LECTURE NOTES ON

CLOUD COMPUTING
(AIT007)

Prepared by

Mr. A. Praveen

Assistant Professor

INFORMATION TECHNOLOGY

INSTITUTE OF AERONAUTICAL ENGINEERING
(Autonomous)
Dundigal, Hyderabad -500 043

UNIT – 1

Systems Modelling, Clustering and Virtualization

1. Scalable Computing Over the Internet

1.1 Scalability: Scalability is the capability of a system or network or process to handle a growing amount of works like database storage, software usage and so on [1]. A scalable system should be able to handle the ever-increasing data, levels of computations and should be efficient.

1.2 NOTE: Generally, a computer uses a centralized system to solve the problems. A parallel and distributed computing system uses multiple computers to solve large scale problems over the Internet [2].

1.3 Parallel Computing: Execution of many processes is carried out simultaneously in this case. Large problems can be divided into smaller ones, solved at the same time and integrated later.

1.4 Distributed Computing: A distributed system is a model in which components located on connected computers (through a network) interchange/monitor their actions by passing messages. Distributed computing may refer to systems situated at different physical locations or different actions being performed on the same system.

Distributed Computing is centred on data and based on networks.
NOTE: ***Data Center*** is a centralised repository and distribution of data and information organised around a particular concept (ex: Telecommunications, Health data, business data etc.). A typical data center may have a capacity in Petabytes.

1.5 Internet Computing: Data centers and super computer sites must be upgraded to meet the demands of millions of users who utilize the Internet. High Performance Computing (**HPC**), which was a standard for measuring the system performance, is no longer used. High Throughput Computing (**HTC**) came into existence with emergence of computing clouds. Here, the systems are parallel and distributed.

1.6 Platform Evolution:

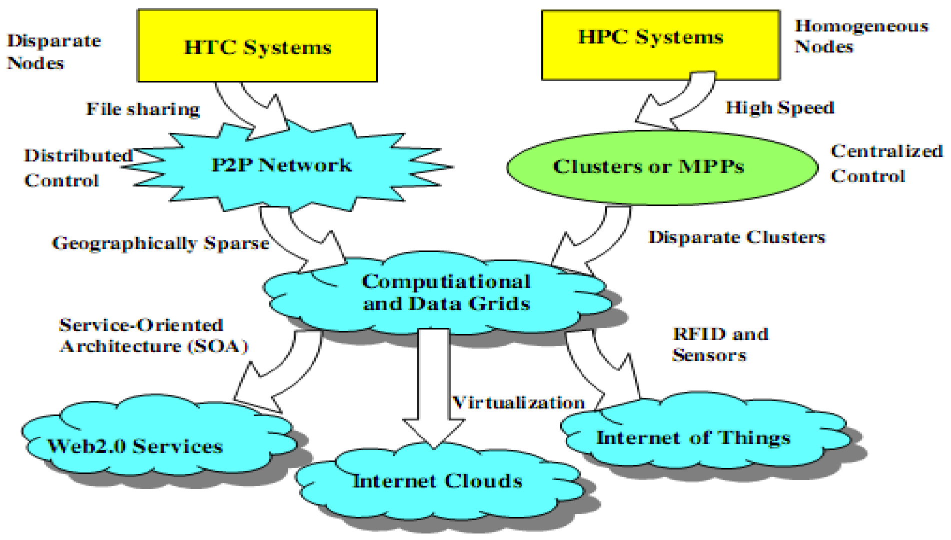


Figure 1.1 [2]: Evolutionary Trend towards parallel, distributed and cloud computing

Computer technology has gone through five generations of development, each spanning at 10 to 20 years. By the start of 1990s, the use of HPC and HTC systems has sky-rocketed. These use clusters, grids, Internet and clouds.

The general trend is to control shared web resources and massive data over the Internet. In the above figure 1.1, we can observe the evolution of HPC and HTC systems.

NOTE: HPC contains super computers which are gradually replaced by clusters of inter-cooperating systems that share the data among them. A **cluster** is a collection of homogeneous computers, which are physically connected.

HTC shows the formation of peer-to-peer (P2P) networks for distributed file sharing and apps. A P2P system is built over many client machines and is globally distributed. This leads to formation of computational grids or data grids.

1.7 High Performance Computing (HPC): HPC stressed upon the speed performance. The speed of HPC systems has increased from Gflops to Pflops (FLOP=> **Fl**oating Point **O**perations **P**er Second) these days, driven by the requirements from different fields like science, engineering, medicine and others [3]. The systems that generally have high speed are super computers, main frames and other servers.

It should be noted here that the number of users (in HPC) is limited – less than 10% of all the users. The majority of the market now uses servers, PCs or mobile devices that conduct Internet searches and other assigned tasks.

1.8 High Throughput Computing: The market-oriented computing is now going through a strategic change from HPC to HTC paradigm (concept). HTC concentrates more on high-flux computing (ex: Internet searches, web apps used by many users simultaneously). The performance goal has shifted from speed of the device to the number of tasks completed per unit of time (throughput).

HTC needs not only to improve the speed but also to solve other problems like time availability, cost, security and reliability.

1.9 New Computing Concepts: It can be seen from Figure 1.1that SOA (Software Oriented Architecture) has made the web services available for all tasks. The Internet Clouds have become a major factor to consider for all types of tasks. Three new paradigms have come into existence:

(a) **Radio-Frequency Identification (RFID)**: This uses electro-magnetic fields to automatically identify and track tags attached to objects [4]. These tags contain electronically stored information.

(b) **Global Positioning System (GPS)**: It is a global navigation satellite system that provides the geographical location and time information to a GPS receiver [5].

(c) **Internet of Things (IoT)**: It is the internetworking of different physical devices (vehicles, buildings etc.) embedded with electronic devices (sensors), software, and network connectivity [6]. Data can be collected and exchanged through this network (IoT).

1.10 Computing Paradigm Distinctions:

(a) **Centralized Computing**: All computer resources like processors, memory and storage are centralized in one physical system. All of these are shared and inter-connected and monitored by the OS.

(b) **Parallel Computing**: All processors are tightly coupled with centralized shared memory or loosely coupled with distributed memory (parallel processing). Inter processor communication is achieved by message passing. This methodology is known as parallel computing.
NOTE: Coupling is the inter-dependence between software/hardware modules.

(c) **Distributed Computing**: A distributed system consists of multiple autonomous computers with each device having its own private memory. They interconnect among themselves by the usage of a computer network. Here also, information exchange is accomplished by message passing.

(d) **Cloud Computing**: An Internet Cloud of resources can either be a centralized or a distributed computing system. The cloud applies parallel or distributed computing or both. Cloud can be built by using physical or virtual resources over data centers. CC is also called as utility/ service/concurrent computing.

1.11 NOTE: IoT is a networked connection of general objects used everyday including computers, systems and sensors. IoT is supported by Internet Clouds to access any ‘thing’ at any place at any time. Internet Computer is a larger concept that covers all computing paradigms, emphasizing on distributed and cloud computing.

1.12 *Explanation on the recent surge in networks of clusters, data grids.* Internet Clouds are the result of moving desktop computing to service-oriented computing using server clusters and huge databases at data centers.

In the future, both HPC and HTC will demand multicore processors that can handle large number of computing threads per core. Both concentrate upon parallel and distributed computing. The main work lies in the fields of throughput, efficiency, scalability and reliability.

Main Objectives:

(a) **Efficiency**: Efficiency is decided by speed, programming and throughput demands' achievement.

(b) **Dependability**: This measures the reliability from the chip to the system at different levels. Main purpose here is to provide good QoS (Quality of Service).

(c) **Adaption in the Programming Model**: This measures the ability to support unending number of job requests over massive data sets and virtualized cloud resources under different models.

(d) **Flexibility:** It is the ability of distributed systems to run in good health in both HPC (science/engineering) and HTC (business).

1.13 **Degrees of 'Parallelism'**:

(a) **Bit-level parallelism** (BLP) 8 bit, 16, 32, and 64.

(b) **Instruction-level parallelism** (ILP): The processor executes multiple instructions simultaneously. Ex: Pipelining, supercomputing, VLIW (very long instruction word), and multithreading.

Pipelining: Data processing elements are connected in series where output of one element is input to the next.

Multithreading: Multithreading is the ability of a CPU or a single core in a multi-core processor to execute multiple processes or threads concurrently, supported by the OS.

(c) **Data-level Parallelism** (DLP): Here, instructions are given like arrays (single instruction, multiple data SIMD). More hardware support is needed.

(d) **Task-level Parallelism (TLP):** It is a process of execution where different threads (functions) are distributed across multiple processors in parallel computing environments.

(e) **Job-level Parallelism (JLP):** Job level parallelism is the highest level of parallelism where we concentrate on a lab or computer center to execute as many jobs as possible in any given time period [7]. To achieve this, we purchase more systems so that more jobs are running at any one time, even though any one user's job will not run faster.

1.14 **Usage of CC**: It is used in different fields for different purposes. All applications demand computing economics, web-scale data collection, system reliability, and scalable performance. Ex: Distributed transaction processing is practiced in the banking industry. Transactions represent 90 percent of the existing market for reliable banking systems. [Give an example of demonetization to increase Internet transactions.]

Table 1.1 Applications of High-Performance and High-Throughput Systems

Domain	**Specific Applications**
Science and engineering	Scientific simulations, genomic analysis, etc.
	Earthquake prediction, global warming, weather forecasting, etc.
Business, education, services industry, and health care	Telecommunication, content delivery, e-commerce, etc.
	Banking, stock exchanges, transaction processing, etc.
	Air traffic control, electric power grids, distance education, etc.
	Health care, hospital automation, telemedicine, etc.
Internet and web services, and government applications	Internet search, data centers, decision-making systems, etc.
	Traffic monitoring, worm containment, cyber security, etc.
	Digital government, online tax return processing, social networking, etc.
Mission-critical applications	Military command and control, intelligent systems, crisis management, etc.

Table 1.1 [2]

1.15 Major computing paradigms and available services/capabilities are coming together to produce a technology convergence of cloud/utility computing where both HPC and HTC are utilised to achieve objectives like reliability and scalability. They also aim to reach autonomic operations that can be self-organized and support dynamic recovery. Ex: Interpretation of sensor data, effectors like Google Home and Amazon Echo, smart home devices etc.

CC focuses on a business model where a customer receives different computing resources (storage, service, security etc.) from service providers like AWS, EMC, Salesforce.com.

A new hype (exciting) cycle is coming into picture where different important and significant works needed by the customer are offered as services by CC. Ex: SaaS, IaaS, Security as a Service, DM as a Service etc. Many others are also along the pipeline.

Figures 1.2 and 1.3 [2] depict various actions discussed above (as in 2010).

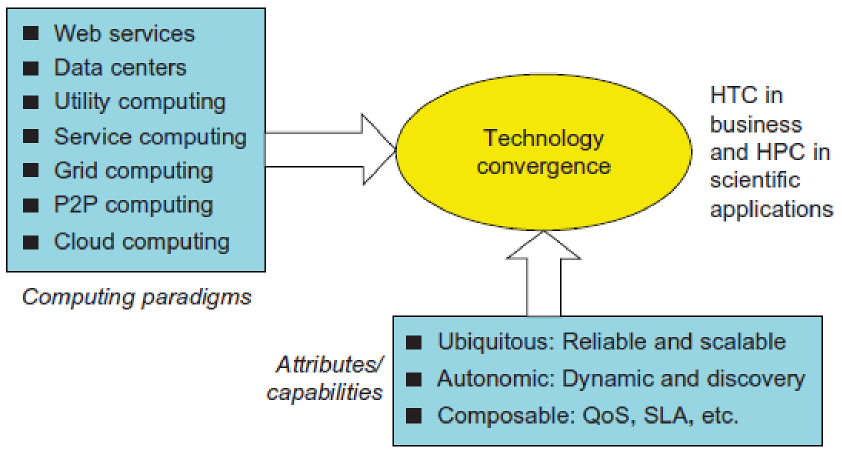


FIGURE 1.2

The vision of computer utilities in modern distributed computing systems.

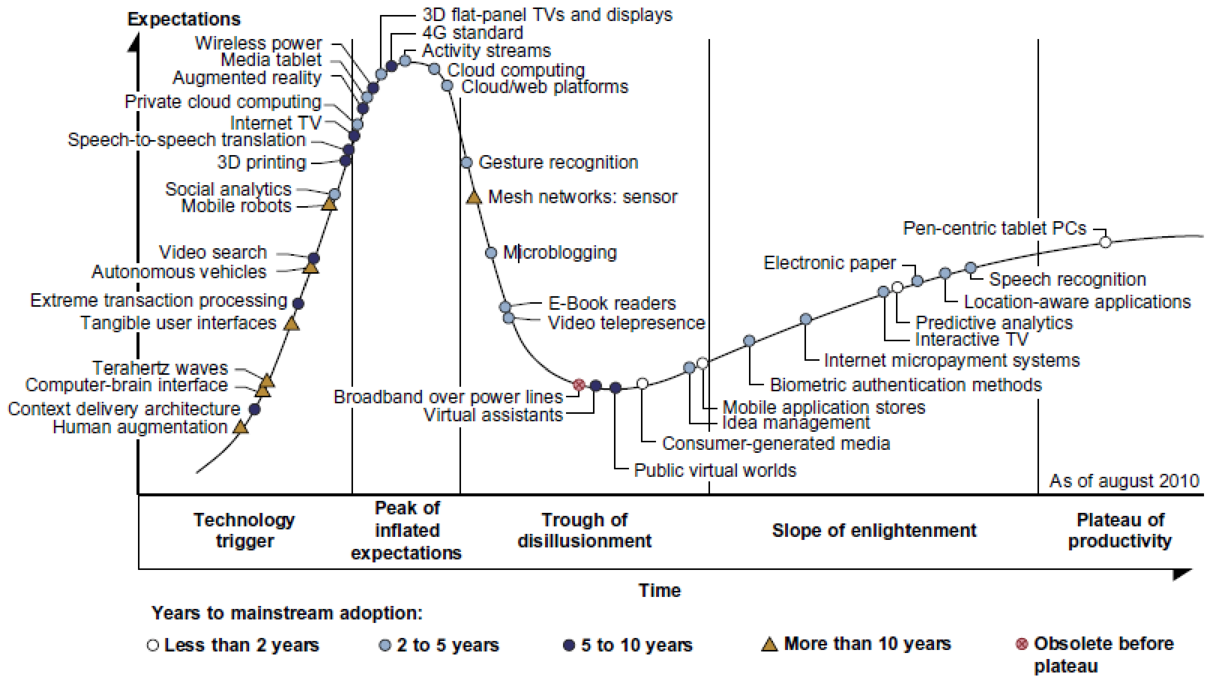


FIGURE 1.3

Hype cycle for Emerging Technologies, 2010

1.16 **Internet of Things**: The IoT [8] refers the networked interconnection of everyday objects, tools, devices or computers. It can be seen as a wireless network of sensors that interconnect all things we use in our daily life. RFID and GPS are also used here. The IoT demands universal addressability of all the objects or things that may be steady or moving.
These objects can be interconnected, can exchange data and interact with each other by the usage of suitable applications (web/mobile). In the IoT era, CC can be used efficiently and in a secure way to provide different services to the humans, computers and other objects. Ex: Smart cities, inter-connected networks, self-controlling street lights/traffic lights etc.

1.17 **NOTE**: CPS means cyber–physical system where physical objects and computational processes interact with each other. Ex: Wrest bands to monitor BP. CPS merges the 3Cs which are computation, communication and control to provide intelligent feedbacks between the cyber and physical worlds.

2. Technologies for Network based Systems

2.1 **Multi-core CPUs and Multithreading Technologies**: Over the last 30 years the speed of the chips and their capacity to handle variety of jobs has increased at an exceptional rate. This is crucial to both HPC and HTC system development. Note that the processor speed is measured in MIPS (millions of instructions per second) and the utilized network bandwidth is measured in Mbps or Gbps.

2.2 **Advances in CPU Processors**: The advanced microprocessor chips (by Intel, NVIDIA, AMD, Qualcomm etc.) assume a multi-core architecture with dual core, quad core or more processing cores. They exploit parallelism at different levels. Moore’s law has proven accurate at these levels. Moore's law is the observation that the number of transistors in a dense integrated circuit doubles approximately every two years.

2.3 **Multi-core CPU**: A multi-core processor is a single computing component with two or more independent actual processing units (called "cores"), which are units that read and execute program instructions [9]. (Ex: add, move data, and branch). The multiple cores can run multiple instructions at the same time, increasing overall speed for programs open to parallel computing.

2.4 **Many-core GPU**: (Graphics Processing Unit) Many-core processors are specialist multi-core processors designed for a high degree of parallel processing, containing a large number of simpler, independent processor cores [10]. Many-core processors are used extensively in embedded computers and high-performance computing. (Main frames, super computers).

2.5 **GPU Computing:** A GPU is a graphics co-processor mounted on a computer's graphics card to perform high level graphics tasks in video editing apps. (Ex: Intel Xeon, NVIDIA). A modern GPU chip can be built with hundreds of processing cores. These days, parallel GPUs or GPU clusters are gaining more attention.

Starting as co-processors attached to the CPU, the GPUs these days possess 128 cores on a single chip (NVIDIA). Hence they have 1024 threads (128*8) executing tasks concurrently, on a single GPU. This can be termed as massive parallelism at multicore and multi-threading levels. GPUs are not restricted to videos only – they can be used in HPC systems to super computers for handling high level calculations in parallel.

2.6 **GPU Programming Model**: Figure 1.7 and 1.8 [2] show the interaction between a CPU and GPU in performing parallel execution of floating-point operations concurrently.

Floating-point operations involve floating-point numbers and typically take longer to execute than simple binary integer operations. A GPU has hundreds of simple cores organised as multiprocessors. Each core can have one or more threads. The CPU instructs the GPU to perform massive data processing where the bandwidth must be matched between main memory and GPU memory.

