpenPEX (Open Provisioning and EXecution Environment) was constructed around the notion of using advance reservations as the primary method for allocatingVMinstances.

OpenPEX provides the following features: multi-platform (Java) controller; Web portal and Web services (REST) interfaces; Citrix XenServer backend; advance reservation of capacity with negotiation.

oVirt: oVirt is an open-source VI manager, sponsored by Red Hat's Emergent Technology group oVirt provides the following features: Fedora Linux-based controller packaged as a virtual appliance; Web portal interface; KVM backend.

Platform ISF: Infrastructure Sharing Facility (ISF) is the VI manager offering from Platform Computing. The company, mainly through its LSF family of products, has been serving the HPC market for several years ISF provides the following features: Linux-based controller packaged as a virtual appliance; Web portal interface; dynamic resource allocation; advance reservation of capacity; high availability.

VMWare vSphere and vCloud: vSphere is VMware's suite of tools aimed at transforming IT infrastructures into private clouds. In the vSphere architecture, servers run on the ESXi platform. A separate server runs vCenter Server, which centralizes control over the entire virtual infrastructure. Through the vSphere Client software, administrators connect to vCenter Server to perform various tasks. The Distributed Resource Scheduler (DRS) makes allocation decisions based on predefined rules and policies. It continuously monitors the amount of resources available to VMs and, if necessary, makes allocation changes to meet VM requirements. In the storage virtualization realm, vStorage VMFS is a cluster file system to provide aggregate several disks in a single volume. VMFS is especially optimized to store VM images and virtual disks. It supports storage equipment that use Fibre Channel or iSCSI SAN. vSphere provides the following features: Windows-based controller (vCenter Server); CLI, GUI, Web portal, and Web services interfaces; VMware ESX, ESXi backend; VMware vStorage VMFS storage virtualization; interface to external clouds (VMware vCloud partners); virtual networks (VMWare Distributed Switch); dynamic resource allocation (VMware DRM); high availability; data protection (VMWare Consolidated Backup).

INFRASTRUCTURE AS A SERVICE PROVIDERS

Public Infrastructure as a Service providers commonly offer virtual servers containing one or more CPUs, running several choices of operating systems and a customized software stack. In addition, storage space and communication facilities are often provided.

Features

IaaS offerings can be distinguished by the availability of specialized features that influence the cost_benefit ratio to be experienced by user applications when moved to the cloud. The most relevant features are: (i) geographic distribution of data centers; (ii) variety of user interfaces and APIs to access the system; (iii) specialized components and services that aid particular applications (e.g., loadbalancers, firewalls); (iv) choice of virtualization platform and operating systems; and (v) different billing methods and period (e.g., prepaid vs. post-paid, hourly vs. monthly).

Geographic Presence: To improve availability and responsiveness, a provider of worldwide services would typically build several data centers distributed around the world. For example, Amazon Web Services presents the concept of "availability zones" and "regions" for its EC2

service. Availability zones are “distinct locations that are engineered to be insulated from failures in other availability zones and provide inexpensive, low-latency network connectivity to other availability zones in the same region.” Regions, in turn, “are geographically dispersed and will be in separate geographic areas or countries

User Interfaces and Access to Servers: Ideally, a public IaaS provider must provide multiple access means to its cloud, thus catering for various users and their preferences. Different types of user interfaces (UI) provide different levels of abstraction, the most common being graphical user interfaces (GUI), command-line tools (CLI), and Web service (WS) APIs.

Advance Reservation of Capacity: Advance reservations allow users to request for an IaaS provider to reserve resources for a specific time frame in the future, thus ensuring that cloud resources will be available at that time. However, most clouds only support best-effort requests; that is, users requests are server whenever resources are available.

Automatic Scaling and Load Balancing: Elasticity is a key characteristic of the cloud computing model. Applications often need to scale up and down to meet varying load conditions. Automatic scaling is a highly desirable feature of IaaS clouds. It allow users to set conditions for when they want their applications to scale up and down, based on application-specific metrics such as transactions per second, number of simultaneous users, request latency, and so forth. When the number of virtual servers is increased by automatic scaling, incoming traffic must be automatically distributed among the available servers. This activity enables applications to promptly respond to traffic increase while also achieving greater fault tolerance.

Service-Level Agreement: Service-level agreements (SLAs) are offered by IaaS providers to express their commitment to delivery of a certain QoS. To customers it serves as a warranty. An SLA usually include availability and performance guarantees. Additionally, metrics must be agreed upon by all parties as well as penalties for violating these expectations. Most IaaS providers focus their SLA terms on availability guarantees, specifying the minimum percentage of time the system will be available during a certain period.

Hypervisor and Operating System Choice: Traditionally, IaaS offerings have been based on heavily customized open-source Xen deployments. IaaS providers needed expertise in Linux, networking, virtualization, metering, resource management, and many other low-level aspects to successfully deploy and maintain their cloud offerings.

Case Studies

Amazon Web Services: Amazon WS4 (AWS) is one of the major players in the cloud computing market. It pioneered the introduction of IaaS clouds in 2006. It offers a variety cloud services, most notably: S3 (storage), EC2 (virtual servers), Cloudfront (content delivery), Cloudfront Streaming (video streaming), SimpleDB (structured datastore), RDS (Relational Database), SQS (reliable messaging), and Elastic MapReduce (data processing). The ElasticCompute Cloud (EC2) offers Xen-based virtual servers (instances) that can be instantiated from Amazon Machine Images (AMIs). Instances are available in a variety of sizes, operating systems, architectures, and price. CPU capacity of instances is measured in Amazon Compute Units and, although fixed for each instance, vary among instance types from 1 (small instance) to 20 (high CPU instance). Each instance provides a certain amount of nonpersistent disk space; a persistence disk service (Elastic Block Storage) allows attaching virtual disks to instances with space up to 1TB. Elasticity can be achieved by combining the CloudWatch, Auto Scaling, and

Elastic Load Balancing features, which allow the number of instances to scale up and down automatically based on a set of customizable rules, and traffic to be distributed across available instances. Fixed IP address (Elastic IPs) are not available by default, but can be obtained at an additional cost.

Flexiscale: Flexiscale is a UK-based provider offering services similar in nature to Amazon Web Services. Flexiscale cloud provides the following features: available in UK; Web services (SOAP), Web-based user interfaces; access to virtual server mainly via SSH (Linux) and Remote Desktop (Windows); 100% availability SLA with automatic recovery of VMs in case of hardware failure; per hour pricing; Linux and Windows operating systems; automatic scaling (horizontal/vertical).

Joyent: Joyent's Public Cloud offers servers based on Solaris containers virtualization technology. These servers, dubbed accelerators, allow deploying various specialized software-stack based on a customized version of Open- Solaris operating system, which include by default a Web-based configuration tool and several pre-installed software, such as Apache, MySQL, PHP, Ruby on Rails, and Java. Software load balancing is available as an accelerator in addition to hardware load balancers. A notable feature of Joyent's virtual servers is automatic vertical scaling of CPU cores, which means a virtual server can make use of additional CPUs automatically up to the maximum number of cores available in the physical host.
The Joyent public cloud offers the following features: multiple geographic locations in the United States; Web-based user interface; access to virtual server via SSH and Web-based administration tool; 100% availability SLA; per month pricing; OS-level virtualization Solaris containers; Open- Solaris operating systems; automatic scaling (vertical).

GoGrid: GoGrid, like many other IaaS providers, allows its customers to utilize a range of pre-made Windows and Linux images, in a range of fixed instance sizes. GoGrid also offers "value-added" stacks on top for applications such as high-volume Web serving, e-Commerce, and database stores. It offers some notable features, such as a "hybrid hosting" facility, which combines traditional dedicated hosts with auto-scaling cloud server infrastructure. As part of its core IaaS offerings, GoGrid also provides free hardware load balancing, auto-scaling capabilities, and persistent storage, features that typically add an additional cost for most other IaaS providers.

Rackspace Cloud Servers: Rackspace Cloud Servers is an IaaS solution that provides fixed size instances in the cloud. Cloud Servers offers a range of Linux-based pre-made images. A user can request different-sized images, where the size is measured by requested RAM, not CPU.

PLATFORM AS A SERVICE PROVIDERS

Public Platform as a Service providers commonly offer a development and deployment environment that allow users to create and run their applications with little or no concern to low-level details of the platform. In addition, specific programming languages and frameworks are made available in the platform, as well as other services such as persistent data storage and in memory caches.

Features

Programming Models, Languages, and Frameworks: Programming models made available by IaaS providers define how users can express their applications using higher levels of abstraction and efficiently run them on the cloud platform. Each model aims at efficiently solving a

particular problem. In the cloud computing domain, the most common activities that require specialized models are: processing of large dataset in clusters of computers (MapReduce model), development of request-based Web services and applications; definition and orchestration of business processes in the form of workflows (Workflow model); and high-performance distributed execution of various computational tasks.
For user convenience, PaaS providers usually support multiple programming languages. Most commonly used languages in platforms include Python and Java (e.g., Google AppEngine), .NET languages (e.g., Microsoft Azure), and Ruby (e.g., Heroku). Force.com has devised its own programming language (Apex) and an Excel-like query language, which provide higher levels of abstraction to key platform functionalities.
A variety of software frameworks are usually made available to PaaS developers, depending on application focus. Providers that focus on Web and enterprise application hosting offer popular frameworks such as Ruby on Rails, Spring, Java EE, and .NET.

Persistence Options: A persistence layer is essential to allow applications to record their state and recover it in case of crashes, as well as to store user data. Web and enterprise application developers have chosen relational databases as the preferred persistence method. These databases offer fast and reliable structured data storage and transaction processing, but may lack scalability to handle several petabytes of data stored in commodity computers. In the cloud computing domain, distributed storage technologies have emerged, which seek to be robust and highly scalable, at the expense of relational structure and convenient query languages.

Case Studies

Aneka: Aneka is a .NET-based service-oriented resource management and development platform. Each server in an Aneka deployment (dubbed Aneka cloud node) hosts the Aneka container, which provides the base infrastructure that consists of services for persistence, security (authorization, authentication and auditing), and communication (message handling and dispatching). Cloud nodes can be either physical server, virtual machines (XenServer and VMware are supported), and instances rented from Amazon EC2. The Aneka container can also host any number of optional services that can be added by developers to augment the capabilities of an Aneka Cloud node, thus providing a single, extensible framework for orchestrating various application models.
Several programming models are supported by such task models to enable execution of legacy HPC applications and MapReduce, which enables a variety of data-mining and search applications. Users request resources via a client to a reservation services manager of the Aneka master node, which manages all cloud nodes and contains scheduling service to distribute request to cloud nodes.

App Engine: Google App Engine lets you run your Python and Java Web applications on elastic infrastructure supplied by Google. App Engine allows your applications to scale dynamically as your traffic and data storage requirements increase or decrease. It gives developers a choice between a Python stack and Java. The App Engine serving architecture is notable in that it allows real-time auto-scaling without virtualization for many common types of Web applications. However, such auto-scaling is dependent on the application developer using a limited subset of the native APIs on each platform, and in some instances you need to use specific Google APIs such as URLFetch, Datastore, and memcache in place of certain native API calls. For example, a deployed App Engine application cannot write to the file system directly (you must use the Google Datastore) or open a socket or access another host directly (you must use Google URL fetch service). A Java application cannot create a new Thread either.

Microsoft Azure: Microsoft Azure Cloud Services offers developers a hosted. NET Stack (C#, VB.Net, ASP.NET). In addition, a Java & Ruby SDK for .NET Services is also available. The Azure system consists of a number of elements. The Windows Azure Fabric Controller provides auto-scaling and reliability, and it manages memory resources and load balancing. The .NET Service Bus registers and connects applications together. The .NET Access Control identity providers include enterprise directories and Windows LiveID. Finally, the .NET Workflow allows construction and execution of workflow instances.

Force.com: In conjunction with the Salesforce.com service, the Force.com PaaS allows developers to create add-on functionality that integrates into main Salesforce CRM SaaS application. Force.com offers developers two approaches to create applications that can be deployed on its SaaS plaform: a hosted Apex or Visualforce application. Apex is a proprietary Java-like language that can be used to create Salesforce applications. Visualforce is an XML-like syntax for building UIs in HTML, AJAX, or Flex to overlay over the Salesforce hosted CRM system. An application store called AppExchange is also provided, which offers a paid & free application directory.

Heroku: Heroku is a platform for instant deployment of Ruby on Rails Web applications. In the Heroku system, servers are invisibly managed by the platform and are never exposed to users. Applications are automatically dispersed across different CPU cores and servers, maximizing performance and minimizing contention. Heroku has an advanced logic layer than can automatically route around failures, ensuring seamless and uninterrupted service at all times.

CHALLENGES AND RISKS

Despite the initial success and popularity of the cloud computing paradigm and the extensive availability of providers and tools, a significant number of challenges and risks are inherent to this new model of computing. Providers, developers, and end users must consider these challenges and risks to take good advantage of cloud computing. Issues to be faced include user privacy, data security, data lock-in, availability of service, disaster recovery, performance, scalability, energy-efficiency, and programmability.

Security, Privacy, and Trust: Security and privacy affect the entire cloud computing stack, since there is a massive use of third-party services and infrastructures that are used to host important data or to perform critical operations. In this scenario, the trust toward providers is fundamental to ensure the desired level of privacy for applications hosted in the cloud. Legal and regulatory issues also need attention. When data are moved into the Cloud, providers may choose to locate them anywhere on the planet. The physical location of data centers determines the set of laws that can be applied to the management of data. For example, specific cryptography techniques could not be used because they are not allowed in some countries. Similarly, country laws can impose that sensitive data, such as patient health records, are to be stored within national borders.

Data Lock-In and Standardization: A major concern of cloud computing users is about having their data locked-in by a certain provider. Users may want to move data and applications out from a provider that does not meet their requirements. However, in their current form, cloud computing infrastructures and platforms do not employ standard methods of storing user data and applications. Consequently, they do not interoperate and user data are not portable.

The answer to this concern is standardization. In this direction, there are efforts to create open standards for cloud computing. The Cloud Computing Interoperability Forum (CCIF) was formed by organizations such as Intel, Sun, and Cisco in order to "enable a global cloud computing ecosystem whereby organizations are able to seamlessly work together for the purposes for wider industry adoption of cloud computing technology." The development of the Unified Cloud Interface (UCI) by CCIF aims at creating a standard programmatic point of access to an entire cloud infrastructure. In the hardware virtualization sphere, the Open Virtual Format (OVF) aims at facilitating packing and distribution of software to be run on VMs so that virtual appliances can be made portable—that is, seamlessly run on hypervisor of different vendors.

Availability, Fault-Tolerance, and Disaster Recovery: It is expected that users will have certain expectations about the service level to be provided once their applications are moved to the cloud. These expectations include availability of the service, its overall performance, and what measures are to be taken when something goes wrong in the system or its components. In summary, users seek for a warranty before they can comfortably move their business to the cloud. SLAs, which include QoS requirements, must be ideally set up between customers and cloud computing providers to act as warranty. An SLA specifies the details of the service to be provided, including availability and performance guarantees. Additionally, metrics must be agreed upon by all parties, and penalties for violating the expectations must also be approved.

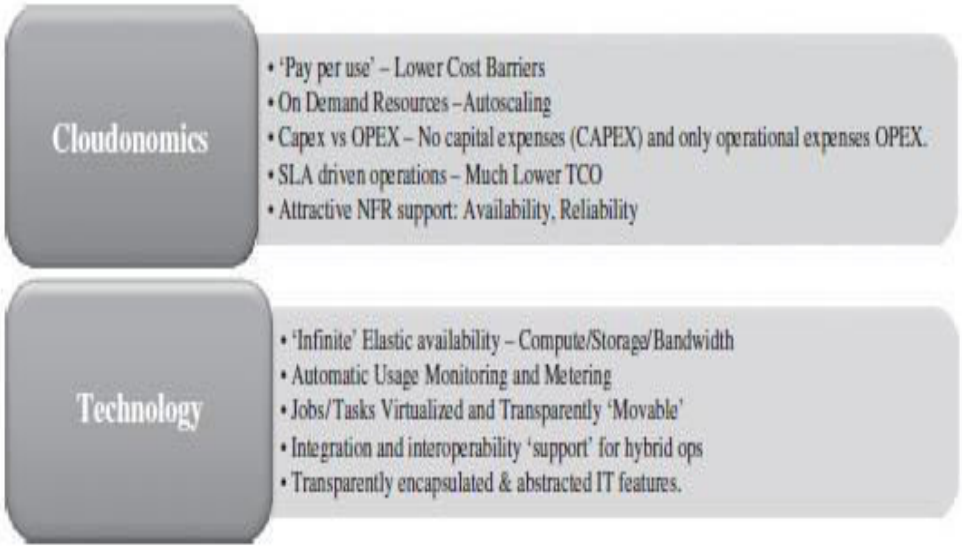
Resource Management and Energy-Efficiency: One important challenge faced by providers of cloud computing services is the efficient management of virtualized resource pools. Physical resources such as CPU cores, disk space, and network bandwidth must be sliced and shared among virtual machines running potentially heterogeneous workloads. The multi-dimensional nature of virtual machines complicates the activity of finding a good mapping of VMs onto available physical hosts while maximizing user utility. Dimensions to be considered include: number of CPUs, amount of memory, size of virtual disks, and network bandwidth. Dynamic VM mapping policies may leverage the ability to suspend, migrate, and resume VMs as an easy way of preempting low-priority allocations in favor of higher-priority ones. Migration of VMs also brings additional challenges such as detecting when to initiate a migration, which VM to migrate, and where to migrate. In addition, policies may take advantage of live migration of virtual machines to relocate data center load without significantly disrupting running services. In this case, an additional concern is the trade-off between the negative impact of a live migration on the performance and stability of a service and the benefits to be achieved with that migration. Another challenge concerns the outstanding amount of data to be managed in various VM management activities. Such data amount is a result of particular abilities of virtual machines, including the ability of traveling through space (i.e., migration) and time (i.e., checkpointing and rewinding), operations that may be required in load balancing, backup, and recovery scenarios. In addition, dynamic provisioning of new VMs and replicating existing VMs require efficient mechanisms to make VM block storage devices (e.g., image files) quickly available at selected hosts. Data centers consumer large amounts of electricity. According to a data published byHP[4], 100 server racks can consume 1.3MWof power and another 1.3 MW are required by the cooling system, thus costing USD 2.6 million per year. Besides the monetary cost, data centers significantly impact the environment in terms of CO2 emissions from the cooling systems

Migrating into a Cloud

Cloud computing: "It is a techno-business disruptive model of using distributed large-scale data centers either private or public or hybrid offering customers a scalable virtualized infrastructure or an abstracted set of services qualified by service-level agreements (SLAs) and charged only by the abstracted IT resources consumed." Most enterprises today are powered by captive data centers. In most large or small enterprises today, IT is the backbone of their operations. Invariably for these large enterprises, their data centers are distributed across various geographies. They comprise systems and software that span several generations of products sold by a variety of IT vendors. In order to meet varying loads, most of these data centers are provisioned with capacity beyond the peak loads experienced. If the enterprise is in a seasonal or cyclical business, then the load variation would be significant. Thus what is observed generally is that the provisioned capacity of IT resources is several times the average demand. This is indicative of significant degree of idle capacity. Many data center management teams have been continuously innovating their management practices and technologies deployed to possibly squeeze out the last possible usable computing resource cycle through appropriate programming, systems configurations, SLAs, and systems management. Cloud computing turned attractive to them because they could pass on the additional demand from their IT setups onto the cloud while paying only for the usage and being unencumbered by the load of operations and management

The promise of the cloud computing services

In small and medium enterprises, cloud computing usage for all additional cyclical IT needs has yielded substantial and significant economic savings. This economics and the associated trade-offs, of leveraging the cloud computing services, now popularly called "cloudonomics," for satisfying enterprise's seasonal IT loads has become a topic of deep interest amongst IT managers and technology architects



As shown in Figure 2.1, the promise of the cloud both on the business front (the attractive cloudonomics) and the technology front widely aided the CxOs to spawn out several non-mission critical IT needs from the ambit of their captive traditional data centers to the appropriate cloud service. Invariably, these IT needs had some common features: They were typically Web-oriented; they represented seasonal IT demands; they were amenable to parallel batch processing;

they were non-mission critical and therefore did not have high security demands. They included scientific applications too. Several small and medium business enterprises, however, leveraged the cloud much beyond the cautious user. Many startups opened their IT departments exclusively using cloud services—very successfully and with high ROI. Having observed these successes, several large enterprises have started successfully running pilots for leveraging the cloud. Many large enterprises run SAP to manage their operations.

Why Migrate

- Business Reasons
- Technological Reasons

What can be Migrated

- Application
- Code
- Design
- Architecture
- Usage

The migration of an enterprise application is best captured by the following:

$$P \rightarrow P'_C + P'_I \rightarrow P'_{OFC} + P'_I$$

where P is the application before migration, running in captive data center

P'_C is the application part after migration into a (hybrid) cloud

P'_I is the part of application being run in the captive local data center

P'_{OFC} is the application part optimized for cloud.

Invariably, migrating into the cloud is driven by economic reasons of cost cutting in both the IT capital expenses (Capex) as well as operational expenses (Opex).

If the average costs of using an enterprise application on a cloud is substantially lower than the costs of using it in one's captive data center and if the cost of migration does not add to the burden on ROI, then the case for migration into the cloud is strong.

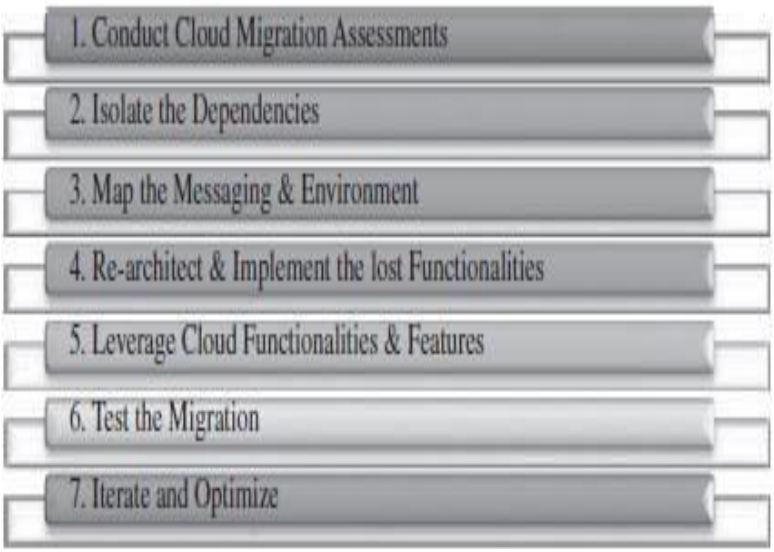
Apart from these costs, other factors that play a major role in the cloudonomics of migration are

- the licensing issues (for perhaps parts of the enterprise application)
- the SLA compliances, and the pricing of the cloud service offerings.

THE SEVEN-STEP MODEL OF MIGRATION INTO A CLOUD

Migration initiatives into the cloud are implemented in phases or in stages

- **Assessment**: Proof of concepts or prototypes for various approaches to the migration along with the leveraging of pricing parameters enables one to make appropriate assessments. These assessments are about the cost of migration as well as about the ROI that can be achieved in the case of production version
- **Isolation**: The next process step is in isolating all systemic and environmental dependencies of the enterprise application components within the captive data center. This, in turn, yields a picture of the level of complexity of the migration
- **Mapping**: Generating the mapping constructs between what shall possibly remain in the local captive data center and what goes onto the cloud.
- **Re-architect**: A substantial part of the enterprise application needs to be rearchitected, redesigned, and reimplemented on the cloud
- **Augment**: leverage the intrinsic features of the cloud computing service to augment our enterprise application in its own small ways
- **Test**: validate and test the new form of the enterprise application with an extensive test suite that comprises testing the components of the enterprise application on the cloud as well



- **Optimize**: These test results could be positive or mixed. In the latter case, we iterate and optimize as appropriate. After several such optimizing iterations, the migration is deemed successful.

Migration risks

Migration risks for migrating into the cloud fall under two broad categories:

The general migration risks :

- performance monitoring and tuning,
- the compliance with standards and governance issues; the IP and licensing issues;
- the quality of service (QoS) parameters as well as the corresponding SLAs committed to;
- the ownership, transfer, and storage of data in the application;
- the portability and interoperability issues which could help mitigate potential vendor lock-ins

The security-related migration risks :

- obtaining the right execution logs as well as retaining the rights to all audit trails at a detailed level
- matters of multi-tenancy and the impact of IT data leakage in the cloud computing environments

Integration as a service (IaaS)

Why Integration?

- Increasingly business applications are deployed in clouds to reap the business and technical benefits.
- On the other hand, there are still innumerable applications and data sources locally stationed and sustained primarily due to the security reason.
- The question here is how to create a seamless connectivity between those hosted and on-premise applications to empower them to work together.

How Integration is done?

- Integration as a service (IaaS) is the budding and distinctive capability of clouds in fulfilling the business integration requirements.
- IaaS overcomes these challenges by smartly utilizing the time-tested business-to-business (B2B) integration technology as the valueadded bridge between SaaS solutions and in-house business applications.

SaaS INTEGRATION

- Cloud-centric integration solutions are being developed and demonstrated for showcasing their capabilities for integrating enterprise and cloud applications.
- Now with the arrival and adoption of the transformative and disruptive paradigm of cloud computing, every ICT products are being converted into a collection of services to be delivered via the open Internet
- In that line, the standards-compliant integration suites are being transitioned into services so that any integration need of any one from any part of the world, can be easily, cheaply and rapidly met.

Integration as a Service (IaaS) : Migration of the functionality of a typical enterprise application integration (EAI) hub / enterprise service bus (ESB) into the cloud for providing for smooth data transport between any enterprise and SaaS applications.

- Users subscribe to IaaS as they would do for any other SaaS application.
- cloud middleware will be made available as a service.

For service integration, it is enterprise service bus (ESB) and for data integration, it is enterprise data bus (EDB).

There are Message oriented middleware (MOM) and message brokers for integrating decoupled applications through message passing and pick up.

Events are coming up fast and there are complex event processing (CEP) engines that receive a stream of diverse events from diverse sources, process them at real-time to extract and figure out the encapsulated knowledge, and accordingly select and activate one or more target applications.

- Cloud infrastructure is not very useful without SaaS applications that run on top of them, and SaaS applications are not very valuable without access to the critical corporate data that is typically locked away in various corporate systems.
- So, for cloud applications to offer maximum value to their users, they need to provide a simple mechanism to import or load external data, export or replicate their data for reporting or analysis purposes, and finally keep their data synchronized with on-premise applications.

Why SaaS Integration is hard?

Reasons:

Limited Access: Access to cloud resources (SaaS, PaaS, and the infrastructures) is more limited than local applications. Once applications move to the cloud, custom applications must be designed to support integration because there is no longer that low level of access. Enterprises putting their applications in the cloud or those subscribers of cloud-based business services are dependent on the vendor to provide the integration hooks and APIs.

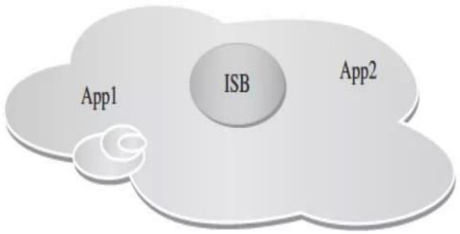
Dynamic Resources: Cloud resources are virtualized and service-oriented. That is, everything is expressed and exposed as a service. Due to the dynamism factor infrastructural changes are liable for dynamic changes. These would clearly impact the integration model.

Performance: Clouds support application scalability and resource elasticity. However the network distances between elements in the cloud are no longer under our control. Because of the round trip latency, the cloud integration performance is bound to slow down

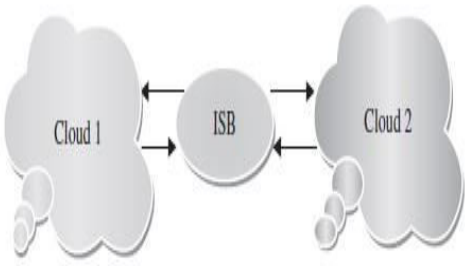
NEW INTEGRATION SCENARIOS

Three major integration scenarios

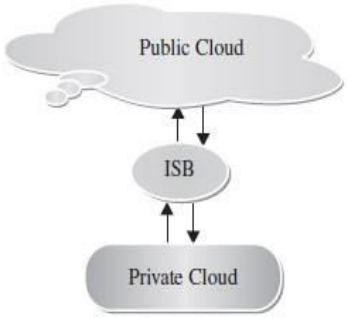
- **Within a Public Cloud**: Two different applications are hosted in a cloud. The role of the cloud integration middleware (say cloud-based ESB or internet service bus (ISB)) is to seamlessly enable these applications to talk to each other. These applications can be owned by two different companies. They may live in a single physical server but run on different virtual machines



- **Homogeneous Clouds:** The applications to be integrated are positioned in two geographically separated cloud infrastructures. The integration middleware can be in cloud 1 or 2 or in a separate cloud. There is a need for data and protocol transformation and they get done by the ISB



- **Heterogeneous Clouds :** One application is in public cloud and the other application is in private cloud



The Integration Methodologies

There are three types for cloud integration

- **Traditional Enterprise Integration Tools can be empowered with special connectors to access Cloud-located Applications**: With a persistent rise in the necessity towards accessing and integrating cloud applications, special drivers, connectors and adapters are being built and incorporated on the existing integration platforms to enable bidirectional connectivity with the participating cloud services.
- **Traditional Enterprise Integration Tools are hosted in the Cloud:** This approach is similar to the first option except that the integration software suite is now hosted in any third-party cloud infrastructures so that the enterprise does not worry about procuring and managing the hardware or installing the integration software
- **Integration-as-a-Service (IaaS) or On-Demand Integration Offerings:** These are SaaS

applications that are designed to deliver the integration service securely over the Internet and are able to integrate cloud applications with the on-premise systems, cloud-to-cloud applications.

Characteristics of Integration Solutions and Products

- **Connectivity** refers to the ability of the integration engine to engage with both the source and target systems using available native interfaces
- **Semantic Mediation** refers to the ability to account for the differences between application semantics between two or more systems. Semantics means how information gets understood, interpreted and represented within information systems
- **Data Mediation** converts data from a source data format into destination data format. Coupled with semantic mediation, data mediation or data transformation is the process of converting data from one native format on the source system, to another data format for the target system
- **Data Migration** is the process of transferring data between storage types, formats, or systems. Data migration means that the data in the old system is mapped to the new systems
- **Data Security** means the ability to insure that information extracted from the source systems has to securely be placed into target systems
- **Data Integrity** means data is complete and consistent. Thus, integrity has to be guaranteed when data is getting mapped and maintained during integration operations, such as data synchronization between on-premise and SaaS-based systems.
- **Governance** refers to the processes and technologies that surround a system or systems, which control how those systems are accessed and leveraged

Products And Platforms

- **Jitterbit** is a fully graphical integration solution that provides users a versatile platform and a suite of productivity tools to reduce the Integration efforts sharply.
  It can be used standalone or with existing EAI infrastructures, enabling users to create new projects or consume and modify existing ones offered by the open source community or service provider.
  The Jitterbit solution enables the cool integration among confidential and corporate data, enterprise applications, web services, XML data sources, legacy systems, simple and complex flat files.
  Jitterbit is comprised of two major components:
  - **Jitterbit Integration Environment**: An intuitive point-and-click graphical UI that enables to quickly configure, test, deploy and manage integration projects on the Jitterbit server.
  - **Jitterbit Integration Server** A powerful and scalable run-time engine that processes all the integration operations, fully configurable and manageable from the Jitterbit application.

- **Boomi Software**: Boomi AtomSphere is an integration service that is completely on-demand and connects any combination of SaaS, PaaS, cloud, and on-premise applications without the burden of installing and maintaining software packages or appliances. Anyone can securely build, deploy and manage simple to complex integration processes using only web browser.
- **Bungee Connect:** Bungee Connect enables cloud computing by offering an application development and deployment platform that enables highly interactive applications integrating multiple data sources and facilitating instant deployment. Built specifically for cloud development, Bungee Connect reduces the efforts to integrate (mashup) multiple web services into a single application.
- **OpSource Connect:** Expands on the OpSource Services Bus (OSB) by providing the infrastructure for two-way web services interactions, allowing customers to consume and publish applications across a common web services infrastructure.
- **SnapLogic**: SnapLogic is a capable, clean, and uncluttered solution for data integration that can be deployed in enterprise as well as in cloud landscapes. The free community edition can be used for the most common point-to-point data integration tasks

  The Pervasive DataCloud: Pervasive Data Cloud is the first multi-tenant platform for delivering the following

  - Integration as a Service (IaaS) for both hosted and on-premises applications and data sources
  - Packaged turnkey integration
  - Integration that supports every integration scenario
  - Connectivity to hundreds of different applications and data sources

Other Products

- Bluewolf
- Online MQ
- CloudMQ
- Linxter

Enterprise cloud computing

Enterprise cloud computing is the alignment of a cloud computing model with an organization's business objectives (profit, return on investment, reduction of operations costs) and processes.

Cloud computing is composed of five essential characteristics:

- on-demand self-service
- broad network access

- resource pooling,
- rapid elasticity
- measured service

The ways in which these characteristics are manifested in an enterprise context vary according to the deployment model employed.

Deployment Models for Enterprise Cloud Computing

- **Public clouds** are provided by a designated service provider for general public under a utility based pay-per-use consumption model. The cloud resources are hosted generally on the service provider's premises
- **Private clouds** are built, operated, and managed by an organization for its internal use only to support its business operations exclusively
- **Virtual private clouds** are a derivative of the private cloud deployment model but are further characterized by an isolated and secure segment of resources, created as an overlay on top of public cloud infrastructure using advanced network virtualization capabilities. Some of the public cloud vendors that offer this capability include Amazon Virtual Private
  Cloud, OpSource Cloud and Skytap Virtual Lab
- **Community clouds** are shared by several organizations and support a specific community that has shared concerns (e.g., mission, security requirements, policy, and compliance considerations). They may be managed by the organizations or a third party and may exist on premise or off premise . One example of this is OpenCirrus formed by HP, Intel, Yahoo, and others
- **Managed clouds** arise when the physical infrastructure is owned by and/or physically located in the organization's data centers with an extension of management and security control plane controlled by the managed service provider
- **Hybrid clouds** are a composition of two or more clouds (private, community,or public) that remain unique entities but are bound together by standardized or proprietary technology that enables data and application portability. Some examples of these offerings include Amazon Virtual Private Cloud, Skytap Virtual Lab, and CohesiveF T VPN-Cubed.

Adoption and Consumption Strategies

The selection of strategies for enterprise cloud computing is critical for IT capability as well as for the earnings and costs the organization experiences, motivating efforts toward convergence of business strategies and IT. Critical questions toward this convergence in the enterprise cloud paradigm

- Will an enterprise cloud strategy increase overall business value?

- Are the effort and risks associated with transitioning to an enterprise cloud strategy worth it?
- Which areas of business and IT capability should be considered for the enterprise cloud?
- Which cloud offerings are relevant for the purposes of an organization?
- How can the process of transitioning to an enterprise cloud strategy be piloted and systematically executed?