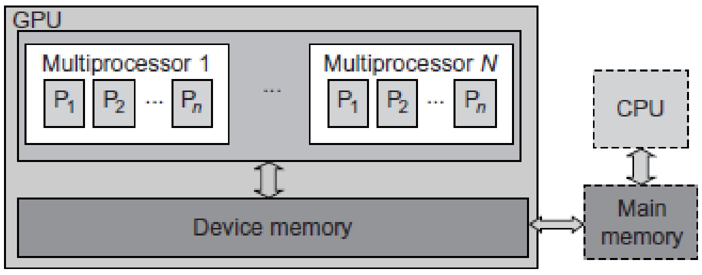


FIGURE 1.7

The use of a GPU along with a CPU for massively parallel execution in hundreds or thousands of processing cores.

2.7 **NOTE**: Bandwidth is the bit-rate of available or consumed information capacity expressed typically in metric multiples of bits per second. Variously, bandwidth may be characterized as network bandwidth, data bandwidth, or digital bandwidth.

2.8 In future, thousand-core GPUs may feature in the field of Eflops/10^{18} flops systems.

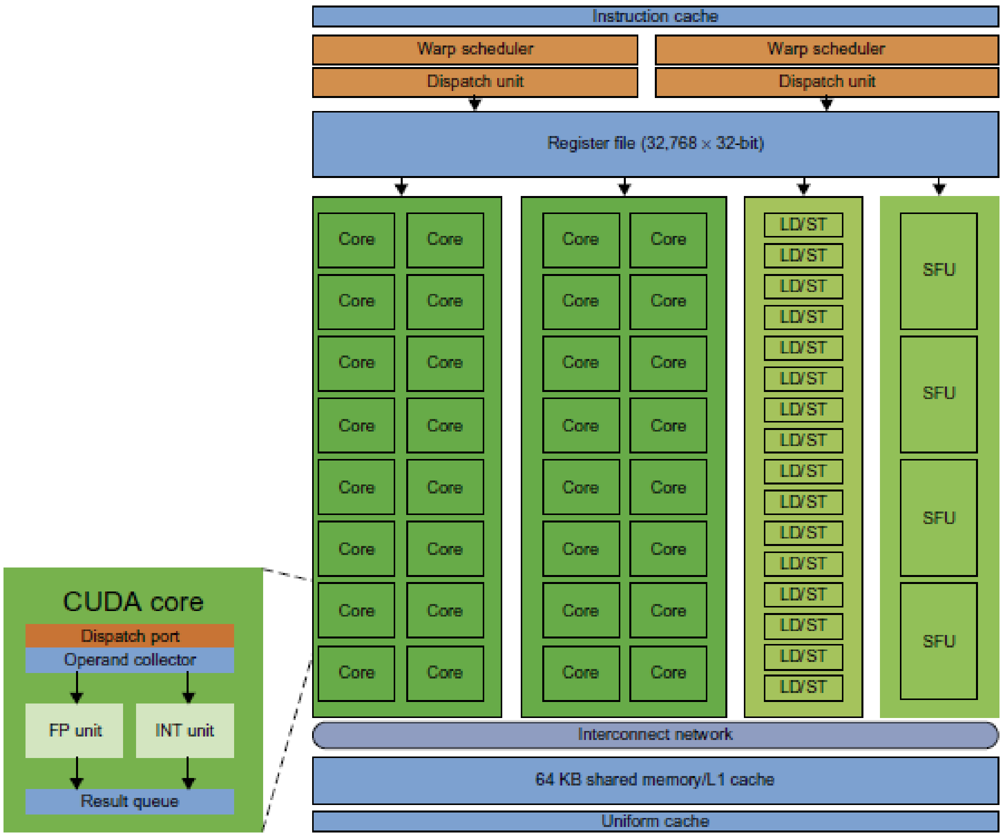


FIGURE 1.8

NVIDIA Fermi GPU built with 16 streaming multiprocessors (SMs) of 32 CUDA cores each; only one SM Is shown. More details can be found also in [49].

2.9 Power Efficiency of the GPU: The major benefits of GPU over CPU are power and massive parallelism. Estimation says that 60 Gflops/watt per core is needed to run an exaflops system. [One exaflops is a thousand petaflops or a quintillion, 10^{18}, floating point operations per second]. A GPU chip requires one-tenth less of the power that a CPU requires. (Ex: CPU: 100, GPU: 90).

CPU is optimized (use most effectively) for latency (time between request and response) in caches and memory; GPU is optimized for throughput with explicit (open) management of on-chip memory.

Both power consumption and software are the future challenges in parallel and distributed systems.

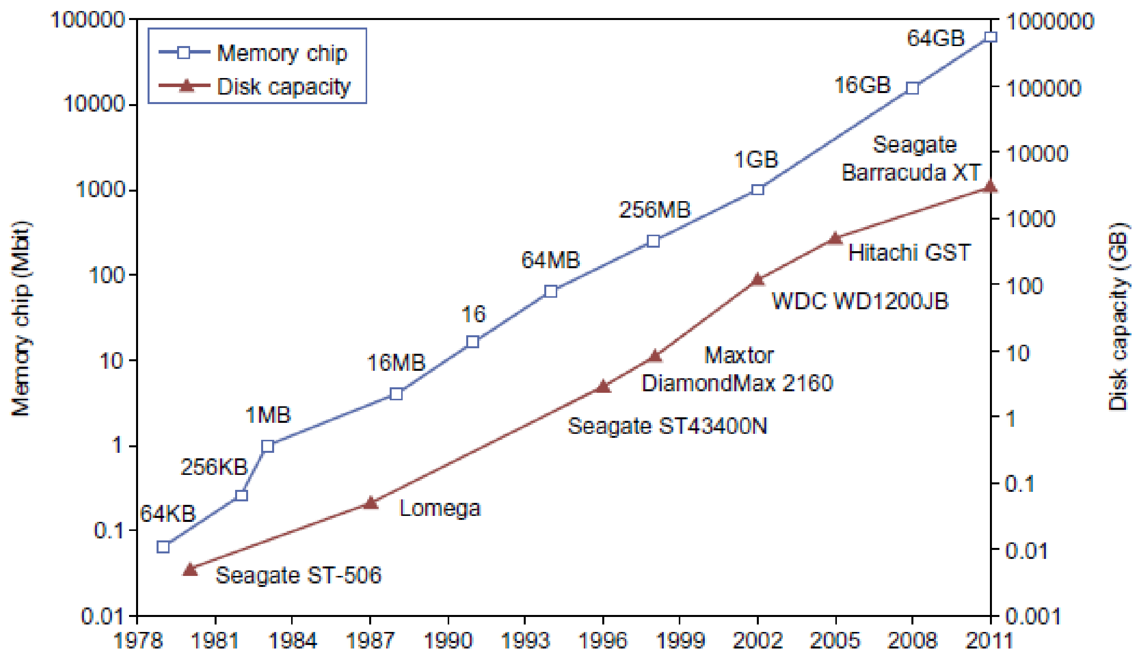


FIGURE 1.10

Improvement in memory and disk technologies over 33 years. The Seagate Barracuda XT disk has a capacity of 3 TB in 2011.

2.10 Memory, Storage and WAN:

(a) **Memory Technology**: The upper curve in Figure 1.10 shows the growth of DRAM chip capacity from 16 KB to 64 GB. [**SRAM** is Static RAM and is 'static' because the memory does not have to be continuously refreshed like **Dynamic RAM**. SRAM is faster but also more expensive and is used inside the CPU. The traditional RAMs in computers are all DRAMs]. For hard drives, capacity increased from 260 MB to 3 TB and lately 5 TB (by Seagate). Faster processor speed and higher memory capacity will result in a wider gap between processors and memory, which is an ever-existing problem.

(b) **Disks and Storage Technology:** The rapid growth of flash memory and solid-state drives (SSD) also has an impact on the future of HPC and HTC systems. An SSD can handle 300,000 to 1 million write cycles per block, increasing the speed and performance. Power consumption should also be taken care-of before planning any increase of capacity.

(c) **System-Area Interconnects**: The nodes in small clusters are interconnected by an Ethernet switch or a LAN. As shown in Figure 1.11 [2], a LAN is used to connect clients to servers. A Storage Area Network (SAN) connects servers to network storage like disk arrays. Network Attached Storage (NAS) connects clients directly to disk arrays. All these types of network appear in a large cluster built with commercial network components (Cisco, Juniper). If not much data is shared (overlapped), we can build a small cluster with an Ethernet Switch + copper cables to link to the end machines (clients/servers).

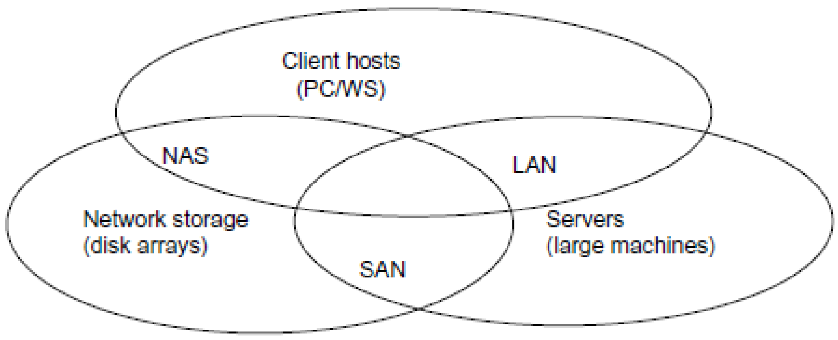


FIGURE 1.11

Three interconnection networks for connecting servers, client hosts, and storage devices; the LAN connects client hosts and servers, the SAN connects servers with disk arrays, and the NAS connects clients with large storage systems in the network environment.

(d) **WAN**: We can also notice the rapid growth of Ethernet bandwidth from 10 Mbps to 1 Gbps and still increasing. Different bandwidths are needed for local, national, and international levels of networks. It is also estimated that computers will be used concurrently in the coming future and higher bandwidth will certainly add more speed and capacity to aid the cloud/distributed computing. Note that most data centers use gigabit Ethernet as interconnect in their server clusters.

2.11 Virtual Machines and Middleware: A typical computer has a single OS image at a time. This leads to a rigid architecture that tightly couples apps to a specific hardware platform i.e., an app working on a system might not work on another system with another OS (non-portable).
To build large clusters, grids and clouds, we need to increase the capacity of computing, storage and networking resources in a virtualized manner. A cloud of limited resources should aggregate all these dynamically to bring out the expected results.

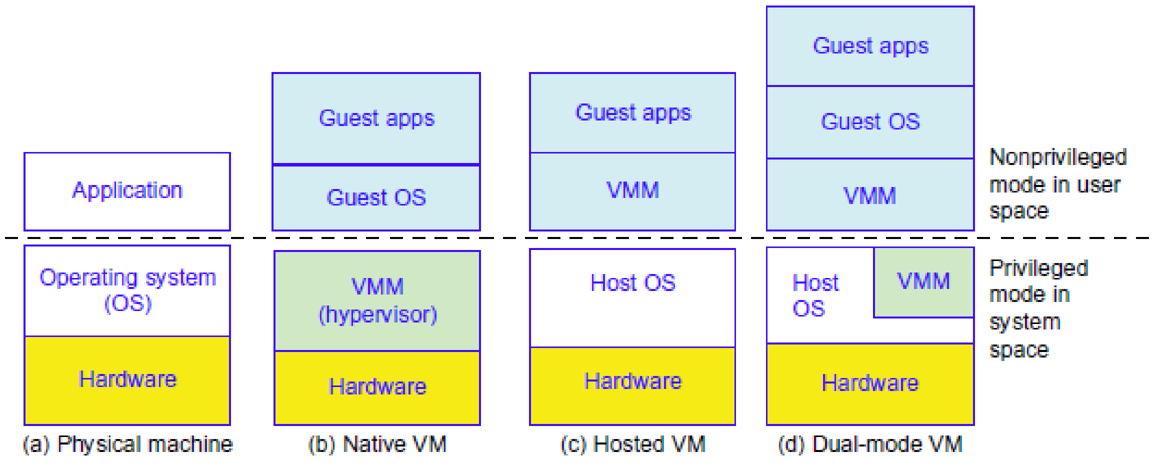


FIGURE 1.12

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

(Courtesy of M. Abde-Majeed and S. Kulkarni, 2009 USC)

(a) **Virtual Machines**: As seen in Figure 1.12 [2], the host machine is equipped with a physical hardware. The VM is built with virtual resources managed by a guest OS to run a specific application (Ex: VMware to run Ubuntu for Hadoop). Between the VMs and the host platform we need a middleware called VM Monitor (VMM). A **hypervisor** (VMM) is a program that allows different operating systems to share a single hardware host. This approach is called bare-metal VM because a hypervisor handles CPU, memory and I/O directly. VM can also be implemented with a dual mode as shown in Figure 1.12 (d). Here, part of VMM runs under user level and another part runs under supervisor level.

NOTE: The VM approach provides hardware independence of the OS and apps. The VM can run on an OS different from that of the host computer.

(b) **VM Primitive operations**: A VMM operation provides VM abstraction to the guest OS. The VMM can also export an abstraction at full virtualization so that a standard OS can run it as it would on physical hardware. Low level VMM operations are indicated in Figure 1.13 [2].

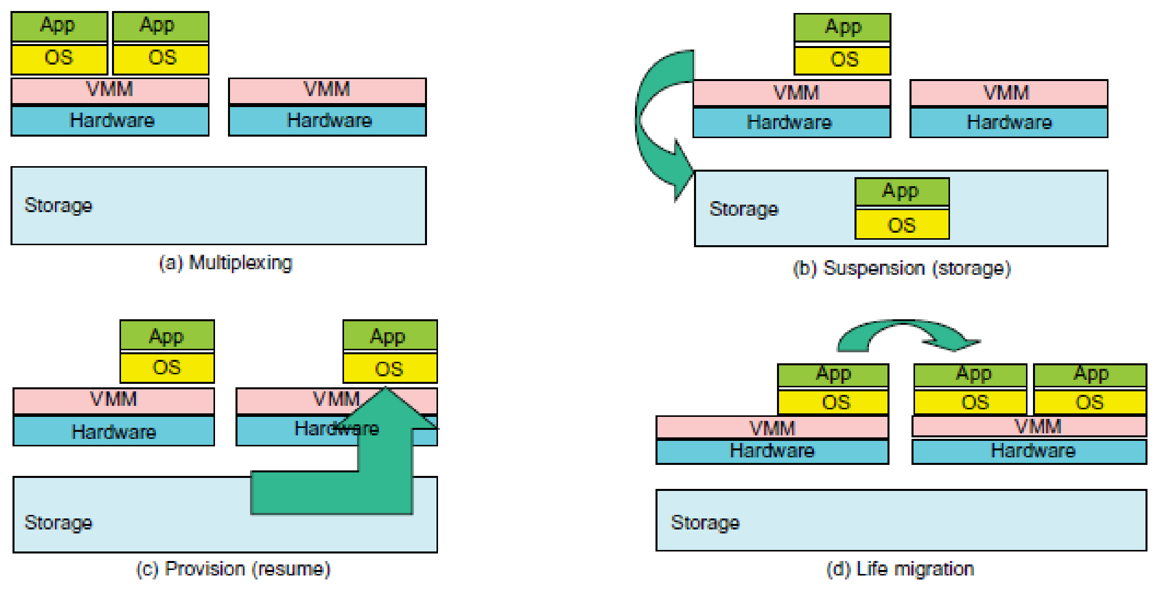


FIGURE 1.13

VM multiplexing, suspension, provision, and migration in a distributed computing environment.

(Courtesy of M. Rosenblum, Keynote address, ACM ASPLOS 2006 [41])

- The VMs can be multiplexed between hardware machines as shown in 1.13 (a)
- A VM can be suspended and stored in a stable storage as shown in 1.13(b)
- A suspended VM can be resumed on a new hardware platform as shown in 1.13 (c)
- A VM can be migrated from one hardware platform to another as shown in 1.13 (d)

Advantages:

- These VM operations can enable a VM to work on any hardware platform.
- They enable flexibility (the quality of bending easily without breaking) in porting distributed app executions.
- VM approach enhances the utilization of server resources – multiple server functions can be integrated on the same hardware platform to achieve higher system efficiency. [VMware claims that server resource utilization has increased from 5-15% to 60-80%].
- Eliminates server crashes due to VM usage or shows more transparency in the operations that are being carried out.

(c) **Virtual Infrastructures**: Virtual Infrastructure connects resources to distributed applications in such a way that a resource needed by an app is exactly mapped to it. This decreases the costs and increases efficiency and server response.

2.12 Data Center Virtualization for Cloud Computing: Cloud architecture is built with products like hardware and network devices. Almost all cloud platforms use x86 (Family of Intel 8086 processors). Low-cost terabyte disks and gigabit Ethernet are used to build data centers. A data center takes into consideration the performance/price ratio instead of only speed.

(a) **Data Center Growth and Cost Breakdown**: Large data centers are built with thousands of servers and smaller ones have hundreds of the same. The cost of maintaining a data center has increased and much of this money is spent on management and maintenance which did not increase with time. Electricity and cooling also consume much of the allocated finance.

(b) **Low Cost Design Philosophy:** High-end switches or routers that provide high bandwidth networks cost more and do not match the financial design of cloud computing. For a fixed budget, typical switches and networks are more desirable.

Similarly, usage of x86 servers is more preferred over expensive mainframes. Appropriate software 'layer' should be able to balance between the available resources and the general requirements like network traffic, fault tolerance, and expandability. [*Fault tolerance* is the property that enables a system to continue operating properly even when one or more of its components have failed].

(c) **Convergence of Technologies**: CC is enabled by the convergence of technologies in four areas:

- Hardware virtualization and multi-core chips
- Utility and grid computing
- SOA, Web 2.0 and Web Service integration
- Autonomic Computing and Data Center Automation

Web 2.0 is the second stage of the development of the Internet, where static pages transformed into dynamic and the growth of social media.

Data is increasing by leaps and bounds every day, coming from sensors, simulations, web services, mobile services and so on. Storage, acquisition and access of this huge amount of data sets requires standard tools that support high performance, scalable file systems, DBs, algorithms and visualization. With science becoming data-centric, storage and analysis of the data plays a huge role in the appropriate usage of the data-intensive technologies.

Cloud Computing is basically focused on the massive data that is flooding the industry. CC also impacts the e-science where multi-core and parallel computing is required. To achieve the goals in these fields, one needs to work on workflows, databases, algorithms and virtualization issues.