These questions are addressed from two strategic perspectives:

(1) Adoption (2) Consumption

Adoption strategy : an organization makes a decision to adopt a cloud computing model based on fundamental drivers for cloud computing—scalability, availability, cost and convenience

- **Scalability - Driven Strategy:** The objective is to support increasing workloads of the organization without investment and expenses exceeding returns. The conditions are that the effort, costs (CAPEX and OPEX) and time involved in accessing and installing IT capability on a CDC are less than going through a standard hardware and software procurement and licensing process
- **Availability - Driven Strategy:** Availability has close relations to scalability but is more concerned with the assurance that IT capabilities and functions are accessible, usable and acceptable by the standards of users. This is hence the objective of this basic enterprise cloud strategy.
- **Market-Driven Strategy**: This strategy is more attractive and viable for small, agile organizations that do not have (or wish to have) massive investments in their IT infrastructure. The objective here is to identify and acquire the "best deals" for IT capabilities as demand and supply change, enabling ongoing reductions in OPEX and CAPEX.
- **Convenience-Driven Strategy**: The objective is to reduce the load and need for dedicated system administrators and to make access to IT capabilities by users easier, regardless of their location and connectivity (e.g. over the Internet). The expectation is that the cost of obtaining IT capabilities from a CDC and making them accessible to users is significantly lower than the cost of having a dedicated administrator

Consumption Strategy:

The consumption strategies make a distinction between data and application logic because there are questions of programming models used, data sensitivity, software licensing and expected response times that need to be considered.

There are four consumptions strategies identified, where the differences in objectives, conditions and actions reflect the decision of an organization to trade-off hosting costs, controllability and resource elasticity of IT resources for software and data

- **Software Provision**. This strategy is relevant when the elasticity requirement is high for software and low for data, the controllability concerns are low for software and high for

data, and the cost reduction concerns for software are high, while cost reduction is not a priority for data, given the high controllability concerns for data, that is, data are highly sensitive

- **Storage Provision**. This strategy is relevant when the elasticity requirements is high for data and low for software, while the controllability of software is more critical than for data. This can be the case for data intensive applications, where the results from processing in the application are more critical and sensitive than the data itself. Furthermore, the cost reduction for data resources is a high concern, hereas cost for software, given its criticality, is not an issue for the organization within reasonable means.
- **Solution Provision**. This strategy is relevant when the elasticity and cost reduction requirements are high for software and data, but the controllability requirements can be entrusted to the CDC. It is not the case that controllability is an insignificant requirement; it is rather the case that the organization trusts the CDC sufficiently to manage access and usage control of its software and data
- **Redundancy Services**. This strategy can be considered as a hybrid enterprise cloud strategy, where the organization switches between traditional, software, storage or solution management based on changes in its operational conditions and business demands

The strategy is referred to as the "redundancy strategy" because the CDC is used for situations such as disaster recovery, fail-over and load balancing

Software, storage or solution services can be implemented using redundancy, such that users are redirected for the purpose of maintaining availability of functionality or performance/response times experienced by the user of the service.

UNIT – 3

Virtual Machines Provisioning and Migration Services

Introduction and Inspiration

Cloud computing builds on

- service-oriented architecture (SOA)
- grid computing and
- virtualization technology

Offers Infrastructure as a service(IaaS) to the end users as a public utility service based on pay-as-you-use and on-demand computing models

The provisioning of the cloud infrastructure in data centers is a prerequisite

The provisioning for systems and applications on a large number of physical machines is time-consuming process with low assurance on deployment's time and cost

Two core services enable the users to get the best out of the IaaS model in public and private cloud setups

- Virtual machine provisioning
- Migration services

When installing a new server for a certain workload to provide a service for a client, the following steps are required

- Check the inventory for a new machine
- Get one, format, install OS required, install services
- A server is needed along with lots of security batches and appliances

With the emergence of virtualization technology and the cloud computing IaaS model, the same task can be achieved in few minutes.
To provision a virtual server through a self-service interface with small steps to get what we desire with the required specifications

- Provisioning this machine in a public cloud like Amazon Elastic Compute Cloud (EC2)
- Or using a virtualization management software package
- Or a private cloud management solution installed at data center inside the Organization and within the private cloud setup

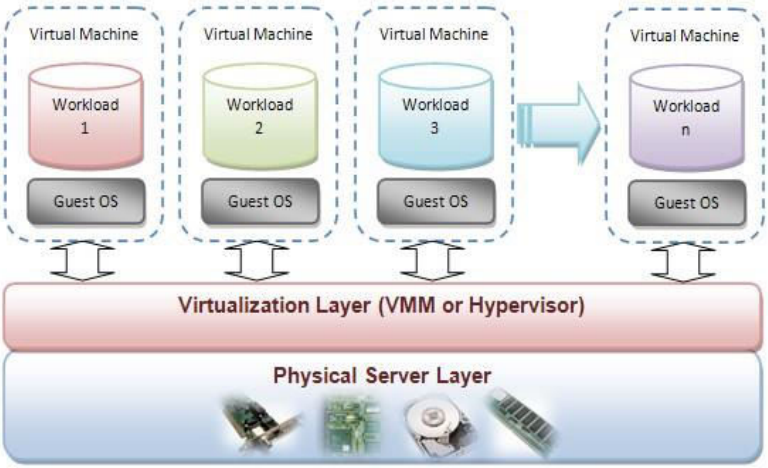
A need for performing a server's upgrade or performing maintenance tasks is an expensive operation to maintain or upgrade a main server that has lots of applications and users

With the advance of the revolutionized virtualization technology and migration services associated with hypervisors' capabilities these tasks (maintenance, upgrades, patches, etc.) need no time to accomplish

- Provisioning a new virtual machine is a matter of minutes
- Migrations of a virtual machine is a matter of milliseconds

Virtualization Technology Overview

- Virtualization facilitates the providing and management of the dynamic data center's infrastructure
- An essential and enabling technology of cloud computing environments
- Virtualization can be defined as the abstraction of the four computing resources : Storage, processing power, memory, and network or I/O
- Conceptually similar to emulation where a system pretends to be another system
- Virtualization is a system pretending to be two or more of the same system



- The virtualization layer will partition the physical resource of the underlying physical server into multiple virtual machines with different workloads
- It Schedules, allocates the physical resource, makes a virtual machine think it totally owns the whole underlying hardware's physical resource Processor, disks, RAMs, etc.
- Virtual machine's technology manages resources in cloud computing environments
- Improves the utilization of such resources by multiplexing many virtual machines on one physical host - Server consolidation
- Machines can be scaled up and down on demand with a high level of resources' abstraction
- Virtualization enables high, reliable, and agile deployment mechanisms and management of services, providing on-demand cloning and live migration services which improve reliability

Public Cloud and Infrastructure Services

Public cloud or external cloud

- Resources are dynamically provisioned via publicly accessible Web applications/Web services (SOAP or RESTful interfaces) from an off-site third-party provider
- Shares resources and bills on a fine-grained utility computing basis
- The user pays only for the capacity of the provisioned resources at a particular time
- Examples for vendors who publicly provide IaaS
  Amazon Elastic Compute Cloud (EC2), GoGrid, Joyent, Accelerator, Rackspace, AppNexus, FlexiScale and Manjrasoft Aneka

Public Cloud and Infrastructure Services

- **Amazon Elastic Compute Cloud (EC2) is an IaaS service**

▫ Provides elastic compute capacity in the cloud
▫ Leveraged via Web services (SOAP or REST), a Web-based AWS (Amazon Web Service) management console, or the EC2 command line tools
▫ Provides hundreds of pre-made AMIs (Amazon Machine Images) with a variety of operating systems and pre-loaded software i.e., Linux, OpenSolaris, or Windows

▫ Provides complete control of computing resources run on Amazon's computing and infrastructure environment easily

▫ Reduces the time required for obtaining and booting a new server's instances to minutes

▫ Allows a quick scalable capacity and resources, up and down as the computing requirements change

▫ Offers different instances' size according to

- The resources' needs (small, large, and extra large)
- The high CPU's needs it provides (medium and extra large high CPU instances)
- High-memory instances (extra large, double extra large, and quadruple extra large instance)

Private Cloud and Infrastructure Services

• A private cloud aims at providing public cloud functionality, but on private resources
▫ Maintaining control over an organization's data and resources to meet security and governance's requirements in an organization
▫ A highly virtualized cloud data center located inside the organization's firewall
▫ Also be a private space dedicated for the company within a cloud vendor's data center designed to handle the organization's workloads
Private clouds exhibit the following characteristics:
▫ Allow service provisioning and compute capability for an organization's users in a self-service manner
▫ Automate and provide well-managed virtualized environments
▫ Optimize computing resources, and servers' utilization ▫ Support specific workloads

Examples for vendors and frameworks that provide Iaas in private setups
▫ Eucalyptus (elastic utility computing architecture linking your programs to useful systems)
▫ Open Nebula

A third type of cloud setup named **Hybrid cloud**
▫ A combination of private/internal and external cloud resources existing together by enabling outsourcing of noncritical services and functions in public cloud and keeping the critical ones internal

Main function of Hybrid cloud is to release resources from a public cloud and handle sudden demand usage called cloud bursting

Distributed Management of Virtualization

- Virtualization needs powerful management capabilities
- Many commercial, open source products and research projects are being developed to dynamically provision virtual machines e.g., Open Nebula , IBM Virtualization Manager, Joyent utilizing the physical infrastructure
- Some commercial and scientific infrastructure cloud computing initiatives provide remote interfaces for controlling and monitoring virtual resources e.g., Globus VWS, Eucalyptus and Amazon
- The **RESERVOIR** initiative provides Grid interfaces and protocols that enable the required interoperability between the clouds or infrastructure's providers

High Availability

- A system design protocol and an associated implementation ensures a certain absolute degree of operational continuity during a given measurement period
- Availability refers to the ability of a user's community to access the system
  - Submitting new work, updating or altering existing work, or collecting the results of the previous work
- Unavailable: A user cannot access the system

Services should be available all the time along with some planned/unplanned downtime according to a certain SLA

SLA formalizes the service availability objectives and requirements The monthly availability or downtime of a service, to calculate the service's credits to match the billing cycles

- Business critical services are often categorized as high availability services achieving the lowest possible amount of planned and unplanned downtime
- High availability allows virtual machines to automatically be restarted in case of an underlying hardware failure or individual VM failure.
- If one of servers fails, the VMs will be restarted on other virtualized servers in the resource pool restoring the essential services with minimal service interruption

Cloud and Virtualization Standardization Efforts

Standardization is important to ensure interoperability

The prevalent standards that make cloud computing and virtualization possible

- **Distributed Management Task Force (DMTF)** have produced standards for almost all the aspects of virtualization technology. DMTF initiated the VMAN (Virtualization Management) Initiative. It delivers broadly supported interoperability and portability standards for managing the virtual computing lifecycle
- **VMAN's OVF (Open Virtualization Format)** is a collaboration between industry key players Dell, HP, IBM, Microsoft, XenSource, and VMware. OVF provides a common format to package and securely distribute virtual appliances across multiple virtualization platforms. VMAN profiles define a consistent way of managing a heterogeneous virtualized environment

OCCI and OGF

Another standardization effort has been initiated by **Open Grid Forum (OGF)** to deliver a standard API for cloud IaaS

Open Cloud Computing Interface Working Group (OCCI-WG)

- Dedicated for delivering an API specification for the remote management of cloud computing's infrastructure for allowing the development of interoperable tools for common tasks including deployment, autonomic scaling, and monitoring.
- Covering a high-level functionality required for managing the lifecycle of virtual machines/workloads, running on virtualization technologies/containers and supporting service elasticity

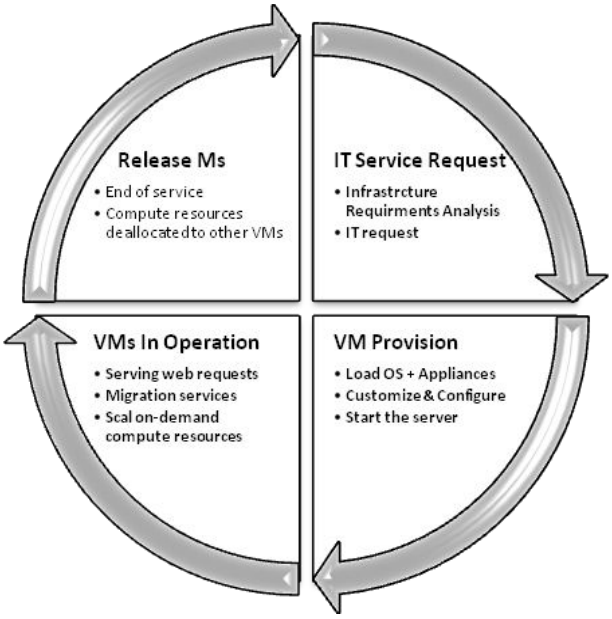
The new API for interfacing IaaS cloud computing facilities will allow

- **Consumers** to interact with cloud computing infrastructure on an ad hoc basis
- **Integrators** to offer advanced management services
- **Aggregators** to offer a single common interface to multiple providers
- **Providers** to offer a standard interface that is compatible with the available tools
- **Vendors** of grids/clouds to offer standard interfaces for dynamically scalable service's delivery in their products

Virtual Machines Provisioning and Manageability

Life cycle of VM and its major possible states of operation

- Starts by a request delivered to the IT department stating the requirement for creating a new server for a particular service – Request is processed by the IT administration by seeing the servers' resource pool, matching these resources with the requirements, starting the provision of the needed virtual machine
- Once it is provisioned and started it is ready to provide the required service according to an SLA
- A time period after which the VM is released and resources freed



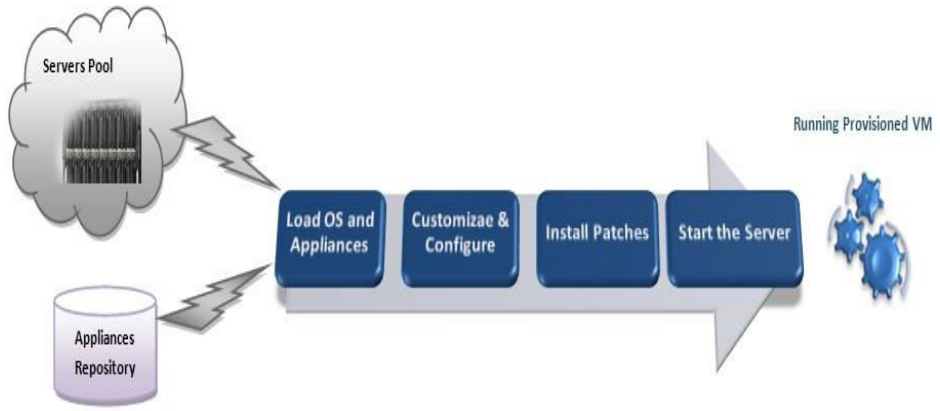
VM Provisioning Process

The common and normal steps of provisioning a virtual server

- Select a server from a pool of available servers (Physical servers with enough capacity) along with the appropriate OS template
- Load the appropriate software operating system, device drivers, middleware, and the needed applications for the service required

- Customize and configure the machine to configure an associated network and storage resources e.g., IP address, Gateway
- The virtual server is ready to start with its newly loaded software
- Server provisioning is defining server's configuration based on the organization requirements, a hardware, and software component, processor, RAM, storage, networking, operating system, applications, etc.
- Virtual machines can be provisioned by manually installing an operating system, by using a preconfigured VM template, by cloning an existing VM, or by importing a physical server or a virtual server from another hosting platform

VM Provisioning Process



Virtual Machine Migration Services

- Migration service is the process of moving a virtual machine from one host server or storage location to another
- Different techniques of VM migration
  - Hot/life migration
  - cold/regular migration
- Live storage migration of a virtual machine
- In this process, all key machines' components, are completely virtualized e.g., CPU, storage disks, networking, memory facilitating the entire state of a virtual machine to be captured by a set of easily moved data files

Migrations Techniques

• Live migration also called hot or real-time migration: The movement of a virtual machine from one physical host to another while being powered on without any noticeable effect from the end user's point of view (a matter of milliseconds)

Live Migration

- Facilitates proactive maintenance upon failure; the potential problem can be resolved before the disruption of service occurs
- Used for load balancing. Work is shared among computers optimize the utilization of available CPU resources

Live migration's mechanism

- How memory and virtual machine states are being transferred through the network from one host A to another host B
- e.g., the Xen hypervisor
- The process has been viewed as a transactional interaction between the two hosts involved

Stage 0: Pre-Migration

• An active virtual machine exists on the physical host A

Stage 1: Reservation

- A request is issued to migrate an OS from host A to B
- The necessary resources exist on B and on a VM container of that size

Stage 2: Iterative Pre-Copy

- During the first iteration, all pages are transferred from A to B
- Subsequent iterations copy only those pages dirtied during the previous transfer phase

Stage 3: Stop-and-Copy

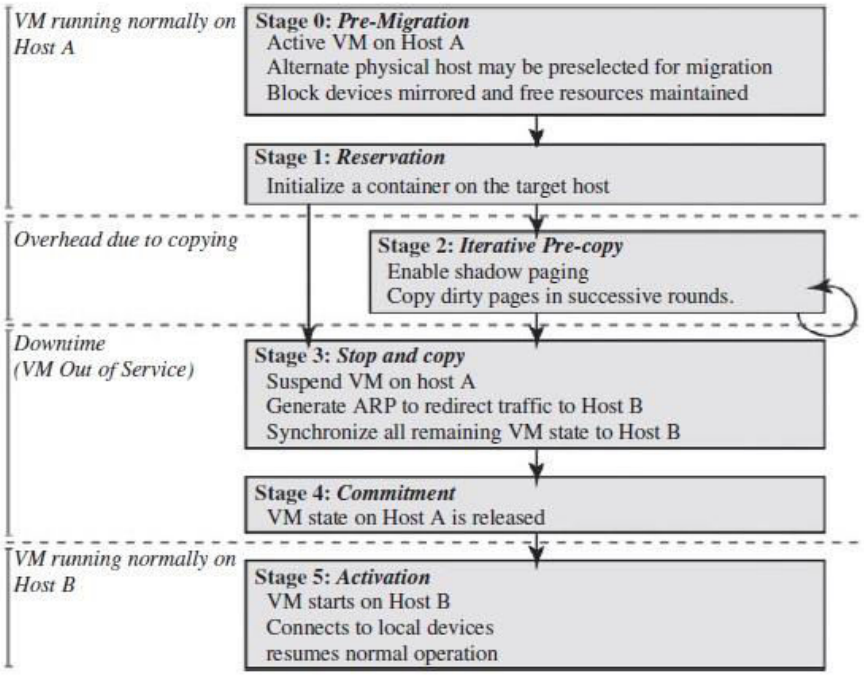
- Running OS instance at A is suspended
- The network traffic is redirected to B
- CPU state and any remaining inconsistent memory pages are then transferred
- At the end of this stage, there is a consistent suspended copy of the VM at both A and B.
- Copy at A is considered primary and is resumed in case of failure

Stage 4: Commitment

- Host B indicates to A that it has successfully received a consistent OS image
- Host A acknowledges this message as a commitment of the migration transaction
- Host A may now discard the original VM
- Host B becomes the primary host

Stage 5: Activation

- The migrated VM on B is now activated
- Post-migration code runs to reattach the device's drivers to the new machine and advertise moved IP addresses



This approach to failure management ensures

- At least one host has a consistent VM image at all times during migration
- The original host remains stable until the migration commits and the VM may be suspended and resumed on that host with no risk of failure

A migration request essentially attempts to move the VM to a new host on any sort of failure, execution is resumed locally aborting the migration

VM Management and Provisioning tools

- Provide the live migration of VM facility
- e.g., VMware VMotion and Citrix XenServer XenMotion

• VMware Vmotion

- Allows users to automatically optimize and allocate an entire pool of resources for maximum hardware utilization, flexibility, and availability
- To perform hardware's maintenance without scheduled downtime along with migrating virtual machines away from failing or underperforming servers

• Citrix XenServer XenMotion

- Inherited from the Xen live migrate utility
- Provides the IT administrator with the facility to move a running VM from one Xen Server to another in the same pool without interrupting the service, making it a highly available service
- A good feature to balance the workloads on the virtualized environment

Cold migration:

- The migration of a powered-off virtual machine
- Associated disks can be moved from one data store to another
- The virtual machines are not required to be on a shared storage

Differences between **Hot**(Live) migration and **Cold** migration

Live migration needs a shared storage for virtual machines in the server's pool, but cold migration does not

In live migration for a virtual machine between two hosts, there would be certain CPU compatibility checks to be applied, in cold migration this checks do not apply

The cold migration process is simple

- The configuration files are moved from the source host to the destination host's associated storage area including the NVRAM file (BIOS settings), log files, as well as the disks of the virtual machine
- The virtual machine is registered with the new host
- After the migration is completed, the old version of the virtual machine is deleted from the source host

• **Live Storage Migration of Virtual Machine**

– Moving the virtual disks or configuration file of a running virtual machine to a new data store without any interruption in the availability of the virtual machine's service

PROVISIONING IN THE CLOUD CONTEXT

- Amazon EC2 is a widely known example for vendors that provide public cloud services. Also, Eucalyptus and Open-Nebula are two complementary and enabling technologies for open source cloud tools, which play an invaluable role in infrastructure as a service and in building private, public, and hybrid cloud architecture.
- The Amazon EC2 (Elastic Compute Cloud) is a Web service that allows users to provision new machines into Amazon's virtualized infrastructure in a matter of minutes; using a publicly available API
- EC2 instance is typically a virtual machine with a certain amount of RAM, CPU, and storage capacity.

Amazon EC2 provides its customers with three flexible purchasing models to make it easy for the cost optimization:

- **On-Demand instances**: which allow you to pay a fixed rate by the hour with no commitment.
- **Reserved instances**: which allow you to pay a low, one-time fee and in turn receive a significant discount on the hourly usage charge for that instance. It ensures that any reserved instance you launch is guaranteed to succeed (provided that you have booked them in advance). This means that users of these instances should not be affected by any transient limitations in EC2 capacity.
- **Spot instances**: which enable you to bid whatever price you want for instance capacity, providing for even greater savings, if your applications have flexible start and end times.

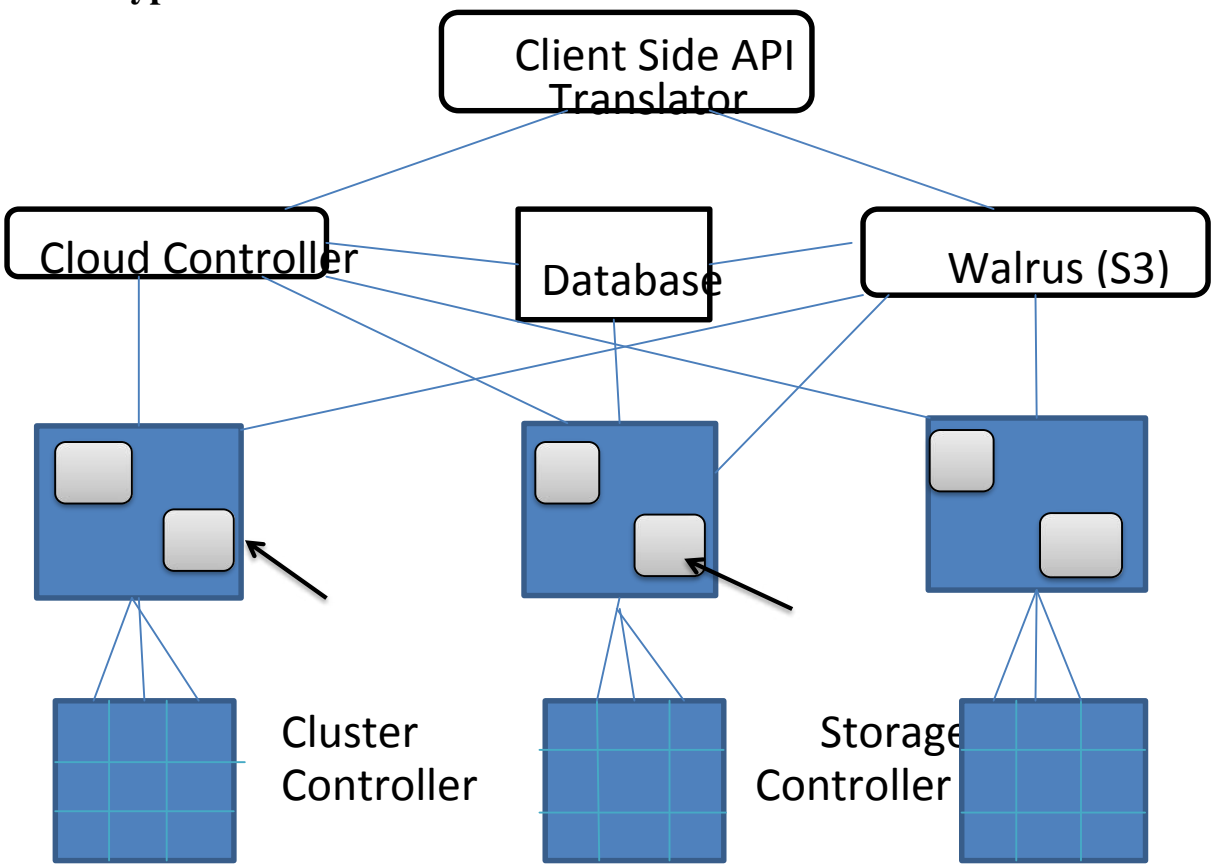
Amazon **Elastic Load Balancer** is another service that helps in building faulttolerant applications by automatically provisioning incoming application workload across available Amazon EC2 instances and in multiple availability zones.

Eucalyptus is an open-source infrastructure for the implementation of cloud computing on computer clusters. It is considered one of the earliest tools developed as a surge computing (in which data center's private cloud could augment its ability to handle workload's spikes by a design that allows it to send overflow work to a public cloud) tool. Its name is an acronym for "**elastic utility computing architecture for linking your programs to useful systems."**

Eucalyptus features :

- Interface compatibility with EC2, and S3 (both Web service and Query/REST interfaces).
- Simple installation and deployment.
- Support for most Linux distributions (source and binary packages).
- Support for running VMs that run atop the Xen hypervisor or KVM.
- Support for other kinds of VMs, such as VMware, is targeted for future releases.
- Secure internal communication using SOAP with WS security.
- Cloud administrator's tool for system's management and user's accounting.
- The ability to configure multiple clusters each with private internal network addresses into a single cloud.
- Eucalyptus aims at fostering the research in models for service's provisioning, scheduling, SLA formulation, and hypervisors' portability.

Eucalyptus Architecture



- **Node controller (NC)** controls the execution, inspection, and termination of VM instances on the host where it runs.
- **Cluster controller (CC)** gathers information about and schedules VM execution on specific node controllers, as well as manages virtual instance network.
- **Storage controller (SC)** is a put/get storage service that implements Amazon's S3 interface and provides a way for storing and accessing VM images and user data.
- **Cloud controller (CLC)** is the entry point into the cloud for users and administrators. It queries node managers for information about resources, makes high-level scheduling decisions, and implements them by making requests to cluster controllers.
- **Walrus (W)** is the controller component that manages access to the storage services within Eucalyptus. Requests are communicated to Walrus using the SOAP or REST-based interface

Management of VMs for Cloud Infrastructures

IaaS cloud providers share five characteristics:

- they provide on-demand provisioning of computational resources
- they use virtualization technologies to lease these resources
- they provide public and simple remote interfaces to manage those resources
- they use a pay-as-you-go cost model, typically charging by the hour and
- 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.

Virtualization technologies have been the key enabler of many of these salient characteristics of IaaS clouds by giving providers a more flexible and generic way of managing their resources.

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

Virtual machines require a fair amount of configuration, including preparation of the machine's software environment and network configuration and 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 user.

Several VI management solutions have emerged over time, such as platform ISF and VMware VSphere

Virtual Machine Management Activity of RESERVOIR addresses three problems

- Distributed management of virtual machines
- Reservation-based provisioning of virtualized resource and
- Provisioning to meet SLA commitments

Distributed Management of Virtual Machines

To efficiently schedule resources, VI managers must be able to support flexible and complex scheduling policies and must leverage the ability of VMs to suspend, resume, and migrate.

- **Reservation-Based Provisioning of Virtualized Resources** Provisioning of virtualized resources beforehand turns out not to be 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 deployed the services. So, we have cloud providers, service owners and service users.

Service owners will enter into SLAs with their end users, covering guarantees such as the timeliness.

Cloud providers are not directly exposed to the service semantics or the SLAs that service owners may contract with their end users.

The cloud provider's task is to make sure that resource allocation requests are satisfied with specific probability and timeliness.
These 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.

DISTRIBUTED MANAGEMENT OF VIRTUAL INFRASTRUCTURES

OpenNebula – an open source virtual infrastructure engine - 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

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

VM Model and Life Cycle

The life cycle of a VM within Open Nebula follows several stages:

- **Resource Selection**. Once a VM is requested to Open Nebula, a feasible placement plan for the VM must be made. Open Nebula'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
- **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.
- **VM Creation**. The VM is booted by the resource hypervisor.
- **VM Migration**. The VM potentially gets migrated to a more suitable resource (e.g., to optimize the power consumption of the physical resources).
- **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

VM Management

Open Nebula manages a VMs life cycle by combining three different management areas:

- **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;
- **Image management** by transferring the VM images from an image repository to the selected resource and by creating on-the-fly temporary images; and
- **Networking** by creating local area networks (LAN) to interconnect the VMs and tracking the MAC addresses leased in each network.

Virtualization : Open Nebula 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. By decoupling the Open Nebula core from the virtualization technologies through the use of a driver-based architecture, adding support for additional virtual machine managers only requires writing a driver for it.

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.

Open Nebula uses the following concepts for its image management model

- **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 Open Nebula 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:
  - **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.

    Open Nebula 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 as targeting for cloning or not.
  - **Save/remove**. If the save qualifier is disabled, once the VM has been shut down, 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.

SCHEDULING TECHNIQUES FOR ADVANCE RESERVATION OF CAPACITY

Advance reservations lead to utilization problems, caused by the need to vacate resources before a reservation can begin.

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.

While preemption can be accomplished trivially 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.

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 their ability to be suspended, potentially migrated, and 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 was created to develop a scheduler that can efficiently support advance reservations efficiently by using 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.
- **Leasing Model:** Lease is defined 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

- 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.

Capacity Management To Meet SLA Commitments

IaaS providers perform two complementary management tasks:

- capacity planning to make sure that SLA obligations are met as contracted with the service providers and
- continuous optimization of resource utilization given specific workload to make the most efficient use of the existing capacity

In an IaaS model it is expected from the service provider that it sizes capacity demands for its service. If resource demands are provided correctly and are indeed satisfied upon request, then desired user experience of the service will be guaranteed.
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.

There are three main approaches

- **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
- **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

  Elasticity rules 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 yougo 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 observable in the black box mode. These auto-scaling policies are useful for transparently scaling and de-scaling services..
- **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.

Cluster as a Service(CaaS)

RVWS Design

- While Web services have simplified resource access and management, it is not possible to know if the resource(s) behind the Web service is (are) ready for requests. Clients need to exchange numerous messages with required Web services to learn the current activity of resources and thus face significant overhead loss if most of the Web services prove ineffective
- clients still have to locate the services themselves.
- Finally, the Web services have to be stateful so that they are able to best reflect the current state of their resources.

This was the motivation for creating the **RVWS** framework (Resources Via Web Services)

RVWS combines dynamic attributes, stateful Web services (aware of their past activity), stateful and dynamic WSDL documents, and brokering into a single, effective, service-based framework

Dynamic Attribute Exposure

There are two categories of dynamic attributes addressed in the RVWS framework

- State
- Characteristic

State attributes cover the current activity of the service and its resources, thus indicating readiness.

Example: A Web service that exposes a cluster (itself a complex resource) would most likely have a dynamic state attribute that indicates how many nodes in the cluster are busy and how many are idle.

Characteristic attributes cover the operational features of the service, the resources behind it, the quality of service (QoS), price and provider information.

Example: A possible characteristic is an array of support software within the cluster.

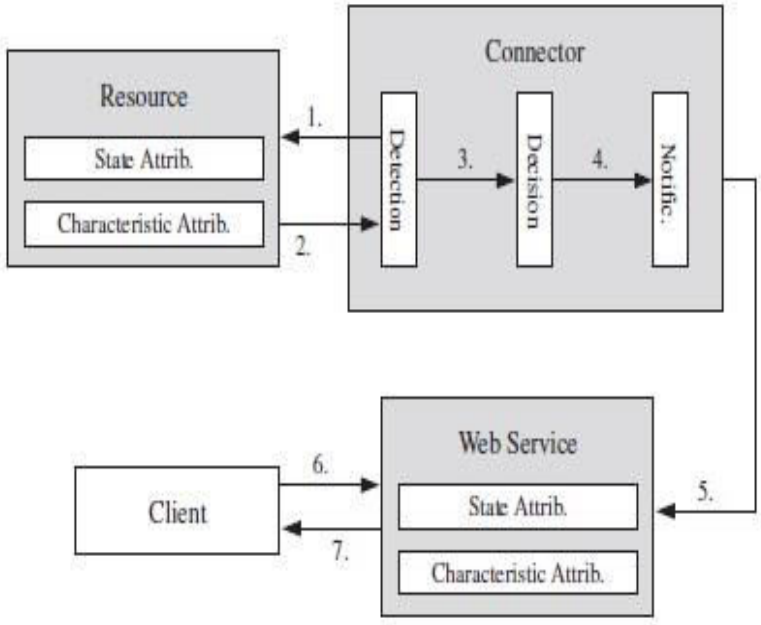


FIGURE 7.1. Exposing resource attributes.

To keep the stateful Web service current, a Connector is used to detect changes in resources and then inform the Web service.

The Connector has three logical modules:

- Detection
- Decision
- Notification.

The Detection module routinely queries the resource for attribute information (1-2). Any changes in the attributes are passed to the Decision module (3) that decides if the attribute change is large enough to warrant a notification. This prevents excessive communication with the Web service. Updated attributes are passed on to the Notification module (4), which informs the stateful Web service (5) that updates its internal state. When clients requests the stateful WSDL document (6), the Web service returns the WSDL document with the values of all attributes (7) at the request time.

Stateful WSDL Document Creation

- When exposing the dynamic attributes of resources, the RVWS framework allows Web services to expose the dynamic attributes through the WSDL documents of Web services.

- The Web Service Description Language (WSDL) governs a schema that describes a Web service and a document written in the schema
- All information of service resources is kept in a new WSDL section called Resources.
- For each resource behind the Web service, a ResourceInfo section exists.
- Each ResourceInfo section has a resource-id attribute and two child sections: state and characteristic.
- All resources behind the Web service have unique identifiers. When the Connector learns of the resource for the first time, it publishes the resource to the Web service.

```
<definitions xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
  <resources>
      <resource-info identifier="resourceID">
          <state>
              <description name="" attribute1="value1" ...
                           attributen="valuen">
                  ...Other description Elements...
              </description>
              ...Other description Elements...
          </state>
          <characteristics>
              <description name="" />
              ...Other description Elements...
          </characteristics>
      </resource-info>
      ...Other resource-info elements
  </resources>
  <types>...</types>
   message name="MethodSoapIn">...</message>
  <message name="MethodSoapOut">...</message>
  <portType name="CounterServiceSoap">...</portType>
  <binding name="CounterServiceSoap"
           type="tns:CounterServiceSoap">...</wsdl:binding>
  <wsdl:service name="CounterService">...</wsdl:service>
</wsdl:definitions>
```

FIGURE 7.2. New WSDL section.

Publication in RVWS

To help ease the publication and discovery of required services with stateful WSDL documents, a Dynamic Broker was proposed.
The goal of the Dynamic Broker is to provide an effective publication and discovery service based on service, resource, and provider dynamic attributes.

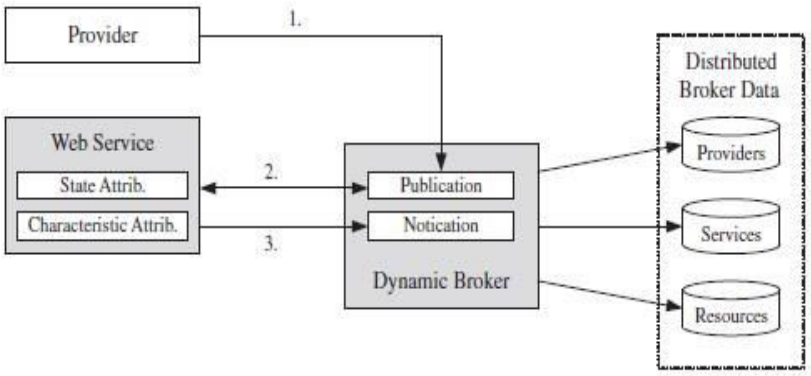


FIGURE 7.3. Publication.

When publishing to the Broker (1), the provider sends attributes of the Web service to the Dynamic Broker.

The dynamic attributes indicate the functionality, cost, QoS, and any other attributes the provider wishes to have published about the service. Furthermore, the provider is able to publish information about itself, such as the provider's contact details and reputation.

After publication (1), the Broker gets the stateful WSDL document from the Web service (2).

After getting the stateful WSDL document, the Dynamic Broker extracts all resource dynamic attributes from the stateful WSDL documents and stores the resource attributes in the resources store.

The Dynamic Broker then stores the (stateless) WSDL document and service attributes from (1) in the service store. Finally, all attributes about the provider are placed in the providers store.

As the Web service changes, it is able to send a notification to the Broker (3) which then updates the relevant attribute in the relevant store.

Automatic Discovery and Selection

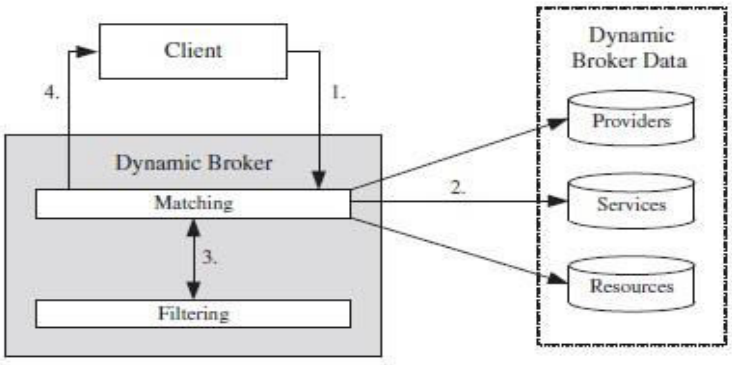


FIGURE 7.4. Matching parameters to attributes.

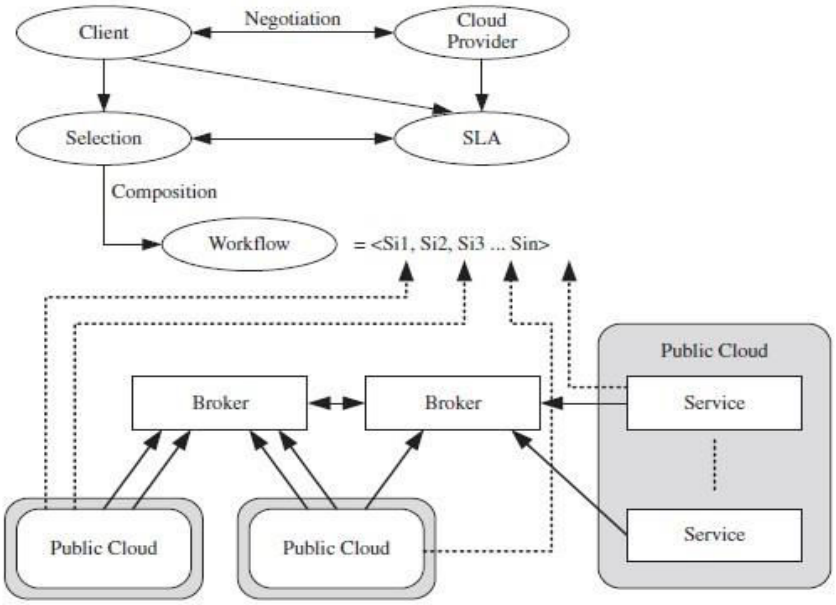


FIGURE 7.5. Dynamic discovery and selection.

When discovering services, the client submits to the Dynamic Broker three groups of requirements

- service
- resource
- provider.

The Dynamic Broker compares each requirement group on the related data store (2). Then, after getting matches, the Broker applies filtering (3). As the client using the Broker could vary from human operators to other software units, the resulting matches have to be filtered to suit the client. Finally, the filtered results are returned to the client (4).

Cluster as a Service

The purpose of the CaaS Technology is to ease the publication, discovery, selection, and use of existing computational clusters.

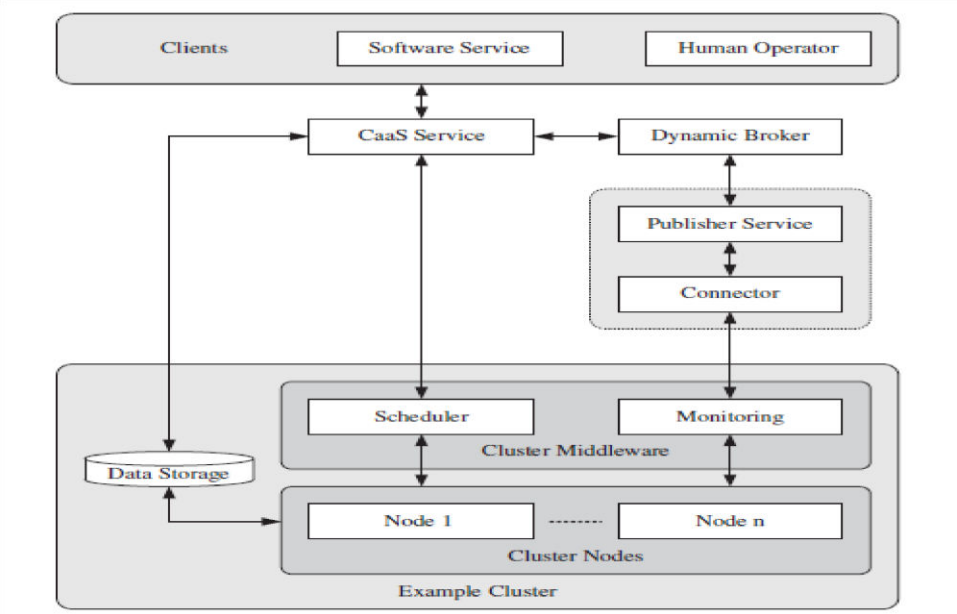
A typical cluster is comprised of three elements:

- Nodes
- Data storage and
- Middleware.

The middleware virtualizes the cluster into a single system image; thus resources such as the CPU can be used without knowing the organization of the cluster.

- **Scheduler** : manages the allocation of jobs to nodes

- **Monitor** : monitors the activity of the cluster
- **Publisher Web service** was to expose the dynamic attributes of the cluster via the stateful WSDL document.



- Furthermore, the Publisher service is published to the Dynamic Broker so clients can easily discover the cluster.
- To find clusters, the CaaS Service makes use of the **Dynamic Broker.**

The role of the CaaS Service is to

- provide easy and intuitive file transfer tools so clients can upload jobs and download results and
- offer an easy to use interface for clients to monitor their jobs.

The CaaS Service communicates with the cluster's scheduler, thus freeing the client from needing to know how the scheduler is invoked when submitting and monitoring jobs.

CaaS Service Design

The CaaS service can be described as having four main tasks:

- cluster discovery and selection
- result organization
- job management and
- file management.

Each module in the CaaS Service encapsulates one of the tasks and is able to communicate with other modules to extend its functionality.

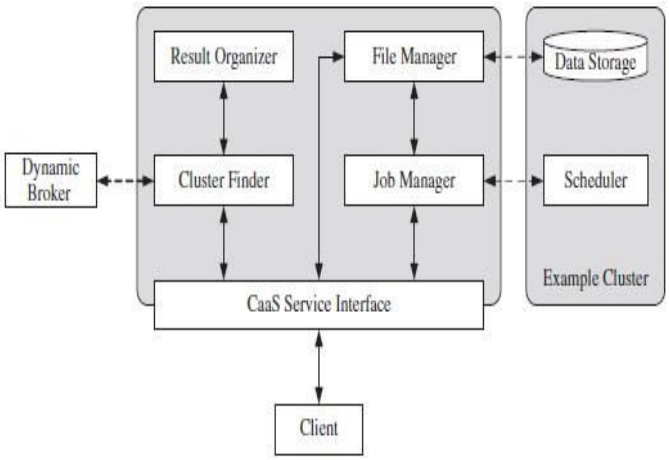


FIGURE 7.8. CaaS Service design.

- The modules inside the CaaS Web service are only accessed through an interface
- Invoking an operation on the CaaS Service Interface (discovery, etc.) invokes operations on various modules.

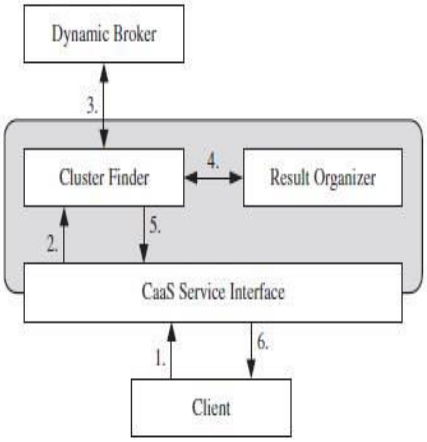


FIGURE 7.9. Cluster discovery.

Before a client uses a cluster, a cluster must be discovered and selected first.

- clients submit cluster requirements in the form of attribute values to the CaaS Service Interface. (The requirements range from the number of nodes in the cluster to the installed software (both operating systems and software APIs).
- The CaaS Service Interface invokes the Cluster Finder module
- Cluster Finder module communicates with the Dynamic Broker (and returns service matches (if any).
- To address the detailed results from the Broker, the Cluster Finder module invokes the Results Organizer module
- Results Organizer module takes the Broker results and returns an organized version that is returned to the client. The organized results instruct the client what clusters satisfy the specified requirements.

- After reviewing the results, the client chooses a cluster.

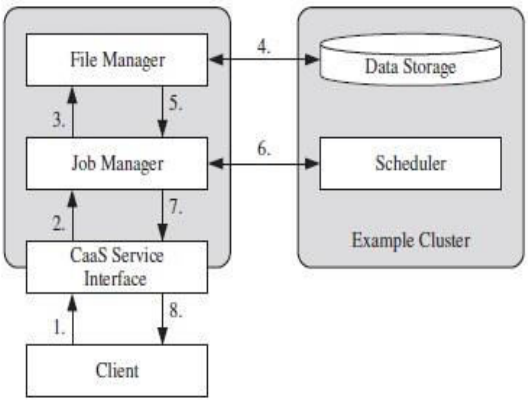


FIGURE 7.10. Job submission.

- All required data, parameters, such as estimated runtime, are uploaded to the CaaS Service (1).
- Once the file upload is complete, the Job Manager is invoked (2).
- It resolves the transfer of all files to the cluster by invoking the File Manager (3) that makes a connection to the cluster storage and commences the transfer of all files (4).
- Upon completion of the transfer, the outcome is reported back to the Job Manager (5).
- On failure, a report is sent and the client can decide on the appropriate action to take. If the file transfer was successful, the Job Manager invokes the scheduler on the cluster (6).
- If the outcome of the scheduler (6) is successful, the client is then informed (7-8). The outcome includes the response from the scheduler, the job identifier the scheduler gave to the job, and any other information the scheduler provides.

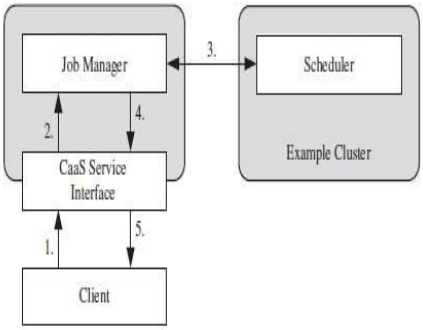


FIGURE 7.11. Job monitoring.

- During execution, clients should be able to view the execution progress of their jobs.
- The client contacts the CaaS service interface (1) that invokes the Job Manager module (2).

- No matter what the operation is (check, pause, or terminate), the Job Manager only has to communicate with the scheduler (3) and
- reports back a successful outcome to the client (4-5).

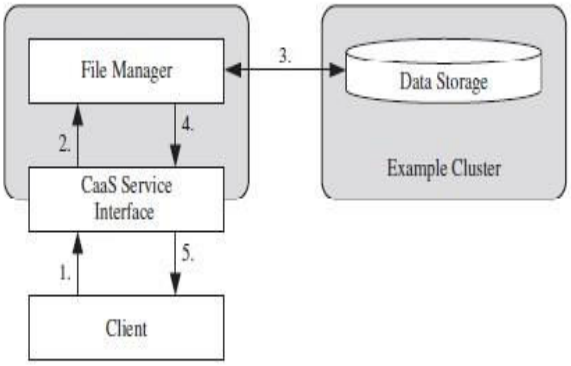


FIGURE 7.12. Job result collection.

Result Collection:

- Clients start the error or result file transfer by contacting the CaaS Service Interface (1) that then invokes the File Manager (2) to retrieve the files from the cluster's data storage (3).
- If there is a transfer error, the File Manager attempts to resolve the issue first before informing the client.
- If the transfer of files (3) is successful, the files are returned to the CaaS Service Interface (4) and then the client (5).
- When returning the files, URL link or a FTP address is provided so the client can retrieve the files.

Secure Distributed Data Storage in Cloud Computing
Moving From LANs to WANs

- Most designs of distributed storage take the form of either storage area networks (SANs) or network-attached storage (NAS) on the LAN level, such as the networks of an enterprise, a campus, or an organization.
- SANs are constructed on top of block-addressed storage units connected through dedicated high-speed networks.
- In contrast, NAS is implemented by attaching specialized file servers to a TCP/IP network and providing a file-based interface to client machine
- However, such a security system would not be robust enough to secure the data in distributed storage applications at the level of wide area networks, specifically in the cloud computing environment

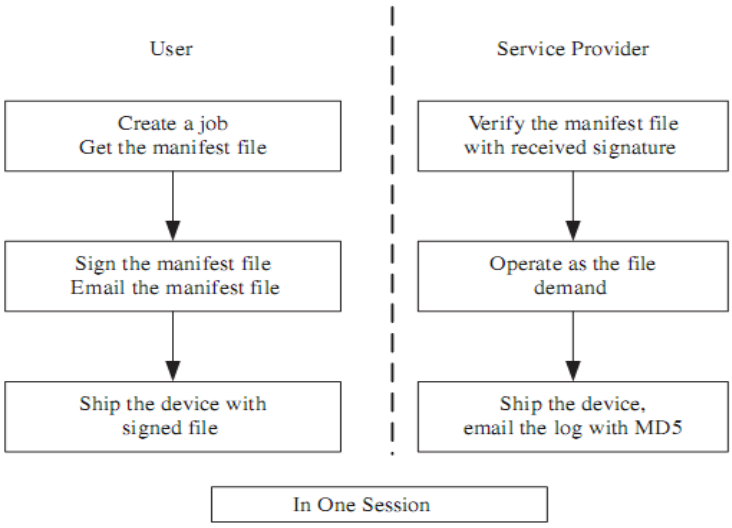
Existing Commercial Cloud Services

Data storage services on the platform of cloud computing are fundamentally provided by applications/software based on the Internet

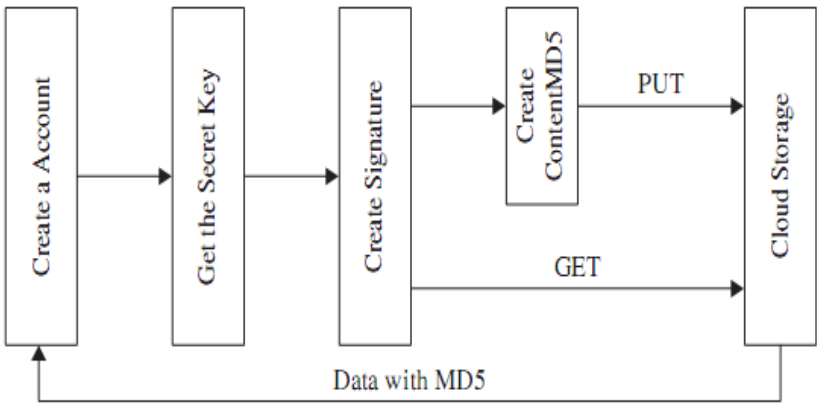
In normal network-based applications, user authentication, data confidentiality, and data integrity can be solved through IPSec proxy using encryption and digital signature. The key exchanging issues can be solved by SSL proxy.

These methods have been applied to today's cloud computing to secure the data on the cloud and also secure the communication of data to and from the cloud

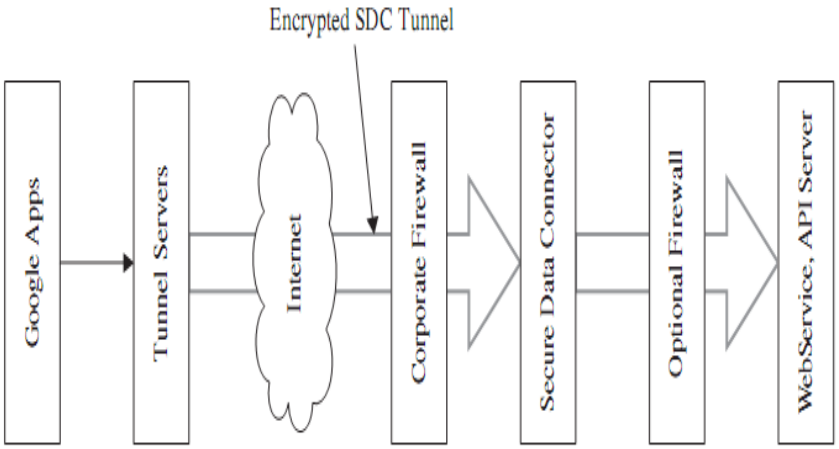
AWS Data Processing Procedure



Microsoft Windows Azure



Google App Engine



- The SDC constructs an encrypted connection between the data source and Google Apps. As long as the data source is in the Google Apps domain to the Google tunnel protocol servers, when the user wants to get the data, he/she will first send an authorized data requests to Google Apps, which forwards the request to the tunnel server.
- The tunnel servers validate the request identity. If the identity is valid, the tunnel protocol allows the SDC to set up a connection, authenticate, and encrypt the data that flows across the Internet. At the same time, the SDC uses resource rules to validate whether a user is authorized to access a specified resource.
- When the request is valid, the SDC performs a network request. The server validates the signed request, checks the credentials, and returns the data if the user is authorized
- From the perspective of cloud storage services, data integrity depends on the security of operations while in storage in addition to the security of the uploading and downloading sessions.
- The uploading session can only ensure that the data received by the cloud storage is the data that the user uploaded;
- The downloading session can guarantee the data that the user retrieved is the data cloud storage recorded.
- Unfortunately, this procedure applied on cloud storage services cannot guarantee data integrity
- First, assume that Alice, a company CFO, stores the company financial data at a cloud storage service provided by Eve.
- And then Bob, the company administration chairman, downloads the data from the cloud.

There are three important concerns in this simple procedure:

- **1. Confidentiality**. Eve is considered as an untrustworthy third party, Alice and Bob do not want reveal the data to Eve.
- **2.Integrity**. As the administrator of the storage service, Eve has the capability to play with the data in hand. How can Bob be confident that the data he fetched from Eve are the

same as what was sent by Alice? Are there any measures to guarantee that the data have not been tampered by Eve?

- **3**. **Repudiation**. If Bob finds that the data have been tampered with, is there any evidence for him to demonstrate that it is Eve who should be responsible for the fault? Similarly, Eve also needs certain evidence to prove her innocence.

Solutions for Missing Link

Third Authority Certified(TAC)

Secret Key Sharing(SKS)

Four Solutions

- Neither TAC nor SKS
- With SKS but without TAC
- With TAC but without SKS
- With Both TAC and SKS

Neither TAC nor SKS.

Uploading Session

1. *User*: Sends data to service provider with MD5 checksum and MD5 Signature by User (MSU).
2. *Service Provider*: Verifies the data with MD5 checksum, if it is valid, the service provider sends back the MD5 and MD5 Signature by Provider (MSP) to user.
3. MSU is stored at the user side, and MSP is stored at the service provider side.

Once the uploading operation finished, both sides agreed on the integrity of the uploaded data, and each side owns the MD5 checksum and MD5 signature generated by the opposite site.

Downloading Session

1. *User*: Sends request to service provider with authentication code.
2. *Service Provider*: Verifies the request identity, if it is valid, the service provider sends back the data with MD5 checksum and MD5 Signature by Provider (MSP) to user.
3. User verifies the data using the MD5 checksum.

With SKS but without TAC.

Uploading Session

1. *User*: Sends data to service provider with MD checksum 5.
2. *Service Provider*: Verifies the data with MD5 checksum, if it is valid, the service provider sends back the MD5 checksum.
3. The service provider and the user share the MD5 checksum with SKS.

Then, both sides agree on the integrity of the uploaded data, and they share the agreed MD5 checksum, which is used when disputation happens.

Downloading Session

1. *User*: Sends request to the service provider with authentication code.
2. *Service Provider*: Verifies the request identity, if it is valid, the service provider sends back the data with MD5 checksum.
3. User verifies the data through the MD5 checksum.

When disputation happens, the user or the service provider can take the shared MD5 together, recover it, and prove his/her innocence.

With Both TAC and SKS.

Uploading Session

1. *User*: Sends data to the service provider with MD5 checksum.
2. *Service Provider*: verifies the data with MD5 checksum.
3. Both the user and the service provider send MD5 checksum to TAC.
4. TAC verifies the two MD5 checksum values. If they match, the TAC distributes MD5 to the user and the service provider by SKS.

Both sides agree on the integrity of the uploaded data and share the same MD5 checksum by SKS, and the TAC own their agreed MD5 signatures.

Downloading Session

1. *User*: Sends request to the service provider with authentication code.
2. *Service Provider*: Verifies the request identity, if it is valid, the service provider sends back the data with MD5 checksum.
3. User verifies the data through the MD5 checksum.

TECHNOLOGIES FOR DATA SECURITY IN CLOUD COMPUTING

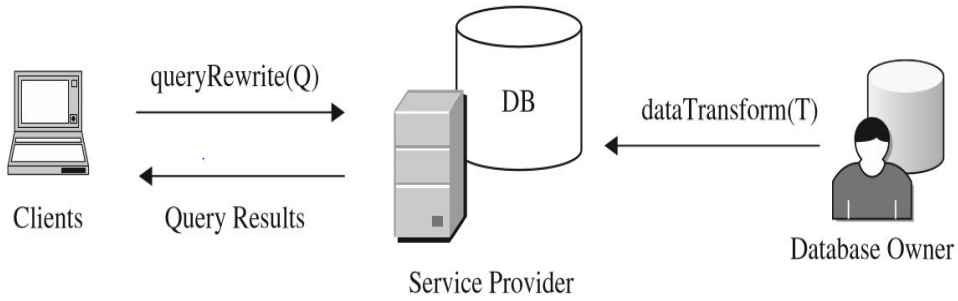
Unique issues of the cloud data storage platform from a few different perspectives

- **Database Outsourcing and Query Integrity Assurance**
  - Storing data into and fetching data from devices and machines behind a cloud are essentially a novel form of database outsourcing
- **Data Integrity in Untrustworthy Storage**
  - The fear of losing data or data corruption
  - Relieve the users' fear by providing technologies that enable users to check the integrity of their data

- **Web-Application-Based Security**
  - Once the dataset is stored remotely, a Web browser is one of the most convenient approaches that end users can use to access their data on remote services
  - Web security plays a more important role for cloud computing
- **Multimedia Data Security**
  - With the development of high-speed network technologies and large bandwidth connections, more and more multimedia data are being stored and shared in cyber space
  - The security requirements for video, audio, pictures, or images are different from other applications

Database Outsourcing and Query Integrity Assurance

- Database outsourcing has become an important component of cloud computing as
  - The cost of transmitting a terabyte of data over long distances has decreased significantly
  - The total cost of data management is five to ten times higher than the initial acquisition costs
  - A growing interest in outsourcing database management tasks to third parties can provide these tasks for a much lower cost due to the economy of scale
  - The benefits of reducing the costs for running Database Management Systems (DBMS) independently enabling enterprises to concentrate on their main businesses
- The general architecture of a database outsourcing environment with clients
  - The database owner outsources its data management tasks
  - Clients send queries to the untrusted service provider
    - The data is preprocessed, encrypted, and stored at the service provider
  - For evaluating queries, a user rewrites a set of queries against the data to queries against the encrypted database



- The outsourcing of databases to a third-party service provider raises
  - Two security concerns
    - Data privacy and
    - Query integrity

Data Privacy Protection

- A method to execute SQL queries over encrypted databases

        - To process as much of a query as possible by the service providers, without having to decrypt the data
        - Decryption and the remainder of the query processing are performed at the client side
    - An order-preserving encryption scheme for numeric values
- **Query Integrity Assurance**
    - Query integrity examines the trustworthiness of the hosting environment
    - When a client receives a query result from the service provider
        - Assures that the result is both correct and complete
        - Correct means that the result must originate in the owner's data and not has been tampered with
        - Complete means that the result includes all records satisfying the query
    - A solution named dual encryption
        - Ensure query integrity without requiring the database engine to perform any special function beyond query processing

Data Integrity in Untrustworthy Storage

The fear of loss of control on their data is one of the major concerns that prevent end users from migrating to cloud storage services

Different motivations for a storage service provider could become untrustworthy

- To cover the consequence of a mistake in operation
- Or deny the vulnerability in the system after the data have been stolen by an adversary

Before cloud computing, several remote data storage checking protocols have been suggested

In practice, a remote data possession checking protocol has to satisfy the following five requirements

Requirement #1

- The verifier has to possess a complete copy of the data to be checked
- In practice, it does not make sense for a verifier to keep a duplicated copy of the content to be verified
- Storing a more concise contents digest of the data at the verifier should be enough

Requirement #2

- The protocol has to be very robust considering the untrustworthy prover
- A malicious prover is motivated to hide the violation of data integrity
- The protocol should be robust enough that such a prover ought to fail in convincing the verifier

Requirement #3

- The amount of information exchanged during the verification operation should not lead to high communication overhead

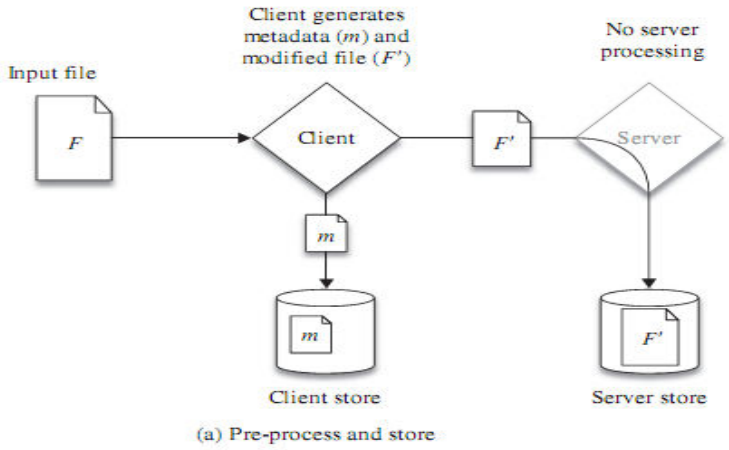
Requirement #4

- The protocol should be computationally efficient

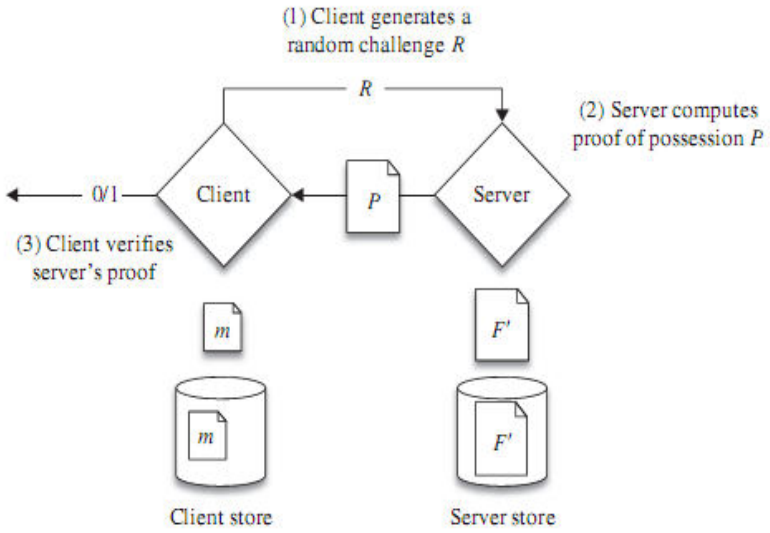
Requirement #5

- It ought to be possible to run the verification an unlimited number of times

- **A PDP-Based Integrity Checking Protocol**
  - Based on the provable data procession (PDP) technology
    - Allows users to obtain a probabilistic proof from the storage service providers
    - Used as evidence that their data have been stored there
  - The proof could be generated by the storage service provider by accessing only a small portion of the whole dataset
  - The amount of the metadata that end users are required to store is also small, i.e., O(1)
    - Such a small amount data exchanging procedure lowers the overhead in the communication channels



(a) Pre-process and store



(b) Verify server possession

- The flowcharts of the protocol for provable data possession
  - The data owner, namely the client, executes the protocol to verify that a dataset is stored in an outsourced storage machine
    - As a collection of *n* blocks
  - Before uploading the data into the remote storage

        - The data owner pre-processes the dataset and a piece of metadata is generated
        - The metadata are stored at the data owner's side
    - The dataset will be transmitted to the storage server
    - The cloud storage service stores the dataset
        - Sends the data to the user in responding to queries from the data owner in the future
    - As part of pre-processing procedure, the data owner (client) may conduct operations on the data
        - e.g., Expanding the data or generating additional metadata to be stored at the cloud server side
    - The data owner could execute the PDP protocol before the local copy is deleted
        - To ensure that the uploaded copy has been stored at the server machines successfully
    - The data owner may encrypt a dataset before transferring them to the storage machines

Web-Application-Based Security

- In cloud computing environments, resources are provided as a service over the Internet in a dynamic, virtualized and scalable way.
- Hence, web security plays an important role in the era of cloud computing
- The web site server is the first gate that guards the cloud resources. Any web security vulnerability will be amplified at the level of whole cloud

Types of attacks:

- Authentication
- Authorization,
- Client-side attacks
- Command Execution
- Information Disclosure
- Logical Attacks
- **Authentication:** It is the process of verifying a claim that a subject made to act on behalf of a given principal.
- Authentication attacks target a web site's method of validating the identity of a user, service or application

These attacks could be

- **Brute Force:** This attack employs an automated process to guess a person's username and password by trail and error
- **Insufficient Authentication:** Some sensitive content or functionality are protected by hiding the specific location in obscure string but still remains accessible directly through a specific URL
- **Password Recovery:** This service will automatically recover the username and password to the user if he or she can answer some questions defined as part of the registration process. If the recovery questions are either easily guessed or can be skipped, this website is considered to be Weak Password Recovery Validation

- **Authorization:** It is used to verify if an authenticated subject can perform a certain operation
- Ex: Only certain users are allowed to access specific content or functionality
- Authentication should precede Authorization

Types of Attacks

- **Insufficient Authorization**: occurs when a web site does not protect sensitive content of functionality with proper access control restrictions.
- **Credential/Session Prediction attack**: This attack deduces or guesses the unique session ID of a session to hijack or impersonate a user.
- **Insufficient Session Expiration**: occurs when an attacker is allowed to reuse old session credentials or session IDs for authorization
- **Session Fixation** forces a user's session ID to an arbitrary value via cross-site Scripting. Once the user logs in, the attacker uses the predefined session ID to impersonate the victim's identity
- **Client side attacks**: These attacks trap user to click a link in a malicious web page and then use the trust relationship expectations of the victim for the real web site

Types of Attacks:

- **Content Spoofing**: The malicious web page can trick a user into typing username and password and will use this information to impersonate the user.
- **Cross-Site Scripting(CSS)**: This attack launches attacker supplied executable code in the victim's browser. Since the code runs within the security context of the hosting web site, the code has the ability to read, modify, and transmit any sensitive data.
- **Command Execution**: These attacks exploit server-side vulnerabilities to execute remote commands on the web site
- **Information Disclosure**: These attacks acquire sensitive information about a web site revealed by developer comments, error messages, or well-known file name conventions
- **Logical Attacks:** Logical attacks involve exploitation of a web application's logic flow. A common logical attack is Denial of Service(DOS) attack. Dos attacks will attempt to consume all available resources in the web server such as CPU, memory, disk space by abusing the functionality provided by the web site.

Multimedia Data Security Storage

With the rapid developments of multimedia technologies, more and more multimedia contents are being stored and delivered over many kinds of devices, databases and networks

Multimedia Data Security plays an important role in the data storage to protect multimedia data.

- **Protection from unauthorized Replication**: Contents replication is required to generate and keep multiple copies of certain multimedia contents. Although the replication can improve the system performance, the unauthorized replication causes some problems such as contents copyright, waste of replication cost and extra control overheads
- **Protection from unauthorized Replacement**: As storage capacity is limited, a replacement process must be carried out when the capacity exceeds its limit. If an unauthorized replacement happens, the content which the user doesn't want to delete will be removed resulting in an accident of the data loss

- **Protection from unauthorized Pre-fetching:** If a content can be predicted to be requested by the user in future requests, this content will be fetched from the server database to the end user before this user requests it, in order to decrease the response time. Although Pre-fetching is efficient, unauthorized pre-fetching should be avoided to make the system to fetch the necessary content.