Cloud Computing is a transformative approach since it promises more results than a normal data center. The basic interaction with the information is taken up in a different approach to obtain a variety of results, by using different types of data to end up with useful analytical results.
It should also be noted that a cloud provides sources on demand at the infrastructure, platform, or software level. At platform level, MapReduce offers a new programming model that transparently handles data parallelism with natural fault tolerance capability. Iterative MapReduce extends MapReduce to support a broader range of DM algorithms.

A typical cloud runs on an extremely large cluster of standard PCs. In each cluster node, multithreading is practised with a large number of cores in many-core GPU clusters. Hence, data science, cloud computing and multi-core computing are coming together to revolutionize the next generation of computing and take up the new programming challenges.

2.13 System Models for Cloud Computing: Distributed and Cloud Computing systems are built over a large number of independent computer nodes, which are interconnected by SAN, LAN or WAN. Few LAN switches can easily connect hundreds of machines as a working cluster. A WAN can connect many local clusters to form large cluster of clusters. In this way, millions of computers can be brought together by using the above mentioned methodology, in a hierarchical manner.

Large systems are highly scalable, and can reach web-scale connectivity either physically or logically. Table 1.2 [2] below shows massive systems classification as four groups: clusters, P2P networks, computing grids and Internet clouds over large data centers. These machines work collectively, cooperatively, or collaboratively at various levels.

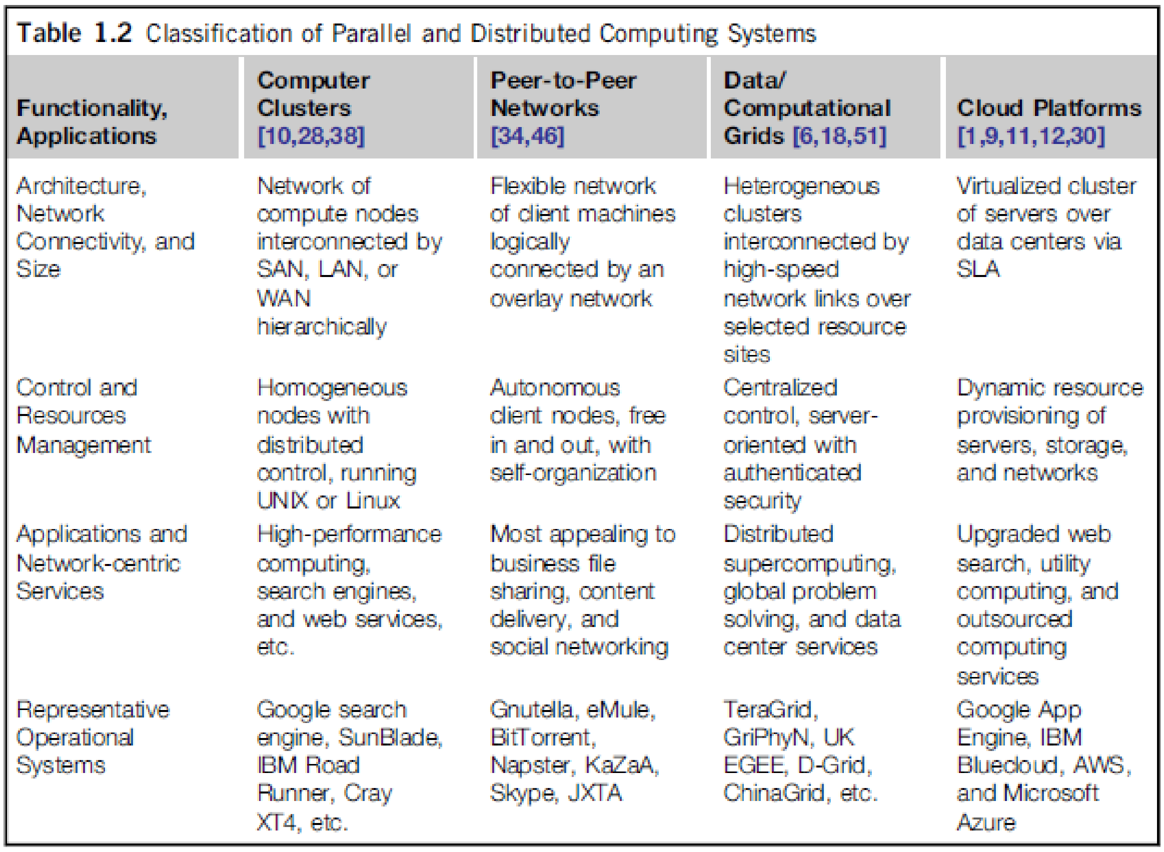


Table 1.2 Classification of Parallel and Distributed Computing Systems

Functionality, Applications	**Computer Clusters [10,28,38]**	**Peer-to-Peer Networks [34,46]**	**Data/ Computational Grids [6,18,51]**	**Cloud Platforms [1,9,11,12,30]**
Architecture, Network Connectivity, and Size	Network of compute nodes interconnected by SAN, LAN, or WAN hierarchically	Flexible network of client machines logically connected by an overlay network	Heterogeneous clusters interconnected by high-speed network links over selected resource sites	Virtualized cluster of servers over data centers via SLA
Control and Resources Management	Homogeneous nodes with distributed control, running UNIX or Linux	Autonomous client nodes, free in and out, with self-organization	Centralized control, server-oriented with authenticated security	Dynamic resource provisioning of servers, storage, and networks
Applications and Network-centric Services	High-performance computing, search engines, and web services, etc.	Most appealing to business file sharing, content delivery, and social networking	Distributed supercomputing, global problem solving, and data center services	Upgraded web search, utility computing, and outsourced computing services
Representative Operational Systems	Google search engine, SunBlade, IBM Road Runner, Cray XT4, etc.	Gnutella, eMule, BitTorrent, Napster, KaZaA, Skype, JXTA	TeraGrid, GriPhyN, UK EGEE, D-Grid, ChinaGrid, etc.	Google App Engine, IBM Bluecloud, AWS, and Microsoft Azure

2.14 Clusters are more popular in supercomputing apps. They have laid the foundation for cloud computing. P2P are mostly used in business apps. Many grids formed in the previous decade have not been utilized per their potential due to lack of proper middleware or well-coded apps.

NOTE: The advantages of cloud computing include its low cost and simplicity for providers and users.

2.15 **Clusters of Cooperative Computers**: A computing cluster consists of inter-connected standalone computers which work jointly as a single integrated computing resource. Particularly, this approach yields good results in handling heavy workloads with large datasets.

(a) The Figure 1.1.5 [2] below shows the architecture of a typical server cluster that has low latency and high bandwidth network. [**Latency** is the delay from input into a system to desired outcome]. For building a large cluster, an interconnection network can be utilized using Gigabit Ethernet, Myrinet or InfiniBrand switches.

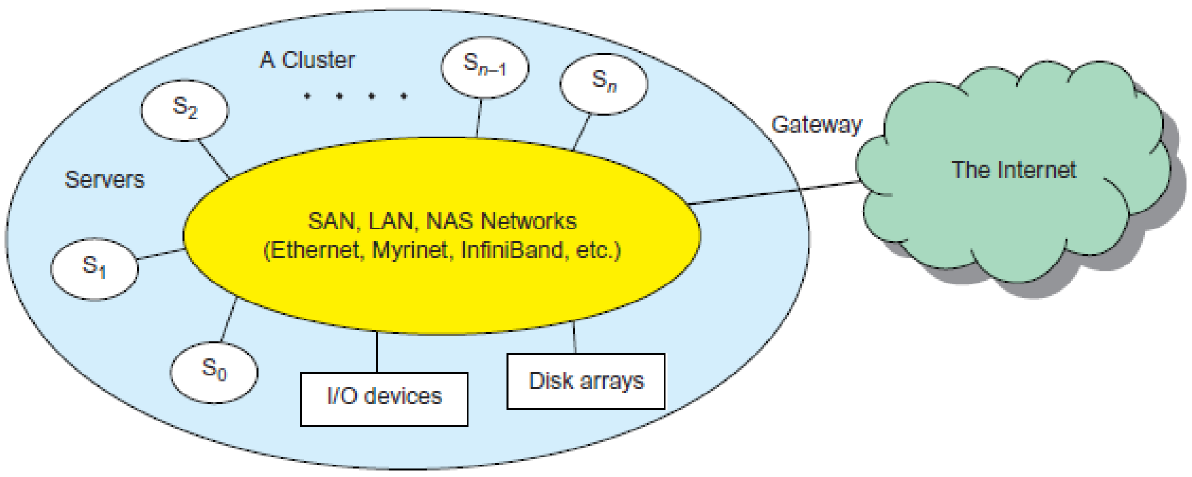


FIGURE 1.15

A cluster of servers interconnected by a high-bandwidth SAN or LAN with shared I/O devices and disk arrays; the cluster acts as a single computer attached to the Internet.

Through a hierarchical construction using SAN, LAN or WAN, scalable clusters can be built with increasing number of nodes. The concerned cluster is connected to the Internet through a VPN (Virtual Private Network) gateway, which has an IP address to locate the cluster. Generally, most clusters have loosely connected nodes, which are autonomous with their own OS.

(b) **Single-System Image (SSI)**: It was indicated that multiple system images should be integrated into a single-system image for a cluster. A cluster-OS is more desired these days, or a middleware to support SSI that includes sharing of CPUs, memory, I/O across all the nodes in the cluster. An SSI is an illusion (something that doesn't exist actually) that shows the integrated resources as a single and powerful resource. SSI can be created by software or hardware. Finally, a cluster is with multiple system images is only a collection of the resources of independent computers that are loosely inter-connected.

(c) **HW, SW and MW Support**: It should be noted that MPPs (Massively Parallel Processing) are clusters exploring high-level parallel processing. The building blocks here are the computer nodes (PCs, Symmetric Multi-Processors (SMPs), work stations or servers), communication software like Parallel Virtual Machine (PVM), Message Passing Interface (MPI), and a network interface card in each node. All the nodes are interconnected by high bandwidth network (Ex: Gigabit Ethernet).

To create SSIs, we need special cluster middleware support. Note that both sequential and parallel apps can run on the cluster but parallel environments give effective exploitation of the resources. Distributed Shared memory (DSM) makes all the data to be shared by all the clusters, thus bringing all the resources into availability of every user. But SSI features are expensive and difficult to achieve; so users generally prefer loosely coupled machines.

(d) **Major Cluster Design Issues**: A cluster-wide OSs or a single OS controlling the cluster virtually is not yet available. This makes the designing and achievement of SSI difficult and expensive. All the apps should rely upon the middleware to bring out the coupling between the machines in cluster or between the clusters. But it should also be noted that the major advantages of clustering are scalable performance, efficient message passing, high system availability, good fault tolerance and a cluster-wide job management which react positively to the user demands.

2.16 **Grid Computing Infrastructures**: Grid computing is designed to allow close interaction among applications running on distant computers simultaneously.

(a) **Computational Grids**: A computing grid provides an infrastructure that couples computers, software/hardware, sensors and others together. The grid can be constructed across LAN, WAN and other networks on a regional, national or global scale. They are also termed as *virtual platforms*.

Computers, workstations, servers and clusters are used in a grid. Note that PCs, laptops and others can be viewed as access devices to a grid system. Figure 1.6 [2] below shows an example grid built by different organisations over multiple systems of different types, with different operating systems.

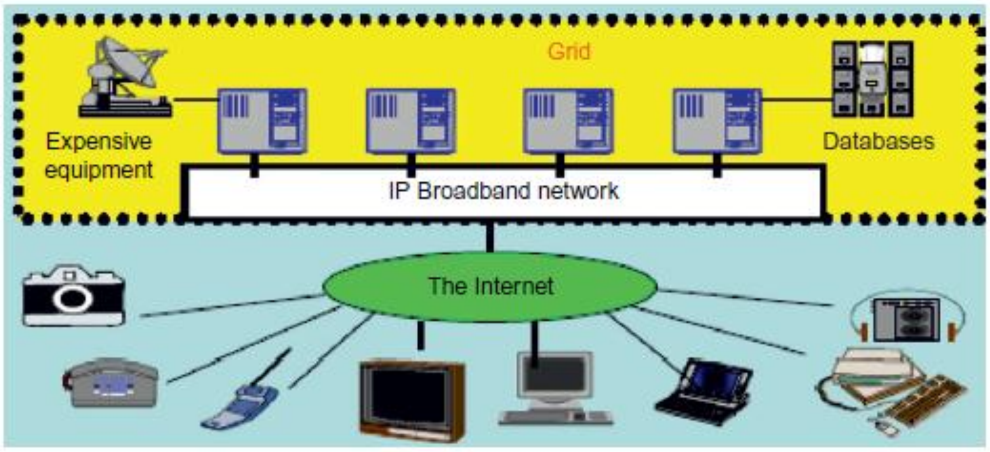


FIGURE 1.16

Computational grid or data grid providing computing utility, data, and information services through resource sharing and cooperation among participating organizations.

(b) Grid Families: Grid technology demands new distributed computing models, software/middleware support, network protocols, and hardware infrastructures. National grid projects are followed by industrial grid platforms by IBM, Microsoft, HP, Dell-EMC, Cisco, and Oracle. New grid service providers (GSPs) and new grid applications have emerged rapidly, similar to the growth of Internet and web services in the past two decades. Grid systems are classified in essentially two categories: computational or data grids and P2P grids. Computing or data grids are built primarily at the national level.

2.17 Peer-to-Peer Network Families: In the basic client-server architecture, the client machines are connected to a central server for different purposes and these are essentially P2P networks. The P2P architecture offers a distributed model of networked systems. Note that P2P network is client-oriented instead of server-oriented.

(a) P2P Systems: Here, every node acts as both a client and a server. Peer machines are those connected to the Internet; all client machines act autonomously to join or leave the P2P system at their choice. No central coordination DB is needed. The system is self-organising with distributed control.

Basically, the peers are unrelated. Each peer machine joins or leaves the P2P network at any time. The participating peers form the *physical network* at any time. This physical network is not a dedicated interconnection but a simple ad-hoc network at various Internet domains formed randomly.

(b) Overlay Networks: As shown in Figure 1.17 [2], an overlay network is a virtual network formed by mapping each physical machine with its ID, through a virtual mapping.

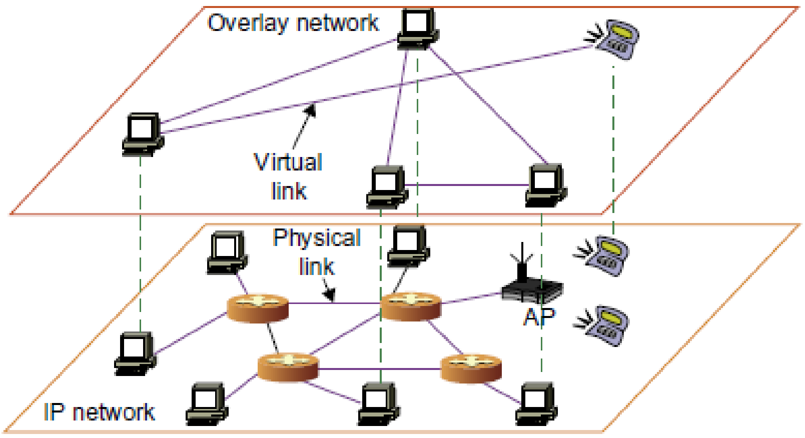


FIGURE 1.17

The structure of a P2P system by mapping a physical IP network to an overlay network built with virtual links.

If a new peer joins the system, its peer ID is added as a node in the overlay network. The P2P overlay network distinguishes the logical connectivity among the peers. The types here are unstructured and structured. Unstructured P2P ON is a random one and has no fixed route of contact – flooding is used to send queries to all nodes. This resulted in sudden increase of network traffic and unsure results. On the other hand, structured ONs follow a pre-determined methodology of connectivity for inserting and removing nodes from the overlay graph.

(c) **P2P Application Families**: There exist 4 types of P2P networks: distributed file sharing, collaborative platform, distributed P2P computing and others. Ex: BitTorrent, Napster, Skype, Geonome, JXTA, .NET etc.

(d) **P2P Computing Challenges:** The main problems in P2P computing are those in hardware, software and network. Many hardware models exist to select from; incompatibility exists between the software and the operating systems; different network connections and protocols make it too complex to apply in real-time applications. Further, data location, scalability, performance, bandwidth etc. are the other challenges.

P2P performance is further affected by routing efficiency and self-organization among the peers. Fault tolerance, failure management, load balancing, lack of trust among the peers (for security, privacy and copyright violations), storage space availability are the other issues that have to be taken care of. But it should also be noted that the distributed nature of P2P network increases robustness since the failure of some peers doesn't affect the full network – fault tolerance is good.

Disadvantages here are that since the total system is not centralized, management of the total network is difficult – anyone can logon and put in any type of data. Security is less.

NOTE: P2P computing or networking is a distributed application architecture that partitions tasks or workloads between peers [11].

It can be concluded that P2P networks are useful for small number of peers but not for large networks with multiple peers.

2.18 Cloud Computing over Internet: **Cloud Computing is defined** by IBM as follows: A cloud is a pool of virtualized computer resources. A cloud can host a variety of different workloads that include batch-style backend jobs and interactive and user-facing applications.

Since the explosion of data the trend of computing has changed – the software apps have to be sent to the concerned data. Previously, the data was transferred to the software for computation. This is the main reason for promoting cloud computing.

A cloud allows workloads to be deployed and scaled out through rapid provisioning of physical or virtual systems. The cloud supports *redundant, self-recovering, and highly scalable programming models* that allow workloads to recover from software or hardware failures. The cloud system also monitors the resource use in such a way that allocations can be rebalanced when required.

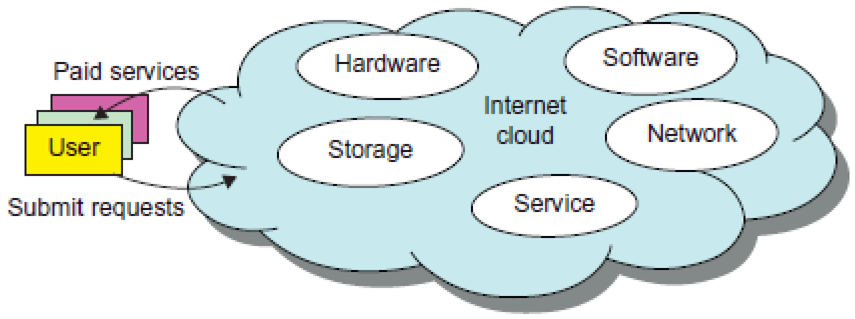


FIGURE 1.18

Virtualized resources from data centers to form an Internet cloud, provisioned with hardware, software, storage, network, and services for paid users to run their applications.