PLATFORM-AS-A-SERVICE

Cloud Computing opens new opportunities to small, medium-sized, and large companies

- Not necessary anymore to bear considerable costs for maintaining the IT infrastructures or to plan for peak demand
- Infrastructure and applications can scale elastically according to the business needs at a reasonable price

Led to the establishment of the concept of Public Cloud

- Represents a publicly accessible distributed system hosting the execution of applications and providing services billed on a pay-per-use basis

Such a solution built on outsourcing the entire IT infrastructure to third parties would not be applicable in many cases

Especially when there are critical operations to be performed and security concerns to consider

With the public cloud distributed anywhere on the planet, legal issues arise

Difficult to rely on a virtual public infrastructure for any IT operation

- Data location and confidentiality are two of the major issues that scare stakeholders to move into the cloud

Data that might be secure in one country may not be secure in another

- Users of cloud services do not know where their information is held

Different jurisdictions can apply

- In Europe, the European Union favors very strict protection of privacy
- In America, laws have virtually limitless powers to access information including that of companies
- The distinctive feature of cloud computing still remains appealing
- The possibility of replicating in-house, i.e., on their own IT infrastructure, the resource and service provisioning model proposed by cloud computing led to the development of the Private Cloud concept

Private clouds are virtual distributed systems that rely on a private infrastructure

- Provide internal users with dynamic provisioning of computing resources

Private clouds have the advantage

- Keeping in-house the core business operations by relying on the existing IT infrastructure
- Reducing the burden of maintaining it once the cloud has been set up
- Security concerns are less critical
- Sensitive information does not flow out of the private infrastructure

But, Private clouds cannot easily scale out in the case of peak demand

The integration with public clouds could be a solution to the increased load

Hybrid clouds are the result of a private cloud growing and provisioning resources from a public cloud

- Best option for the future in many cases

Hybrid clouds exploit existing IT infrastructures

- Maintaining sensitive information within the premises
- Naturally growing and shrinking by provisioning external resources and releasing them when needed
- Security concerns are only limited to the public portion of the cloud

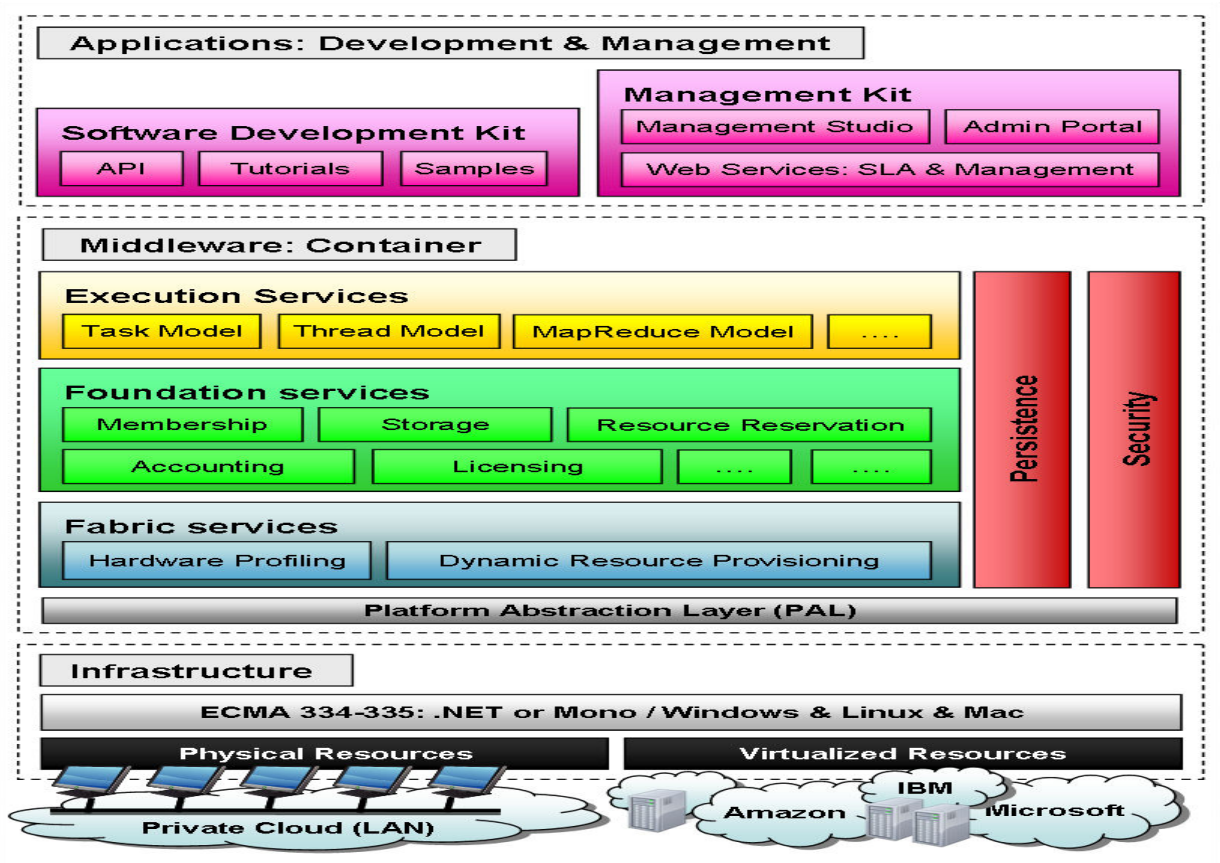
Platform as a Service (PaaS) solutions offer the right tools to implement and deploy hybrid clouds

- Provide enterprises with a platform for creating, deploying, and managing distributed applications on top of existing infrastructures
- In charge of monitoring and managing the infrastructure and acquiring new nodes
- Rely on virtualization technologies in order to scale applications on demand

ANEKA CLOUD PLATFORM

- **Aneka** is a software platform and a framework for developing distributed applications on the cloud
  - Harnesses the computing resources of a heterogeneous network of workstations and servers or data centers on demand
  - Provides developers with a rich set of APIs for transparently exploiting these resources by expressing the application logic with a variety of programming abstractions
  - System administrators can leverage a collection of tools to monitor and control the deployed infrastructure
  - Can be a public cloud available to anyone through the Internet
  - A private cloud constituted by a set of nodes with restricted access within an enterprise
  - or a hybrid cloud where external resources are integrated on demand allowing applications to scale

Aneka Framework Architecture



Aneka is essentially an implementation of the PaaS model

- Provides a runtime environment for executing applications by leveraging the underlying infrastructure of the cloud

– Developers can express distributed applications
  - By using the API contained in the Software Development Kit (SDK)
  - Or by porting existing legacy applications to the cloud
– Such applications are executed on the Aneka cloud
  - Represented by a collection of nodes connected through the network hosting the Aneka container
– The container is the building block of the middleware
  - Represents the runtime environment for executing applications
  - Contains the core functionalities of the system
  - Built up from an extensible collection of services that allow administrators to customize the Aneka cloud

Three classes of services that characterize the container

– **Execution Services**
  - Responsible for scheduling and executing applications
  - Each of the programming models supported by Aneka defines specialized implementations of these services for managing the execution of a unit of work defined in the model
– **Foundation Services**

        - The core management services of the Aneka container
        - In charge of metering applications, allocating resources for execution, managing the collection of available nodes, and keeping the services registry updated
    - **Fabric Services**
        - Constitute the lowest level of the services stack of Aneka
        - Provide access to the resources managed by the cloud
        - The Resource Provisioning Service enables horizontal scaling in the cloud
        - Horizontal scaling is the process of adding more computing nodes to a system
        - Vertical scaling is the process of increasing the computing capability of a single computer resource
        - Resource provisioning makes Aneka elastic
        - Allows to grow or to shrink dynamically to meet the QoS requirements of applications

The container relies on a platform abstraction layer

- Interfaces it with the underlying host whether this is a physical or a virtualized resource
- Makes the container portable over different runtime environments
    - Feature an implementation of the ECMA 334 and ECMA 335 specifications
    - e.g., the .NET framework or Mono

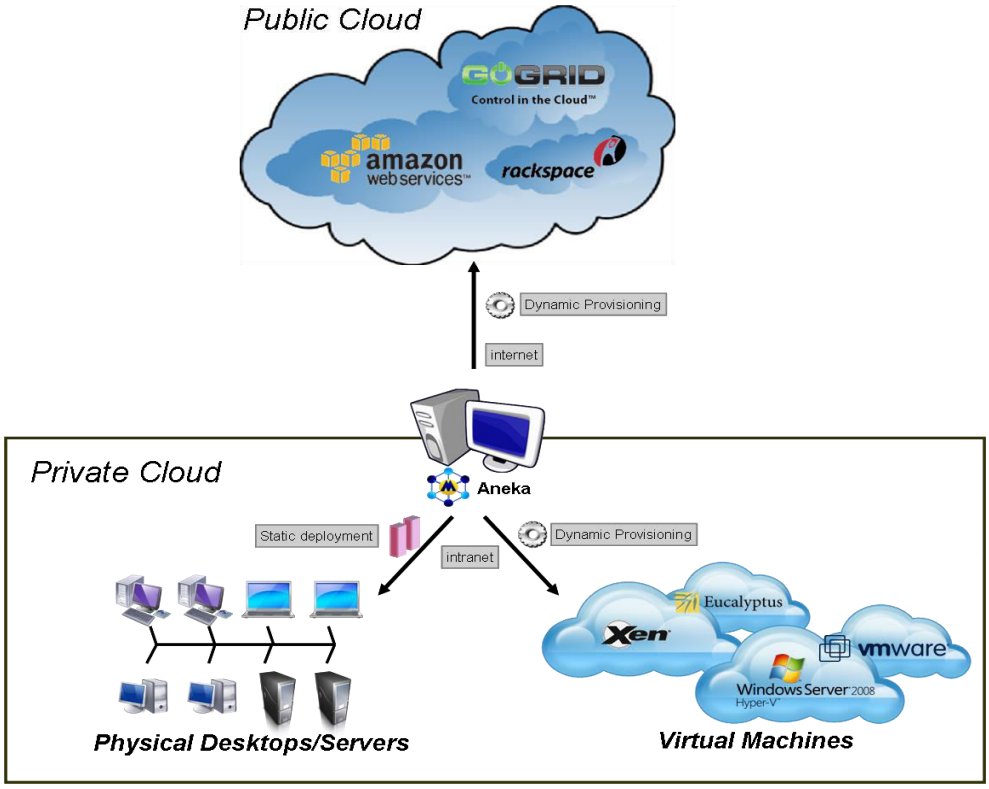
- Aneka also provides a tool for managing the cloud
    - Allowing administrators to start, stop, and deploy instances of the container on new resources and then reconfigure them dynamically to alter the behavior of the cloud

Aneka Resource Provisioning Service

- Cloud computing has the ability to automatically scale out
    - Based on demand and users' quality of service requests
- Aneka is a PaaS
    - Features multiple programming models allowing developers to easily build their distributed applications
    - Provides resource provisioning facilities in a seamless and dynamic fashion
        - Applications managed by the Aneka container can be dynamically mapped to heterogeneous resources
        - Grow or shrink according to the application's needs
    - Achieved by means of the resource provisioning framework
        - Composed primarily of services built into the Aneka fabric layer

A typical scenario that a medium or large enterprise may encounter

- Combines privately owned resources with public rented resources to dynamically increase the resource capacity to a larger scale



- Private resources identify computing and storage elements kept in the premises. They share similar internal security and administrative policies

Aneka identifies two types of private resources

  - Static and dynamic resources
- **Static resources** are constituted by existing physical workstations and servers
  - May be idle for a certain period of time
  - Their membership to the Aneka cloud is manually configured by administrators
  - Does not change over time
- **Dynamic resources** are mostly represented by virtual instances that join and leave the cloud
  - Controlled by resource pool managers that provision and release them when needed

- Public resources reside outside the boundaries of the enterprise
  - Provisioned by establishing a service-level agreement with the external provider

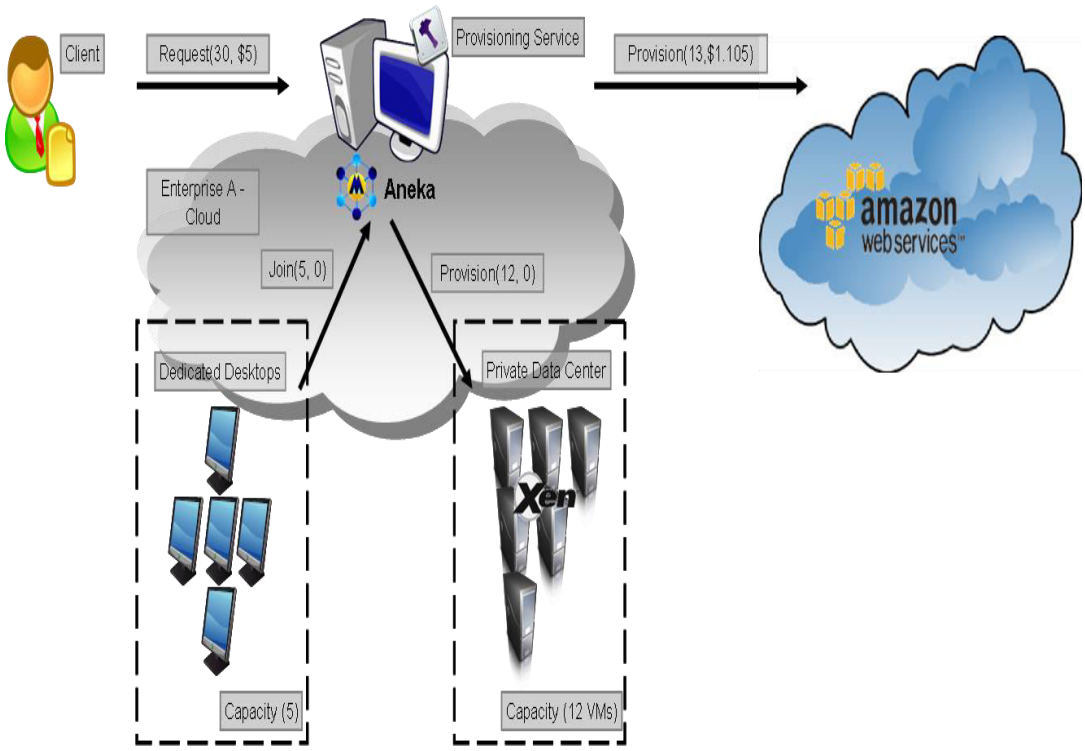
Two classes: on-demand and reserved resources

- **On-demand** resources are dynamically provisioned by resource pools for a fixed amount of time, e.g., an hour
  - With no long-term commitments
  - On a pay-as-you-go basis
- **Reserved resources** are provisioned in advance by paying a low, one-time fee
  - Mostly suited for long-term usage

        - Actually the same as static resources
        - No automation is needed in the resource provisioning service to manage them

Resource Provisioning Scenario

- A possible scenario in which the resource provisioning service becomes important
  - A private enterprise maintains a private cloud consisting of
    - Five physical dedicated desktops from its engineering department
    - A small data center managed by Xen Hypervisor providing virtual machines with the maximum capacity of 12 VMs
  - In most of the cases, this setting is able to address the computing needs of the enterprise
  - In the case of peak computing demand
    - Additional resources can be provisioned by leveraging the virtual public infrastructure
  - A mission critical application could require at least 30 resources to complete within an hour
    - The customer is willing to spend a maximum of 5 dollars to achieve this goal
  - The Aneka Resource Provisioning service becomes a fundamental infrastructure component to address this scenario
    - The Aneka scheduling engine detects that the current capacity in terms of resources, i.e., 5 dedicated nodes is not enough to satisfy the user's QoS requirement and to complete the application on time
    - An additional 25 resources must be provisioned
  - The responsibility of the Aneka Resource Provisioning service to acquire these resources from both the private data center managed by Xen Hypervisor and the Amazon public cloud
  - The provisioning service is configured by default with a cost-effective strategy
    - Privileges the use of local resources instead of the dynamically provisioned and chargeable ones
    - The computing needs of the application require the full utilization of the local data center that provides the Aneka cloud with 12 virtual machines
    - The remaining 13 resources are rented from Amazon for a minimum of one hour
    - Only incurs a few dollars' cost



Another simple strategy for provisioning resources could be minimizing the execution time to let the application finish as early as possible

- Requires Aneka to request more powerful resources from the Amazon public cloud

Instead of provisioning 13 small instances from Amazon, a major number of resources or more powerful resources can be rented by spending the entire budget available for the application

Hybrid Cloud Implementation

No widely accepted standard for provisioning virtual infrastructure from Infrastructure as a Service (IaaS) providers

- Each provider exposes its own interfaces and protocols
- Not possible to seamlessly integrate different providers into one single infrastructure

- The resource provisioning service implemented in Aneka addresses these issues
  - Abstracts away the differences of providers' implementation

Design and Implementation Guidelines

The particular nature of hybrid clouds demands additional and specific functionalities

- **Support for Heterogeneity**
  - Hybrid clouds are produced by heterogeneous resources such as clusters, public or private virtual infrastructures, and workstations

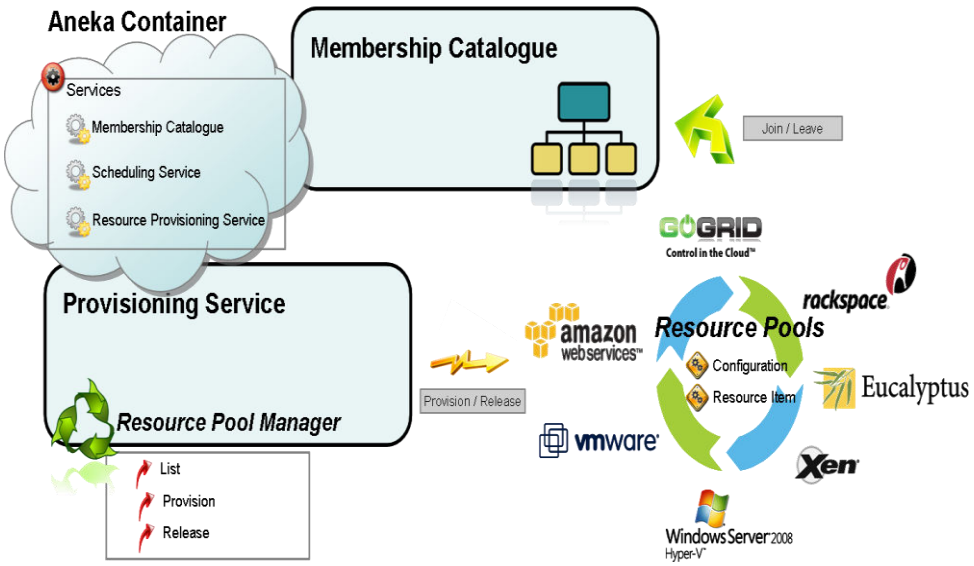
        - A virtual machine manager must be possible to integrate additional cloud service providers, mostly IaaS providers, without major changes to the entire system design and codebase
        - The specific code related to a particular cloud resource provider should be kept isolated behind interfaces and within pluggable components
    - **Support for Dynamic and Open Systems**
        - Hybrid clouds change their composition and topology over time
        - An open and extensible architecture that allows easily plugging new components and rapidly integrating new features is of a great value
        - Dependency injection is a technique that allows configuring and connecting components within a software container, like a Web or an application server, without hard coding their relation but by providing an abstract specification

e.g., a configuration file that specifies which component to instantiate and to connect them together

    - **Support for Basic VM Operation Management**
        - Hybrid clouds integrate virtual infrastructures with existing physical systems
        - Software frameworks that support hypervisor-based execution should implement a minimum set of operations including requesting a virtual instance, controlling its status, terminating its execution, and keeping track of all the instances that have been requested
    - **Support for Flexible Scheduling Policies**
        - The heterogeneity of resources that constitute a hybrid infrastructure naturally demands for flexible scheduling policies
        - Public and private resources can be differently utilized
        - The workload should be dynamically partitioned into different streams according to their security and quality of service (QoS) requirements
    - **Support for Workload Monitoring**
        - A subset of resources is leased and resources can be dismissed if they are no longer necessary
        - Necessary to integrate this feature with scheduling policies that directly or indirectly govern the management and leases of virtual instances
    - The most relevant features for successfully supporting the deployment and the management of hybrid clouds
- Security is transversal to all features
    - A basic recommendation for implementing a security infrastructure for any runtime environment is to use a Defense in Depth security model whenever it is possible
        - Defense in depth is an information assurance (IA) strategy in which multiple layers of defense are placed throughout an information technology (IT) system
    - Even more important in hybrid clouds
        - Both applications and resources can represent treats to each other

Aneka Hybrid Cloud Architecture

- The solution implemented in Aneka
  - The Resource Provisioning Framework represents the foundation on top of which Aneka-based hybrid clouds are implemented
  - The resource provisioning infrastructure is represented by a collection of resource pools provides access to resource providers
  - Managed uniformly through a specific component called a resource pool manager
  - **Resource Provisioning Service**
    - An Aneka-specific service that implements the service interface and wraps the resource pool manager, allowing its integration within the Aneka container
  - **Resource Pool Manager**
    - Manages all the registered resource pools
    - Decides how to allocate resources from those pools
    - Provides a uniform interface for requesting additional resources from any private or public provider
      - Hides the complexity of managing multiple pools to the Resource Provisioning Service



  - **Resource Pool**
    - A container of virtual resources that mostly come from the same resource provider
    - In charge of managing the virtual resources it contains
    - Finally releasing them when they are no longer in use
    - Each vendor exposes its own specific interfaces
    - Encapsulates the specific implementation of the communication protocol required to interact with it
    - Provides the pool manager with a unified interface for acquiring, terminating, and monitoring virtual resources

- The request for additional resources is generally triggered by a scheduler
  - Detects that the current capacity is not sufficient to satisfy the expected quality of services ensured for specific applications
- In this case, a provisioning request is made to the Resource Provisioning Service
  - According to specific policies, the pool manager determines the pool instance(s) that will be used to provision resources and will forward the request to the selected pools
  - Each resource pool will translate the forwarded request by using the specific protocols required by the external provider and provision the resources
- Once the requests are successfully processed
  - The requested number of virtual resources will join the Aneka cloud by registering themselves with the Membership Catalogue Service
    - Keeps track of all the nodes currently connected to the cloud
  - Once joined the cloud, the provisioned resources are managed like any other node
- A release request is triggered by the scheduling service when provisioned resources are no longer in use

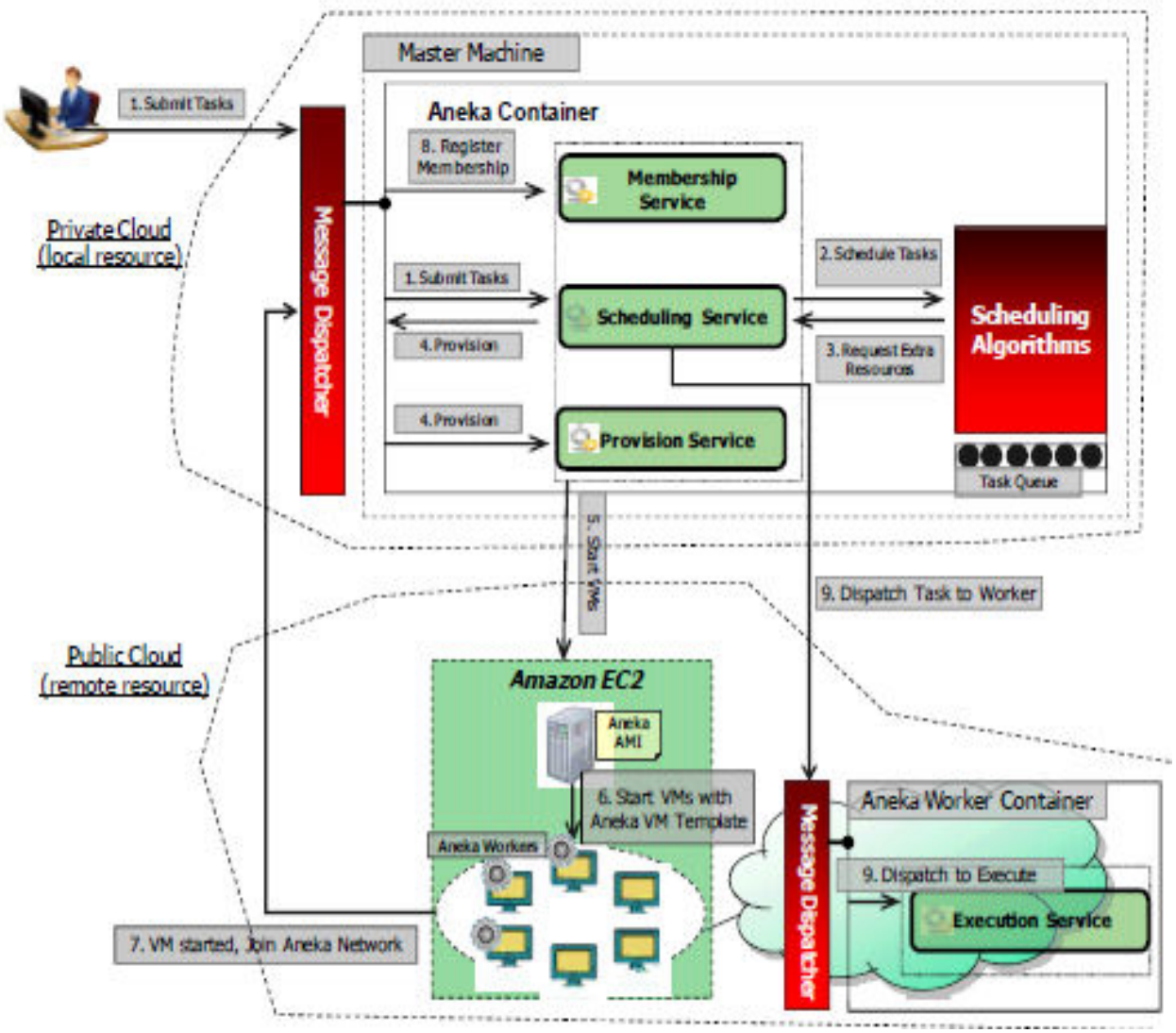
Use Case - The Amazon EC2 Resource Pool

- The implementation of the Amazon EC2 resource pool
  - A practical application of the infrastructure developed
  - Amazon EC2 is one of the most popular cloud resource providers
    - Among the top 10 companies providing cloud computing services
  - EC2 provides a Web service interface for accessing, managing, and controlling virtual machine instances
    - Simplifies the integration of Amazon EC2 with any application
- To interact with Amazon EC2, several parameters are required
  - **User Identity**: Represents the account information used to authenticate with Amazon EC2
    - Constituted by a pair of encrypted keys that are the access key and the secret key
    - These keys can be obtained from the Amazon Web services portal once the user has signed in
    - Required to perform any operation that involves Web service access
  - **Resource Identity**
    - The identifier of a public or a private Amazon Machine Image (AMI) used as template from which to create virtual machine instances
  - **Resource Capacity**
    - Specifies the different type of instance that will be deployed by Amazon EC2
    - Instance types vary according to the number of cores, the amount of memory, and other settings that affect the performance of the virtual machine instance
    - Several types of images are available

        - Those commonly used are: small, medium, and large
        - The capacity of each type of resource has been predefined by Amazon and is charged differently
- This information is maintained in the EC2ResourcePoolConfiguration class
    - Need to be provided by the administrator to configure the pool
- The implementation of EC2ResourcePool is forwarding the request of the pool manager to EC2
    - By using the Web service client and the configuration information previously described
    - Then stores the metadata of each active virtual instance for further use
- To utilize at best the virtual machine instances provisioned from EC2
    - The pool implements a cost-effective optimization strategy
    - A virtual machine instance is charged by using one-hour time blocks
        - According to the current business model of Amazon
        - If a virtual machine instance is used for 30 minutes, the customer is still charged for one hour of usage
    - To provide a good service to applications with a smaller granularity in terms of execution times
        - The EC2ResourcePool class implements a local cache that keeps track of the released instances whose time block is not expired yet
        - These instances will be reused instead of activating new instances from Amazon
- With the cost-effective optimization strategy
    - The pool is able to minimize the cost of provisioning resources from Amazon cloud to achieve high utilization of each provisioned resource

Implementation Steps for Aneka Resource Provisioning Service

- The resource provisioning service is a customized service
    - Used to enable cloud bursting by Aneka at runtime
- One of the application scenarios that utilize resource provisioning
    - To dynamically provision virtual machines from Amazon EC2 cloud
- The general steps of resource provisioning on demand in Aneka
    - The application submits its tasks to the scheduling service
    - Adds the tasks into the scheduling queue



- The scheduling algorithm finds an appropriate match between a task and a resource
    - If the algorithm could not find enough resources for serving all the tasks, it requests extra resources from the scheduling service
- The scheduling service will send a Resource Provision Message to provision service
    - Ask provision service to get X number of resources as determined by the scheduling algorithm
- Upon receiving the provision message
    - The provision service will delegate the provision request to a component called resource pool manager
    - Responsible for managing various resource pools
    - All the work instances will then connect to the Aneka master machine
    - Register themselves with Aneka membership service
    - A resource pool is a logical view of a cloud resource provider
    - The virtual machines can be provisioned at runtime
    - Aneka resource provisioning supports multiple resource pools like Amazon EC2 pool and Citrix Xen server pool

- The resource pool manager knows how to communicate with each pool
    - Provision the requested resources on demand
    - The pool manager starts X virtual machines by utilizing the predefined virtual machine template already configured to run Aneka containers
- A work instance of Aneka will be configured and running once a virtual resource is started
- The scheduling algorithm will be notified by the membership service once those work instances join the network and start allocating pending tasks to them immediately
- Once the application is completed all the provisioned resources will be released by the provision service to reduce the cost of renting the virtual machine

COMETCLOUD- AN AUTONOMIC CLOUD ENGINE

- Clouds typically have highly dynamic demands for resources with highly heterogeneous and dynamic workloads
- Different applications may have very different and dynamic quality of service (QoS) requirements
    - One application may require high throughput
    - Another may be constrained by a budget
    - A third may have to balance both throughput and budget
- To support on-demand scale-up, scale-down, and scale-out
    - Combining public cloud platforms and integrating them with existing grids and data centers
    - Users may want to use resources in their private cloud or data center or grid first before scaling out onto a public cloud
    - Integrating these public cloud platforms with exiting computational grids to provide opportunities for on-demand scale-up and scale-down is **cloud burst**

The Comet Cloud autonomic cloud engine
Goal is to realize a virtual computational cloud with resizable computing capability that

- Integrates local computational environments and public cloud services on-demand
- Provide abstractions and mechanisms to support a range of programming paradigms and applications requirements

- Comet Cloud enables policy-based autonomic cloud bridging and cloud bursting
    - Autonomic cloud bridging enables on-the-fly integration of local computational environments, like data centers and grids, and public cloud services, like Amazon EC2 and Eucalyptus
    - Autonomic cloud bursting enables dynamic application scale-out to address dynamic workloads, spikes in demands, and other extreme requirements

- CometCloud is based on a decentralized coordination substrate

- Supports highly heterogeneous and dynamic cloud/grid infrastructures, integration of public/private clouds, and cloudbursts
- Also used to support a decentralized and scalable task space
    - Coordinates the scheduling of task onto sets of dynamically provisioned workers on available private and/or public cloud resources
    - Submitted by a dynamic set of users
    - Based on their QoS constraints like cost or performance

COMETCLOUD ARCHITECTURE

- Comet Cloud is an autonomic computing engine for cloud and grid environments
    - Based on the Comet decentralized coordination substrate
    - Supports highly heterogeneous and dynamic cloud/grid infrastructures
    - Integration of public/private clouds
    - Autonomic cloudbursts
    - Based on a peer-to-peer substrate that can span enterprise data centers, grids, and clouds

Comet is composed of three layers

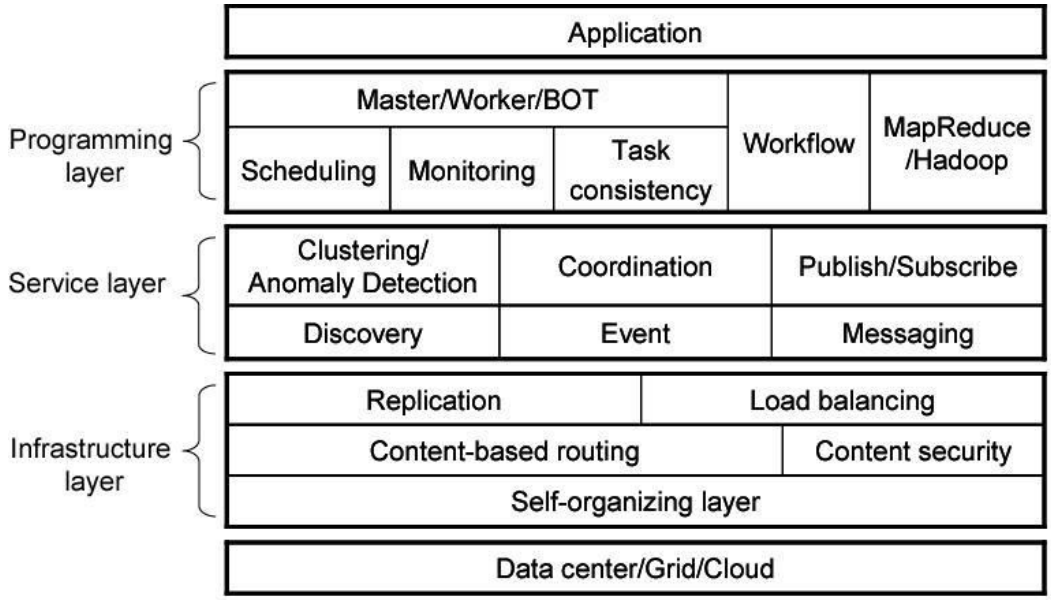
- programming layer,
- service layer, and
- an infrastructure layer

Adapts the Squid information discovery scheme to deterministically map the information space onto the dynamic set of peer nodes

- The resulting structure is a locality preserving semantic distributed hash table (DHT) on top of a self-organizing structured overlay that maintains content locality
- Guarantees that content-based queries are delivered with bounded costs using flexible content descriptors in the form of keywords, partial keywords, and wildcards

- Comet builds a tuple-based coordination space abstraction using Squid
    - Can be associatively accessed by all system peers without requiring the location information of tuples and host identifiers
    - Also provides transient spaces that enable applications to explicitly exploit context locality

Comet Cloud Layered Abstractions



A schematic overview of the Comet Cloud architecture

- The **infrastructure layer** uses the Chord self-organizing overlay
  - The Squid information discovery(peer-to-peer information discovery system) and content-based routing substrate built on top of Chord(distributed lookup protocol: given a key, it maps on to a node)
    - The routing engine supports flexible content-based routing and complex querying using partial keywords, wildcards, or ranges
    - Guarantees that all peer nodes with data elements that match a query/message will be located

Nodes providing resources in the overlay have different roles and
different access privileges based on their credentials and capabilities

- Provides replication and load balancing services
  - Handles dynamic joins and leaves of nodes as well as node failures
- Every node keeps the replica of its successor node's state
  - Reflects changes to this replica whenever its successor notifies it of changes
  - Notifies its predecessor of any changes to its state
- If a node fails, the predecessor node merges the replica into its state
  - Makes a replica of its new successor
- If a new node joins, the joining node's predecessor updates its replica to reflect the joining node's state. The successor gives its state information to the joining node
- To maintain load balancing
  - Load should be redistributed among the nodes whenever a node joins and leaves

The **service layer** provides a range of services to support autonomics at the programming and application level

- Provides a virtual shared-space abstraction as well as associative access primitives