(a) **Internet Clouds**: The idea in CC is to move desktop computing to a service-oriented platform using server clusters and huge DBs at data centers. CC benefits both users and providers by using its low cost and simple resources through machine virtualization. Many user applications are satisfied simultaneously by CC and finally, its design should satisfy the security norms, be trustworthy and dependable. CC is viewed in two ways: a centralized resource pool or a server cluster practising distributed computing.

(b) **The Cloud Landscape**: A distributed computing system is controlled by companies or organisations. But these traditional systems encounter several bottlenecks like constant maintenance, poor utilization, and increasing costs and updates of software or hardware. To get rid of these, CC should be utilized as on-demand computing.

CC offers different types of computing as services:

- **Infrastructure as a Service (IaaS)**: This model provides different infrastructures like servers, storage, networks and the data center fabric (here, databases) to the user on demand. A typical user can deploy and run multiple VMs where guest operating systems can be used for specific applications. Note that that the user cannot manage or control the cloud infrastructure but can specify when tor request and release the concerned resources. Ex: AWS, MS Azure, Cisco Metapod, Google Compute Engine etc.
- **Platform as a Service (PaaS)**: In this model, the user can install his own apps onto a virtualized cloud platform. PaaS includes middleware, DBs, development tools, and some computing languages. It includes both hardware and software. The provider supplies the API and the software tools (ex: Java, Python, .NET). The user need not manage the cloud infrastructure which is taken care of by the provider.
- **Software as a Service (SaaS)**: It is browser-initiated application software paid cloud customers. This model is used in business processes, industry applications, CRM, ERP, HR and collaborative (joint) applications. Ex: Google Apps, Twitter, Facebook, Cloudera, Salesforce etc.

(c) Inter clouds offer four deployment models: private, public, managed and hybrid.

- **Private Cloud**: Private cloud is a type of cloud computing that delivers similar advantages to public cloud, including scalability and self-service, but through a proprietary architecture.

- **Public Cloud**: A public cloud is one based on the standard cloud computing model, in which a service provider makes resources, such as applications and storage, available to the general public over the Internet.
- **Managed Cloud**: Managed cloud hosting is a process in which organizations share and access resources, including databases, hardware and software tools, across a remote network via multiple servers in another location. [12]
- **Hybrid Cloud**: A hybrid cloud is an integrated cloud service utilising both private and public clouds to perform distinct functions within the same organisation. [13]

2.19 **NOTE**: The different service level agreements (SLAs) mean that the security responsibility is shared among all the cloud providers, consumers, and the third-party cloud-enabled software service providers.

2.20 **Software Environments for Distributed Systems and Clouds – SOA:** In grids that use Java/CORBA, an entity is a service or an object. Such architectures build on the seven OSI layers (APSTNDP) that provide networking abstractions. Above this we have a base service environment like .NET, Java etc. and a broker network for CORBA, which enables collaboration between systems on different operating systems, programming languages and hardware [14]. By using this base, one can build a higher level environment reflecting the special features of distributed computing. The same is reflected in the figure 1.20 [2] below.

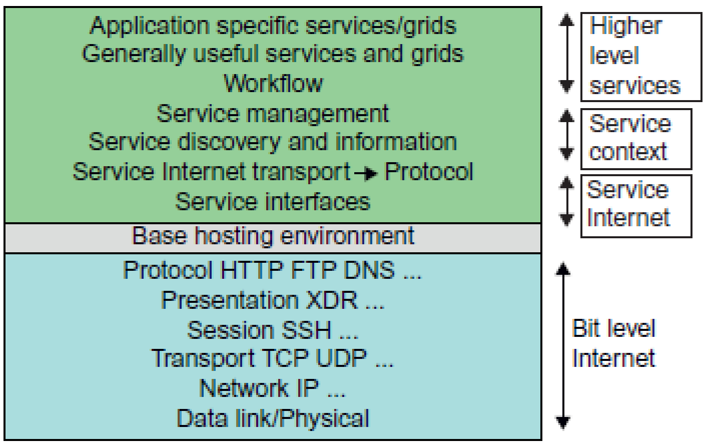


FIGURE 1.20

Layered achitecture for web services and the grids.

(a) **Layered Architecture for Web Services and Grids**: The entity interfaces correspond to the WSDL (web services description language) like XML, Java and CORBA interface definition language (IDL) in the distributed systems. These interfaces are linked with high level communication systems like SOAP, RMI and IIOP. These are based on message-oriented middleware infrastructures like JMS and Web Sphere MQ.

At entity levels, for fault tolerance, the features in (Web Services Reliable Messaging) WSRM and its framework are same as the levels of OSI model. Entity communication is supported by higher level services for services, metadata, and the management of entities, which can be discussed later on. Ex: JNDI, CORBA trading service, UDDI, LDAP and ebXML. Note that the services have a common service: a shared memory. This enables effective exchange of information. This also results in higher performance and more throughputs.

(b) **Web Services and Tools**: Loose Coupling and support of heterogeneous implementations make services (SaaS, IaaS etc.) more attractive than distributed objects. It should be realised that the above figure corresponds to two choices of service architecture: web services or (Representational State Transfer) REST systems.

In web services, the aim is to specify all aspects of the offered service and its environment. This idea is carried out by using SOAP. Consequently, the environment becomes a universal distributed OS with fully distributed capability carried out by SOAP messages. But it should be noted that this approach has had mixed results since the protocol can't be agreed upon easily and even if so, it is hard to implement.

In the REST approach, simplicity is stressed upon, and difficult problems are delegated to the apps. In a web services language, REST has minimal information in the header and the message body carries the needed information. REST architectures are more useful in rapid technology environments. Above the communication and management layers, we can compose new entities or distributed programs by grouping several entities together.

Java and CORBA use RPC methodology through RMI. In grids, sensors represent entities that output data as messages; grids and clouds represent collection of services that have multiple message-based inputs and outputs.

(c) The Evolution of SOA: Software Oriented Architecture applies to building grids, clouds, their combinations and even inter-clouds and systems of systems. The data collections is done through the sensors like ZigBee device, Bluetooth device, Wi-Fi access point, a PC, a mobile phone and others. All these devices interact among each other or with grids, clouds and databases at distant places.

Raw Data → Data → Information → Knowledge → Wisdom → Decisions

(d) Grids Vs Clouds: Grid systems apply static resources, while a cloud stresses upon elastic resources. Differences between grid and cloud exist only in dynamic resource allocation based on virtualization and autonomic computing. A 'grid of clouds' can also be built and can do a better job than a pure cloud because it can support resource allocation. Grid of clouds, cloud of grids, cloud of clouds and inter-clouds are also possible.

2.21 Distributed Operating Systems: To promote resource sharing and fast communication, it is best to have a distributed operating system that can manage the resources efficiently. In distributed systems or more generally, a network needs an operating system itself since it deals with many heterogeneous platforms. But such an OS offers low transparency to the users. It should be noted that middleware can also be used to generate resource sharing but only till we attain a certain level. The third approach is to develop a truly distributed OS to achieve highest efficiency and maximum transparency. Comparison can be seen in Table 1.6 [2].

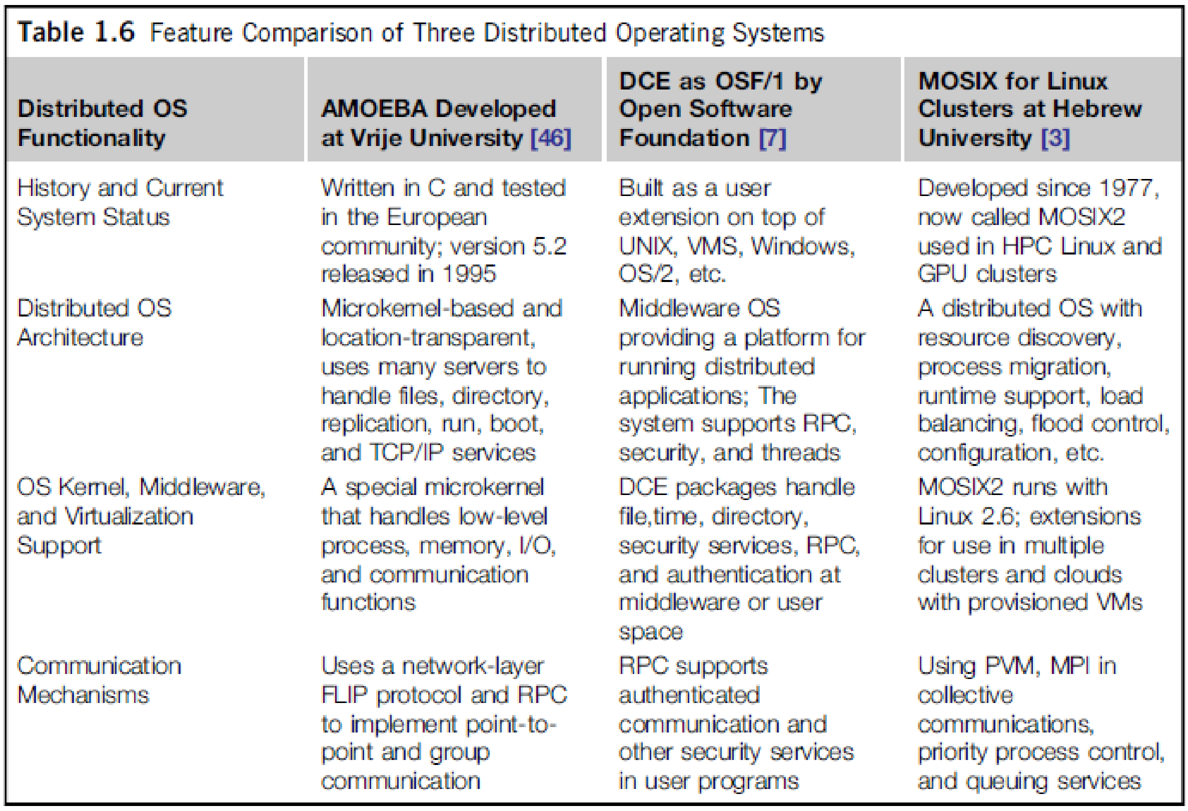


Table 1.6 Feature Comparison of Three Distributed Operating Systems

Distributed OS Functionality	AMOEBA Developed at Vrije University [46]	DCE as OSF/1 by Open Software Foundation [7]	MOSIX for Linux Clusters at Hebrew University [3]
History and Current System Status	Written in C and tested in the European community; version 5.2 released in 1995	Built as a user extension on top of UNIX, VMS, Windows, OS/2, etc.	Developed since 1977, now called MOSIX2 used in HPC Linux and GPU clusters
Distributed OS Architecture	Microkernel-based and location-transparent, uses many servers to handle files, directory, replication, run, boot, and TCP/IP services	Middleware OS providing a platform for running distributed applications; The system supports RPC, security, and threads	A distributed OS with resource discovery, process migration, runtime support, load balancing, flood control, configuration, etc.
OS Kernel, Middleware, and Virtualization Support	A special microkernel that handles low-level process, memory, I/O, and communication functions	DCE packages handle file,time, directory, security services, RPC, and authentication at middleware or user space	MOSIX2 runs with Linux 2.6; extensions for use in multiple clusters and clouds with provisioned VMs
Communication Mechanisms	Uses a network-layer FLIP protocol and RPC to implement point-to-point and group communication	RPC supports authenticated communication and other security services in user programs	Using PVM, MPI in collective communications, priority process control, and queuing services

2.22 Amoeba vs DCE: Distributed Computing Environment is a middleware-based system for DCEs. Amoeba was developed by academicians in Holland. But it should be noticed that DCE, Amoeba and MOSIX2 are all research prototypes used only in academia.

MOSIX2 vs Linux Clusters: MOSIX is a distributed OS, which runs with a virtualization layer in the Linux environment. This layer provides a single-system image to user apps. MOSIX supports both sequential and parallel apps and the resources are discovered and migrated among the Linux nodes. (MOSIX uses Linux Kernel). A MOSIX enabled grid can extend indefinitely as long as interoperation the clusters exists.

Transparency in programming environments that handle user data, OS, and hardware plays a key role in the success of clouds. This concept is divided into 4 levels as seen below [2]: Data, app, OS, and hardware. Users will be able to chose the OS they like as well as the app they like – this is the main concept behind Software as a Service (SaaS).

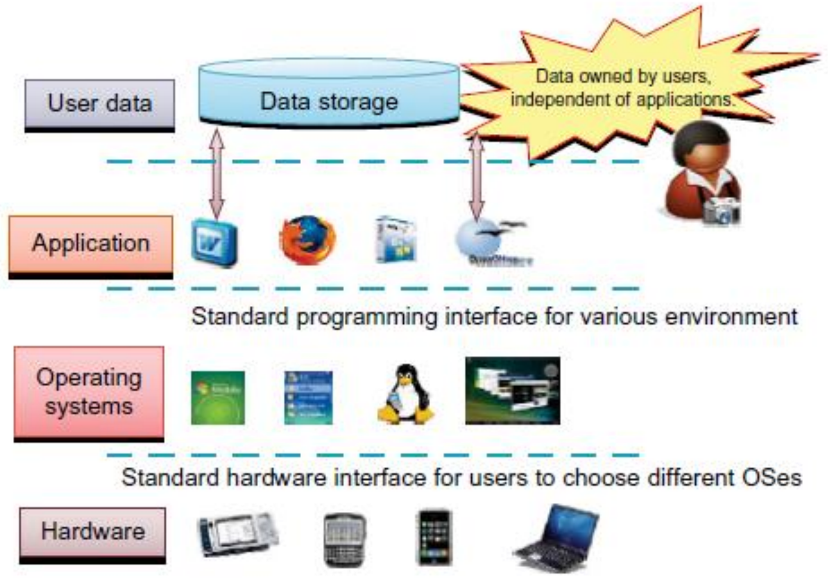

User data
Data storage
Data owned by users, independent of applications.
Application
Standard programming interface for various environment
Operating systems
Standard hardware interface for users to choose different OSes
Hardware

FIGURE 1.22

A transparent computing environment that separates the user data, application, OS, and hardware in time and space – an ideal model for cloud computing.

2.23 **Message-Passing Interface (MPI)**: MPI is a library of sub-programs that can be called from C or FORTRAN to write parallel programs running on a distributed system. The goal here is to represent clusters, grid systems, and P2P systems with upgraded web services and other utility apps. Distributed programming can also be supported by Parallel Virtual Machine (PVM).

2.24 **MapReduce**: it is a web programming model for scalable data processing on large data clusters. It is applied mainly in web-scale search and cloud computing apps. The user specifies a Map function to generate a set of intermediate key/value pairs. Then the user applies a Reduce function to merge all intermediate values with the same (intermediate) key. MapReduce is highly scalable to explore high degrees of parallelism at different job levels and can handle terabytes of data on thousands of client machines. Many MapReduce programs can be executed simultaneously. Ex: Google's clusters.

2.25 **Hadoop Library**: Hadoop enables users to write and run apps over vast amounts of distributed data. Users can easily scale Hadoop to store and process Petabytes of data in the web space. The package is economical (open source), efficient (high level of parallelism) and is reliable (keeps multiple data copies).

2.26 **Open Grid Services Architecture**: OGSA is driven by large-scale distributed computing apps. These apps must provide take into account high degree of resource and data sharing. The key features here are: distributed executed environment, public key infrastructure (PKI) services, trust management and security problems in grid computing.

Globus is a middleware library that implements OGSA standards for resource discovery, allocation and security enforcement.

2.27 **Performance Metrics**: In a distributed system, system throughput is measured in MIPS, Tflops (Tera Floating point Operations per Second) or Transactions per Second (TPS). Other measures also exist: job response and network latency. An interconnection network with low latency and high bandwidth is preferred. The key factors to be considered for performance are OS boot time, compile time, I/O data rate, and the runtime support system used.

2.28 Dimensions of Scalability: System scaling can increase or decrease resources depending on different practical factors.

- **Size Scalability**: This targets higher performance or more functionality by increasing the machine size (cache, processors, memory etc.). We can determine the size scalability by counting the number of processors installed. That is more processors => more ‘size’.
- **Software Scalability**: Upgrades in OS/compilers, adding mathematical libraries, installing new apps, and using more user friendly environments are the factors considered in determining software scalability.
- **Application Scalability**: This refers to matching problem size scalability (increasing data) with machine size scalability (effectively use the resources to obtain the best result possible).
- **Technology Scalability**: Here, systems that can adapt to changes in different aspects of technology like component or network are considered. Three aspects play an important role here: time, space and heterogeneity. Time is concerned with processors, motherboard, power supply packaging and cooling. All these have to be upgraded between 3 to 5 years. Space is related to packaging and energy concerns. Heterogeneity refers to the use of hardware components or software packages from different vendors; this affects scalability the most.

2.29 Scalability versus OS Image Count: In Figure 1.23 [2], scalable performance is estimated against the multiplicity of OS images in distributed systems. Note that scalable performance means we can ever increase the speed of the system by adding more servers of processors, or by enlarging memory size and so on. The OS image is counted by the no. of independent OS images observed in a cluster, grid, P2P network or the cloud.

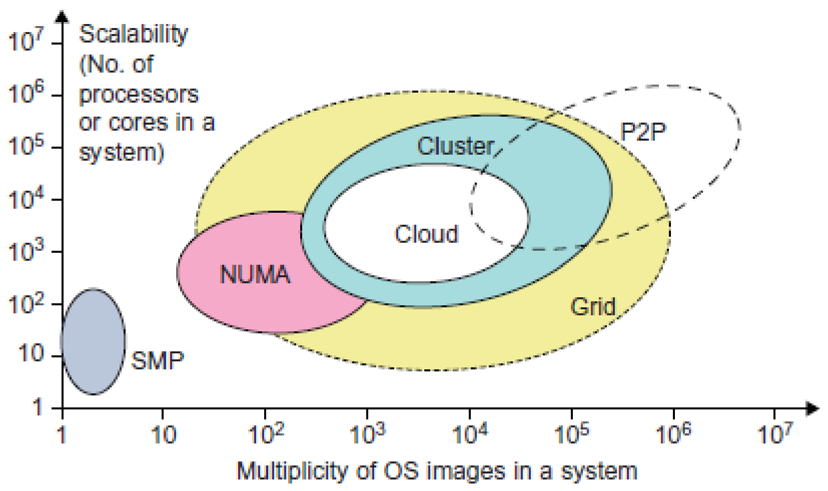


FIGURE 1.23

System scalability versus multiplicity of OS images based on 2010 technology.