- The basic coordination primitives:
  - out(*ts*, *t*): A non blocking operation that inserts tuple *t* into space *ts*
  - in(*ts*, *t'*): A blocking operation that removes a tuple *t* matching template *t'* from the space *ts* and returns it
  - rd(ts, *t'*): A blocking operation that returns a tuple *t* matching template *t'* from the space *ts*; The tuple is not removed from the space
  - The out is for inserting a tuple into the space
  - in and rd are for reading a tuple from the space
  - in removes the tuple after read
  - rd only reads the tuple
- Support range query
  - "*" can be used for searching all tuples
- To address this issue, CometCloud supports dynamically constructed transient spaces
  - Have a specific scope definition
  - e.g., within the same geographical region or the same physical subnet
- The global space is accessible to all peer nodes
  - Acts as the default coordination platform
- Membership and authentication mechanisms are adopted to restrict access to the transient spaces
- The structure of the transient space is exactly the same as the global space
  - An application can switch between spaces at runtime
  - Can simultaneously use multiple spaces
- Also provides asynchronous (publish/subscribe) messaging and eventing services
- On-line clustering services support autonomic management
  - Enable self-monitoring and control
  - Events describing the status or behavior of system components are clustered
  - Used to detect anomalous behaviors

- The **programming** layer provides the basic framework for application development and management
  - Supports a range of paradigms including the master/worker/BOT (Bags of Tasks)
    - Masters generate tasks and workers consume them
    - Masters and workers can communicate via virtual shared space or using a direct connection
  - Scheduling and monitoring of tasks are supported by the application framework
  - The task consistency service handles lost tasks
    - Even though replication is provided by the infrastructure layer, a task may be lost due to network congestion.
    - Since there is no failure, infrastructure level replication may not be able to handle it
  - Lost tasks can be handled by the master
    - e.g., by waiting for the result of each task for a predefined time interval

        - If the master does not receive the result back, regenerating the lost task
        - If the master receives duplicate results for a task, it selects the first one and ignores other subsequent results
    - Other supported paradigms include workflow-based applications as well as Mapreduce and Hadoop

Comet Space

- In Comet, a tuple is a simple XML string
    - The first element is the tuple's tag
        - Followed by an ordered list of elements containing the tuple's fields
        - Each field has a name followed by its value
        - The tag, field names, and values must be actual data for a tuple
        - Can contain wildcards ("*") for a template tuple
    - This lightweight format is flexible enough to represent the information for a wide range of applications
        - Can support rich matching relationships
        - The cross-platform nature of XML makes this format suitable for information exchange in distributed heterogeneous environments
- A tuple in Comet can be retrieved
    - If it exactly or approximately matches a template tuple
    - Exact matching requires the tag and field names of the template tuple to be specified without any wildcard, as in Linda
        - This strict matching pattern must be relaxed in highly dynamic environments
        - Applications like service discovery may not know exact tuple structures
    - Supports retrievals with incomplete structure information using approximate matching
        - Only requires the tag of the template tuple be specified using a keyword or a partial keyword
- Tuple (a) tagged "contact"
    - Has fields "name, phone, email, dep"
    - With values "Smith, 7324451000, smith@gmail.com, ece"
    - Can be retrieved using tuple template (b) or (c)

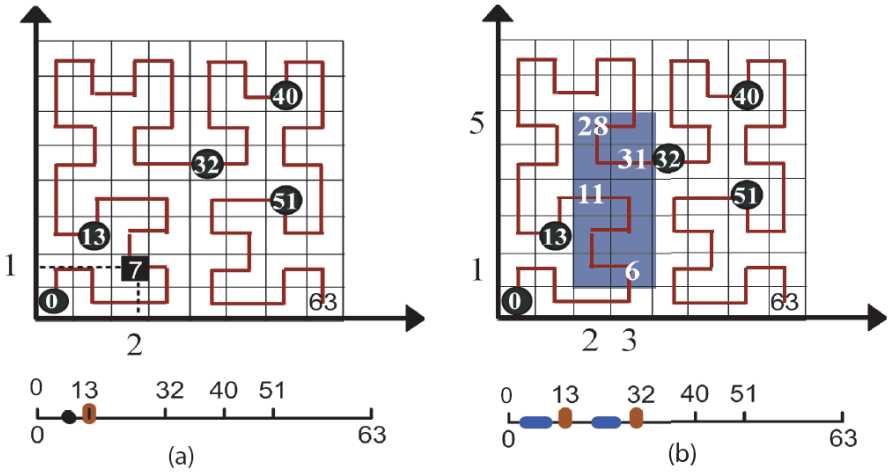
```
<contact>
   <name> Smith </name>
   <phone> 7324451000 </phone>
   <email> smith@gmail.com </email>
   <dep> ece </dep>
</contact>
```
(a)

```
<contact>
   <name> Smith </name>
   <phone> 7324451000 </phone>
   <email>* </email>
   <dep> * </dep>
</contact>
```
(b)

```
<contact>
   <na*> Smith </na*>
   <*>
   <*>
   <dep> ece </dep>
</contact>
```
(c)

- Comet adapts Squid information discovery scheme
    - Employs the Hilbert space-filling curve (SFC)

        - To map tuples from a semantic information space to a linear node index
    - The semantic information space is defined by application users
        - Consisting of based-10 numbers and English words
        - e.g., A computational storage resource may belong to the 3D storage space with coordinates “space,” “bandwidth,” and “cost”
- Each tuple is associated with k keywords selected from its tag and field names
    - The keys of a tuple
    - e.g., The keys of tuple (a) can be “name, phone” in a 2D student information space
    - Tuples are local in the information space
        - If their keys are lexicographically close
        - Or if they have common keywords
    - The selection of keys can be specified by the applications
- A Hilbert SFC is a locality preserving continuous mapping
    - From a k-dimensional (kD) space to a 1D space
    - Points that are close on the curve are mapped from close points in the kD space
    - The Hilbert curve readily extends to any number of dimensions
    - Enables the tuple space to maintain content locality in the index space
- In Comet, the peer nodes form a one-dimensional overlay
    - Indexed by a Hilbert SFC
    - The tuples are mapped from the multi-dimensional information space to the linear peer index space
    - Comet uses the Hilbert SFC constructs the distribute hash table (DHT)
        - For tuple distribution and lookup
    - If the keys of a tuple only include complete keywords
        - The tuple is mapped as a point in the information space and located on at most one node
    - If its keys consist of partial keywords, wildcards, or ranges
        - The tuple identifies a region in the information space
        - This region is mapped to a collection of segments on the SFC
        - Corresponds to a set of points in the index space
    - Each node stores the keys that map to the segment of the curve between itself and the predecessor node
- Five nodes are indexed using SFC from 0 to 63
    - With id shown in solid circle
    - The tuple defined as the point (2, 1) is mapped to index 7 on the SFC
        - Corresponds to node 13
    - The tuple defined as the region (23, 15) is mapped to two segments on the SFC
        - Corresponds to nodes 13 and 32



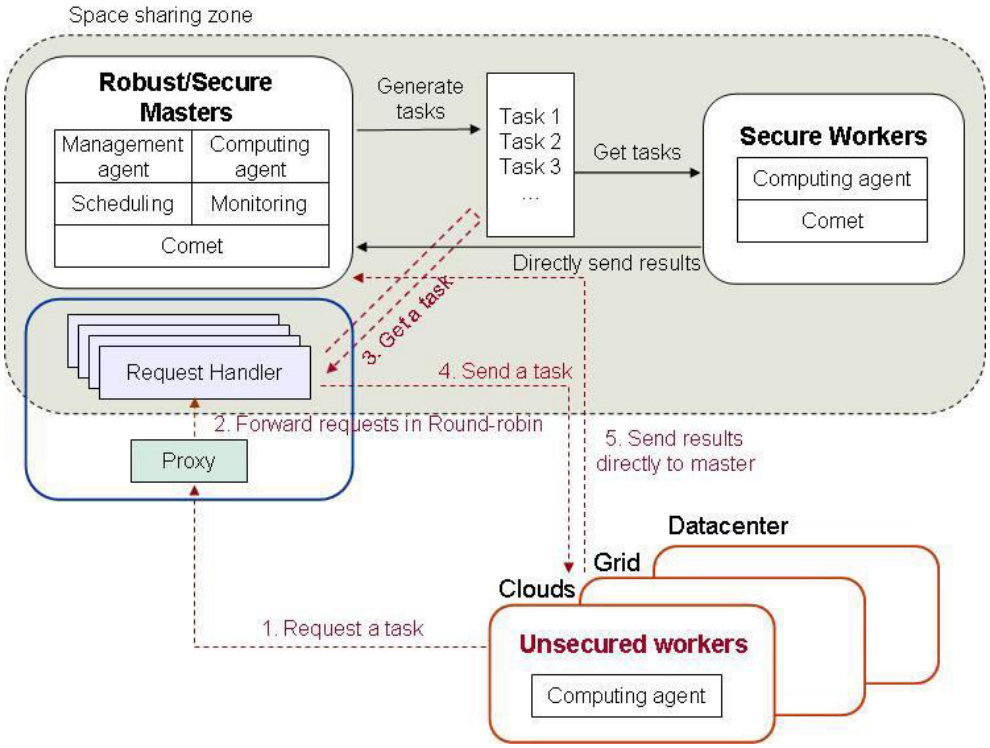
Autonomic Cloudbursting

- The goal of autonomic cloudbursts
  - To seamlessly and securely integrate private enterprise clouds and data centers with public utility clouds on-demand to provide the abstraction of resizable computing capacity
  - Enables the dynamic deployment of application components onto a public cloud
    - Typically run on internal organizational compute resources
    - To address dynamic workloads, spikes in demands, and other extreme requirements
- Typical over-provisioning strategies are no longer feasible
  - The increasing application and infrastructure scales
  - Their cooling, operation, and management costs
  - Autonomic cloudbursts can leverage utility clouds
    - To provide on-demand scale-out and scale-in capabilities based on a range of metrics
- The overall approach for supporting autonomic cloudbursts in CometCloud
  - CometCloudconsiders three types of clouds
    - Based on perceived security/trust
  - Assigns capabilities accordingly
  - The first is a highly trusted, robust, and secure cloud
    - Usually composed of trusted/secure nodes within an enterprise
    - Typically used to host masters and other key (management, scheduling, monitoring) roles
    - Also used to store states
  - In most applications, the privacy and integrity of critical data must be maintained
    - Tasks involving critical data should be limited to cloud nodes that have required credentials
    - The second type of cloud is one composed of nodes with such credentials
    - The cloud of secure workers
  - A privileged Comet space may span these two clouds

- May contain critical data, tasks, and other aspects of the application-logic/workflow

– The final type of cloud consists of casual workers
  - These workers are not part of the space
  - Can access the space through the proxy and a request handler to obtain (possibly encrypted) work units as long as they present required credentials

- Nodes can be added or deleted from any of these clouds by purpose
  – If the space needs to be scale-up to store dynamically growing workload and requires more computing capability
    - Autonomic cloudbursts target secure worker to scale up



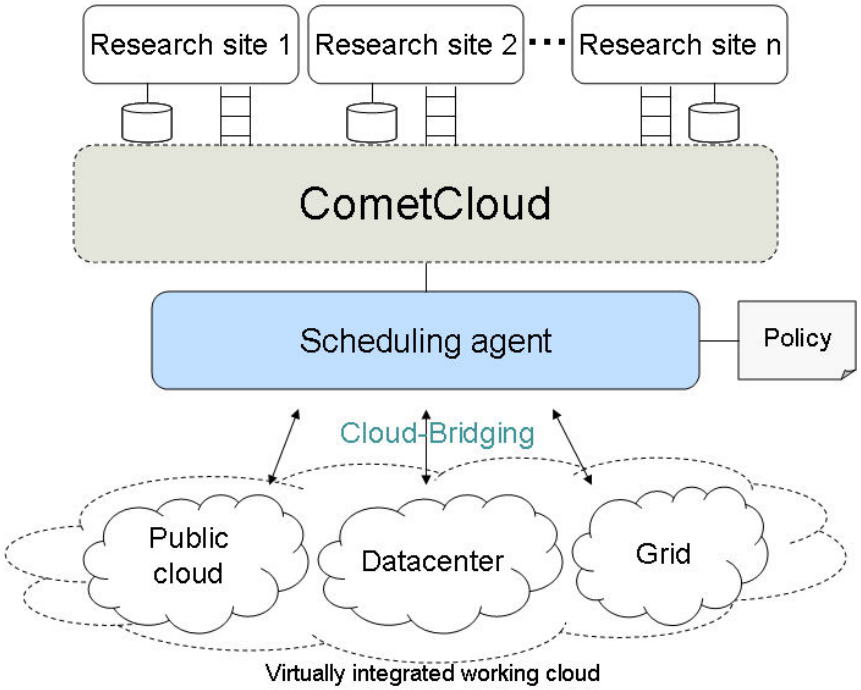
  – Only if more computing capability is required
    - Unsecured workers are added

- Key motivations for autonomic cloudbursts
  – Load Dynamics
    - Application workloads can vary significantly
    - Includes the number of application tasks and the computational requirements of a task
    - The computational environment must dynamically grow (or shrink) in response to these dynamics while still maintaining strict deadlines
  – Accuracy of the Analytics
    - The required accuracy of risk analytics depends on a number of highly dynamic market parameters
    - e.g., the number of scenarios in the Monte Carlo VaR formulation
    - The computational environment must be able to dynamically adapt to satisfy the accuracy requirements while still maintaining strict deadlines

- Collaboration of Different Groups
  - Different groups can run the same application with different dataset policies
  - Policy means user's SLA bounded by their condition such as time frame, budgets, and economic models
  - As collaboration groups join or leave the work, the computational environment must grow or shrink to satisfy their SLA
- Economics
  - Application tasks can have very heterogeneous and dynamic priorities
  - Must be assigned resources and scheduled accordingly
  - Budgets and economic models can be used to dynamically provision computational resources based on the priority and criticality of the application task
  - e.g., Application tasks can be assigned budgets and can be assigned resources based on this budget
  - The computational environment must be able to handle heterogeneous and dynamic provisioning and scheduling requirements
- Failures
  - Due to the strict deadlines involved, failures can be disastrous
  - The computation must be able to manage failures without impacting application quality of service, including deadlines and accuracies

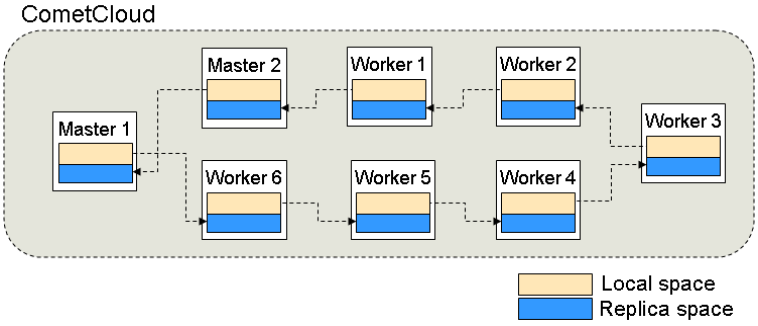
- Autonomic cloudbridging is meant to connect CometCloud and a virtual cloud
  - Consists of public cloud, data center, and grid by the dynamic needs of the application
  - The clouds in the virtual cloud are heterogeneous
    - Have different types of resources and cost policies
  - The performance of each cloud can change over time by the number of current users
    - Types of used clouds, the number of nodes in each cloud, and resource types of nodes should be decided according to the changing environment of the clouds and application's resource requirements
- The operation of the CometCloud-based autonomic cloudbridging
  - The scheduling agent manages autonomiccloudbursts over the virtual cloud
    - There can be one or more scheduling agents
    - Located at a robust/secure master site
  - If each group requires generating tasks with its own data and managing the virtual cloud by its own policy
    - Can have a separate scheduling agent in its master site
  - The requests for tasks generated by the different sites are logged in the CometCloud virtual shared space
    - Spans master nodes at each of the sites
  - These tasks are then consumed by workers
    - May run on local computational nodes at the site, a shared data center, and a grid or on a public cloud infrastructure

- A scheduling agent manages QoS constraints and autonomic cloudbursts of its site
  - According to the defined policy
- The workers can access the space using appropriate credentials
  - Access authorized tasks
- Return results back to the appropriate master indicated in the task itself



- A scheduling agent manages autonomic cloudbridging and guarantees QoS within user policies
  - Autonomic cloudburst is represented by changing resource provisioning not to violate defined policy
- Three types of policies
  - Deadline-Based
    - When an application needs to be completed as soon as possible, assuming an adequate budget, the maximum required workers are allocated for the job
  - Budget-Based
    - When a budget is enforced on the application, the number of workers allocated must ensure that the budget is not violated
  - Workload-Based
    - When the application workload changes, the number of workers explicitly defined by the application is allocated or released
- Fault-Tolerance
  - Fault-tolerance during runtime is critical to keep the application's deadline
  - Support fault-tolerance in two ways
    - In the infrastructure layer and in the programming layer
  - The replication substrate in the infrastructure layer provides a mechanism to keep the same state as that of its successor's state
    - Specifically, coordination space and overlay information

- The overview of replication in the overlay
  - Every node has a local space in the service layer
    - A replica space in the infrastructure layer
  - When a tuple is inserted or extracted from the local space
    - The node notifies this update to its predecessor
    - The predecessor updates the replica space
    - Every node keeps the same replica of its successor's local space.
  - When a node fails, another node in the overlay detects the failurenotifies it to the predecessor of the failed node
  - The predecessor of the failed node merges the replica space into the local space



- This makes all the tuples from the failed node recovered
  - The predecessor node makes a new replica for the local space of its new successor
- Also support fault-tolerance in the programming layer
  - Some tasks can be lost during runtime
    - Because of networkcongestion or task generation during failure
  - The master checks the space periodically and regenerates lost tasks
- Load Balancing
  - Executing application requests on underlying grid resources consists of two key steps
  - The first consists of creating VM instances to host each application request, matching the specific characteristics and requirements of the request
    - Called VM Provisioning
  - The second step is mapping and scheduling these requests onto distributed physical resources
    - Called Resource Provisioning
  - Most virtualized data centers currently provide a set of general-purpose VM classes with generic resource configurations
    - Quickly become insufficient to support the highly varied and interleaved workloads
  - Clients can easily under- or overestimate their needs because of a lack of understanding of application requirements
    - Due to application complexity and/or uncertainty
    - Results in over-provisioning due to a tendency to be conservative

- The decentralized clustering approach specifically addresses the distributed nature of enterprise grids and clouds
  - The approach builds on a decentralized messaging and data analysis infrastructure provides monitoring and density-based clustering capabilities
  - The characterization of different resource classes can be accomplished to provide autonomic VM provisioning
    - By clustering workload requests across data center job queues
  - Has several advantages
    - The capability of analyzing jobs across a dynamic set of distributed queues
    - The nondependency on a priori knowledge of the number of clustering classes
    - The amenity for online application
    - Timely adaptation to changing workloads and resources
  - The robust nature of the approach allows
- To handle changes (joins/leaves) in the job queue servers as well as their failures while maximizing the quality and efficiency of the clustering

UNIT – 4

Architecture forOpen Federated Cloud Computing

Introduction

- With cloud computing,
  - companies can lease infrastructure resources on-demand from a virtually unlimited pool.
  - The "pay as you go" billing model applies charges for the actually used resources per unit time.

A business can optimize its IT investment and improve availability and scalability

Issues in current Clouds

- **Inherently limited scalability of single-provider clouds**
  - Most infrastructure cloud providers today claim infinite scalability
  - In the long term, scalability problems may be expected to aggravate when
    - Cloud providers may serve an increasing number of on-line services, and
    - Services become accessed by massive amounts of global users at all times.
- **Lack of interoperability among cloud providers**

- Contemporary cloud technologies have not been designed with interoperability in mind.
    - Cannot scale through business partnerships across clouds providers.
    - Prevents small and medium cloud infrastructure providers from entering the cloud provisioning market.
- This locks consumers to a single vendor.

- **No Built-In Business Service Management Support**. Business Service Management (BSM) is a management strategy that allows businesses to align their IT management with their high-level business goals. The key aspect of BSM is service-level agreement (SLA) management. Current cloud computing solutions are not designed to support the BSM practices that are well established in the daily management of the enterprise IT departments.

A Use case

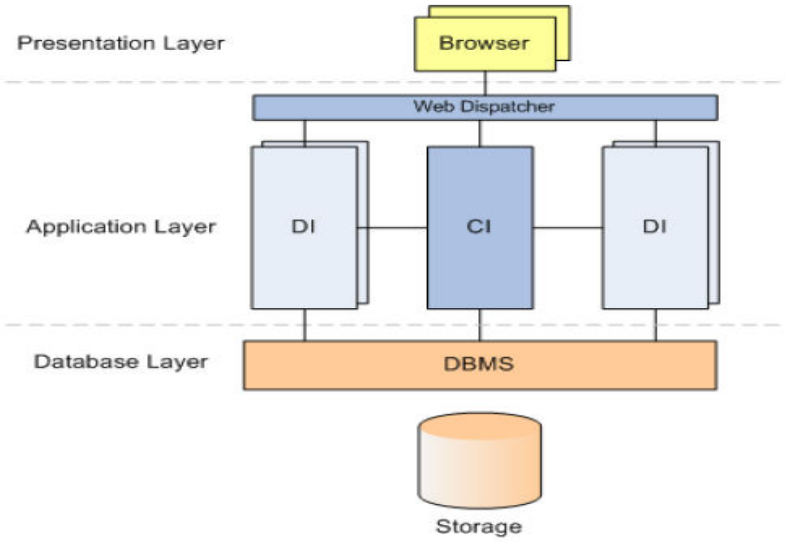


Figure 1: Abstraction of an SAP System

About SAP System

- SAP systems are used for a variety of business applications that differ by version and functionality

  EX. CRM, ERP

An SAP system is a typical three-tier system

- Requests are handled by the SAP Web dispatcher
- In the middle tier, there are two types of components :

- Multiple stateful Dialog Instances (DIs)
- Single Central Instance (CI) that performs central services such as application level locking, messaging, and registration of DIs. The number of DIs can be changed while the system is running to adapt to load
- A single Database Management System (DBMS) serves the SAP system.

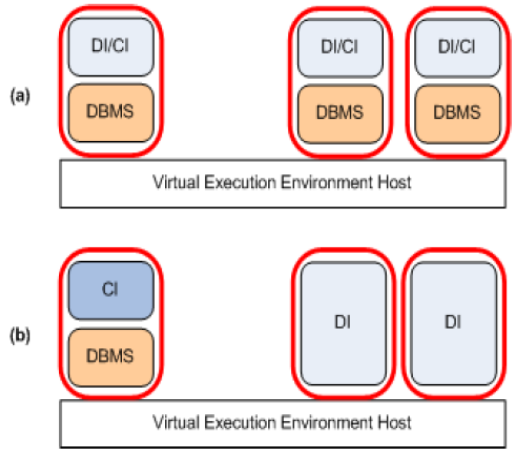


Figure 2: Sample SAP system deployments: (a) all components run in the same virtual execution environment (represented as rounded rectangles); (b) the large components (CI and DBMS) run each on a dedicated virtual execution environment. The Virtual Execution Environment Host refers to the set of components managing the virtual environments.

Primary Requirements from cloud Infrastructure

- **Automated and fast deployment**
  - Automated provisioning of service applications based on a formal contract specifying the infrastructure SLAs.
  - The same contract should be reused to provision multiple instances of the same application for different tenants with different customizations.
- **Dynamic elasticity(seamless)**

Resource allocation parameters of individual virtual execution environments(memory, CPU, network bandwidth, storage) should be adjusted dynamically

  - The number of virtual execution environments should be adjusted dynamically to adapt to the changing load
- **Automated continuous optimization**
  - Continuously optimize alignment of infrastructure resources management with the high-level business goals
- **Virtualization technology independence**
  - Support different virtualization technologies transparently

BASIC PRINCIPLES OF CLOUD COMPUTING

- **Federation**: All cloud computing providers, regardless of how big they are, have a finite capacity. To grow beyond this capacity, cloud computing providers should be able to form federations of providers such that they can collaborate and share their resources.

Any federation of cloud computing providers should allow virtual application to be deployed across federated sites. Furthermore, virtual applications need to be completely location free and allowed to migrate in part or as a whole between sites.

- **Independence** Users should be able to use the services of the cloud without relying on any provider specific tool, and cloud computing providers should be able to manage their infrastructure without exposing internal details to their customers or partners.
- **Isolation**: Cloud computing services are, by definition, hosted by a provider that will simultaneously host applications from many different users.
  Users must be ensured that their resources cannot be accessed by others sharing the same cloud and that adequate performance isolation is in place to ensure that no other user may possess the power to directly effect the service granted to their application.
- **Elasticity:** One of the main advantages of cloud computing is the capability to provide, or release, resources on-demand. These "elasticity" capabilities should be enacted automatically by cloud computing providers to meet demand variations
- **Business Orientation**: Before enterprises move their mission critical applications to the cloud, cloud computing providers will need to develop the mechanisms to ensure quality of service (QoS) and proper support for service-level agreements (SLAs).
- **Trust:** Mechanisms to build and maintain trust between cloud computing consumers and cloud computing providers, as well as between cloud computing providers among themselves, are essential for the success of any cloud computing offering

The RESERVOIR Approach

- The RESERVOIR vision is to
  - enable on-demand delivery of IT services at competitive costs,

  without requiring a large capital investment in infrastructure.
- The model is inspired by a strong desire to liken the delivery of IT services to the delivery of common utilities
  - EX. dynamically acquire electricity from a neighboring facility to meet a spike in demand.
    That is, no single provider serves all customers at all times
  - Next-generation cloud computing infrastructure should support a model that Multiple independent providers can cooperate seamlessly to maximize their benefit.
  - Informally, we refer to the infrastructure that supports this paradigm as a federated cloud.

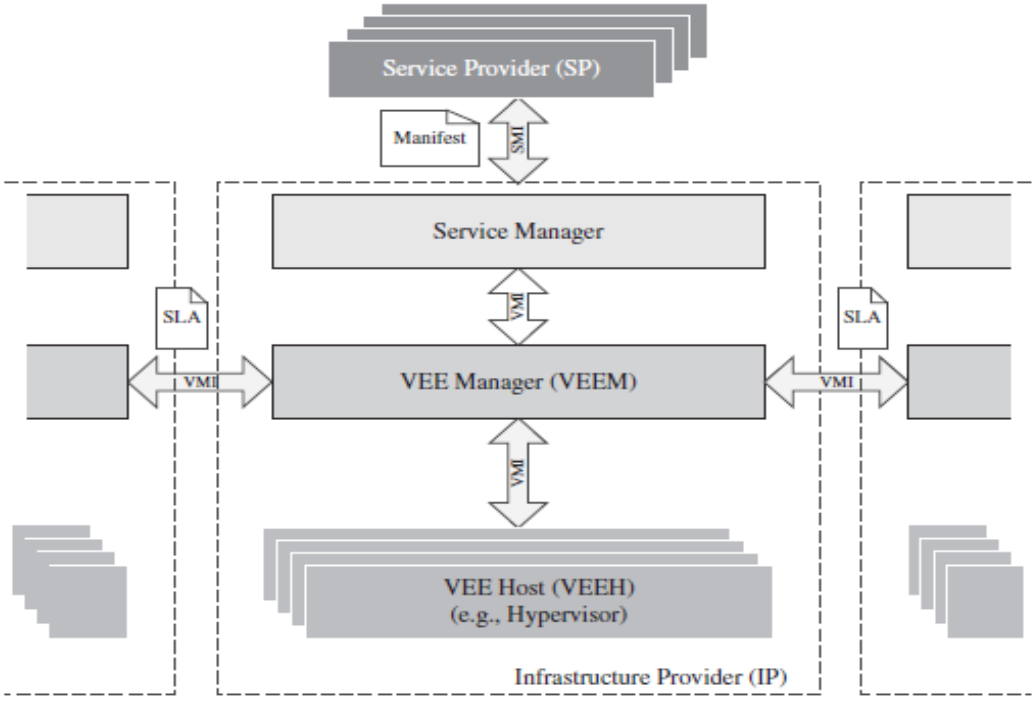
The RESERVOIR Model for Federated Cloud Computing

- Two roles in the RESERVOIR model:
  - **Service providers**:
    - The entities that understand the needs of a particular business
    - Offer service applications to address those needs.

- Do not own the computational resources.

– **Infrastructure providers**:
  - Operate RESERVOIR sites that own and manage the physical infrastructure on which service applications execute.

RESERVOIR Architecture



- A Service Application is a set of software components that work collectively to achieve a common goal.
- Each component of such service applications executes in a dedicated VEE.
- SPs deploy service applications in the cloud by providing to a IP, known as the primary site, with a Service Manifest—that is, a document that defines the structure of the application as well as the contract and SLA between the SP and the IP.
- 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
- **Virtual Execution Environment Host (VEEH)** contains virtualized computational resources, alongside the virtualization layer and all the management enablement components
- It 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.).

- **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 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.
- Different implementations of each layer will be able to interact with each other.
- The **Service Management Interface (SMI)** with its service manifest exposes a standardized interface into the RESERVOIR cloud for service providers.
- The **VEE Management Interface (VMI)** simplifies the introduction of different and independent IT optimization strategies without disrupting other layers or peer VEEMs
- The **VEE Host Interface (VHI)** will support plugging-in of new virtualization platforms (e.g., hypervisors), without requiring VEEM recompilation or restart

Features of Federation Types

- **Framework agreement support**: Framework agreements may either be supported by the architecture or not. If framework agreements are not supported, this implies that federation may only be carried out in a more ad hoc opportunistic manner
- **Opportunistic placement support**: If framework agreements are not supported by the architecture, or if there is not enough spare capacity even including the framework agreements, a site may choose to perform opportunistic placement. It is a process where remote sites are queried on-demand as the need for additional resources arises, and the local site requests a certain SLA-governed capacity for a given cost from the remote sites
- **Advance resource reservation support**: This feature may be used both when there is an existing framework agreement and when opportunistic placement has been performed. Both types of advance reservations are only valid for a certain time, since they impact the utilization of resources at a site. Because of this impact, they should be billed as actual usage during the active time interval.
- **Federated migration support**: The ability to migrate machines across sites defines the federated migration support. There are two types of migration: cold and hot (or live).
  In **cold migration**, the VEE is suspended and experiences a certain amount of downtime while it is being transferred.

Hot or live migration does not allow for system downtime, and it works by transferring the runtime state while the VEE is still running.

- **Cross-site virtual network support**: VEEs belonging to a service are potentially connected to virtual networks, if be requested by the SP. Ideally, these virtual networks will span across sites.

The federation can offer public IP addresses retention post cross-site migration. With fully virtualized networks, this may be a directly supported feature; but even if virtualized networks are not available, it may still be possible to maintain public IP addresses by manipulating routing information

Information disclosure within the federation has also to be taken into account. The sites in the federation may provide information to different degrees (for instance, the information exchange between sites may be larger within the same administrative domain than outside it).

Information regarding deployed VEEs will be primarily via the monitoring system, whereas some information may also potentially be exposed via the VMI as response to a VEE deployment request.

VMI operation support: Depending on the requirements of the federation scenario, only a subset of the VMI operations may be made available.

Which operations are required may be related to the amount of information that is exposed by the remote sites; access to more information may also increase the possibility and need to manipulate the deployed VEEs.

Federation Scenarios

- **The baseline federation scenario** provides opportunistic placement of VEEs at a remote site. Migration is not supported, nor does it resize the VEEs once placed at the remote site. Advanced features such as virtual networks across site boundaries are also not supported.

The basic federation scenario includes framework agreements, cold migration, and retention of public IP addresses.

Notably missing is

(a) support for hot migration and

(b) cross-site virtual network functionality.

This scenario offers a useful cloud computing federation with support for site collaboration in terms of framework agreements without particularly high technological requirements on the underlying architecture in terms of networking support.

- **The advanced federation scenario** offers advanced functionality such as cross-site virtual network support. The feature most notably missing is hot migration, and the monitoring system also does not disclose VEE sub state metadata information.

- **The full featured federation scenario** offers the most complete set of features, including hot migration of VEEs.

SLA Management in Cloud Computing

- **Capacity planning:** The activity of determining the number of servers and their capacity that could satisfactorily serve the application end-user requests at peak loads

  An example scenario where two web applications, application A and application B, are hosted on a separate set of dedicated servers within the enterprise-owned server rooms is shown in Figure 16.1. The planned capacity for each of the applications to run successfully is three servers. As the number of web applications grew, the server rooms in the organization became large and such server rooms were known as data centers. These data centers were owned and managed by the enterprises themselves

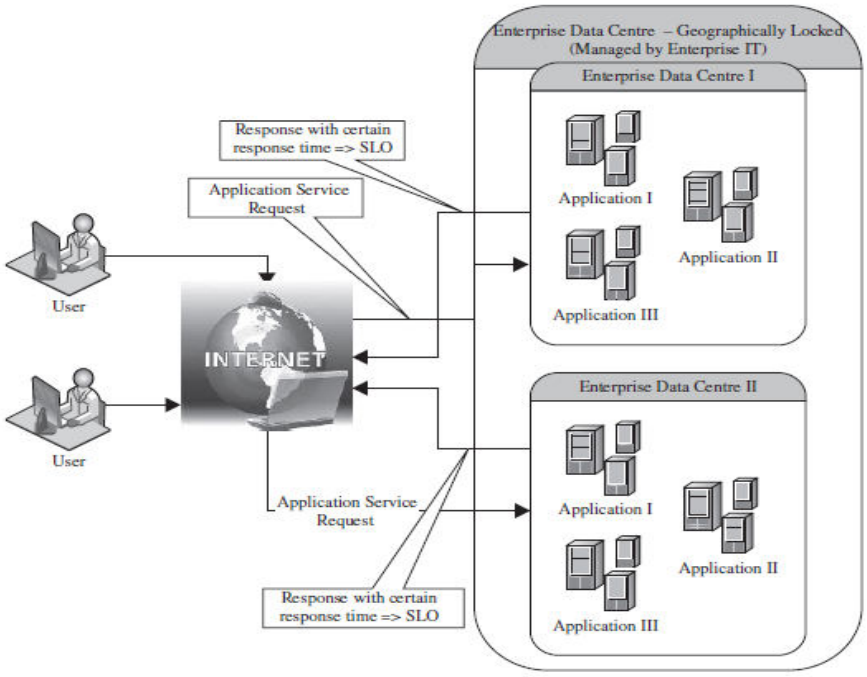


FIGURE 16.1. Hosting of applications on servers within enterprise's data centers.

- As the number and complexity of the web applications grew, enterprises realized that it was economical to outsource the application hosting activity to third-party infrastructure providers
- These providers get the required hardware and make it available for application hosting.
- It necessitated the enterprises to enter into a legal agreement with the infrastructure service providers to guarantee a minimum quality of service (QoS).
- The QoS parameters are related to the availability of the system CPU, data storage, and network for efficient execution of the application at peak loads.
- This legal agreement is known as the **service-level agreement (SLA).**

Example of SLA

- one SLA may state that the application's server machine will be available for 99.9% of the key business hours of the application's end users, also called core time, and 85% of the non-core time.
- Another SLA may state that the service provider would respond to a reported issue in less than 10 minutes during the core time, but would respond in one hour during non-core time.

- These SLAs are known as the infrastructure SLAs, and the infrastructure service providers are known as Application Service Providers (ASPs).
- This scenario is depicted in Figure, wherethe enterprise applications are hosted on the dedicated servers belonging to an ASP.