An SMP (Symmetric multiprocessor) server has a single system image or a single node in a large cluster. NUMA (non-uniform memory access) machines are SMP machines with distributed and shared memory. NUMA machine can run with multiple OS and can scale a hundreds of processors. Note that clusters can be SMP servers or high-end machines with loose coupling. Obviously, clusters have more scalability than NUMA machines.

2.30 Amdahl's Law: Consider the execution of a given program on a uniprocessor workstation with a total execution time of T minutes. Say the program is running in parallel with other servers on a cluster of many processing nodes. Assume that a fraction α of the code must be executed sequentially (sequential bottleneck). Hence, $(1-\alpha)$ 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 for sequential execution time on a single processor and the second term is for parallel execution time on *n* parallel nodes.

Note that all communication overhead, the I/O time and exception handling time are ignored here. Amdahl's Law states that the speedup factor of using n-processor system over the use of a single processor is given by:

Speedup S= $T/[\alpha T + (1-\alpha) T/n] = 1/[\alpha + (1-\alpha)/n]$ ---- (1.1)

The maximum speedup of n can be obtained only if α is reduced to zero or the code can be parallelized with $\alpha = 0$.

As the cluster becomes large (that is $n \rightarrow \infty$), S approaches $1/\alpha$, which is the threshold on the speedup of S. Note that the threshold is independent of n. The sequential bottleneck is the portion of the code that cannot be parallelized. Ex: The maximum speed achieved is 4, if $\alpha=0.25$ or $1-\alpha=0.75$, even if a user uses hundreds of processors. This law deduces that we should make the sequential bottleneck as small as possible.

2.31 Problem with fixed workload: In Amdahl's law, same amount of workload was assumed for both sequential and parallel execution of the program with a fixed problem size or dataset. This was called fixed-workload speedup by other scientists. To execute this fixed-workload on *n* processors, parallel processing leads to a system efficiency E which is given by:

$E = S/n = 1/[\alpha n + 1-\alpha]$ ---- (1.2)

Generally, the system efficiency is low, especially when the cluster size is large. To execute a program on cluster with n=256 nodes, and $\alpha=0.25$, efficiency E = 1/[0.25x256 + 0.75] = 1.5%, which is very low. This is because only a few processors, say 4, are kept busy whereas the others are kept idle.

2.32 Gustafson's Law: To obtain higher efficiency when using a large cluster, scaling the problem size to match the cluster's capability should be considered. The speedup law proposed by Gustafson is also referred to as scaled-workload speedup.

Let W be the workload in a given program. When using an *n*-processor system, the user scales the workload to $W' = \alpha W + (1-\alpha)nW$. Note that only the portion of the workload that can be parallelized is scaled n times in the second term. This scaled workload W' is the sequential execution time on a single processor. The parallel execution time W' on n processors is defined by a scaled-workload speedup as:

$S' = W'/W = [\alpha W + (1-\alpha) nW]/W = \alpha + (1-\alpha) n$ ---- (1.3)

This speedup is known as Gustafson's law. By fixing the parallel execution time at level W, we can obtain the following efficiency:

$E' = S'/n = \alpha/n + (1-\alpha)$ ---- (1.4)

Taking previous workload values into consideration, efficiency can be improved for a 256-node cluster to $E' = 0.25/256 + (1-0.25) = 0.751$. For a fixed workload Amdahl's law must be used and for scaled problems users should apply Gustafson's law.

NOTE: In addition to performance, system availability and application flexibility are two other important design goals in a distributed computing system. They can be found in 2.33.

2.33 System Availability: High availability (HA) is needed in all clusters, grids, P2P networks and cloud systems. 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) ---- (1.5)

System availability depends on many factors like hardware, software and network components. Any failure that will lead to the failure of the total system is known as a *single point of failure*. It is the general goal of any manufacturer or user to bring out a system with no single point of failure. For achieving this goal, the factors that need to be considered are: adding hardware redundancy, increasing component reliability and designing testability. In the Figure 1.24 [2] below, the effects of system availability are estimated by scaling the system size in terms of no. of process cores in the system.

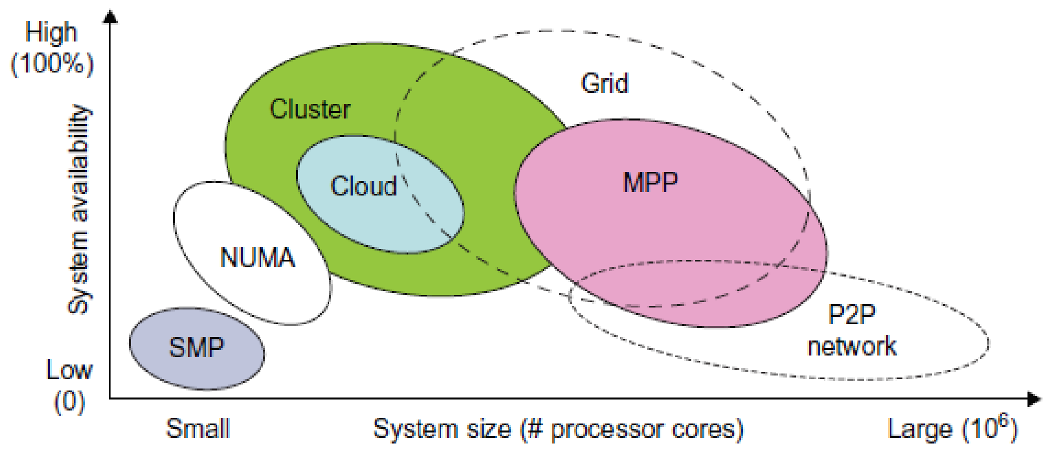


FIGURE 1.24

Estimated system availability by system size of common configurations in 2010.

2.34 As a distributed system increases in size, availability decreases due to a higher chance of failure and difficulty in isolating the features. Both SMP and MPP are likely to fail under centralized resources with one OS. NUMA machines are a bit better here since they use multiple OS.

Note here that private clouds are created out of virtualized data centers; hence a cloud has availability similar to that of a cluster. A grid is a cluster of clusters. Therefore, clusters, clouds and grids have decreasing availability as the system increases in size.

2.35 Threats to networks and systems:

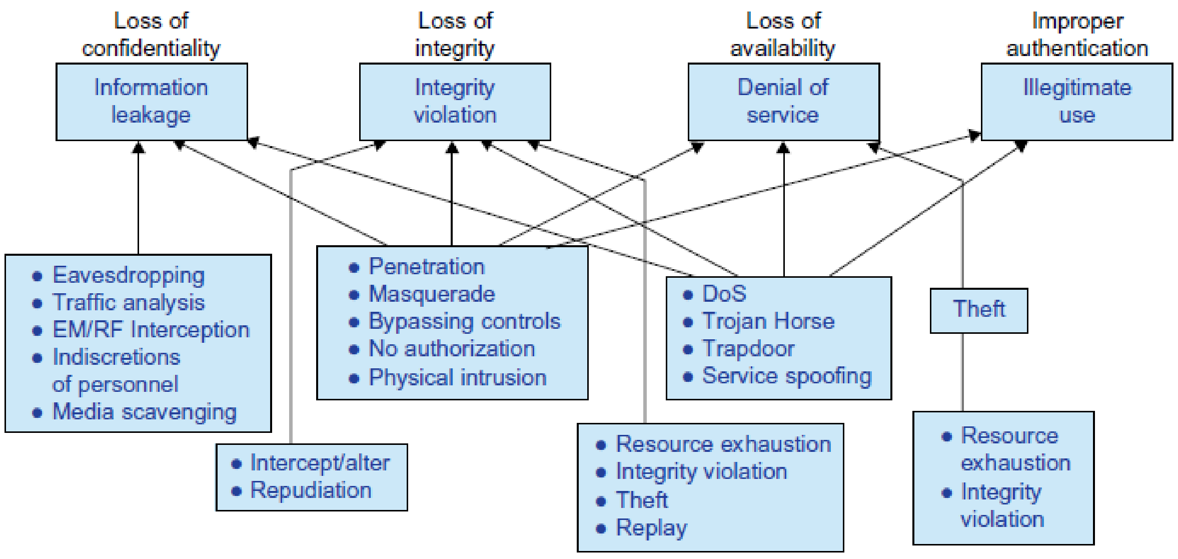


FIGURE 1.25

Various system attacks and network threats to the cyberspace, resulting 4 types of losses.

The Figure 1.25 [2] presents a summary of various attack types and the damaged caused by them to the users. Information leaks lead to a loss of confidentiality. Loss of data integrity can be caused by user alteration, Trojan horses, service spoofing attacks, and Denial of Service (DoS) – this leads of loss of Internet connections and system operations. Users need to protect clusters, grids, clouds and P2P systems from malicious intrusions that may destroy hosts, network and storage resources. Internet anomalies found generally in routers, gateways and distributed hosts may hinder (hold back) the usage and acceptance of these public resources.

2.36 **Security Responsibilities**: The main responsibilities include confidentiality, integrity and availability for most Internet service providers and cloud users. In the order of SaaS, PaaS and IaaS, the providers increase/transfer security control to the users. IN brief, the SaaS model relies on the cloud provider for all the security features. On the other hand, IaaS wants the users to take control of all security functions, but their availability is still decided by the providers. Finally, the PaaS model divides the security aspects in this way: data integrity and availability is with the provider while confidentiality and privacy control is the burden of the users.

2.37 **Copyright Protection**: Collusive (secret agreement) piracy is the main source of copyright violation within the boundary of a P2P network. Clients may illegally share their software, allotted only to them, with others thus triggering piracy. One can develop a proactive (control the situation before damage happens) content poisoning scheme to stop colluders (conspirers) and pirates, detect them and stop them to proceed in their illegal work.

2.38 **System Defence Technologies**: There exist three generations of network defence. In the first generation, tools were designed to prevent intrusions. These tools established themselves as access control policies, cryptographic systems etc. but an intruder can always slip into the system since there existed a weak link every time. The second generation detected intrusions in a timely manner to enforce remedies. Ex: Firewalls, intrusion detection systems (IDS), public key infrastructure (PKI) services (banking, e-commerce), reputation systems etc. The third generation provides more intelligent responses to intrusions.

2.39 **Data Protection Infrastructure**: Security infrastructure is required to protect web and cloud services. At the user level, one needs to perform trust negotiation and reputation aggregation over all users. At the app end, we need to establish security precautions and intrusion detection systems to restrain virus, worm, malware, and DDoS attacks. Piracy and copyright violations should also be detected and contained. These can be studied in detail later when the three types of clouds are encountered and the general services offered by the cloud are discussed.

2.40 **Energy Efficiency in Distributed Computing**: The primary goals in parallel and distributed computing systems are HP and HT and also performance reliability (fault tolerance and security). New challenges encountered in this area (distributed power management-DPM) these days include energy efficiency, workload and resource outsourcing. In the forth-coming topics, the energy consumption issues in servers and HPC systems are discussed.

Energy consumption in parallel and distributed computing raises different issues like monetary (financial), environmental and system performance issues. The megawatts of power needed for PFlops has to be within the budget control and the distributed usage of resources has to be planned accordingly. The rising of temperature due to more usage of the resources (cooling) is also to be addressed.

2.41 **Energy Consumption of Unused Servers**: To run a data center, a company has to spend huge amount of money for hardware, software, operational support and energy every year. Hence, the firm should plan accordingly to make maximum utilization of the available resources and yet the financial and cooling issues should not cross their limits. For all the finance spent on a data center, it should also not lie down idle and should be utilized or leased for useful work.

Idle servers can save a lot of money and energy; so the first step in IT departments is to identify the unused or underused servers and plan to utilize their resources in a suitable manner.

2.42 **Reducing Energy in Active Servers**: In addition to identifying unused/underused servers for energy savings, we should also apply necessary techniques to decrease energy consumption in active distributed systems. These techniques should not hinder the performance of the concerned system. Power management issues in distributed computing can be classified into four layers, as seen in Figure 1.26 [2].

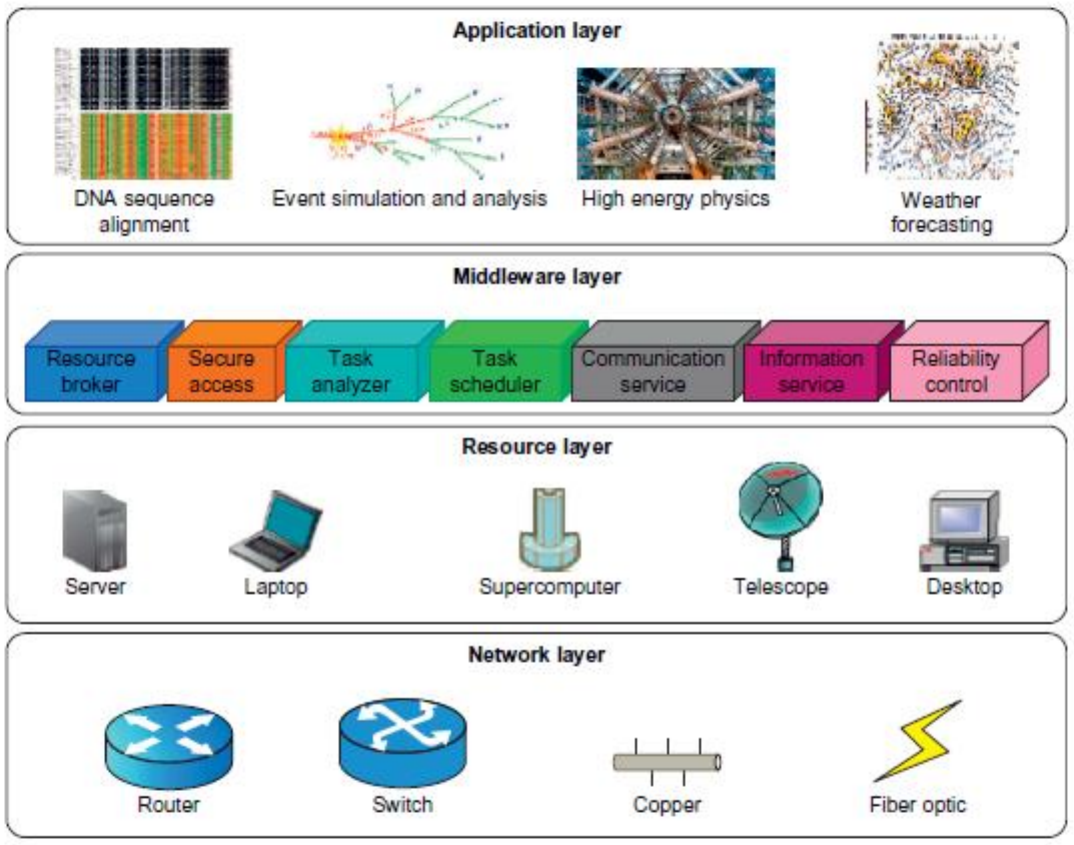


FIGURE 1.26

Four operational layers of distributed computing systems.

2.43 **Application Layer**: Most apps in different areas like science, engineering, business, financial etc. try to increase the system's speed or quality. By introducing energy-conscious applications, one should try to design the usage and consumption in a planned manner such that the apps manage to use the new multi-level and multi-domain energy management methodologies without reducing the performance. For this goal, we need to identify a relationship between the performance and energy consumption areas (correlation). Note that these two factors (compute and storage) are surely correlated and affect completion time.

2.44 **Middleware layer**: The middleware layer is a connection between application layer and resource layer. This layer provides resource broker, communication service, task analyzer & scheduler, security access, reliability control, and information service capabilities. It is also responsible for energy-efficient techniques in task scheduling. In distributed computing system, a balance has to be brought out between efficient resource usage and the available energy.

2.45 **Resource Layer**: This layer consists of different resources including the computing nodes and storage units. Since this layer interacts with hardware devices and the operating systems, it is responsible for controlling all distributed resources. Several methods exist for efficient power management of hardware and OS and majority of them are concerned with the processors.

Dynamic power management (**DPM**) and dynamic voltage frequency scaling (**DVFS**) are the two popular methods being used recently. In DPM, hardware devices can switch from idle modes to lower power modes. In DVFS, energy savings are obtained based on the fact that power consumption in CMOS [15] (Complementary Metal-Oxide Semiconductor) circuits have a direct relationship with frequency and the square of the voltage supply. [$\boldsymbol{P = 0.5\, CV^2 f}$] Execution time and power consumption can be controlled by switching among different voltages and frequencies.

2.46 Network Layer: The main responsibilities of the network layer in distributed computing are routing and transferring packets, and enabling network services to the resource layer. Energy consumption and performance are to measured, predicted and balanced in a systematic manner so as to bring out energy-efficient networks. Two challenges exist here:

- The models should represent the networks systematically and should possess a full understanding of interactions among time, space and energy.
- New and energy-efficient algorithms have to be developed to rope in the advantages to the maximum scale and defend against the attacks.

Data centers are becoming more important in distributed computing since the data is ever-increasing with the advent of social media. They are now another core infrastructure like power grid and transportation systems.

2.47 DVFS Method for Energy Efficiency: This method enables the exploitation of idle time (slack time) encountered by an inter-task relationship. The slack time associated with a task is utilized to the task in a lower voltage frequency. The relationship between energy and voltage frequency in CMOS circuits is calculated by:

$$E = C_{eff} f v^2 t$$

$$f = K \frac{(v - v_t)^2}{v} \qquad \text{---- (1.6)}$$

where v, C_{eff}, K and v_t are the voltage, circuit switching capacity, a technology dependent factor and threshold voltage; t is the execution time of the task under clock frequency f. By reducing v and f, the energy consumption of the device can also be reduced.

References

- https://en.wikipedia.org/wiki/Scalability
- Kai Hwang et al, Distributed and Cloud Computing – From Parallel Processing to the Internet of Things, Morgan Kaufmann, Elsevier, 2012.
- https://en.wikipedia.org/wiki/High-throughput_computing
- https://en.wikipedia.org/wiki/Radio-frequency_identification
- https://en.wikipedia.org/wiki/Global_Positioning_System
- https://en.wikipedia.org/wiki/Internet_of_things
- https://www.phy.ornl.gov/csep/ca/node10.html
- https://en.wikipedia.org/wiki/Internet_of_things#cite_note-31
- https://en.wikipedia.org/wiki/Multi-core_processor
- https://en.wikipedia.org/wiki/Manycore_processor
- https://en.wikipedia.org/wiki/Peer-to-peer
- https://www.techopedia.com/definition/29016/managed-cloud-hosting
- http://www.interoute.com/cloud-article/what-hybrid-cloud
- https://en.wikipedia.org/wiki/Common_Object_Request_Broker_Architecture
- https://en.wikipedia.org/wiki/CMOS

UNIT – 2

Virtual Machines and Virtualization of Clusters and Data Centers

3. The massive usage of virtual machines (VMs) opens up new opportunities for parallel, cluster grid, cloud and distributed computing. Virtualization enables the users to share expensive hardware resources by multiplexing (i.e., multiple analog/digital are combined into one signal over a shared medium [2]) VMs on the same set of hardware hosts like servers or data centers.
4. **Implementation Levels of Virtualization**: Virtualization is a concept by which several VMs are multiplexed into 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 (OS and apps) can be virtualized at various layers of functionality.

   The main idea is to separate hardware from software to obtain greater efficiency from the system. Ex: Users can gain access to more memory by this concept of VMs. With sufficient storage, any computer platform can be installed in another host computer [1], even if processors' usage and operating systems are different.

   4.1 Levels of Virtualization Implementation: A traditional computer system runs with a host OS specially adjusted for its hardware architecture. This is depicted in Figure 3.1a [1].

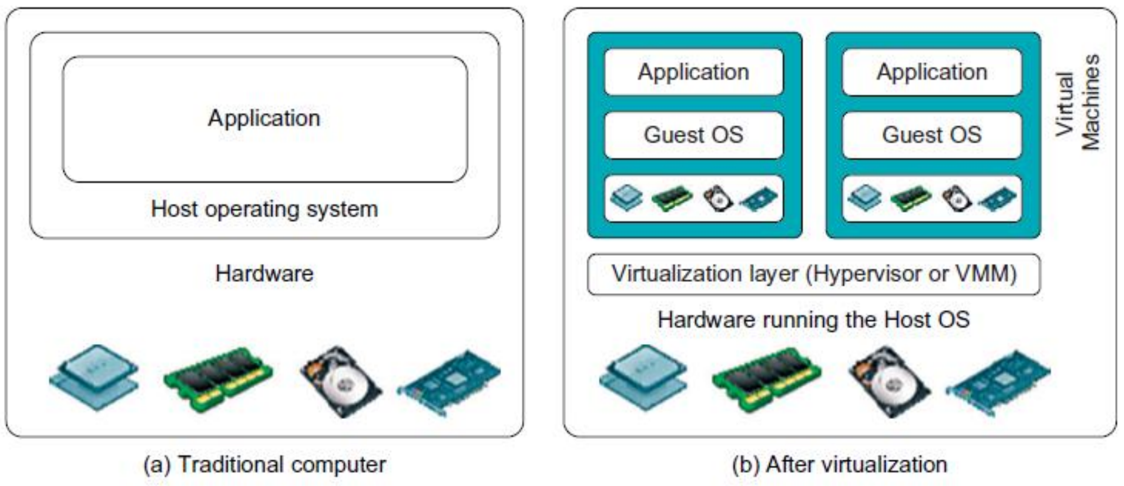


FIGURE 3.1

The architecture of a computer system before and after virtualization, where VMM stands for virtual machine monitor.

   After virtualization, different user apps managed by their own OS (i.e., guest OS) can run on the same hardware, independent of the host OS. This is often done by adding a virtualization layer as shown in Figure 3.1b [2].

   This virtualization layer is called VM Monitor or hypervisor. The VMs can be seen in the upper boxes where apps run on their own guest OS over a virtualized CPU, memory and I/O devices.

   4.2 The main function of the software layer for virtualization is to virtualize the physical hardware of a host machine into virtual resources to be saved by the VMs. The virtualization software creates the abstract of VMs by introducing a virtualization layer at various levels of a computer. General virtualization layers include the instruction set architecture (ISA) level, hardware level, OS level, library support level, and app level. This can be seen in Figure 3.2 [1]. The levels are discussed below.

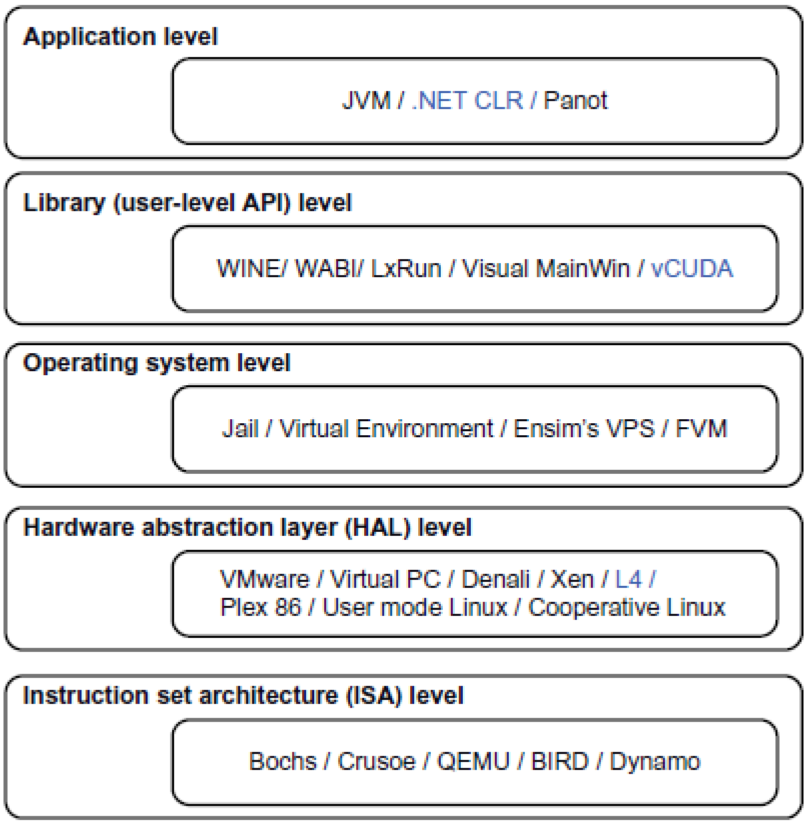

Application level
JVM / .NET CLR / Panot
Library (user-level API) level
WINE/ WABI/ LxRun / Visual MainWin / vCUDA
Operating system level
Jail / Virtual Environment / Ensim's VPS / FVM
Hardware abstraction layer (HAL) level
VMware / Virtual PC / Denali / Xen / L4 / Plex 86 / User mode Linux / Cooperative Linux
Instruction set architecture (ISA) level
Bochs / Crusoe / QEMU / BIRD / Dynamo

FIGURE 3.2

Virtualization ranging from hardware to applications in five abstraction levels.

4.2.1 **Instruction Set Architecture Level**: At the ISA level, virtualization is performed by emulation (imitate) of the given ISA by the ISA of the host machine. Ex: MIPS binary code can run on an x86-based host machine with the help of ISA simulation. Instruction emulation leads to virtual ISAs created on any hardware machine.

Basic level of emulation can be traced at code interpretation. An interpreter (line-by-line compiler) program works on the instructions one-by-one and this process is slow. To speedup, *dynamic binary translation* can be used where it translates blocks of dynamic source instructions to target instructions. The basic blocks can also be extended to program traces or super blocks to increase translation efficiency. This emulation requires binary translation and optimization. Hence, a Virtual-ISA requires a processor specific translation layer to the compiler.

4.2.2 **Hardware Abstraction Level**: Hardware level virtualization is performed on the bare hardware. This approach generates a virtual hardware environment and processes the hardware in a virtual manner. The idea is to virtualize the resources of a computer by utilizing them concurrently. Ex: IBM Xen hypervisor (VMM) runs Linux or other guest OS applications. [Discussed later]

4.2.3 **OS Level:** This refers to an abstraction layer between the OS and the user apps. The OS level virtualization creates isolated *containers* on a single physical server and OS instances to utilize software and hardware in data centers. The containers behave like real servers. OS level virtualization is used in creating virtual hosting environments to allocate hardware resources among a large number of 'distrusting' users. It can also be used to indirectly merge server hardware by moving resources on different hosts into different containers or VMs on one server.

4.2.4 **NOTE:** ***Containers*** [3] use the host operating system as their base, and not the hypervisor. Rather than virtualizing the hardware (which requires full virtualized operating system images for each guest), containers virtualize the OS itself, sharing the host OS kernel and its resources with both the host and other containers.

4.2.5 **Library Support Level**: Most applications use APIs exported by user-level libraries rather than lengthy system calls by the OS. Virtualization with library interfaces is possible by controlling the communication link between apps and the rest of the system through API hooks.
Ex: (a) **Wine** (recursive acronym for *Wine Is Not an Emulator*) is a free and open source compatibility layer software application that aims to allow applications designed for MS-Windows to run on Linux OS.

(b) vCUDA by NVIDIA. (CUDA – No acronym)

NOTE: Library [4] in computing is a collection of non-volatile (stable) resources used by computer programs to develop software. These include configuration data (organised data), documentation, help data, message templates, code subroutines classes and specifications.

4.2.6 **User-App Level**: An app level virtualization brings out a real VM; this process is also known as process level virtualization. Generally HLL VMs are used where virtualization layer is an app above the OS; it can run programs written and compiled to an abstract machine definition. Ex: JVM and .NET CLR (Common Language Runtime).

Other forms of app level virtualization are app isolation, app sandboxing or app streaming. Here, the app is wrapped in a layer and is isolated from the host OS and other apps. This makes the app more much easier to distribute and remove from user workstations. Ex: LANDesk (an app virtualization platform) – this installs apps as self-contained, executable files in an isolated environment. No actual installation is required and no system modifications are needed.

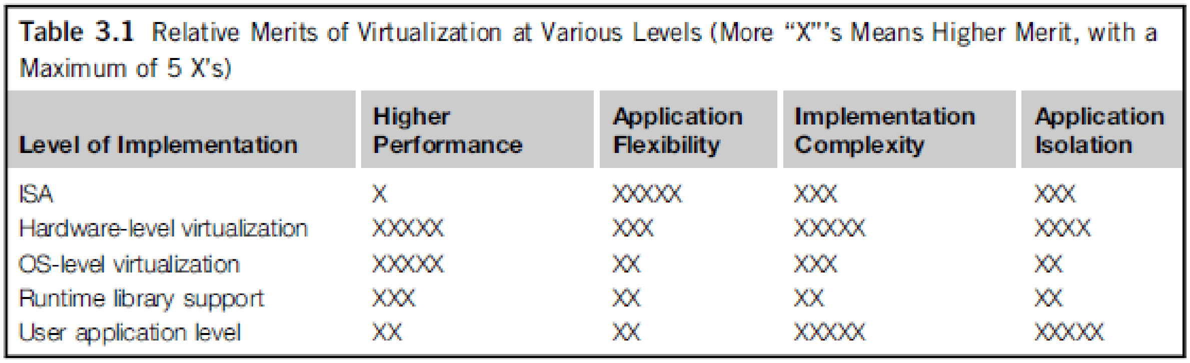


Table 3.1 Relative Merits of Virtualization at Various Levels (More "X"'s Means Higher Merit, with a Maximum of 5 X's)

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

Note from Table 3.1 [1] that hardware and OS support will yield the highest performance. At the same time, the hardware and app levels are most expensive to implement. User isolation is difficult to archive and ISA offers best flexibility.

4.3 VMM Design Requirement and Providers: As seen before, hardware-level virtualization inserts a layer between real hardware and traditional OS. This layer (VMM/hypervisor) manages the hardware resources of the computer effectively. By the usage of VMM, different traditional operating systems can be used with the same set of hardware simultaneously.

4.4 Requirements for a VMM:

(a) For programs, a VMM should provide an identical environment, same as the original machine.
(b) Programs running in this environment should show only minor decreases in speed.
(c) A VMM should be in complete control of the system resources.

Some differences might still be caused due to availability of system resources (more than one VM is running on the same system) and differences caused by timing dependencies.

The hardware resource requirements (like memory) of each VM is reduced, but the total sum of them is greater that of the real machine. This is needed because of any other VMs that are concurrently running on the same hardware.

A VMM should demonstrate efficiency in using the VMs. To guarantee the efficiency of a VMM, a statistically dominant subset of the virtual processor's instructions needs to be executed directly by the real processor with no intervention by the VMM. A comparison can be seen in Table 3.2 [1]:

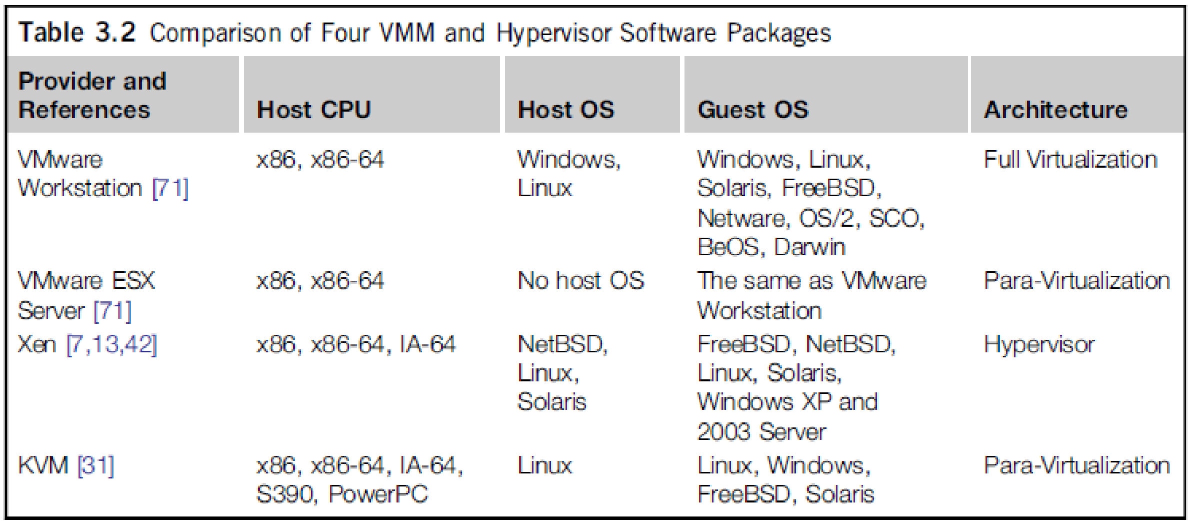


Table 3.2 Comparison of Four VMM and Hypervisor Software Packages

Provider and References	Host CPU	Host OS	Guest OS	Architecture
VMware Workstation [71]	x86, x86-64	Windows, Linux	Windows, Linux, Solaris, FreeBSD, Netware, OS/2, SCO, BeOS, Darwin	Full Virtualization
VMware ESX Server [71]	x86, x86-64	No host OS	The same as VMware Workstation	Para-Virtualization
Xen [7,13,42]	x86, x86-64, IA-64	NetBSD, Linux, Solaris	FreeBSD, NetBSD, Linux, Solaris, Windows XP and 2003 Server	Hypervisor
KVM [31]	x86, x86-64, IA-64, S390, PowerPC	Linux	Linux, Windows, FreeBSD, Solaris	Para-Virtualization

The aspects to be considered here include (1) The VMM is responsible for allocating hardware resources for programs; (2) a program can't access any resource that has not been allocated to it; (3) at a certain juncture, it is not possible for the VMM to regain control of the resources already allocated. Note that all processors might not satisfy these requirements of a VMM.

A VMM is tightly related to the architectures of the processors. It is difficult to implement a VMM on some types of processors like x86. If a processor is not designed to satisfy the requirements of a VMM, the hardware should be modified – this is known as hardware assisted virtualization.

4.5 Virtualization Support at the OS Level: CC is transforming the computing landscape by shifting the hardware and management costs of a data center to third parties, like banks. The challenges of CC are: (a) the ability to use a variable number of physical machines and VM instances depending on the needs of the problem. Ex: A work may need a single CPU at an instance but multi-CPUs at another instance (b) the slow operation of instantiating new VMs.

As of now, new VMs originate either as fresh boots or as replicates of a VM template – unaware of the current status of the application.

4.6 Why OS Level Virtualization (Disadvantages of hardware level virtualization):

(a) It is slow to initiate a hardware level VM since each VM creates its own image from the beginning.
(b) Redundancy content is high in these VMs.
(c) Slow performance and low density
(d) Hardware modifications maybe needed

To provide a solution to all these problems, OS level virtualization is needed. It inserts a virtualization layer inside the OS to partition the physical resources of a system. It enables multiple isolated VMs within a single OS kernel. This kind of VM is called a Virtual Execution Environment (VE) or Virtual Private System or simply a **container**. From the user's point of view, a VE/container has its own set of processes, file system, user accounts, network interfaces (with IP addresses), routing tables, firewalls and other personal settings.

Note that though the containers can be customized for different people, they share the same OS kernel. Therefore this methodology is also called single-OS image virtualization. All this can be observed in Figure 3.3 [1].