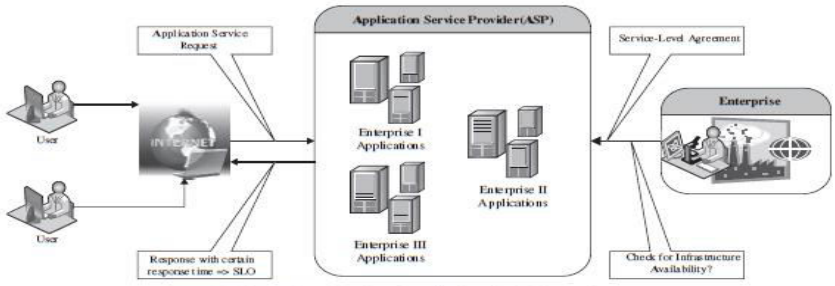


FIGURE 16.2. Dedicated hosting of applications in third party data centers.

- To reduce the redundancies and increase the server utilization in data centers, ASPs started co-hosting applications with complementary workload patterns. Co-hosting of applications means deploying more than one application on a single server
- Virtualization technologies have been proposed to overcome the problems of security and performance isolation
- Adoption of virtualization technologies required ASPs to get more detailed
- insight into the application runtime characteristics with high accuracy. Based on these characteristics, ASPs can allocate system resources more efficiently to these applications on-demand, so that application-level metrics can be monitored and met effectively. These metrics are request rates and response times.

  Therefore, different SLAs than the infrastructure SLAs are required. These SLAs are called application SLAs. These service providers are known as Managed Service Providers (MSP) because the service providers were responsible for managing the application availability too

  This scenario is shown in Figure, where both application A and application B share the same set of virtualized servers.

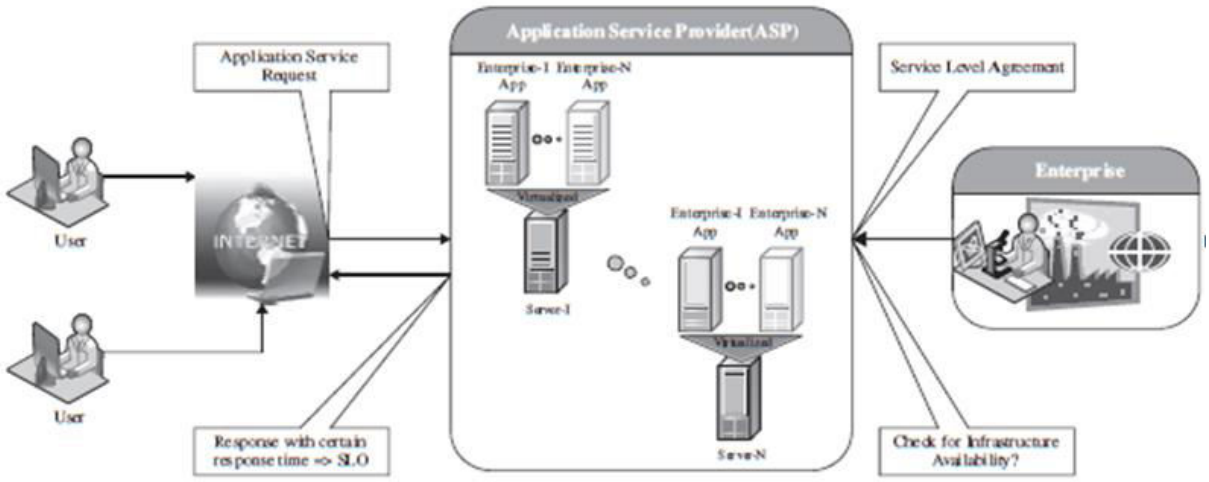


FIGURE 16.4. Shared hosting of applications on virtualized servers within ASP's data centers.

TRADITIONAL APPROACHES TO SLO MANAGEMENT

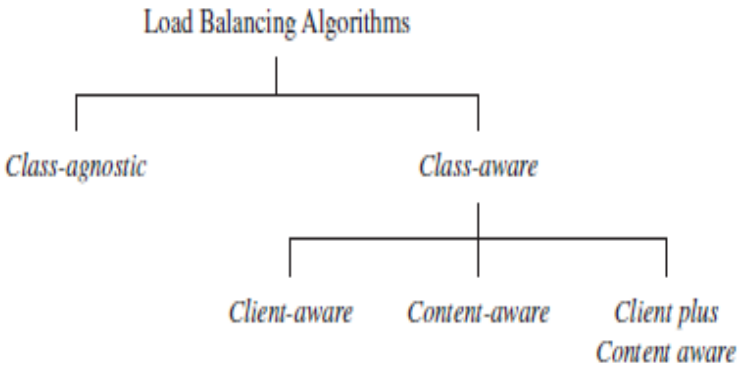
Traditionally, **load balancing** techniques and **admission control** mechanisms have been used to provide guaranteed quality of service (QoS) for hosted web applications

- **Load Balancing**: The objective of a load balancing is to distribute the incoming requests onto a set of physical machines, each hosting a replica of an application, so that the load on the machines is equally distributed.

The load balancing algorithm executes on a physical machine that interfaces with the clients. This physical machine, also called the front-end node, receives the incoming requests and distributes these requests to different physical machines for further execution.

This set of physical machines is responsible for serving the incoming requests and are known as the back-end nodes.

Categories of Load Balancing Algorithms

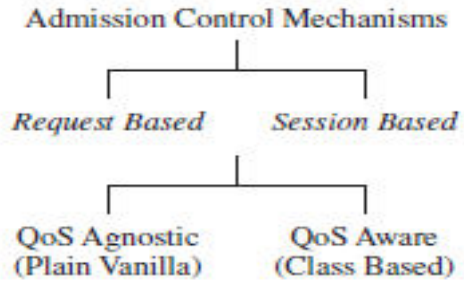


• **Class-agnostic**: The front-end node is neither aware of the type of client from which the request originates nor aware of the category (e.g., browsing, selling, payment, etc.) to which the request belongs to.

• **Class-aware**: The front-end node must additionally inspect the type of client making the request and/or the type of service requested before deciding which back-end node should service the request.
• **Admission Control:** Deciding the set of requests that should be admitted into the application server when the server experiences "very" heavy loads.

During overload situations, since the response time for all the requests would invariably degrade if all the arriving requests are admitted into the server, it would be preferable to be selective in identifying a subset of requests that should be admitted into the system so that the overall pay-off is high.

• The objective of admission control mechanisms, therefore, is to police the incoming requests and identify a subset of incoming requests that can be admitted into the system when the system faces overload situations.



- **Request-based admission control** algorithms reject new requests if the servers are running to their capacity.

The disadvantage with this approach is that a client's session may consistof multiple requests that are not necessarily unrelated. Consequently, some requests are rejected even if there are others that are honored.

- **Session-based admission control** mechanisms try to ensure that longer sessions are completed and any new sessions are rejected. Accordingly, once a session is admitted into the server, all future requests belonging to that session are admitted as well, even though new sessions are rejected by the system.

- Furthermore, the decision to reject a request can depend on the type of user making the request or the nature of the request being made.
- For example, a new request or a new session initiated by a high-priority user may be admitted while the requests from low priority users are rejected.
- Similarly, requests that are likely to consume more system resources can be rejected during overload situations.
- Such admission control mechanisms are called **QoS-aware control** mechanisms

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.
- It outlines the broad understanding between the service provider and the service consumer for conducting business and forms the basis for maintaining a mutually beneficial relationship.
- From a legal perspective, the necessary terms and conditions that bind the service provider to provide services continually to the service consumer are formally defined in SLA.
- SLA can be modeled using web service-level agreement (WSLA) language Specification. Service-level parameter, metric, function, measurement directive, service-level objective, and penalty are some of the important components of WSLA

Key Components of a Service-Level Agreement

• Service-Level Parameter	Describes an observable property of a service whose value is measurable
• Metrics	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
• Measurement directives	These specify how to measure a metric

There are two types of SLAs from the perspective of application hosting

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-allocatingcomputing resources among the co-located applications. Therefore, the service providers are also responsible for ensuring to meet their customer's application SLOs

Key Contractual Elements of an Infrastructural SLA

• Hardware availability	99% uptime in a calendar month
• Power availability	99.99% of the time in a calendar month
• Data center network	99.99% of the time in a calendar month availability
• Backbone network	99.999% of the time in a calendar month availability
• Service credit for unavailability	Refund of service credit prorated on downtime period
• Outage notification	Notification of customer within 1 hr of complete downtime

guarantee

- Internet latency — When latency is measured at 5-min intervals to an upstream guarantee provider, the average doesn't exceed 60 msec
- Packet loss guarantee — Shall not exceed 1% in a calendar month

Key contractual components of an application SLA

- Service-level parameter — Web site response time (e.g., max of 3.5 sec per user request)

metric

Latency of web server (WS) (e.g., max of 0.2 sec per request)

Latency of DB (e.g., max of 0.5 sec per query)

- Function — Average latency of WS=(latency of web server 1+latency of web server 2) /2

Web site response time= Average latency of web server+latency of database

- Measurement directive — DB latency available via http://mgmtserver/em/latency

WS latency available via http://mgmtserver/ws/instanceno/ latency

- Service-level objective — Service assurance web site latency , 1 sec when concurrent connection , 1000
- Penalty — 1000 USD for every minute while the SLO was breached

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 thefollowing five phases:

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

• **Contract Definition**: 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, thisphase 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.

- **Operationalization:** SLA operation consists of
- **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. As part of accounting, the application's actual performance and the performance guaranteed as a part of SLA is reported. Apart from the frequency and the duration of the SLA breach, it should also provide the penalties paid for each SLA violation
- **SLA enforcement** involves taking appropriate action when the runtime monitoring detects a SLA violation. Such actions could be notifying the concerned parties, charging the penalties besides other things.
  The different policies can be expressed using a subset of the Common Information Model (CIM). The CIM model is an open standard that allows expressing managed elements of data center via relationships and common objects.
- **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.

SLA MANAGEMENT IN CLOUD

SLA management of applications hosted on cloud platforms involves five phases.

1. Feasibility
2. On-boarding
3. Pre-production
4. Production
5. Termination

I. **Feasibility Analysis**: MSP conducts the feasibility study of hosting an application on their cloud platforms.

This study involves three kinds of feasibility:

1. Technical feasibility
2. Infrastructure feasibility, and
3. Financial feasibility

1. **Technical Feasibility**: The technical feasibility of an application implies determining the following:

1. Ability of an application to scale out.
2. Compatibility of the application with the cloud platform being used within the MSP's data center.
3. The need and availability of a specific hardware and software required for hosting and running of the application.
4. Preliminary information about the application performance and whether they can be met by the MSP.

2. **Infrastructure feasibility** involves determining the availability of infrastructural resources in sufficient quantity so that the projected demands of the application can be met

3. **Financial feasibility** study involves determining the approximate cost to be incurred by the MSP and the price the MSP charges the customer so that the hosting activity is profitable to both of them

A feasibility report consists of the results of the above three feasibility studies. The report forms the basis for further communication with the customer.

Once the provider and customer agree upon the findings of the report, the outsourcing of the application hosting activity proceeds to the next phase, called "onboarding" of application.

Only the basic feasibility of hosting an application has been carried in this phase. However, the detailed runtime characteristics of the application are studied as part of the on-boarding activity.

II. **On-Boarding of Application:** Once the customer and the MSP agree in principle to host the application based on the findings of the feasibility study, the application is moved from the customer servers to the hosting platform. Moving an application to the MSP's hosting platform is called **on-boarding**

As part of the on-boarding activity, the MSP understands the application runtime characteristics using runtime profilers.

This helps the MSP to identify the possible SLAs that can be offered to the customer for that application.

This also helps in creation of the necessary policies (also called rule sets) required to guarantee the SLOs mentioned in the application SLA.

The application is accessible to its end users only after the onboarding activity is completed

- On-boarding activity consists of the following steps:

a. Packing of the application for deploying on physical or virtual environments. Application packaging is the process of creating deployable components on the hosting platform

b. The packaged application is executed directly on the physical servers to capture and analyze the application performance characteristics. It allows the functional validation of customer's application. Additionally, it helps to identify the nature of application—that is, whether it is CPU-intensive or I/O intensive or network-intensive and the potential performance bottlenecks

c. The application is executed on a virtualized platform and the application performance characteristics are noted again. Important performance characteristics like the application's ability to scale (out and up) and performance bounds (minimum and maximum performance) are

noted

d. Based on the measured performance characteristics, different possible SLAs are identified. The resources required and the costs involved for each SLA are also computed

e. Once the customer agrees to the set of SLOs and the cost, the MSP starts creating different policies required by the data center for automated management of the application. These policies are of three types:

(1) Business policies help prioritize access to the resources in case of contentions. Business policies are in the form of weights for different customers or group of customers.

(2) Operational policies are the actions to be taken when different thresholds/conditions are reached. Also, the actions when thresholds/ conditions/triggers on service-level parameters are breached or about to be breached are defined.

(3) Provisioning: The corrective action could be different types of provisioning such as scale-up, scale-down, scale-out, scale-in, and so on, of a particular tier of an application. Additionally, notification and logging action are also defined.

Operational policies (OP) are represented in the following format:

OP = collection of <Condition, Action>

Ex: OP <average latency of web server > 0.8 sec, scale-out the web-server tier>

It means, if average latency of the web server is more than 0.8 sec then automatically scale out the web-server tier. On reaching this threshold, MSP should increase the number of instances of the web server.

A provisioning policy (PP) is represented as

PP = collection of <Request, Action>

For example, a provisioning policy to start a web site consists of the following sequence: start database server, start web-server instance 1, followed by start the web-server instance 2, and so on.

On defining these policies, the packaged applications are deployed on the cloud platform and the application is tested to validate whether the policies are able to meet the SLA requirements.

This step is iterative and is repeated until all the infrastructure conditions necessary to satisfy the application SLA are identified.

Once the different infrastructure policies needed to guarantee the SLOs mentioned in the SLA are completely captured, the on-boarding activity is said to be completed.

III. Preproduction: Once the determination of policies is completed, the application is hosted in a simulated production environment.

It facilitates the customer to verify and validate the MSP's findings on application's runtime characteristics and agree on the defined SLA.

Once both parties agree on the cost and the terms and conditions of the SLA, the customer sign-off is obtained.

On successful completion of this phase the MSP allows the application to go on-live.

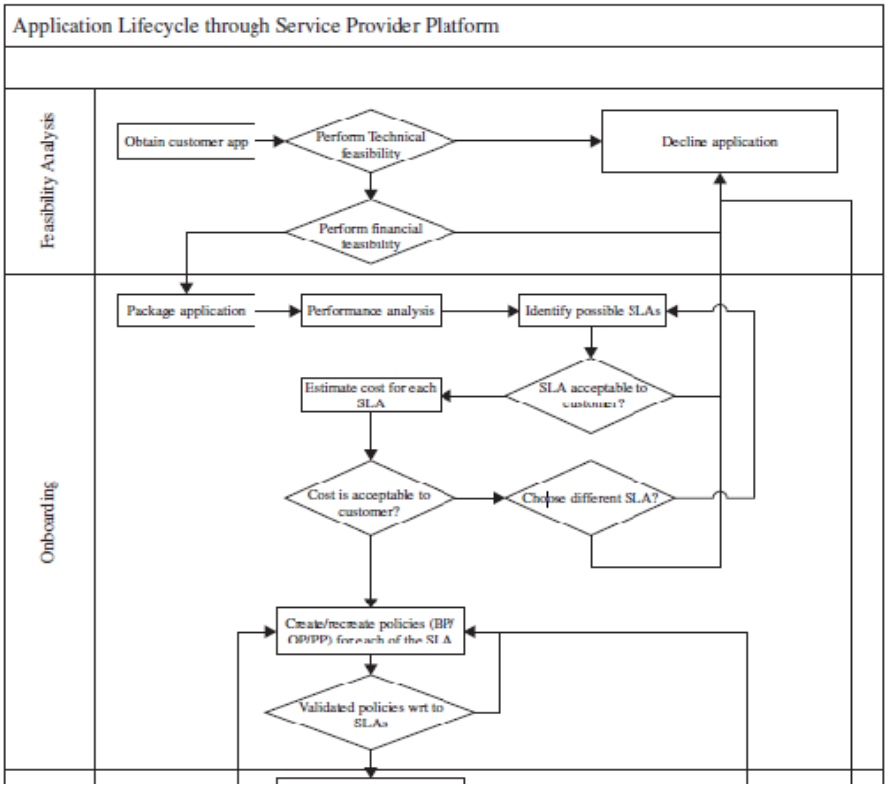
IV. Production: In this phase, the application is made accessible to its end users under the agreed SLA.

However, there could be situations when the managed application tends to behave differently in a production environment compared to the preproduction environment.

This in turn may cause sustained breach of the terms and conditions mentioned in the SLA.

Additionally, customer may request the MSP for inclusion of new terms and conditions in the SLA. If the application SLA is breached frequently or if the customer requests for a new non-agreed SLA, the on-boarding process is performed again.
In the case of the former, on-boarding activity is repeated to analyze the application and its policies with respect to SLA fulfillment.
In case of the latter, a new set of policies are formulated to meet the fresh terms and conditions of the SLA.

V. Termination: When the customer wishes to withdraw the hosted application and does not wish to continue to avail the services of the MSP for managing the hosting of its application, the termination activity is initiated.
On initiation of termination, all data related to the application are transferred to the customer and only the essential information is retained for legal compliance.
This ends the hosting relationship between the two parties for that application, and the customer sign-off is obtained



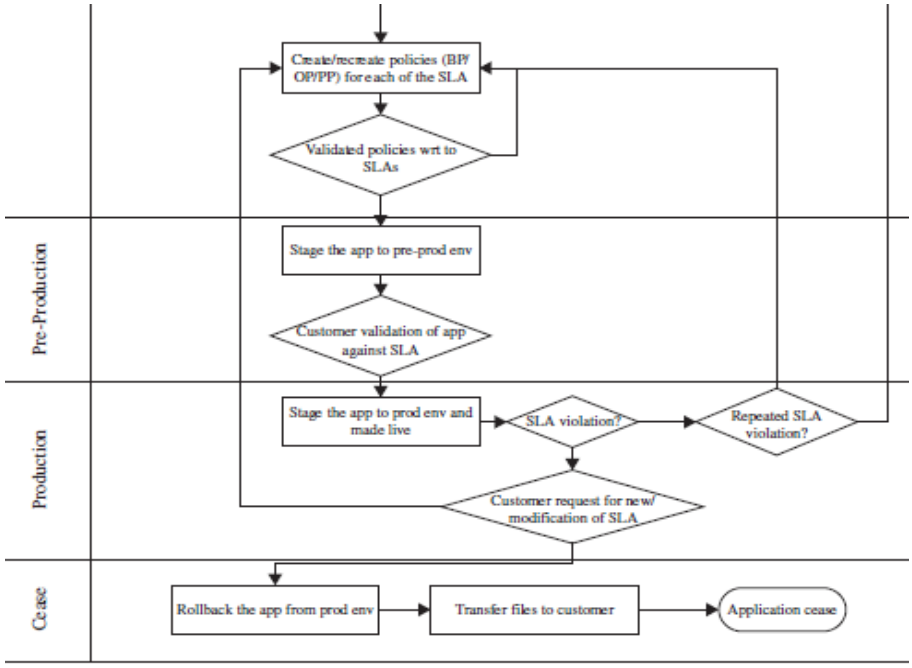


FIGURE 16.7. Flowchart of the SLA management in cloud.

HPC on Clouds Performance Prediction

High-performance computing (**HPC**) is the use of parallel processing for running advanced application programs efficiently, reliably and quickly.

HPC on Cloud

- Cloud computing may be exploited at three different levels: IaaS (Infrastructure as a Service), PaaS (Platform as a Service), and AaaS (Application as a Service).
- In one way or another, all of them can be useful for HPC.
- But, IaaS lets users run applications on fast pay-per-use machines they don't want to buy, to manage, or to maintain.
- An IaaS cloud environment uses a virtualization engine. Basically, this engine provides by means of a hypervisor the illusion of multiple independent replicas of every physical machine in the cloud.
- Furthermore, the total computational power can be easily increased (by additional charge).
- For the HPC user, this solution is undoubtedly attractive:
- no investment in rapidly-obsolescing machines

- no power and cooling nightmares, and
- no system software updates
- HPC users usually exploit parallel hardware, and so they would like to get parallel hardware to execute their explicitly-parallel applications.
- They want to receive from the cloud a (possibly high) number of powerful machines with fast interconnect that could resemble a high-performance computing cluster.
- Stated another way, they exploit the cloud as a provider of cluster on-demand (CoD) systems.
- They obtain clusters that they can configure according to their software requirements.
- This is possible since these are in fact virtual clusters, whose management (even in terms of the number of nodes and their configurations) is completely delegated to the cloud user.

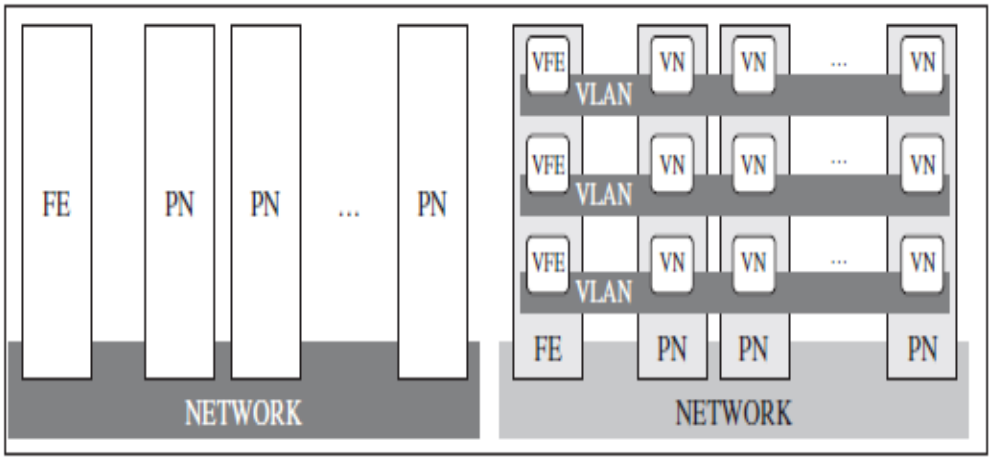
The adoption of the cloud paradigm in HPC is related to the evaluation (and, possibly, to the reduction) of possible performance losses compared to physical HPC hardware.

In clouds, performance penalties may appear at two different levels:

- **Virtual Engine (VE):** These are related to the performance loss introduced by the virtualization mechanism. They are strictly related to the VE technology adopted.
- **Cloud Environment (CE):** These are the losses introduced at a higher level by the cloud environment, and they are mainly due to overheads and to the sharing of computing and communication resources.

The actual hardware used in the cloud, along with the losses at the VE and CE levels, will determine the actual performance of applications running in the cloud

Physical and Virtual Cluster



- A physical cluster is made up of a front-end (typically used only for administration purposes, often the only node with a public IP address) and a number of (physical) processing nodes. These are provided with a single CPU or with multiple CPUs sharing a common memory and I/O resources.
- A physical cluster can execute multiple jobs in parallel, by assigning to every job a subset of the total number of CPUs.
- A parallel application running in a physical cluster is composed of processes.
- To exploit all the available computing resources, the application should use at least a number of processes equal to the number of available CPUs (or, in the case of concurrent jobs, equal to the number of CPU exclusively reserved for the job).
- The main problem with physical clusters is that all jobs running on the cluster, whether concurrent or non-concurrent, have to share the same operating system (OS), the system and application libraries, and the operating environment.
- The frequently recurring requirements for mutually exclusive or incompatible libraries and support software make physical cluster management a nightmare for system administrators.

A virtual cluster is made up of a virtual front-end and a number of virtual Nodes. Virtual front-ends are obtained by virtualization of a physical front end machine, and virtual nodes are obtained by virtualization of physical processing nodes

Virtual cluster may have an execution environment of its own (OS, libraries, tools, etc.) that is loaded and initialized when the cluster is created.

The advantages of cluster virtualization are:

- Every application can set up a proper execution environment, which does not interfere with all other applications and virtual clusters running on the hardware.
- The network traffic of every virtual cluster is encapsulated in a separate VLAN

Every virtual processing node can host one or several virtual machines(VMs), each running a private OS instance. These may belong to the same or to different virtual clusters

In turn, each VM is provided with several virtual CPUs (VCPUs). A virtual machine manager running in every node makes it possible to share the physical CPUs among the VCPUs defined on the node (which may belong to a single virtual cluster or to several virtual clusters)

Given a physical node provided with n CPUs, there are two possibilities to exploit all the computing resources available:

- Using n VMs (each running its OS instance) with one, or even several, VCPUs;
- Using a single VM with at least n VCPU

Grid vs Cloud

From the HPC point of view

- Cloud environments are a centralized resource of computational power.
- On the other hand, the grid paradigm proposes a distributed approach to computational resources, "gluing" together distributed data centers to build up a computational grid, accessible in a simple and standardized way.
- Their objective is the same: offering computational power to final users. But this is obtained following two different approaches:
- centralized for clouds and
- distributed for grids.

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.
- 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.
- Cloud supporters claim that the cloud is easy to be used, is scalable, and always gives users exactly what they want.
- On the other hand, grid is difficult to be used, does not give performance guarantees, is used by narrow communities of scientists to solve specific problems, and does not actually support interoperability

- Grid supporters answer that grid users do not need a credit card, that around the world there are many examples of successful projects, and that a great number of computing nodes connected across the net execute large scale scientific applications, addressing problems that could not be solved otherwise

Grid and Cloud Integration

The integration of cloud and grid, or at least their integrated utilization, has been proposed since there is a trade-off between application turnaround and system utilization, and sometimes it is useful to choose the rightcompromise between them

Two main approaches have been proposed:

- **Grid on Cloud**. A cloud IaaS (Infrastructure as a Service) approach isadopted to build up and to manage a flexible grid system . Doing so,the grid middleware runs on a virtual machine. Hence the main drawbackof 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

HPC IN THE CLOUDPERFORMANCE-RELATED ISSUES

The adoption of the cloud paradigm for HPC is a flexible way to deploy (virtual) clusters dedicated to execute HPC applications

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

In the latter, every time that a task is started, the user will be charged for the used resources. But it is very hard to know in advance which will be the resource usage and hence the cost. On the other hand, even if the global expense for a physical cluster is higher, once the system has been acquired, all the costs are fixed and predictable

In clouds, performance counts two times. Low performance means not only long waiting times, but also high costs
The total cost is given by
<cost per hour per instance>* <number of instances>*<hours>

The use of alternative cost factors (e.g., the RAM memory allocated, as for GoGrid, leads to completely different considerations and requires different application optimizations to reduce the final cost of execution.

The typical HPC user would like to know how long his application will run on the target cluster and which configuration has the highest performance/cost ratio.

The advanced user, on the other hand, would also know if there is a way to optimize its application so as to reduce the cost of its run without sacrificing performance.

The high-end user, who cares more for performance than for the cost to be sustained, would like instead to know how to choose the best configuration to maximize the performance of his application.

The **system dimensioning** is the choice of the system configuration fit for the user purposes (cost, maximum response time, etc.).
An HPC machine is chosen and acquired, aiming to be at the top of available technology and to be able to sustain the highest system usage that may eventually be required. This can be measured in terms of GFLOPS, in terms of number of runnable jobs, or by other indexes depending on the HPC applications that will be actually executed.

In other words, the dimensioning is made by considering the peak system usage. It takes place at system acquisition time, by examining the machine specifications or by assembling it using hardware components of known performance.

In clouds, instead, the system must be dimensioned by finding out an optimal trade-off between application performance and used resources.

The optimality is a concept that is fairly different, depending on the class of users.
Someone would like to obtain high performance at any cost, whereas others would privilege economic factors

TABLE 17.2. Differences Between "Classical" HPC and HPC in Cloud Environments

Problem	HPC	HPC in Clouds
Cost	Buy-and-maintain paradigm	Pay-per-use paradigm
Performance optimization	Tuning of the application to the hardware	Joint tuning of application and system
System dimensioning	At system acquisition time, using global performance indexes under system administrator control	At every application execution, using application oriented performance indexes, under user control

Supporting HPC in the Cloud
To support HPC applications, a fundamental requirement from a cloud provider is that an adequate service-level agreement (SLA) is granted.

For HPC applications, the SLA should be different from the ones currently offered for the most common uses of cloud systems, oriented at transactional Web applications.

The SLA should offer guarantees useful for the HPC user to predict his application performance behavior and hence to give formal (or semiformal) statements about the parameters involved.

At the state of the art, cloud providers offer their SLAs in the form of a contract (hence in natural language, with no formal specification).

TABLE 17.3. Service-Level Agreement of GoGrid Network

Parameter	Description	GoGrid SLA
Jitter	Variation in latency	< 0.5msec
Latency	Amount of time it takes for a packet to travel from one point to another	< 5 msec
Maximum jitter	Highest permissible jitter within a given period when there is no network outage	10 msec within any 15-min period
Network outage	Unscheduled period during which IP services are not useable due to capacity-constraints or hardware failures	None
Packet loss	Latency in excess of 10 seconds	< 0.1%

BEST PRACTICES IN ARCHITECTINGCLOUD APPLICATIONS IN THEAWS CLOUD

Business Benefits of Cloud Computing

There are some clear business benefits to building applications in the cloud:

- **Almost Zero Upfront Infrastructure Investment**. If we have to build a large scale system, it may cost to invest in real estate, physical security, hardware (racks, servers, routers, backup power supplies), hardware management (power management, cooling), and operations personnel. Because of the high upfront costs, the project would typically require several rounds of management approvals before the project could even get started. Now, with utility-style cloud computing, there is no fixed cost or startup cost.
- **Just-in-Time Infrastructure**: By deploying applications in-the-cloud with just-in-time self-provisioning, we do not have to worry about pre-procuring capacity for large-scale systems. This increases agility, lowers risk, and lowers operational cost because you scale only as you grow and only pay for what you use.

- **More Efficient Resource Utilization:** System administrators usually worry about procuring hardware (when they run out of capacity) and higher infrastructure utilization (when they have excess and idle capacity). With the cloud, they can manage resources more effectively and efficiently by having the applications request and relinquish resources on-demand

- **Usage-Based Costing:** With utility-style pricing, you are billed only for the infrastructure that has been used

- **Reduced Time to Market:** Parallelization is one of the great ways to speed up processing. Having available an elastic infrastructure provides the application with the ability to exploit parallelization in a cost-effective manner reducing time to market

Technical Benefits of Cloud Computing

Some of the technical benefits of cloud computing includes:

- **Automation**—"Scriptable Infrastructure": create repeatable build and deployment systems by leveraging programmable (API-driven) infrastructure.

- **Auto-scaling**: scale applications up and down to match unexpected demand without any human intervention. Auto-scaling encourages automation and drives more efficiency.

- **Proactive Scaling**: Scale application up and down to meet anticipated demand with proper planning understanding of traffic patterns so that costs are kept low while scaling.

- **More Efficient Development Life Cycle**: Production systems may be easily cloned for use as development and test environments. Staging environments may be easily promoted to production.

- **Improved Testability**: We can inject and automate testing at every stage during the development process.without running out of hardware. "instant test lab" can be spawned with preconfigured environments only for the duration of testing phase

- **Disaster Recovery and Business Continuity**: The cloud provides a lower cost option for maintaining a fleet of servers and data storage. With the cloud, advantage of geo-distribution can be used to replicate the environment in other location within minutes

- **"Overflow" the Traffic to the Cloud**: With a few clicks and effective load balancing tactics, a complete overflow-proof application can be created by routing excess traffic to the cloud.

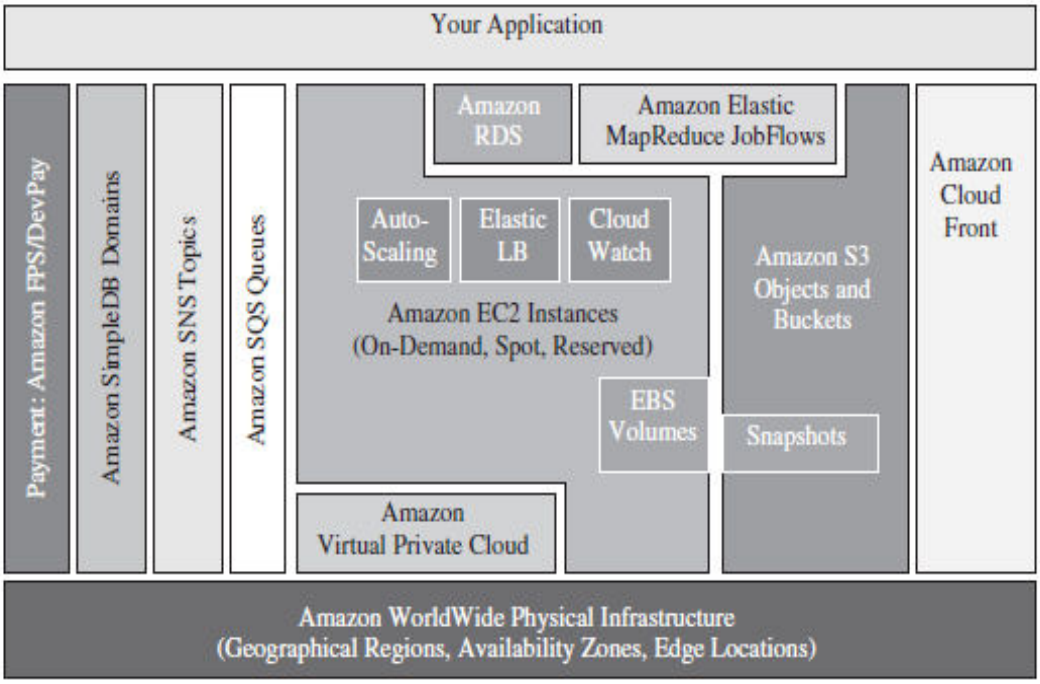
The Amazon Web Services Cloud

- The Amazon Web Services (AWS) cloud provides a highly reliable and scalable infrastructure for deploying Web-scale solutions, with minimal support and

administration costs, and more flexibility than we expect from our own infrastructure, either on-premise or at a datacenter facility

- AWS offers variety of infrastructure services

Amazon Web Services



- **Amazon Elastic Compute Cloud** (Amazon EC2) is a Web service that provides resizable compute capacity in the cloud.

- The operating system, application software, and associated configuration settings can be bundled into an Amazon machine image (AMI) and can use these AMIs to provision multiple virtualized instances as well as decommission them using simple Web service calls to scale capacity up and down quickly, as capacity requirement changes

- Users can purchase either
  (a) on-demand instances, in which they pay for the instances by the hour, r
  (b) reserved instances, in which they pay a low, one-time payment and receive a lower usage rate to run the instance than with an on-demand instance or spot instances where they can bid for unused capacity and further reduce cost

- Instances can be launched in one or more geographical **regions**.

- Each region has multiple **availability zones.**

- Availability zones are distinct locations that are engineered to be insulated from failures in other availability zones and provide inexpensive, low-latency network connectivity to other availability zones in the same region.