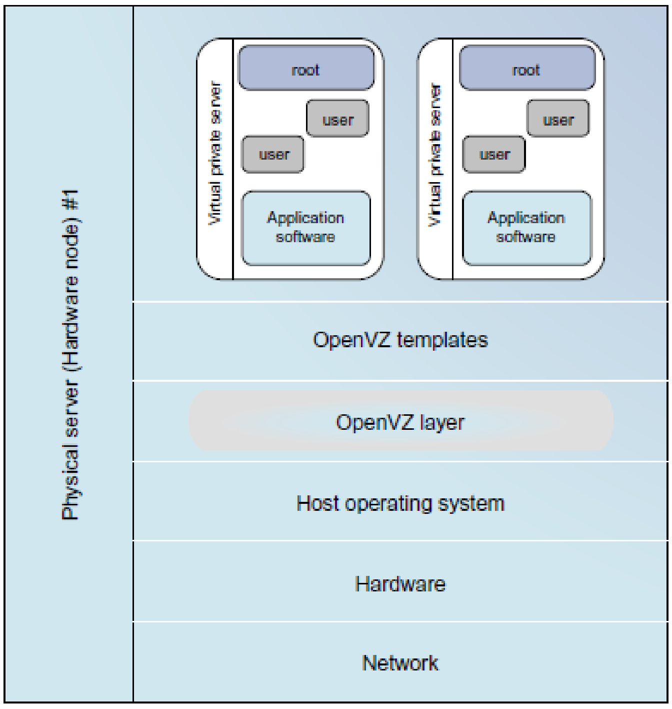


FIGURE 3.3

The OpenVZ virtualization layer inside the host OS, which provides some OS images to create VMs quickly.

4.7 Advantages of OS Extensions:

(a) VMs at the OS level have minimal start-up shutdown costs, low resource requirements and high scalability.
(b) For an OS level VM, the VM and its host environment can synchronise state changes

These can be achieved through two mechanisms of OS level virtualization:

(a) All OS level VMs on the same physical machine share a single OS kernel
(b) The virtualization layer can be designed in way that allows processes in VMs can access as many resources as possible from the host machine, but can never modify them.

4.8 Disadvantages of OS Extension: The main disadvantage of OS extensions is that all VMs at OS level on a single container must have the same kind of guest OS. Though different OS level VMs may have different OS distributions (Win XP, 7, 10), they must be related to the same OS family (Win). A Windows distribution can't run on a Linux based container.

As we can observe in Figure 3.3, the virtualization layer is inserted inside the OS to partition the hardware resources for multiple VMs to run their applications in multiple virtual environments. To implement this OS level virtualization, isolated execution environments (VMs) should be created based on a single OS kernel. In addition, the access requests from a VM must be redirected to the VM's local resource partition on the physical machine. For example, '*chroot*' command in a UNIX system can create several virtual root directories within an OS that can be used for multiple VMs.

To implement the virtual root directories' concept, there exist two ways: (a) duplicating common resources to each VM partition or (b) sharing most resources with the host environment but create private copies for the VMs on demand. It is to be noted that the first method incurs (brings up) resource costs and burden on a physical machine. Therefore, the second method is the apparent choice.

4.9 Virtualization on Linux or Windows Platforms: Generally, the OS-level virtualization systems are Linux-based. Windows based virtualization platforms are not much in use. The Linux kernel offers an abstraction layer to allow software processes to with and operate on resources without knowing the hardware details. Different Linux platforms use patched kernels to provide special support for extended functionality.

Note that most Linux platforms are not tied to a special kernel. In such a case, a host can run several VMs simultaneously on the same hardware. Examples can be seen in Table 3.3 [1].

4.10 Middleware Support for Virtualization: This is the other name for Library-level Virtualization and is also known as user-level Application Binary Interface or API emulation. This type of virtualization can create execution environments for running alien (new/unknown) programs on a platform rather than creating a VM to run the entire OS. The key functions performed here are API call interception and remapping (assign a function to a key).

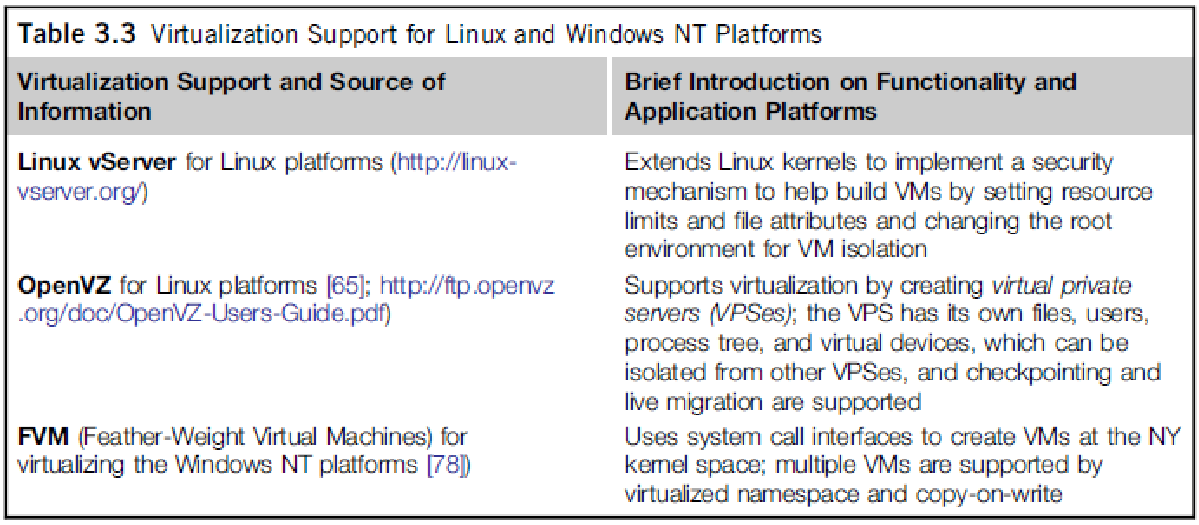


Table 3.3 Virtualization Support for Linux and Windows NT Platforms

Virtualization Support and Source of Information	Brief Introduction on Functionality and Application Platforms
Linux vServer for Linux platforms (http://linux-vserver.org/)	Extends Linux kernels to implement a security mechanism to help build VMs by setting resource limits and file attributes and changing the root environment for VM isolation
OpenVZ for Linux platforms [65]; http://ftp.openvz.org/doc/OpenVZ-Users-Guide.pdf)	Supports virtualization by creating *virtual private servers (VPSes)*; the VPS has its own files, users, process tree, and virtual devices, which can be isolated from other VPSes, and checkpointing and live migration are supported
FVM (Feather-Weight Virtual Machines) for virtualizing the Windows NT platforms [78])	Uses system call interfaces to create VMs at the NY kernel space; multiple VMs are supported by virtualized namespace and copy-on-write

5. **Virtualization Structures/Tools and Mechanisms**: It should be noted that there are three classes of VM architecture [Page 1]. Before virtualization, the OS manages the hardware. After virtualization, a virtualization layer is inserted between the hardware and the OS. Here, the virtualization layer is responsible for converting parts of real hardware into virtual hardware. Different operating systems like Windows and Linux can run simultaneously on the same machine in this manner. Depending on the position of the virtualization layer, several classes of VM architectures can be framed out: Hypervisor Architecture, para-virtualization and host-based virtualization.

5.1 Hypervisor and Xen Architecture: The hypervisor (VMM) supports hardware level virtualization on bare metal devices like CPU, memory, disk and network interfaces. The hypervisor software exists between the hardware and its OS (platform). The hypervisor provides **hypercalls** for the guest operating systems and applications. Depending on the functionality, a hypervisor can assume **micro-kernel** architecture like MS Hyper-V or **monolithic** hypervisor architecture like the VMware ESX for server virtualization.

Hypercall: A hypercall is a software trap from a domain to the hypervisor, just as a syscall is a software trap from an application to the kernel. Domains will use hypercalls to request privileged operations like updating page tables.

Software Trap: A trap, also known as an exception or a fault, is typically a type of synchronous interrupt caused by an exceptional condition (e.g., breakpoint, division by zero, invalid memory access). A trap usually results in a switch to kernel mode, wherein the OS performs some action before returning control to the originating process. A trap in a system process is more serious than a trap in a user process and might be fatal. The term trap might also refer to an interrupt intended to initiate a context switch to a monitor program or debugger.

Domain: It is a group of computers/devices on a network that are administered as a unit with common rules and procedures. Ex: Within the Internet, all devices sharing a common part of the IP address are said to be in the same domain.

Page Table: A page table is the data structure used by a virtual memory system in an OS to store the mapping between virtual addresses and physical addresses.
Kernel: A kernel is the central part of an OS and manages the tasks of the computer and hardware like memory and CPU time.

Monolithic Kernel: These are commonly used by the OS. When a device is needed, it is added as a part of the kernel and the kernel increases in size. This has disadvantages like faulty programs damaging the kernel and so on. Ex: Memory, processor, device drivers etc.

Micro-kernel: In micro-kernels, only the basic functions are dealt with – nothing else. Ex: Memory management and processor scheduling. It should also be noted that OS can't run only on a micro-kernel, which slows down the OS.
[SIM – Micro SIM]

5.2 The size of the hypervisor code of a micro-kernel hypervisor is smaller than that of monolithic hypervisor. Essentially, a hypervisor must be able to convert physical devices into virtual resources dedicated for the VM usage.

5.3 **Xen Architecture**: It is an open source hypervisor program developed by Cambridge University. Xen is a micro-kernel hypervisor, whose policy is implemented by Domain 0.

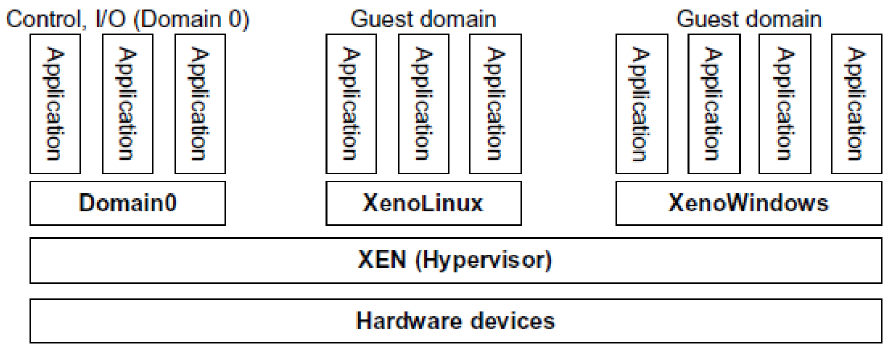


FIGURE 3.5
The Xen architecture's special domain 0 for control and I/O, and several guest domains for user applications.

As can be seen in Figure 3.5 [1], Xen doesn't include any device drivers; it provides a mechanism by which a guest-OS can have direct access to the physical devices. The size of Xen is kept small, and provides a virtual environment between the hardware and the OS. Commercial Xen hypervisors are provided by Citrix, Huawei and Oracle.

The core components of Xen are the hypervisor, kernel and applications. Many guest operating systems can run on the top of the hypervisor; but it should be noted that one of these guest OS controls the others. This guest OS with the control ability is called Domain 0 – the others are called Domain U. Domain 0 is first loaded when the system boots and can access the hardware directly and manage devices by allocating the hardware resources for the guest domains (Domain U).

Say Xen is based on Linux and its security level is some C2. Its management VM is named as Domain 0, which can access and manage all other VMs on the same host. If a user has access to Domain 0 (VMM), he can create, copy, save, modify or share files and resources of all the VMs. This is a huge advantage for the user but concentrating all the resources in Domain 0 can also become a privilege for a hacker. If Domain 0 is hacked, through it, a hacker can control all the VMs and through them, the total host system or systems. Security problems are to be dealt with in a careful manner before handing over Xen to the user.

A machine's lifetime can be thought of as a straight line that progresses monotonically (never decreases or increases) as the s/w executes. During this time, executions are made, configurations are changed, and s/w patches can be applied. VM is similar to tree in this environment; execution can go into N different branches where multiple instances of VM can be done in this tree at any time. VMs can also be allowed to rollback to a particular state and rerun from the same point.

5.4 Binary Translation with Full Virtualization: Hardware virtualization can be categorised into two categories: full virtualization and host-based virtualization.

Full Virtualization doesn't need to modify the host OS; it relies upon binary translation to trap and to virtualize certain sensitive instructions. Normal instructions can run directly on the host OS. This is done to increase the performance overhead – normal instructions are carried out in the normal manner, but the difficult and precise executions are first discovered using a trap and executed in a virtual manner. This is done to improve the security of the system and also to increase the performance.

Binary Translation of Guest OS Requests Using a VMM:

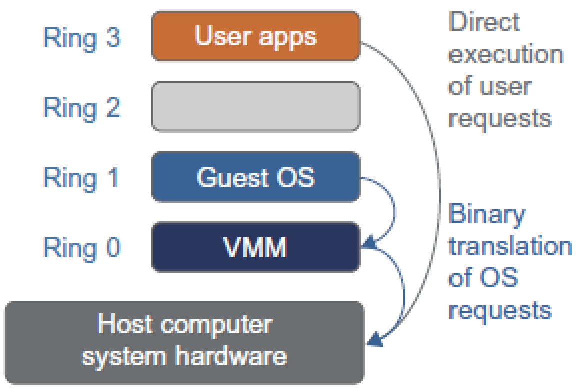

Ring 3
User apps
Ring 2
Ring 1
Guest OS
Ring 0
VMM
Host computer system hardware
Direct execution of user requests
Binary translation of OS requests

FIGURE 3.6

Indirect execution of complex instructions via binary translation of guest OS requests using the VMM plus direct execution of simple instructions on the same host.

This approach is mainly used by VMware and others. As it can be seen in Figure 3.6 [1], the VMware puts the VMM at Ring 0 and the guest OS at Ring 1. The VMM scans the instructions to identify complex and privileged instructions and trap them into the VMM, which emulates the behaviour of these instructions. Binary translation is the method used for emulation (A => 97 => 01100001) [5]. Note that full virtualization combines both binary translation and direct execution. The guest OS is totally decoupled from the hardware and run virtually (like an emulator).

Full virtualization is ideal since it involves binary translation and is time consuming. Binary translation also is cost consuming but it increases the system performance. (Same as 90% of the host).

In a **host-based** virtualization system both host and guest OS are used and a virtualization layer is built between them. The host OS is still responsible for managing the hardware resources. Dedicated apps might run on the VMs and some others can run on the host OS directly. By using this methodology, the user can install the VM architecture without modifying the host OS. The virtualization software can rely upon the host OS to provide device drivers and other low level services. Hence the installation and maintenance of the VM becomes easier.

Another advantage is that many host machine configurations can be perfectly utilized; still four layers of mapping exist in between the guest and host operating systems. This may hinder the speed and performance, in particular when the ISA (Instruction Set Architecture) of a guest OS is different from that of the hardware – binary translation MUST be deployed. This increases in time and cost and slows the system.

5.5 **Para-Virtualization with Compiler Support**: Para-Virtualization modifies the guest operating systems; a para-virtualized VM provides special APIs which take up user apps needing those changes. Para-virtualization tries to reduce the virtualization burden/extra-work to improve the performance – this is done by modifying only the guest OS kernel. This can be seen in Figure 3.7 [1].

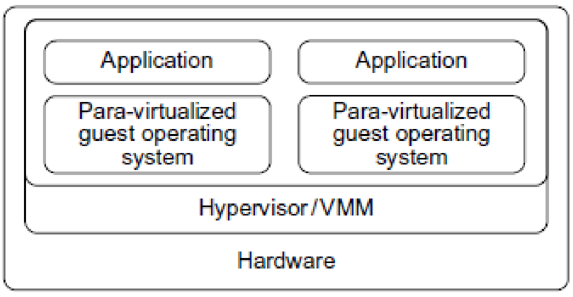

Application
Application
Para-virtualized guest operating system
Para-virtualized guest operating system
Hypervisor/VMM
Hardware

FIGURE 3.7

Para-virtualized VM architecture, which involves modifying the guest OS kernel to replace nonvirtualizable instructions with hypercalls for the hypervisor or the VMM to carry out the virtualization

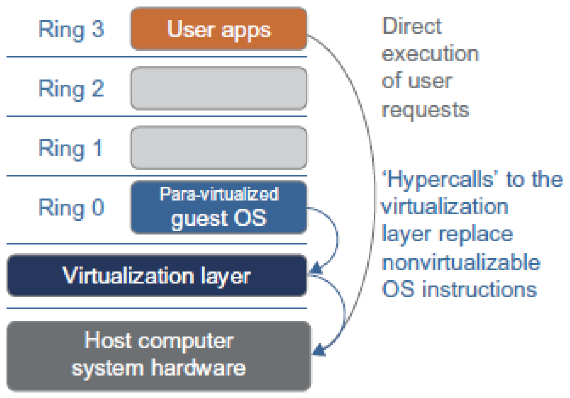

Ring 3
User apps
Ring 2
Ring 1
Ring 0
Para-virtualized guest OS
Virtualization layer
Host computer system hardware
Direct execution of user requests
'Hypercalls' to the virtualization layer replace nonvirtualizable OS instructions

FIGURE 3.8

The use of a para-virtualized guest OS assisted by an intelligent compiler to replace nonvirtualizable OS instructions by hypercalls.

Ex: In a typical para-virtualization architecture, which considers an x86 processor, a virtualization layer is inserted between h/w and OS. According to the x86 'ring definition' the virtualization layer should also be installed at Ring 0. In Figure 3.8 [1], we can notice that para-virtualization replaces instructions that cannot be virtualized with hypercalls (placing a trap) that communicate directly with the VMM. Notice that if a guest OS kernel is modified for virtualization, it can't run the hardware directly – that should be done through the virtualization layer.

5.6 **Disadvantages of Para-Virtualization**: Although para-virtualization reduces the overhead, it has other problems. Its compatibility (suitability) and portability can be in doubt because it has to support both the modified guest OS and the host OS as per requirements. Also, the maintenance cost of para-virtualization is high since it may require deep kernel modifications. Finally, the performance advantage of para-virtualization is not stable – it varies as per the workload. But compared with full virtualization, para-virtualization is more easy and practical since binary translation is not much considered. Many products utiliza para-virtualization to overcome the less speed of binary translation. Ex: Xen, KVM, VMware ESX.

5.7 **Note**: Kernel based VM (**KVM**): This is a Linux para-virtualization system – it is a part of the Linux kernel. Memory management and scheduling activities are carried out by the existing Linux kernel. Other activities are taken care of by the KVM and this methodology makes it easier to handle than the hypervisor. Also note that KVM is hardware assisted para-virtualization tool, which improves performance and supports unmodified guest operating systems like Windows, Linux, Solaris and others.