- **Elastic IP** addresses allows to allocate a static IP address and programmatically assign it to an instance.
- Users can enable monitoring on an Amazon EC2 instance using **AmazonCloudWatch** in order to gain visibility into resource utilization, operational performance, and overall demand patterns (including metrics such as CPU utilization, disk reads and writes, and network traffic).
- An **auto-scaling group** can be created using the auto-scaling feature to automatically scale capacity on certain conditions based on metric that Amazon CloudWatch collects
- Elastic load balancer can be created using the **Elastic Load Balancing service** to distribute incoming traffic
- **Amazon Elastic Block Storage (EBS)** volumes provide network-attached persistent storage to Amazon EC2 instances. Point-in-time consistent snapshots of EBS volumes can be created and stored on **Amazon Simple Storage Service (Amazon S3).**
- Amazon S3 is highly durable and distributed data store. With a simple Web services interface, users can store and retrieve large amounts of data as objects in buckets (containers) at any time, from anywhere on the Web using standard HTTP verbs.
- Copies of objects can be distributed and cached at **edge locations** around the world by creating a distribution using Amazon Cloud-Front service, a Web service for content delivery (static or streaming content)
- **Amazon SimpleDB**is a Web service that provides the core functionality of a database—real-time lookup and simple querying of structured data—without the operational complexity
- **Amazon Relational Database Service (Amazon RDS)** provides an easy way to set up, operate, and scale a relational database in the cloud. Users can launch a DB instance and get access to a full-featured MySQL database
- **Amazon Simple Queue Service (Amazon SQS)**is a reliable, highly scalable, hosted distributed queue for storing messages as they travel between computers and application components.
- **Amazon Elastic MapReduce**provides a hosted Hadoop framework running on the web-scale infrastructure of Amazon Elastic Compute Cloud (Amazon EC2) and Amazon Simple Storage Service (Amazon S3) and allows to create customized JobFlows. JobFlow is a sequence of MapReduce steps.
- **Amazon Simple Notifications Service (Amazon SNS)** provides a simple way to notify applications or people from the cloud by creating Topics and using a publish-subscribe protocol.

- **Amazon Virtual Private Cloud (Amazon VPC)**allows to extend corporate network into a private cloud contained within AWS. Amazon VPC uses an IPSec tunnel mode that enables to create a secure connection between a gateway in user's data center and a gateway in AWS

- AWS also offers various payment and billing services that leverages Amazon's payment infrastructure. All AWS infrastructure services offer utility-style pricing that require no long term commitments or contracts.

CLOUD BEST PRACTICES

The following best practices will help to build an application in the cloud

- **Design for Failure and Nothing Will Fail**

  Rule of Thumb: Be a pessimist when designing architectures in the cloud; assume things will fail. In other words, always design, implement, and deploy for automated recovery from failure.

  The following strategies can help in event of failure:

  1. Have a coherent backup and restore strategy for your data and automate it.
  2. Build process threads that resume on reboot.
  3. Allow the state of the system to re-sync by reloading messages from queues.
  4. Keep preconfigured and pre-optimized virtual images to support strategies 2 and 3 on launch/boot.
  5. Avoid in-memory sessions or stateful user context; move that to data stores

- AWS-Specific Tactics for Implementing This Best Practice

  1. Failover gracefully using Elastic IPs: Elastic IP is a static IP that is dynamically remappable
  2. Utilize multiple availability zones: Availability zones are conceptually like logical datacenters.
  3. Maintain an Amazon Machine Image so that you can restore and clone environments very easily in a different availability zone; maintain multiple database slaves across availability zones and set up hot replication.
  4. Utilize Amazon Cloud Watch to get more visibility and take appropriate actions in case of hardware failure or performance degradation.
  5. Utilize Amazon EBS and set up jobs so that incremental snapshots are automatically uploaded to Amazon S3 and data are persisted independent of instances.
  6. Utilize Amazon RDS and set the retention period for backups, so that it can perform automated backups.

- **Decouple your Components:** The cloud reinforces the SOA design principle that the more loosely coupled the components of the system, the bigger and better it scales.
- The key is to build components that do not have tight dependencies on each other, so that if one component were to die (fail), sleep (not respond), or remain busy (slow to respond) for some reason, the other components in the system are built so as to continue to work as if no failure is happening.
- Decoupling components, building asynchronous systems, and scaling horizontally become very important in the context of the cloud. It will not only allow scaling out by adding more instances of same component but will also allow to design innovative hybrid models in which a few

components continue to run in on-premise while other components can take advantage of the cloud scale and use the cloud for additional compute-power and bandwidth.

One can build a loosely coupled system using messaging queues. If a queue/ buffer is used to connect any two components together (Loose Coupling), it can support concurrency, high availability, and load spikes.

As a result, the overall system continues to perform even if parts of components are momentarily unavailable

AWS Specific Tactics for Implementing This Best Practice

1. Use Amazon SQS to isolate components

2. Use Amazon SQS as buffers between components

3. Design every component such that it expose a service interface and is responsible for its own scalability in all appropriate dimensions and interacts with other components asynchronously.

4. Bundle the logical construct of a component into an Amazon Machine Image so that it can be deployed more often.

5. Make applications as stateless as possible. Store session state outside of component (in Amazon SimpleDB, if appropriate).

- **Implement Elasticity:** Elasticity can be implemented in three ways:

1. **Proactive Cyclic Scaling:** Periodic scaling that occurs at fixed interval

(daily, weekly, monthly, quarterly).

2. **Proactive Event-Based Scaling:** Scaling just when expecting a big surge of traffic requests due to a scheduled business event (new product launch, marketing campaigns).

3. **Auto-scaling Based on Demand:** By using a monitoring service, system can send triggers to take appropriate actions so that it scales up or down based on metrics (utilization of the servers or network i/o)

To implement elasticity, one has to first automate the deployment process and streamline the configuration and build process. This will ensure that the system can scale without any human intervention.

AWS-Specific Tactics to Automate Infrastructure

1. Define auto-scaling groups for different clusters using the Amazon auto-scaling feature in Amazon EC2.

2. Monitor system metrics (CPU, memory, disk I/O, network I/O) using Amazon CloudWatch and take appropriate actions (launching new AMIs dynamically using the auto-scaling service) or send notifications.

3. Store and retrieve machine configuration information dynamically:

Utilize Amazon SimpleDB to fetch config data during the boo t-time of an instance (e.g., database connection strings). SimpleDB may also be used to store information about an instance such as its IP address, machine name, and role. \

4. Design a build process such that it dumps the latest builds to a bucket in Amazon S3;

5. Invest in building resource management tools (automated scripts, preconfigured images)

6. Bundle Just Enough Operating System and software dependencies into an Amazon Machine Image so that it is easier to manage and maintain. Pass configuration files or parameters at launch time and retrieve user data and instance metadata after launch

7. Reduce bundling and launch time by booting from Amazon EBS volumes and attaching multiple Amazon EBS volumes to an instance. Create snapshots of common volumes and share snapshots among accounts wherever appropriate.

8. Application components should not assume health or location of hardware it is running on. For example, dynamically attach the IP address of a new node to the cluster. Automatically failover to the new cloned instance in case of a failure.

- **Think Parallel:** The cloud makes parallelization effortless
- When it comes to accessing (retrieving and storing) data, the cloud is designed to handle massively parallel operations. In order to achieve maximum performance and throughput, request parallelization should be used.
- Multi-threading requests by using multiple concurrent threads will store or fetch the data faster than requesting it sequentially
- When it comes to processing or executing requests in the cloud, it becomes even more important to leverage parallelization.

A general best practice, in the case of a Web application, is to distribute the incoming requests across multiple Web servers using load balancer

- **Think Parallel:** The cloud makes parallelization effortless
- When it comes to accessing (retrieving and storing) data, the cloud is designed to handle massively parallel operations. In order to achieve maximum performance and throughput, request parallelization should be used.
- Multi-threading requests by using multiple concurrent threads will storeor fetch the data faster than requesting it sequentially

- When it comes to processing or executing requests in the cloud, it becomes even more important to leverage parallelization.
- A general best practice, in the case of a Web application, is to distribute the incoming requests across multiple Web servers using load balancer.
- **AWS Specific Tactics for Parallelization**

  1. Multi-thread your Amazon S3 requests

  2. Multi-thread Amazon SimpleDB GET and BATCHPUT requests

  3. Create a JobFlow using the Amazon Elastic MapReduce Service for each of daily batch processes (indexing, log analysis, etc.) which will compute the job in parallel and save time.

- 4. Use the Elastic Load Balancing service and spread load across multiple Web app servers dynamically.
- **Keep Dynamic Data Closer to the Compute and Static Data Closer to the End User:** In general it's a good practice to keep data as close as possible to compute or processing elements to reduce latency
- If a large quantity of data that need to be processed resides outside of the cloud, it might be cheaper and faster to "ship" and transfer the data to the cloud first and then perform the computation.
- Conversely, if the data are static and not going to change often, it is advisable to take advantage of a content delivery service so that the static data are cached at an edge location closer to the end user (requester), thereby lowering the access latency.

AWS-Specific Tactics for Implementing This Best Practice

1. Ship data drives to Amazon using the Import/Export service. It may be cheaper and faster to move large amounts of data than to upload using the Internet.

2. Utilize the same availability zone to launch a cluster of machines.

3. Create a distribution of Amazon S3 bucket and let Amazon CloudFront caches content in that bucket across all the edge locations around the world.

- **Security Best Practices:** Security should be implemented in every layer of the cloud application architecture.
- **Protect Data in Transit**: If exchange of sensitive or confidential information between a browser and a Web server is needed, configure SSL on server instance. A certificate is needed from an external certification authority. The public key included in the certificate authenticates your server to the browser and serves as the basis for creating the shared session key used to encrypt the data in both directions

- **Protect Data at Rest**. If storing sensitive and confidential data in the cloud, encrypt the data (individual files) before uploading it to the cloud.On Amazon EC2, file encryption depends on the operating system. Amazon EC2 instances running Windows can use the built-in Encrypting File system (EFS) feature available in Windows. This feature will handle the encryption and decryption of files and folders automatically and make the process transparent to the users
- **Manage Multiple Users and their permissions with IAM**. AWS Identity and Access Management (IAM) enables you to create multiple Users and manage the permissions for each of these Users within your AWS Account. A User is an identity (within your AWS Account) with unique security credentials that can be used to access AWS Services. IAM eliminates the need to share passwords or access keys, and makes it easy to enable or disable a User's access as appropriate.

Secure Application. Every Amazon EC2 instance is protected by one or more security groups —that is, named sets of rules that specify which ingress (i.e., incoming) network traffic should be delivered to instance. Users can specify TCP and UDP ports, ICMP types and codes, and source addresses. Security groups give you basic firewall-like protection for running instances

BUILDING CONTENT DELIVERY NETWORKS USING CLOUDS

Numerous "storage cloud" providers (or "Storage as a Service") have recently emerged that can provide Internet-enabled content storage and delivery capabilities in several continents, offering service-level agreement (SLA)-backed performance and uptime promises for their services.

Customers are charged only for their utilization of storage and transfer of content

Storage cloud providers:

- Amazon Simple Storage Service (S3) and CloudFront(CF)
- Nirvanix Storage Delivery Network (SDN)
- Rackspace Cloud Files
- Microsoft Azure Storage,

Numerous "storage cloud" providers (or "Storage as a Service") have recently emerged that can provide Internet-enabled content storage and delivery capabilities in several continents, offering service-level agreement (SLA)-backed performance and uptime promises for their services.

Customers are charged only for their utilization of storage and transfer of content

Storage cloud providers:

- Amazon Simple Storage Service (S3) and CloudFront(CF)

- Nirvanix Storage Delivery Network (SDN)
- Rackspace Cloud Files
- Microsoft Azure Storage,

Amazon Simple Storage and CloudFront

- Amazon S3 was launched in the United States in March 2006 and in Europe in November 2007, opening up the huge infrastructure that Amazon themselves utilize to run their highly successful e-commerce company, Amazon.com.
- In November 2008, Amazon launched CloudFront, a content delivery service that added 14 edge locations (8 in the United States, 4 in Europe, and 2 in Asia).
- However, unlike S3, CloudFront does not offer persistent storage.
- Rather, it is analogous to a proxy cache, with files deployed to the different CloudFront locations based on demand and removed automatically when no longer required
- CloudFront also offers "streaming distributions" that can distribute audio and video content in real time, using the Real-Time Messaging Protocol

(RTMP) instead of the HTTP protocol.

- Amazon provides REST and SOAP interfaces to its storage resources, allowing users the ability to read, write, or delete an unlimited amount of objects, with sizes ranging from 1 byte to 5 gigabytes each.
- Amazon S3 has a storage cost of $0.15 per GB/month in their standard U.S. and EU data centers

Nirvanix Storage Delivery Network

- Nirvanix launched its Amazon S3 competitor, the Nirvanix Storage Delivery Network (SDN), on September 2007.
- The Nirvanix service was notable in that it had an SLA-backed uptime guarantee at a time when Amazon S3 was simply operated on a best-effort service basis.
- Nirvanix differentiates itself in several ways notably by having coverage in four regions, offering automatic file replication over sites in the SDN for performance and redundancy, and supporting file sizes up to 256 GB.
- Nirvanix is priced slightly higher than Amazon's service, and they do not publish their pricing rates for larger customers (2 TB/month).
- Nirvanix provides access to their resources via SOAP or REST interfaces, as well as providing SDK's in Java, PHP Zend, Python, and C#.

Rackspace Cloud Files

- Rackspace (formerly Mosso) Cloud Files provides a self-serve storage and delivery service in a fashion similar to that of the Amazon and Nirvanix offerings.
- The core Cloud Files offering is served from a multizoned, redundant data center in Dallas, Texas. The service is notable in that it also provides CDN integration.
- Rather than building their own CDN extension to the Cloud Files platform as Amazon has done for S3, Rackspace has partnered with a traditional CDN service, Limelight, to distribute files stored on the Cloud Files platform to edge nodes operated by Limelight.
- Unlike Amazon CloudFront, Rackspace does not charge for moving data from the core Cloud Files servers to the CDN edge locations.
- Rackspace provides RESTful APIs as well as API bindings for popular languages such as PHP, Python, Ruby, Java, and .NET.

Azure Storage Service

- Microsoft's Windows Azure platform offers a comparable storage and delivery platform called Azure Storage, which provides persistent and redundant storage in the cloud.
- For delivering files, the Blob service is used to store files up to 50 GB in size. On a per storage account basis, the files can be stored and delivered from data centers in Asia (East and South East), the United States (North Central and South Central), and Europe (North and West).
- Azure Storage accounts can also be extended by a CDN service that provides an additional 18 locations globally across the United States, Europe, Asia, Australia, and South America.
- This CDN extension is still under testing and is currently being offered to customers as a Community Techno logy Preview (CTP) at no charge.
- Most "storage cloud" providers are merely basic file storage and delivery services and do not offer the capabilities of a fully featured CDN such as automatic replication, fail-over, geographical load redirection, and load balancing.
- Furthermore, a customer may need coverage in more locations than offered by a single provider.
- To address this, MetaCDN was introduced, a system that utilizes numerous storage providers in order to create an overlay network that can be used as a high-performance, reliable, and redundant geographically distributed CDN
- **Encoding Services:** Video and audio encoding services are also individually available from cloud vendors.

- Two notable providers are encoding.com and Nirvanix.
- The endoing.com service is a cloud-based video encoding platform that can take a raw video file and generate an encoded file suitable for streaming.

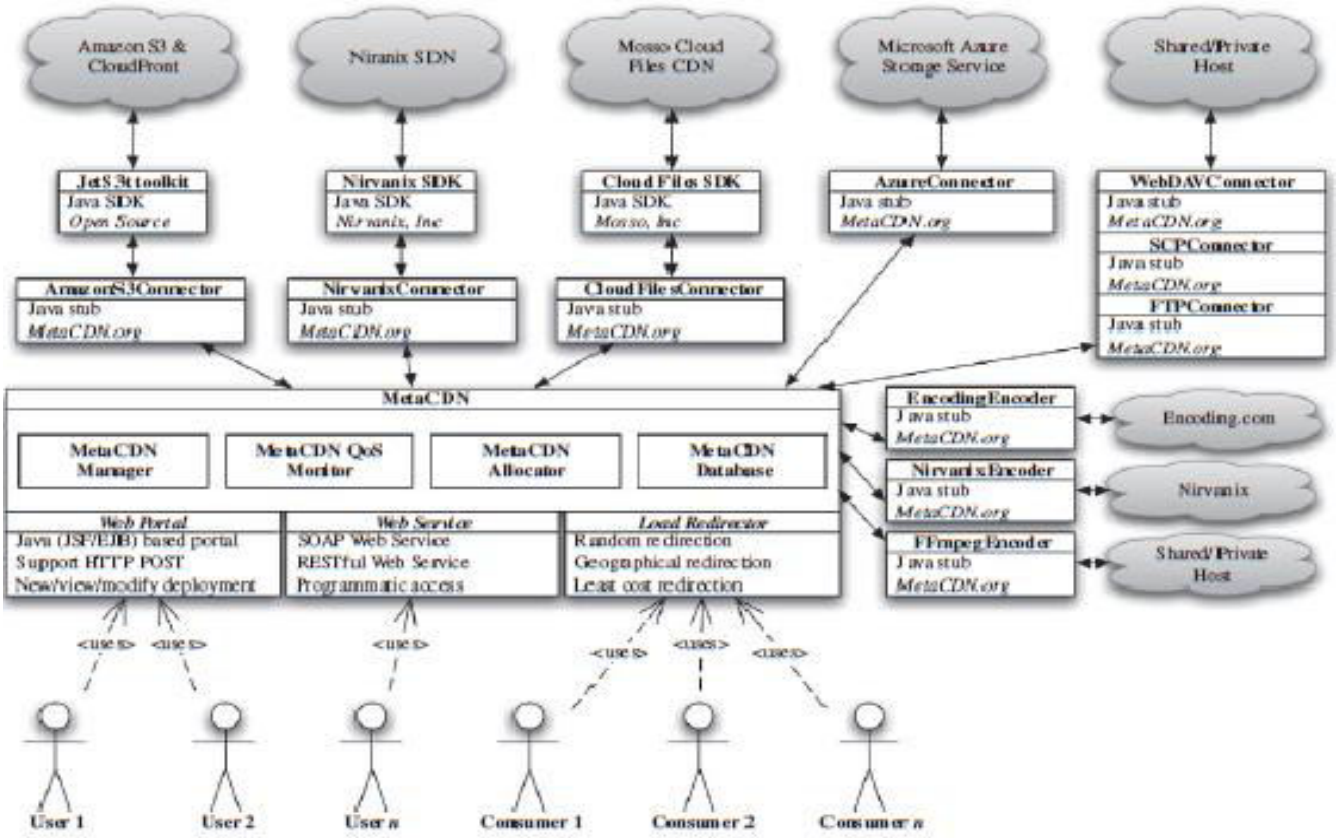


FIGURE 20.2. The MetaCDN architecture.

- The MetaCDN service is presented to end users in two ways. First, it can be presented as a Web portal, which was developed using
  (a) Java Enterprise and Java Server Faces (JSF) technologies, with a
  MySQL back-end to store user accounts and deployments, and
  (b) the capabilities, pricing, and historical performance of service providers.

  The Web portal acts as the entry point to the system and also functions as an application-level load balancer for end users that wish to download content that has been deployed by MetaCDN.

Using the Web portal, users can sign up for an account on the MetaCDN system and enter credentials for any cloud storage or other provider they have an account with.
Once this simple step has been performed, they can utilize the MetaCDN system to intelligently deploy content onto storage providers according to their performance requirements and budget limitations.
The Web portal is most suited for small or ad hoc deployments and is especially useful for less technically inclined content creators.

- The second method of accessing the MetaCDN service is via RESTful Web Services.
- These Web Services expose all of the functionality of the MetaCDN system.
- This access method is most suited for customers with more complex and frequently changing content delivery needs, allowing them to integrate the MetaCDN service in their own origin Web sites and content creation workflows.

Integrating "Cloud Storage" Providers

- The MetaCDN system works by integrating with each storage provider via connectors that provides an abstraction to hide the complexity arising from the differences in how each provider allows access to their systems.
- An abstract class, DefaultConnector, prescribes the basic functionality that each provider could be expected to support, and it must be implemented for all existing and future connectors.
- These include basic operations like creation, deletion, and renaming of replicated files and folders.
- If an operation is not supported on a particular service, then the connector for that service throws a FeatureNotSupportedException.

Overall Design and Architecture of the System

- The MetaCDN service has a number of core components that contain the logic and management layers required to encapsulate the functionality of different upstream storage providers and present a consistent, unified view of the services available to end users.
- These components include the **MetaCDN Allocator**, which (a) selects the optimal providers to deploy content to and (b) performs the actual physical deployment.
- The **MetaCDNQoS monitor** tracks the current and historical performance of participating storage providers
- The **MetaCDN Manager** tracks each user's current deployment and performs various housekeeping tasks.
- The **MetaCDN Database** stores crucial information needed by the MetaCDN portal, ensuring reliable and persistent operation of the system. It also stores information needed by the MetaCDNsystem, such as MetaCDN user details, their credentials for various storage cloud and other providers, and information tracking their (origin) content and any replicas made of such content

- The **MetaCDN Load Redirector** is responsible for directing MetaCDN end users (i.e., content consumers) to the most appropriate file replica, ensuring good performance at all times

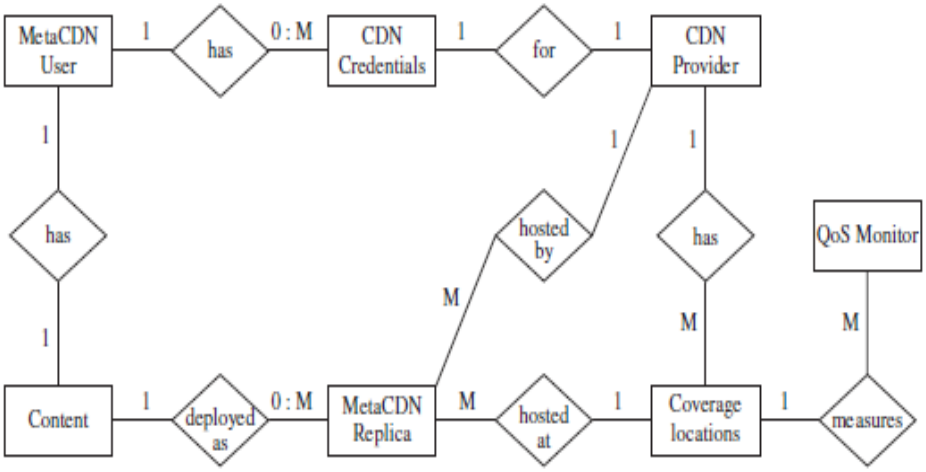


FIGURE 20.5. Entity relationship diagram for the MetaCDN database.

- The **MetaCDN Allocator** allows users to deploy files either directly (uploading a file from their local file system) or from an already publicly accessible origin Web site (sideloading the file, where the backend storage provider pulls the file).
- MetaCDN users are given a number of different deployment options depending on their needs, regardless of whether they access the service via the Web portal or via Web services.
  The available deployment options include:
- **Maximize coverage and performance**, where MetaCDN deploys as many replicas as possible to all available locations. The MetaCDN Load Redirector directs end users to the closest physical replica.
- **Deploy content in specific locations**, where a user nominates regions and MetaCDN matches the requested regions with providers that service those areas. The MetaCDNLoad Redirector directs end users to the closest physical replica.

- **Cost-optimized deployment**, where MetaCDN deploys as many replicas in the locations requested by the user as their storage and transfer budget will allow, keeping them active until that budget is exhausted. The MetaCDN Load Redirector directs end users to the cheapest replica to minimize cost and maximize the lifetime of the deployment.
- **Quality of service (QoS)-optimized deployment**, whereMetaCDN deploys to providers that match specific QoS targets that a user specifies, such as average throughput or response time from a particular location, which is tracked by persistent probing from the MetaCDNQoS monitor. The MetaCDN Load Redirector directs end users to the best-performing replica for their specific region based on historical measurements from the QoS monitor.

- After MetaCDN deploys replicas using one of the above options, it stores pertinent details such as the provider used, the URL of the replica, the desired lifetime of the replica, and the physical location (latitude and longitude) of that deployment in the MetaCDN Database.
- A geolocation service is used to find the latitude and longitude of where the file is stored.

- The **MetaCDNQoS Monitor** tracks the performance of participating providers (and their available storage and delivery locations) periodically, monitoring and recording performance and reliability metrics from a variety of locations, which is used for QoS-optimized deployment matching.
- This component also ensures that upstream providers are meeting their service-level agreements (SLAs), and it provides a logging audit trail to allow end users to claim credit in the event that the SLA is broken

- The **MetaCDN Manager** has a number of housekeeping responsibilities.
- First, it ensures that all current deployments are meeting QoS targets of users that have made QoS optimized deployments.
- Second, it ensures that replicas are removed when no longer required (i.e., the "deploy until" date set by the user has expired), ensuring that storage costs are minimized at all times.
- Third, for users that have made cost-optimized deployments, it ensures that a user's budget has not been exceeded, by tracking usage (i.e., storage and downloads) from auditing information provided by upstream providers.

Resource Cloud Mashups

The initial cloud providers simply opened their existing infrastructure to the customers and thus exploited their respective proprietary solutions. Implicitly, the offered services and hence the according API are specific to the service provider and cannot be used in other environments.

This, however, poses major issues for customers, as well as for future providers.

- **Interoperability and Vendor Lock-In**. Since most cloud offerings are proprietary, customers adopting the according services or adapting their respective applications to these environments are implicitly bound to the respective provider. Movement between providers is restricted by the effort the user wants to vest into porting the capabilities to another environment, implying in most cases reprogramming of the according applications. This makes the user dependent not only on the provider's decisions, but also on his/her failures Since the solutions and systems are proprietary, neither customer nor provider can cross the boundary of the infrastructure and can thus not compensate the issues by making use of additional external resources

- **Cloud Mashup:** Integrating multiple cloud infrastructures into a single platform, which can be accessed via a common web service is called a cloud resource mashup

A Need for Cloud Mashups

- By integrating multiple cloud infrastructures into a single platform, reliability and scalability is extended by the degree of the added system(s).

Platform as a Service (PaaS) providers often offer specialized capabilities to their users via a dedicated API, such as Google App Engine providing Additional features for handling (Google)

documents, and MSAzure is focusing particularly on deployment and provisioning of Web services, and so on.

Through aggregation of these special features, additional, extended capabilities can be achieved (given a certain degree of interoperability), ranging from extended Storage and computation facilities (IaaS) to combined functions, Such as analytics and functionalities

The Cloud Computing Expert Working Group refers to such integrated cloud systems with aggregated capabilities across the individual infrastructures as Meta-Clouds and Meta-Services

Benefits of Cloud Mashups

- **User-Centric Clouds**: Most cloud provisioning is user- and context-agnostic; the user will always get the same type of service, access route

As clouds develop into application platforms , context such as user device properties or location becomes more and more relevant: Device types designate the execution capabilities (even if remote), their connectivity requirements and restrictions, and the location . Each of these aspects has a direct impact on how the cloud needs to handle data and application location, communication

By offering such capabilities across cloud infrastructures, the service provider will be able to support, in particular, mobile users in a better way. Similar issues and benefits apply as for roaming. Along the same way, the systems need to be able to communicate content and authentication information to allow users to connect equally from any location.

Legislation and contractual restrictions may prevent unlimited data replication, access, and shifting between locations

- **Multimedia Streaming**: In order to maintain and provide data as a stream, the platform provider must ensure that data availability is guaranteed without disruptions. This implies that not only data location is reallocated dynamically according to the elasticity paradigm but also the data stream—potentially taking the user context into consideration again

Such business entities must hence not only aggregate information in potentially a user-specific way, but also identify the best sources, handle the streams of these sources, and redirect them according to user context

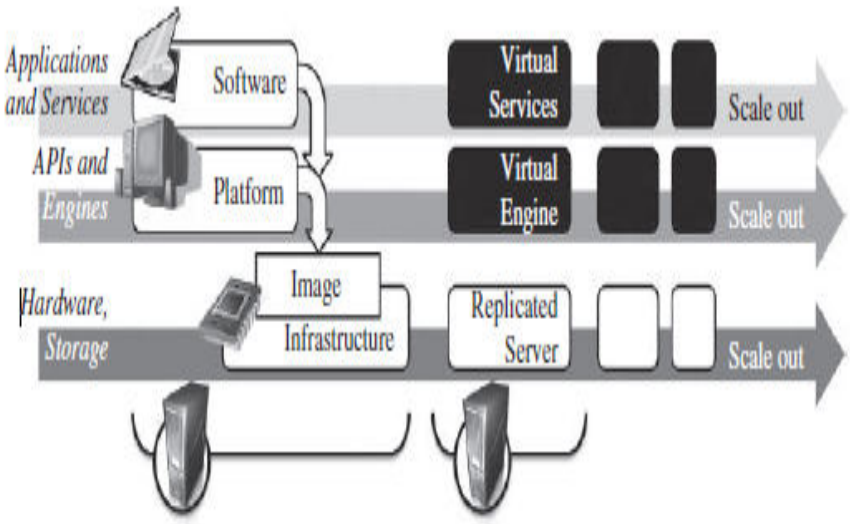
To realize a mashup requires at least:

- A compatible API/programming model, respectively an engine that can parse the APIs of the cloud platforms to be combined (PaaS).
- A compatible virtual machine, respectively an image format that all according cloud infrastructures can host (IaaS).

- Interoperable or transferrable data structures that can be interpreted by all engines and read by all virtual machines involved. This comes as a side effect to the compatibility aspects mentioned above.

By addressing interoperability from bottom up—that is, from an infrastructure layer first—resources in a PaaS and SaaS cloud mashup could principally shift the whole image rather than the service/module. The actual programming engine running on the PaaS cloud, respectively the software exposed asservices, could be shifted within an IaaS cloud as complete virtual machines, given that all resources can read the according image format

Encapsulated Virtual Environments



Intelligent Image Handling

- A straightforward cloud environment management system would replicate any hosted system in a different location the moment the resources become insufficient
- for example, when too many users access the system concurrently and execute a load balance between the two locations. Similarly, an ideal system would down-scale the replicated units once the resource load is reduced again

In order to treat any cloud type as essentially an infrastructure environment, the system requires additional information about how to segment the exposed service(s) and thus how to replicate it (them).

Segmenting the Service. Any process exploiting the capabilities of the cloud essentially consists of the following parts: the user-specific data (state), the scalable application logic, the not-scalable underlying engine or supporting logic, the central dataset, and the execution environment.

In order to allow infrastructure clouds to handle (platform) services in a efficient manner, the management system must be able to identify which parts are needed and can be replicated in order to scale out, respectively which ones can and should be destroyed during scale-down;

for example, it would not be sensible to destroy the whole image if only one user (of many) logs out from the machine.

Life Cycle of a Segmented Cloud Image. With segmented main services in anIaaS environment, the system can now scale up and down in a efficient manner across several resource providers: Any service requires that its base environment is available on the machines it gets replicated to. As soon as the hosted engine wants to scale beyond the boundaries of the local machine, a new physical machine has to be identified ready to host the new instances—in the simplest case, another machine will already provide the respective hosting image.

Intelligent Data Management : Efficient data management for large-scale environments Random segmentation and distribution of data files uses a strategy which takes

(1) the semantic contents of the datasets and

(2) the requirements of users/applications into account (i.e., data shall be distributed according to the interest in the data/information).

For this reason, users, devices, and applications need to be modeled by capturing relevant context parameters (e.g., the actual position and network properties) as well as analyzing application states with respect to upcoming data retrieval and/or processing needs In addition, storage resources, platforms, and infrastructures (i.e., entire virtual images) shall also be continuously monitored, so as to react on sudden bottlenecks immediately

REALIZING RESOURCE MASHUPS

In order to realize efficient cloud mashups on an infrastructure level, distributed data and segmented image management have to be combined in order to handle the additional size created by virtualizing the machine.

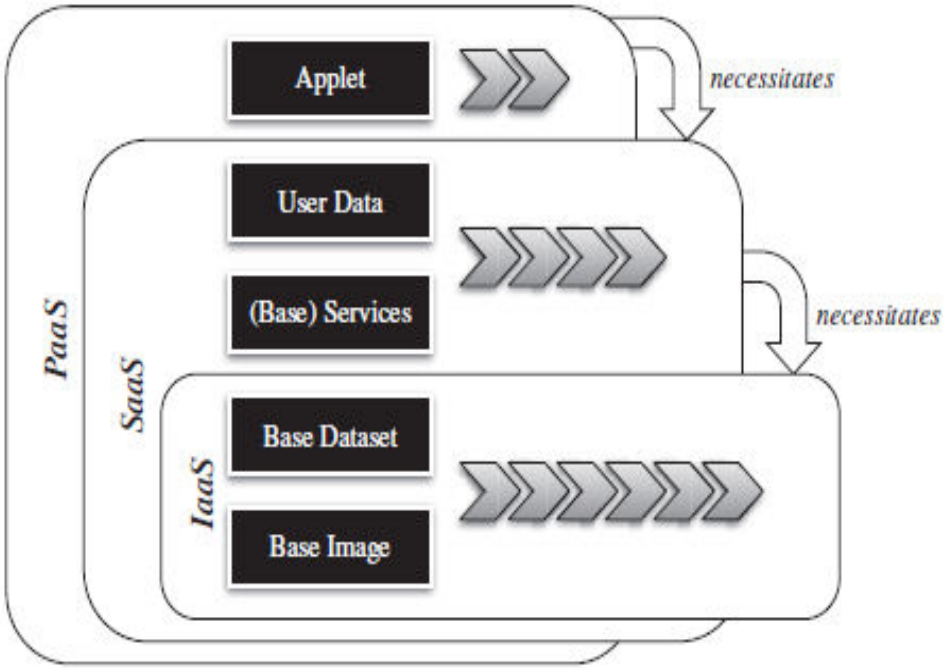
we can distinguish between the base image set consisting of

(a) the setup environment and any engine (if required),

(b) the base dataset that may be customer-specific (but not user- specific), such as general data that are provided to the user, but also and more importantly the applet or service base that is provided to each user equally, and

(c) the user-specific information which may differ per access and which may only be available on a single machine.

The relationship between IaaS, SaaS, and PaaS during scaling



- **IaaS Provisioning:** Infrastructures are typically provided in the form of an image containing the full computational environment or consist of a dynamic dataset, which is typically made available to all users equally. Scaling out involves either replication of the image/data set (horizontal scaling) or increasing the available storage size (vertical scale).

Horizontal scaling thereby typically implies that the full dataset is replicated, while vertical scaling may lead to data segmentation and distribution.

- **PaaS Provisioning**: Multiple different sets have to be managed during scale-out, depending on the original cause to increase the resource load
- **SaaS Provisioning**: Several resources in the cloud environment can host the base image and allow different SaaS customers to make use of these machines.
- In other words, machines with the respective compatible base image can host the replicated service instances, rather than having to duplicate the full image all the time.
- when no machine with a compatible base image is available anymore, a new resource has to be loaded with an image that meets the current scale-out requirements best
- The maintenance of replicated datasets in SaaS environments requires more efforts and carefulness because synchronization between multiple instances of the same dataset on the same image might result in inconsistent states, and thus supervision of duplicated data sets is highly recommended

UNIT - 5

Organizational Readiness and Change Management in the Cloud Age

In order to effectively enable and support enterprise business goals and strategies, information technology (IT) must adapt and continually change. IT must adopt emerging technologies to facilitate business to leverage the new technologies to create new opportunities, or to gain productivity and reduce cost. Managing emerging technologies is always a complex issue, and managers must balance the desire to create competiveness through innovation with the need to manage the complex challenges presented by these emerging technologies .

BASIC CONCEPT OF ORGANIZATIONAL READINESS

Change can be challenging; it brings out the fear of having to deal with uncertainties. This is the **FUD syndrome: Fear, Uncertainty, and Doubt Drivers For Changes: A Framework To Comprehend the Competitive Environment:**

The five driving factors for change encapsulated by the framework are:

- Economic (global and local, external and internal)
- Legal, political, and regulatory compliance
- Environmental (industry structure and trends)
- Technology developments and innovation
- Socio cultural (markets and customers)

The five driving factors for change is an approach to investigate, analyze, and forecast the emerging trends of a future, by studying and understanding the five categories of drivers for change.

The results will help the business to make better decisions, and it will also help shape the short- and long-term strategies of that business.

It is this process that helps reveal the important factors for the organization's desirable future state, and it helps the organization to comprehend which driving forces will change the competitive landscape in the industry the business is in, identify critical uncertainties, and recognize what part of the future is predetermined such that it will happen regardless how the future will play out

A driving force or factor is a conceptual tool; it guides us to think deeply about the underlying issues that impact our well-being and success

Economic (Global and Local, External and Internal): Economic factors are usually dealing with the state of economy, both local and global in scale. To be successful, companies have to live with the paradox of having new market and business opportunities globally.

Following are sample questions that could help in understanding Economic factors

- What is the current economic situation?
- What will the economy looks like in 1 year, 2 years, 3 years, 5 years, and so on?
- What are some of the factors that will influence the future economic outlook?
- Is capital easy to access?
- How does this technology transcend the existing business model?
- Buy vs. build? Which is the right way?
- What is the total cost of ownership (TCO)?

Legal, Political, and Regulatory Compliance:

The objective is to be a good corporate citizen and industry leader and to avoid the potential cost of legal threats from external factors.

The following are sample questions

- What are the regulatory compliance requirements?
- What is the implication of noncompliance?
- What are the global geopolitical issues?

Environmental (Industry Structure and Trends): Environmental factors usually deal with the quality of the natural environment, human health, and safety.

The following are sample questions

- What is the implication of global warming concern?
- Is a green data center over-hyped?
- How can IT initiatives help and support organizational initiatives to reduce carbon footprint?
- Can organizations and corporations leverage information technology, including cloud computing to pursue sustainable development

Technology Developments and Innovation: Technological innovations are the single most important contributing factor in sustained economic growth.

The following are sample questions

- When will the IT industry standards be finalized? By who? Institute of Electrical and Electronics Engineers (IEEE)?
- Who is involved in the standardization process?

- Who is the leader in cloud computing technology?
- What about virtualization of application, operating system (platform)pair (i.e., write once, run anywhere)?
- How does this emerging technology (cloud computing) open up new areas for innovation?
- How can an application be built once so it can configure dynamically in real time to operate most effectively, based on the situational constraint (e.g., out in the cloud somewhere, you might have bandwidth constraint to transfer needed data)?
- What is the guarantee from X Service Providers (XSP) that the existing applications will still be compatible with the future infrastructure (IaaS)?
- Will the data still be executed correctly?

Socio cultural (Markets and Customers): Societal factors usually deal with the intimate understanding of the human side of changes and with the quality of life in general. Survival of the industry, and therefore of the companies, demands that Companies combine with former competitors and transform into new species.

The following are sample questions

- 1. The new direction of the firm (where we want to go today)
- 2. The urgency of the change needed
- 3. What the risks are to
    a. Maintain status quote
    b. Making the change
- 4. What the new role of the employee will be
- 5. What the potential rewards are

One of the important value propositions of cloud computing should be to explain to the decision maker and the users the benefits of:

- Buy and not build
- No need for a large amount of up-front capital investment
- Opportunity to relieve your smartest people from costly data-center operational activities; and switch to focus on value-added activities
- Keep integration (technologies) simple

Lewin's Change Management Model

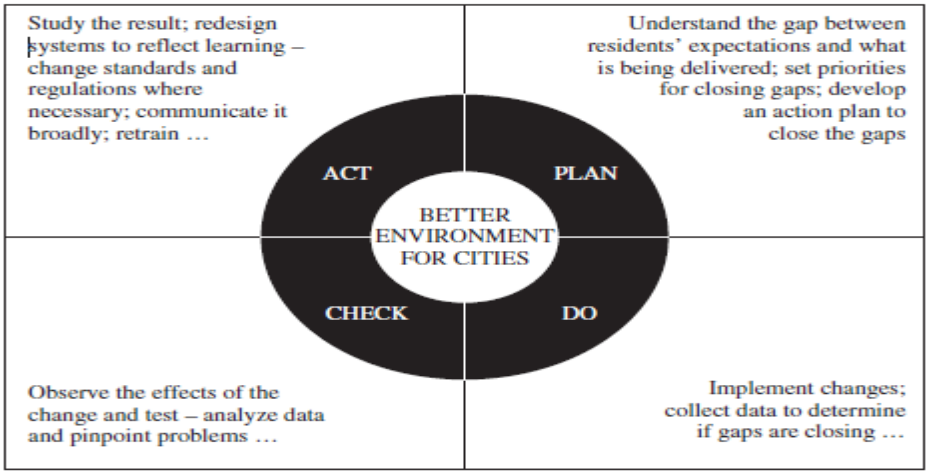
- Kurt Lewin, a psychologist by training, created this change model in the 1950s.
- Lewin observed that there are three stages of change, which are: Unfreeze, Transition, and Refreeze.
- It is recognized that people tend to become comfortable in this "freeze" or "unchanging/stable" environment, and they wish to remain in this "safe/comfort" zone.
- Any disturbance/disruption to this unchanging state will cause pain and become uncomfortable.
- In order to encourage change, it's necessary to unfreeze the environment by motivating people to accept the change. The motivational value has to be greater than the pain in order to entice people to accept the change
- As Lewin put it, "Motivation for change must be generated before change can occur. One must be helped to reexamine many cherished assumptions about oneself and one's relations to others."
- This is the unfreezing stage from which change begins
- The transition phase is when the change (plan) is executed and actual change is being implemented. Since these "activities" take time to be completed, the process and organizational structure may also need to change, specific jobs may also change. The most resistance to change may be experienced during this transition period. This is when leadership is critical for the change process to succeed, and motivational factors are paramount to project success.
- The last phase is Refreeze; this is the stage when the organization once again becomes unchanging/frozen until the next time a change is initiated

Deming Cycle (Plan, Do, Study, Act)

- The Deming cycle is also known as the PDCA cycle;
- it is a continuous improvement (CI) model comprised of four sequential subprocesses; Plan, Do, Check, and Act.
- This framework of process and system improvement was originally conceived by Walter Shewhart in the 1930s and was later adopted by Edward Deming.

- The PDCA cycle is usually implemented as an evergreen process, which means that the end of one complete pass (cycle) flows into the beginning of the next pass and thus supports the concept of continuous quality improvement.
- Edward Deming proposed in the 1950s that business processes and systems should be monitored, measured, and analyzed continuously to identify variations and substandard products and services, so that corrective actions can be taken to improve on the quality of the products or services delivered to the customers
- PLAN: Recognize an opportunity and plan a change.
- DO: Execute the plan in a small scale to prove the concept.
- CHECK: Evaluate the performance of the change and report the results to sponsor.
- ACT: Decide on accepting the change and standardizing it as part of the process. Incorporate what has been learned from the previous steps to plan new improvements, and begin a new cycle.

Deming's PDCA cycle



A Proposed Working Model: CROPS: Change Management framework

- In the IT world, a project portfolio management system gives management timely critical information about projects so they can make better decisions; re-deploy resources due to changing priorities, and keep close tabs on progress. However, as the modern economy moves from product and manufacturing centric to a more information and knowledge base focus, the change management process needs to reflect that people are truly the most valuable asset of the organization. Usually, an organization experiences strong resistance to change.Employees are afraid of the uncertainty, they feel comfortable with the stable state and do not want to change, and are afraid to lose their power if things change. The best approaches to address resistance are through increased and sustained communications and education. The champion of change, usually the leader—for

example, the Chief Information Officer (CIO) of the organization—should communicate the Why aggressively and provide a Vision of Where he wants to go today.

CROPS working model:

Culture, Rewards, Organization and Structures, Process, Skills and Competencies (CROPS) framework.

- Culture: Corporate culture is a reflection of organizational (management and employees) values and belief

Elements of organizational culture may include:

- Stated values and belief
- Expectations for member behavior
- Customs and rituals
- Stories and myths about the history of the organization
- Norms—the feelings evoked by the way members interact with each other, with outsiders, and with their environment
- Metaphors and symbols—found embodied in other cultural elements
- Rewards and Management System. This management system focuses on how employees are trained to ensure that they have the right skills and tools to do the job right. It identifies how to measure employee job performance and how the company compensates them based on their performance. Reward is the most important ingredient that shapes employees' value and beliefs.
- Organization and Structures. How the organization is structured is largely influenced by what the jobs are and how the jobs are performed. The design of the business processes govern what the jobs are, and when and where they get done.

Business processes need to align with organizational vision, mission, and strategies in order to create customer and shareholder values. Therefore, all the components of the CROPS framework are interrelated.

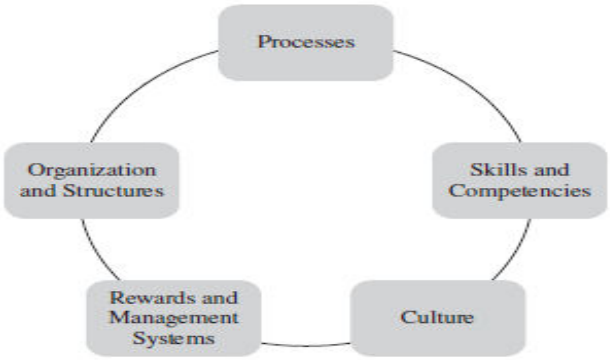
- Process: Thomas Davenport defined a business process or business method as a "collection of related, structured activities or tasks that produce a specific service or product (serve a particular goal) for a particular customer or customers".

Hammer and Champy's definition can be considered as a subset of Davenport's. They define a process as "a collection of activities that takes one or more kinds of input and creates an output that is of value to the customer."

A process is where the work gets done, and value creation occurs through transforming input into output

- Skills and Competencies: Specialized skills that become part of the organizational core competency enable innovation and create a competitive edge. Organizations that invest in research and development which emphasize investing in people's training and well-being will shape a winning strategy.

CROPS framework



CHANGE MANAGEMENT MATURITY MODEL (CMMM)

- A Change Management Maturity Model (CMMM) helps organizations to

(a) analyze, understand, and visualize the strength and weakness of the firm's change management process and

(b) identify opportunities for improvement and building competitiveness

- The working model is based on CMM (Capability Maturity Model), originally developed by American Software Engineering Institute (SEI) in cooperation with Mitre Corporation.
- CMM is a model of process maturity for software development, but it has since been adapted to different domains. The CMM model describes a five-level process maturity continuum
- The business value of CMMM can be expressed in terms of improvements in business efficiency and effectiveness.
- All organizational investments are business investments, including IT investments.
- The resulting benefits should be measured in terms of business returns.
- Therefore, CMMM value can be articulated as the ratio of business performance to CMMM investment;

for example

- ROIT(CMMM) = Estimated total business performanceimprovement

Total CMMM investment(TCO)

where

- ROIT: Observed business value or total return on investment from IT initiative (CMMM)
- Business performance improvement
    - Reduce error rate
    - Increase customer/user satisfaction
        - Customer retention
        - Employee retention
    - Increase market share and revenue
    - Increase sales from existing customer
    - Improve productivity
- CMMM investment
    - Initial capital investment
- Total cost of ownership (TCO) over the life of the investment (solution)

A Working Model: Change Management Maturity Model(CMMM)

- **Level 5**

Description: Optimized

Specific to CMMM: At this level of process maturity, the focus is on improving process performance

Characteristics of Organization: Operational excellence/organizational competency. Change management as part of the core competency. Culturally, employee accepts that change is constant and in a rapid rate

Path to Next Higher Level: Achieve strategic/operational excellence. Extensive training exists at all level of organization

Key Results and Benefits: Better business and IT strategic alignment. Enabling innovation. Create competitiveness

- **Level 4**

Description: Managed

Specific to CMMM: Adopted specific change management methodology and process. Centralized and standardized change management control and tracking to manage risks and sustain quality of products and services.

Characteristics of Organization: Organization and management can find ways to change, evolve, and adapt the process to particular project needs; with minimal or no impact to quality of products or services being delivered as measured against SLA

Path to Next Higher Level: Continuous process improvement. Effective business and IT strategic alignment

Key Results and Benefits: Achieve higher level of quality. Higher degree of customer/user satisfaction. Reduce costs. Higher profitability. Increase revenue and market share.

- **Level 3**

Description: Defined

Specific to CMMM: Standardizing change management processes and practices

Characteristics of Organization: Processes at this level are defined and documented. Some process improvement projects initiate overtime.

Key Results and Benefits: Better appreciation of value of IT. Better business and IT integration.

- **Level 2**

Description: Repeatable

Specific to CMMM: Accept the importance of change management process. No standardization/centralization of change management process and practice. Poor change authorization and tracking scheme

Characteristics of Organization: It is characteristic of processes at this level that some processes are repeatable.

Path to Next Higher Level: Standardize and centralize change management process.

Key Results and Benefits: Project failure rate is still too high. Changes are still very disruptive to business operation.

- **Level 1**

Description: Ad hoc (disruptive)

Specific to CMMM: No change management processes. No specific or informal change management process and practice exist anywhere. Change can be made with no control at all; there is no approval mechanism, no track record and no single party accountable for the failure.

Characteristics of Organization: Chaotic, Reactive, Disruptive, Uncontrolled, Unstable, Constantly operate in a firefighting mode.

Path to Next Higher Level: Adopt formal change management practice.

Key Results and Benefits: No awareness of the benefits of adopting change management and best practice. Project failures are too often and too costly. No understanding of risk management, and do not have the capacity to manage and minimize disruption to IT and business due to change and/or the failure of the uncontrolled changes.

ORGANIZATIONAL READINESS SELF-ASSESSMENT (WHO, WHEN, WHERE, AND HOW)

- An organizational assessment is a process intending to seek a better understanding of the as-is (current) state of the organization.
- It also defines the roadmap (strategies and tactics) required to fill the gap and to get the organization moving toward where it wants to go (future state) from its current state.
- The process implies that the organization needs to complete the strategy analysis process first and to formulate the future goals and objectives that support the future direction of the business organization.
- The organizational assessment can be conducted by either an internal or external professional, depending on whether the expertise is available.
- Before the actual assessment begins, the champion of change (perhaps the CEO of the organization) is advised to articulate the vision of the firm, where the organization wants to go tomorrow, and how it intends to get there.

During an effective organization readiness assessment, it is desirable to achieve the following:

- Articulate and reinforce the reason for change.
- Determine the as-is state.
- Identify the gap (between future and current state).
- Anticipate and assess barriers to change.
- Establish action plan to remove barriers.

Involve the right people to enhance buy-in:

- It is critical to involve all the right people (stakeholders) across the organization, and not just management and decision-makers, as participants in any organization assessment

Asking the “right questions” is also essential.

The assessment should provide insight into challenges and help determine some of these key questions:

- How big is the gap?
- Does organization have the capacity to execute and implement changes?
- How will employees respond to the changes?
- Are all employees in the organization ready to adopt changes that help realize the vision?
- What are the critical barriers to success?
- Are business partners ready to support the changes?

DATA SECURITY IN THE CLOUD

Information in a cloud environment has much more dynamism and fluidity than information that is static on a desktop or in a network folder. Nature of cloud computing dictates that data are fluid objects, accessible from a multitude of nodes and geographic locations and, as such, must have a data security methodology that takes this into account while ensuring that this fluidity is not compromised. The idea of content-centric or information-centric protection, being an inherent part of a data object is a development out of the idea of the "de-perimerization" of the enterprise. This idea was put forward by a group of Chief Information Officers (CIOs) who formed an organization called the **Jericho Forum** .

The **Jericho Forum** was founded in 2004 because of the increasing need for data exchange between companies and external parties—

for example: employees using remote computers; partner companies; Customers.

The idea of creating, essentially, de-centralized perimeters, where the perimeters are created by the data object itself, allows the security to move with the data, as opposed to retaining the data within a secured and static wall

CLOUD COMPUTING AND DATA SECURITY RISK

Cloud computing is a development that is meant to allow more open accessibility and easier and improved data sharing. Data are uploaded into a cloud and stored in a data center, for access by users from that data center; or in a more fully cloud-based model, the data themselves are created in the cloud and stored and accessed from the cloud (again via a data center).

The most obvious risk in this scenario is that associated with the storage of that data. A user uploading or creating cloud-based data include those data that are stored and maintained by a third-party cloud provider such as Google, Amazon, Microsoft, and so on.

This action has several risks associated with it:

- Firstly, it is necessary to protect the data during upload into the data center to ensure that the data do not get hijacked on the way into the database.

- Secondly, it is necessary to the stores the data in the data center to ensure that they are encrypted at all times.
- Thirdly, and perhaps less obvious, the access to those data need to be controlled; this control should also be applied to the hosting company, including the administrators of the data center.
- In addition, an area often forgotten in the application of security to a data resource is the protection of that resource during its use

Data security risks are compounded by the open nature of cloud computing. Access control becomes a much more fundamental issue in cloud-based systems because of the accessibility of the data

Information-centric access control (as opposed to access control lists) can help to balance improved accessibility with risk, by associating access rules with different data objects within an open and accessible platform, without losing the Inherent usability of that platform.

A further area of risk associated not only with cloud computing, but also with traditional network computing, is the use of content after access.

The risk is potentially higher in a cloud network, for the simple reason that the information is outside of your corporate walls.

Data-centric mashups are those that are used to perform business processes around data creation and dissemination—by their very nature, can be used to hijack data, leaking sensitive information and/or affecting integrity of that data. Cloud computing, more than any other form of digital communication technology, has created a need to ensure that protection is applied at the inception of the information, in a content centric manner, ensuring that a security policy becomes an integral part of that data throughout its life cycle.

Encryption is a vital component of the protection policy, but further controls over the access of that data and on the use of the data must be met. In the case of mashups, the controlling of access to data resources, can help alleviate the security concerns by ensuring that mashup access is authenticated. Linking security policies, as applied to the use of content, to the access control method offer a way of continuing protection of data, post access and throughout the life cycle; this type of data security philosophy must be incorporated into the use of cloud computing to alleviate security risks.

CLOUD COMPUTING AND IDENTITY

Digital identity holds the key to flexible data security within a cloud Environment. A digital identity represents who we are and how we interact with others on-line.

Access, identity, and risk are three variables that can become inherently connected when applied to the security of data, because access and risk are directly proportional: As access increases, so then risk to the security of the data increases. Access controlled by identifying the actor attempting the access is the most logical manner of performing this operation. Ultimately,

digital identity holds the key to securing data, if that digital identity can be programmatically linked to security policies controlling the post-access usage of data.

Identity, Reputation, and Trust

Reputation is a real-world commodity; that is a basic requirement of human- to-human relationships:

Our basic societal communication structure is built upon the idea of reputation and trust.

Reputation and its counter value, trust, is easily transferable to a digital realm:

eBay, for example, having partly built a successful business model on the strength of a ratings system, builds up the reputation of its buyers and sellers through successful (or unsuccessful) transactions. These types of reputation systems can be extremely useful when used with a digital identity. They can be used to associate varying levels of trust with that identity, which in turn can be used to define the level (granular variations) of security policy applied to data resources that the individual wishes to access.

User-Centric Identity: Digital identities are a mechanism for identifying an individual, particularly within a cloud environment; identity ownership being placed upon the individual is known as user-centric identity. It allows users to consent and control how their identity (and the individual identifiers making up the identity, the claims) is used.

This reversal of ownership away from centrally managed identity platforms (enterprise-centric) has many advantages. This includes the potential to improve the privacy aspects of a digital identity, by giving an individual the ability to apply permission policies based on their identity and to control which aspects of that identity are divulged. An identity may be controllable by the end user, to the extent that the user can then decide what information is given to the party relying on the identity.

Information Card: Information cards permit a user to present to a Web site or other service (relying party) one or more claims, in the form of a software token, which may be used to uniquely identify that user. They can be used in place of user name/ passwords, digital certificates, and other identification systems, when user identity needs to be established to control access to a Web site or other resource, or to permit digital signing.

Information cards are part of an identity meta-system consisting of:

1. **Identity providers (IdP),** who provision and manage information cards, with specific claims, to users.

2. **Users** who own and utilize the cards to gain access to Web sites and other resources that support information cards.

3. **An identity selector/service**, which is a piece of software on the user's desktop or in the cloud that allows a user to select and manage their cards.

4. **Relying parties**. These are the applications, services, and so on, that can use an information card to authenticate a person and to then authorize an action such as logging onto a Web site, accessing a document, signing content, and so on

Each information card is associated with a set of claims which can be used to identify the user. These claims include identifiers such as name, email address, post code. Only the claim types are stored in cards issued by an identity provider; The claim values are stored by the provider, creating a more secure and privacy-rich system. One of the most positive aspects of an information card is the user-centric nature of the card. An information card IdP can be set up so that the end users themselves can self-issue a card, based on the required claims that they themselves input—the claims being validated if needed. Alternatively, the claims can be programmatically input by the IdP via a Web service or similar, allowing the end user to simply enter the information card site and download the card.

Using Information Cards to Protect Data

Information cards are built around a set of open standards devised by a consortium that includes Microsoft, IBM, Novell, and so on. The original remit of the cards was to create a type of single sign on system for the Internet, to help users to move away from the need to remember multiple passwords. However, the information card system can be used in many more ways. Because an information card is a type of digital identity, it can be used in the same way that other digital identities can be used. For example, an information card can be used to digitally sign data and content and to control access to data and content. One of the more sophisticated uses of an information card is the advantage given to the cards by way of the claims system.

Claims are the building blocks of the card and are dynamic in that they can be changed either manually or programmatically. A security policy could be applied to a data resource that will be enacted when a specific information card claim is presented to it: If this claim changes, the policy can subsequently change. For example, a policy could be applied to a Google Apps document specifying that access is allowed for user A when they present their information card with claim "security clearance level = 3" and that post access, this user will be able to view this document for 5 days and be allowed to edit it. The same policy could also reflect a different security setting if the claim changes, say to a security clearance level = 1; in this instance the user could be disallowed access or allowed access with very limited usage rights.

Weakness and Strengths of Information Cards

The dynamic nature of information cards is the strength of the system, but the weakness of information cards lies in the authentication. The current information card identity provisioning services on offer include Microsoft Geneva, Parity, Azigo, Higgins Project, Bandit, and Avoco Secure. Each offers varying levels of card authentication and are chosen from Username and password, Kerberos token, x509 digital certificate, and personal card.

Each of these methods has drawbacks. For example,

- username and password is less secure and also not transparent.

- X509 digital certificates can be difficult for less technical users to install and use

New developments in information card authentication are on the industry roadmap, including Live ID, OpenID, and out-of-band (also referred to as out-of-wallet"). This latter option offers much higher levels of authentication and thus security, but does have drawbacks in terms of transparency. GPS location authentication can also be added to the list of authentication choices to control access to resources. Based on geographic location of the person attempting access, this could become a particularly important feature for cloud-based data, which can potentially be accessed anywhere in the world but may be constrained by compliance with industry legal requirements. An identity meta-system based on interoperable standards of issuance and authentication, such as an information card, is an absolute requirement for digital identity to be successfully used across borders. Information cards can potentially provide such a framework, because they are based on the idea of an identity Meta system. The goal of which is to connect individual identity systems resulting in cards issued by a given host being compatible across the entire system. The Oasis Foundation, which is nonprofit organization that is striving to establish open standards for IT, has formed a working committee to enable the use of information cards to universally manage personal digital identities.

Legal Issues in cloud computing

Significant issues regarding privacy of data and data security exist, specifically as they relate to protecting personally identifiable information of individuals, but also as they relate to protection of sensitive and potentially confidential business information either directly accessible through or indirectly from the cloud systems. Complex jurisdictional issues may arise due to the potential for data to reside in disparate or multiple geographies. This geographical diversity is inherent in cloud service offerings. This means that both virtualization of and physical locations of servers storing and processing data may potentially impact what country's law might govern in the event of a data breach or intrusion into cloud systems. Jurisdictional matters also determine the country's law that is applicable to data and information that may be moved geographically among data centers around the world at any given point in time

DATA PRIVACY AND SECURITY ISSUES

U.S. Data Breach Notification Requirements

- Data breach is a loss of unencrypted electronically stored personal information.
- This information is usually some combination of name and financial information (e.g., credit card number, Social Security Number).
- A breach can occur in many ways—for example, by having a server compromised, loss of a thumb drive, or theft of a laptop or cell phone.
- Avoidance of a data breach is important to both cloud providers and users of cloud services because of the significant harm, both to the user and to the provider, when a breach occurs.

From the **user's viewpoint**, if personal information is compromised, there is a risk of identity theft and of credit or debit card fraud. From the **provider's viewpoint**, financial harm, potential for lawsuits, Federal Trade Commission (FTC) investigations, loss of customers, and damage to reputation are all likely results of when a data breach occurs.

Data breaches can be expensive. A breach generally results in a company notification of persons across the country when their information has been compromised. For purposes of data breach law, data in the cloud are treated no differently than any other electronically stored information. Cloud providers that have had their systems compromised will be required to notify affected persons and will have to coordinate with the cloud users who provided the data in order to do so

- **U.S. Federal Law Compliance**

Gramm Leach Bliley Act: **Financial Privacy Rule**. The Gramm Leach Bliley Act (GLB) requires that financial institutions implement procedures to ensure the confidentiality of personal information and to protect against unauthorized access to the information.

As part of the requirement to prevent unauthorized access to information, financial institutions must take steps to protect information provided to a service provider.

A service provider under GLB may be any number of individuals or companies that provide services to the financial institution and would include a cloud provider handling the personal information of a financial institution's customers.

- **The Role of the FTC: Safeguards Rule and Red Flags Rule**. At the United States federal level, the Federal Trade Commission (FTC) working under the auspices of the FTC Act has been given authority to protect consumers and their personal information. The Safeguards Rule mandated by GLB and enforced by the FTC requires that all businesses significantly involved in the provision of financial services and products have a written security plan to protect customer information.

The plan must include the following elements:

- Designation of one or more employees to coordinate its information security program;
- Identification and assessment of the risks to customer information in each

relevant area of the company's operation, and evaluation of the effectiveness of the current safeguards for controlling these risks;

- Designing and implementing a safeguards program, and regularly monitoring and testing it;
- Selection of service providers that can maintain appropriate safeguards; and
- Evaluation and adjustment of the program in light of relevant circumstances, including

(a) changes in the firm's business or operations or

(b) the results of security testing and monitoring.

In 2007, as part of the Fair and Accurate Credit Transaction Act of 2003 (FACT), the FTC promulgated the **Red Flag Rules**. These rules are intended to curb identity theft by having financial institutions identify potential "red flags" for activities conducted through the organization's systems that could lead to identity theft.

The rules apply to financial institutions or those that hold credit accounts

- **Health Insurance Portability and Accountability Act & HITECH Act**. The Health Information Technology for Economic and Clinical Health Act (HITECH ACT) requires notification of a breach of unencrypted health records for all covered entities that are required to comply with the Health insurance Portability and Accountability Act of 1996 (HIPAA)

- **USA PATRIOT Act**. Shortly after September 11, 2001, the United States Congress passed the "Uniting and Strengthening America by Providing

Appropriate Tools Required to Intercept and Obstruct Terrorism Act" (USA PATRIOT Act) of 2001. The USA PATRIOT Act has significant implications for the cloud provider seeking to maintain the privacy of data it holds.

The Act allows the installation of devices to record all routing, addressing, and signaling information kept by a computer.

The Act also extends the U.S. government's ability to gain access to personal financial information and student information stored in electronic systems without any suspicion of wrongdoing of the person whose information it seeks.

International Data Privacy Compliance European Union Data Privacy Directive. In 1995, the European Union (EU) passed the "European Union Directive on the Protection of Individuals with regard to the Processing of Personal Data and the Movement of such Data Privacy Directive" (Directive). The Directive mandated that countries that are part of the EU pass a data protection law covering both government and private entities that process business and consumer data.

The Directive covers written, oral, electronic, and Internet-based data that reside in the EU.

- Argentina's regime is similar to the EU approach.

- Brazil has a constitutional right to privacy. But Brazil has no comprehensive data privacy law; instead it relies on a patchwork of sectoral laws.

- China's constitution refers to privacy indirectly, but the country has very few specific laws.

- On the other hand, Hong Kong has a Personal Data Ordinance that covers public and private data processors and both electronic and non-electronic records.

- India, a popular destination for outsourcing, recognizes a right to privacy against entities in the public sector, but has enacted only a limited number of privacy statutes with scant coverage for the private sector

- **Canada's Personal Information Protection and Electronic Documents Act (PIPEDA).** PIPEDA is intended to "support and promote electronic commerce by protecting personal information that is collected, used, or disclosed in certain circumstances. . ."

- Organizations are held accountable for the protection of personal information it transfers to third parties, whether such parties are inside or outside of Canada.

- Since PIPEDA requires that the contractual arrangements provide a "comparable level of protection while the information is being processed by a third-party

CLOUD CONTRACTING MODELS

Licensing Agreements Versus Services Agreements

- **Summary of Terms of a License Agreement**. A traditional software license agreement is used when a licensor is providing a copy of software to a licensee for its use (which is usually non-exclusive). This copy is not being sold or transferred to the licensee, but a physical copy is being conveyed to the licensee.

- The software license is important because it sets forth the terms under which the software may be used by the licensee

- It also provides a mechanism for the licensor of the software to (among other things) retrieve the copy it provided to the licensee in the event that the licensee

(a) stops complying with the terms of the license agreement or

(b) stops paying the fee the licensee charges for the license. In the case of infringement the license agreement provides a mechanism for the licensor to repair, replace, or remove the software from the licensee's possession

- **Summary of Terms of a Service Agreement**. A service agreement, on the other hand, is not designed to protect against the perils of providing a copy of software to a user.

It is primarily designed to provide the terms under which a service can be accessed or used by a customer.

The service agreement may also set forth quality parameters around which the service will be provided to the users.

Since the software service is controlled by the provider, the attendant risks and issues associated with transferring possession of software without transferring ownership do not exist

- **On-Line Agreements Versus Standard Contracts** There are two contracting models under which a cloud provider will grant access to its services.

The first, the **on-line agreement**, is a click wrap agreement with which a cloud user will be presented before initially accessing the service. A click wrap is the agreement the user enters into when he/she checks an "I Agree" box, or something similar at the initiation of the service relationship.

The agreement is not subject to negotiation and is generally thought to be a contract of adhesion

- The second model, the standard, negotiated, signature-based contract will have its place as well—over time.
- As larger companies move to the cloud (especially the public cloud), or more mission-critical applications or data move to the cloud, the cloud user will most likely require the option or a more robust and user-friendly agreement. The cloud user will push for a negotiated agreement.

Jurisdictional Issues Raised by Virtualization and Data Location

- **Jurisdiction** is defined as a court's authority to judge acts committed in a certain territory.

The geographical location of the data in a cloud computing environment will have a significant impact on the legal requirements for protection and handling of the data.

- **Virtualization and Multi-tenancy**

Virtualization. Computer virtualization in its simplest form is where one physical server simulates being several separate servers. Some **benefits of virtualization** are need for less hardware and consumption of less power across the virtualized enterprise.

Virtualization also provides greater utilization and maximization of hardware processing power. Because of these benefits, virtualization should lower expenses associated with operating a data center. Virtualization across a single or multiple data centers makes it difficult for the cloud user or the cloud provider to know what information is housed on various machines at any given time. The emphasis in the virtualized environment is on maximizing usage of available resources no matter where they reside

- **Multi-tenancy.** Multi-tenancy refers to the ability of a cloud provider to deliver software as-a-service solutions to multiple client organizations (or tenants) from a single, shared instance of the software. The cloud user's information is virtually, not physically, separated from other users. The major benefit of this model is cost-effectiveness for the cloud provider. Some risks or issues with the model for the cloud user include the potential for one user to be able to access data belonging to another user and difficulty to back up and restore data

The Issues Associated with the Flexibility of Data-Location

One of the benefits of cloud computing from the cloud provider's perspective is the ability of the cloud provider to move data among its available data center resources as necessary to maximize the efficiencies of it overall system. From a technology perspective, this ability to move data is a reasonably good solution to the problem of under utilized machines.

Data Protection. In fact, in the cloud environment it is possible that the same data may be stored in multiple locations at the same time. For example, real time-transaction data may be in one geographic location while the backup or disaster recovery systems may be elsewhere. It is also likely that the agreement governing the services says nothing about data location. From a legal perspective, flexibility of data location potentially challenges the governing law provision in the contract. If the law specified in the contract (e.g., the contract says that laws of Thailand will govern this agreement) requires a certain treatment of the data, but the law of the jurisdiction where the data resides (e.g., data center in Poland) requires another treatment, there is an inherent conflict that must be resolved. This conflict exists regardless of whether the storage is temporal, and as part of the processing of the data, or long-term storage that might be a service in itself (i.e., infrastructure as a service), or part of a software or platform as a service offering.

Other Jurisdiction Issues

- Confidentiality and Government Access to Data.

  Each jurisdiction (and perhaps states or provinces within a jurisdiction) has its own regime to protect the confidentiality of information. In the cloud environment, given the potential movement of data among multiple jurisdictions, the data housed in a jurisdiction is subject to the laws of that jurisdiction, even if its owner resides elsewhere. Given the inconsistency of confidentiality protection in various jurisdictions, a cloud user may find that its sensitive data are not entitled to the protection with which the cloud user may be familiar, or that to which it contractually agreed. A government's ability to access data is also directly connected to the jurisdiction in which the data reside. If the jurisdiction has laws that permit its government to get access to data (with or without notice to the cloud user or the individual or entity that owns the data), that data may be subject to interception by the government.

- Subcontracting. A cloud provider's use of a third-party subcontractor to carry out its business may also create jurisdictional issues. The existence or nature of a subcontracting relationship is most likely invisible to the cloud user. If, in the performance of the services, there was a lapse that was due to the subcontractor's performance, the location of the subcontractor or the data acted on by the subcontractor will be difficult for a cloud user to ascertain. As a result, the risk associated with the acts of or the locations of the subcontractor are difficult to measure by the cloud user.

- International Conflicts of Laws

The body of law known as “conflict of laws” acknowledges that the laws of different countries may operate in opposition to each other, even as those laws relate to the same subject matter. In such an event, it is necessary to decide which country’s law will be applied. Every nation is sovereign within its own territory. That means that the laws of that nation affect all property and people within it, including all contracts made and actions carried out within its borders. In a cloud environment, the conflicts of laws issues make the cloud provider’s decisions regarding cross-geography virtualization and multi-tenancy, the cloud user’s lack of information regarding data location, and the potential issues with geographically diverse subcontractors highly relevant.