5.8 **Virtualization of CPU, Memory and I/O Devices**: Processors employ a special running mode and instructions, known as hardware-assisted virtualization. Through this, the VMM and guest OS run in different modes; all sensitive instructions of the guest OS and its apps are caught by the 'trap' in the VMM.

5.8.1 **H/W Support for Virtualization**: Modern operating systems and processors permit multiple processes to run simultaneously. A protection mechanism should exist in the processor so that all instructions from different processes will not access the hardware directly – this will lead to a system crash.

All processors should have at least two modes – user and supervisor modes to control the access to the hardware directly. Instructions running in the supervisor mode are called privileged instructions and the others are unprivileged.

Ex: VMware Workstation

5.8.2 **CPU Virtualization**: A VM is a duplicate of an existing system; majority of instructions are executed by the host processor. Unprivileged instructions run on the host machine directly; other instructions are to be handled carefully. These critical instructions are of three types: privileged, control-sensitive and behaviour-sensitive.

Privileged=> Executed in a special mode and are trapped if not done so.

Control-Sensitive=> Attempt to change the configuration of the used resources

Behaviour-Sensitive=> They have different behaviours in different situations (high load or storage or capacity)

A CPU is VZ only if it supports the VM in the CPU's user mode while the VMM runs in a supervisor's mode. When the privileged instructions are executed, they are trapped in the VMM. In this case, the VMM acts as a mediator between the hardware resources and different VMs so that correctness and stability of the system are not disturbed. It should be noted that not all CPU architectures support VZ.

Process:

- System call triggers the 80h interrupt and passes control to the OS kernel.
- Kernel invokes the interrupt handler to process the system call
- In Xen, the 80h interrupt in the guest OS concurrently causes the 82h interrupt in the hypervisor; control is passed on to the hypervisor as well.
- After the task is completed, the control is transferred back to the guest OS kernel.

5.8.3 **Hardware Assisted CPU VZ**: Since full VZ or para-VZ is complicated, this new methodology tries to simplify the situation. Intel and AMD add an additional mode called privilege mode level to the x86 processors. The OS can still run at Ring 0 and hypervisor at Ring 1. Note that all privileged instructions are trapped at the hypervisor. Hence, no modifications are required in the VMs at OS level.

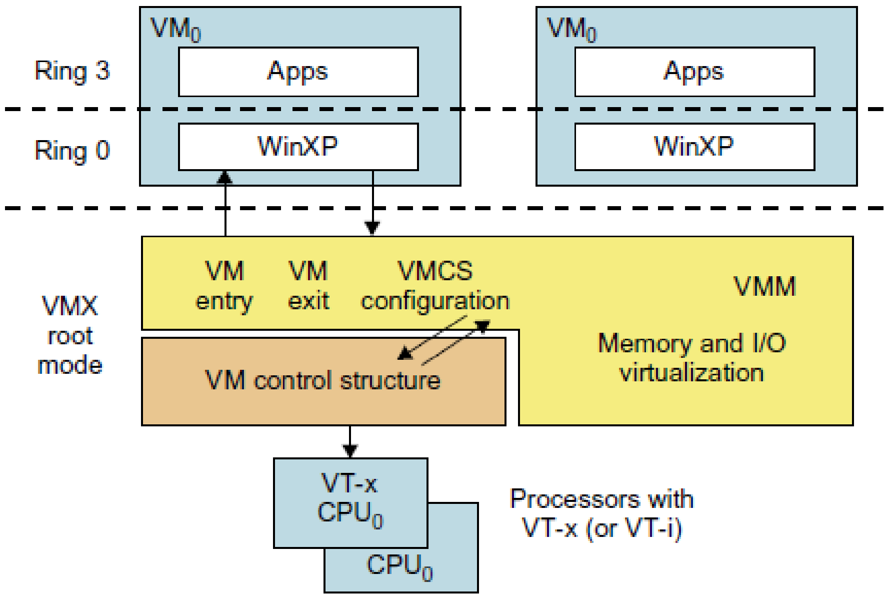


FIGURE 3.11

Intel hardware-assisted CPU virtualization.

VMCS=> VM Control System

VMX=> A virtual router

5.8.4 **Memory Virtualization**: In the **traditional** methodology, the OS maintains mappings between virtual memory to machine memory (MM) using page tables, which is a one-stage mapping from virtual memory to MM.

Virtual memory is a feature of an operating system (OS) that allows a computer to compensate for shortages of physical memory by temporarily transferring pages of data from random access memory (**RAM**) to disk storage.

Machine Memory [6] is the upper bound (threshold) of the physical memory that a host can allocate to the VM. All modern x86 processors contain memory management unit (MMU) and a translation look-aside buffer (TLB) to optimize (use in the best way) the virtual memory performance.

In a **virtual execution environment**, virtual memory VZ involves sharing the physical system memory in RAM and dynamically allocating it to the physical memory of the VMs.

Stages:

- Virtual memory to physical memory
- Physical memory to machine memory.

Other Points: MMU should be supported, guest OS controls to monitor mapping of virtual addresses to physical memory address of the VMs. All this is depicted in Figure 3.12 [1].

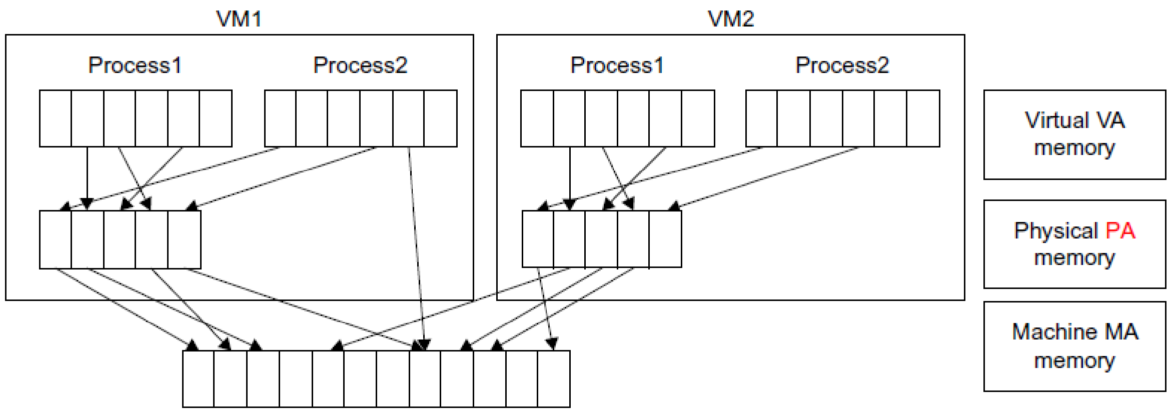


FIGURE 3.12

Two-level memory mapping procedure.

VA-Virtual Address; PA-Physical Address; MA-Machine Address

Each page table of a guest OS has a page table allocated for it in the VMM. The page table in the VMM which handles all these is called a **shadow page table**. As it can be seen all this process is nested and inter-connected at different levels through the concerned address. If any change occurs in the virtual memory page table or TLB, the shadow page table in the VMM is updated accordingly.

5.8.5 **I/O Virtualization**: This involves managing of the routing of I/O requests between virtual devices and shared physical hardware. The there are three ways to implement this are full device emulation, para-VZ and direct I/O.

- **Full Device Emulation**: This process emulates well-known and real-world devices. All the functions of a device or bus infrastructure such as device enumeration, identification, interrupts etc. are replicated in the software, which itself is located in the VMM and acts as a virtual device. The I/O requests are trapped in the VMM accordingly. The emulation approach can be seen in Figure 3.14 [1].

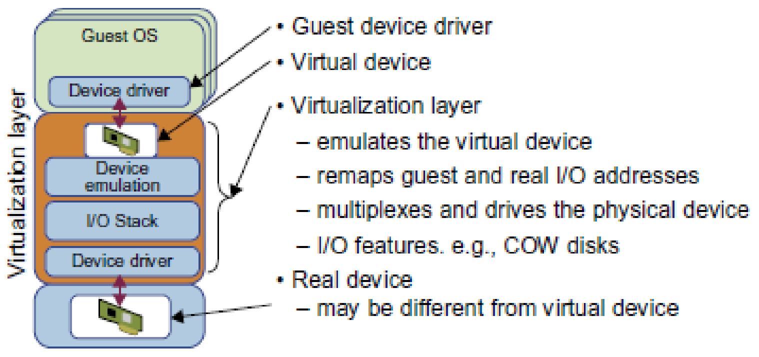

Guest OS
Device driver
Device emulation
I/O Stack
Device driver
Virtualization layer
• Guest device driver
• Virtual device
• Virtualization layer
– emulates the virtual device
– remaps guest and real I/O addresses
– multiplexes and drives the physical device
– I/O features. e.g., COW disks
• Real device
– may be different from virtual device

FIGURE 3.14

Device emulation for I/O virtualization implemented inside the middle layer that maps real I/O devices into the virtual devices for the guest device driver to use.

- **Para-VZ**: This method of I/O VZ is taken up since software emulation runs slower than the hardware it emulates. In para-VZ, the frontend driver runs in Domain-U; it manages the requests of the guest OS. The backend driver runs in Domain-0 and is responsible for managing the real I/O devices. This methodology (para) gives more performance but has a higher CPU overhead.

- **Direct I/O VZ**: This lets the VM access devices directly; achieves high performance with lower costs. Currently, it is used only for the mainframes.

Ex: **VMware Workstation for I/O VZ:** ***NIC=> Network Interface Controller***

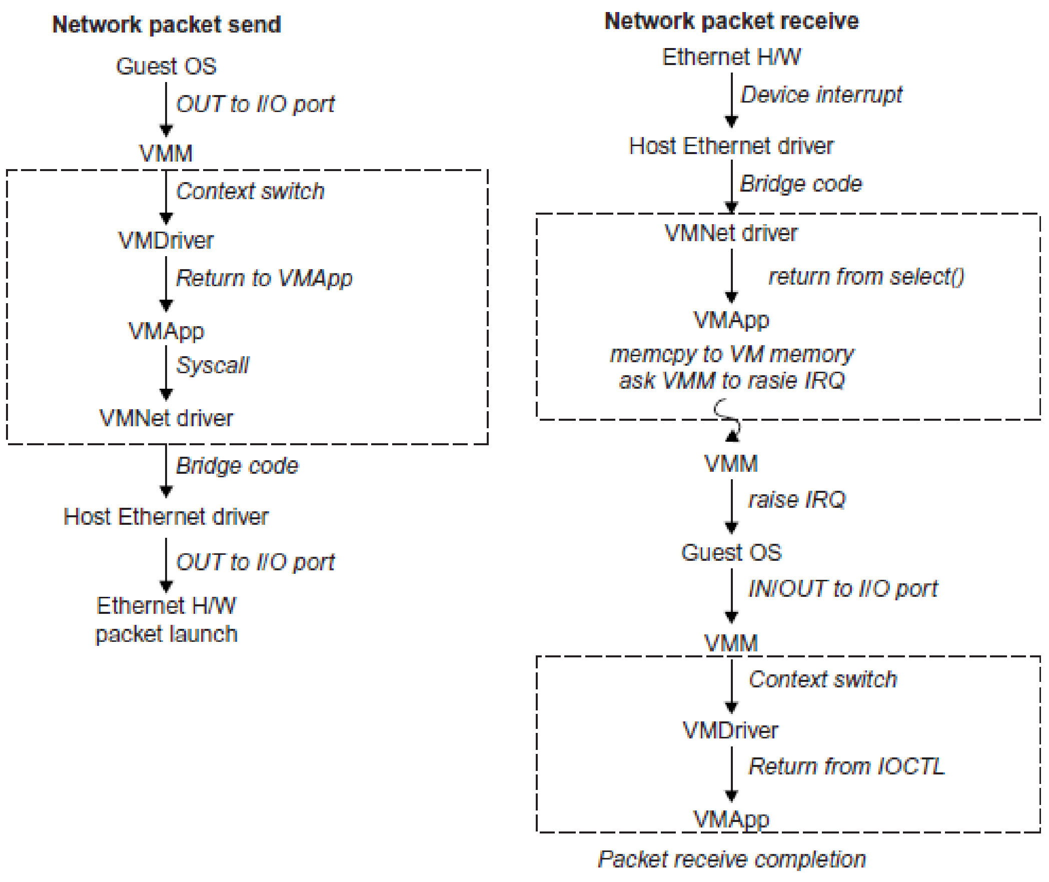


FIGURE 3.15

Functional blocks involved in sending and receiving network packets.

5.9 **Virtualization in Multi-Core Processors**: Virtualizing a multi-core processor is more complicated than that of a uni-core processor. Multi-core processors have high performance by integrating multiple cores in a chip, but their virtualization poses a new challenge. The main difficulties are that apps must be utilized in a parallelized way to use all the cores and this task must be accomplished by software, which is a much higher problem.

To reach these goals, new programming models, algorithms, languages and libraries are needed to increase the parallelism.

5.10 **Physical versus Virtual Processor Cores**: A multi-core virtualization method was proposed to allow hardware designers to obtain an abstraction of the lowest level details of all the cores. This technique alleviates (lessens) the burden of managing the hardware resources by software. It is located under the ISA (Instruction Set Architecture) and is unmodified by the OS or hypervisor. This can be seen in Figure 3.16 [1].

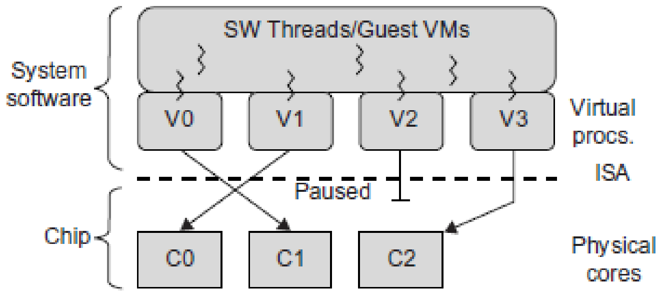
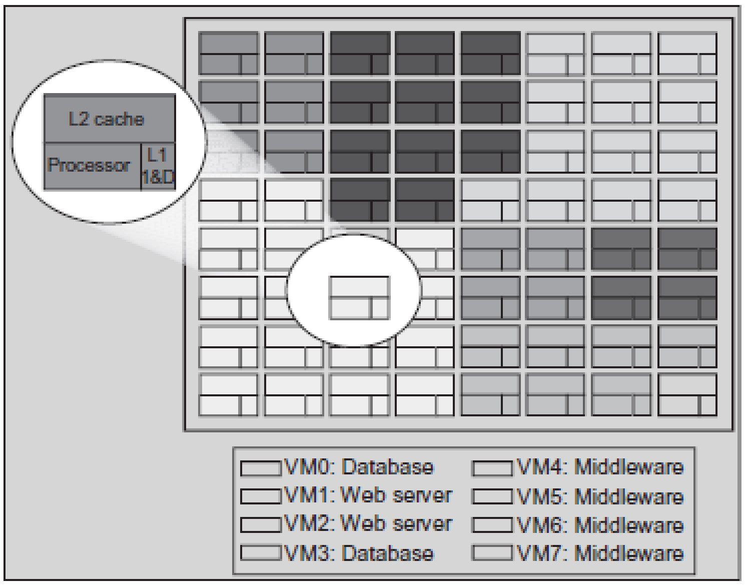


FIGURE 3.16

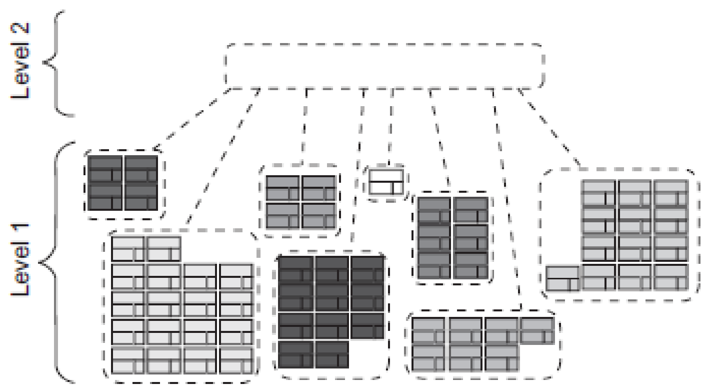
Multicore virtualization method that exposes four VCPUs to the software, when only three cores are actually present.

5.11 **Virtual Hierarchy**: The emerging concept of many-core chip multiprocessors (CMPs) is a new computing landscape (background). Instead of supporting time-sharing jobs on one or few cores, abundant cores can be used in a space-sharing – here single or multi-threaded jobs are simultaneously assigned to the cores. Thus, the cores are separated from each other and no interferences take place. Jobs go on in parallel, for long time intervals. To optimize (use effectively) the workloads, a virtual hierarchy has been proposed to overlay (place on top) a coherence (consistency) and caching hierarchy onto a physical processor. A virtual hierarchy can adapt by itself to fit how to carry out the works and share the workspace depending upon the workload and the availability of the cores.

The CMPs use a physical hierarchy of two or more cache levels that statically determine the cache (memory) allocation and mapping. A **virtual hierarchy is** a cache hierarchy that can adapt to fit the workloads. First level in the hierarchy locates data blocks close to the cores to increase the access speed; it then establishes a shared-cache domain, establishes a point of coherence, thus increasing communication speed between the levels. This idea can be seen in Figure 3.17(a) [1].



(a) Mapping of VMs into adjacent cores



(b) Multiple virtual clusters assigned to various workloads

FIGURE 3.17

CMP server consolidation by space-sharing of VMs into many cores forming multiple virtual clusters to execute various workloads.

Space sharing is applied to assign three workloads to three clusters of virtual cores: VM0 and VM3 for DB workload, VM1 and VM2 for web server workload, and VM4-VM7 for middleware workload. Basic assumption here is that a workload runs in its own VM. But in a single OS, space sharing applies equally. To encounter this problem, Marty and Hill suggested a two-level virtual coherence and caching hierarchy. This can be seen in Figure 3.17(b) [1]. Each VM operates in its own **virtual cluster** in the first level which minimises both access time and performance interference. The second level maintains a globally shared memory.

A virtual hierarchy adapts to space-shared workloads like multiprogramming and server consolidation.

UNIT – 3

Cloud Platform Architecture

6. **Cloud Computing and Service Models**: In recent days, the IT industry has moved from manufacturing to offering more services (service-oriented). As of now, 80% of the industry is 'service-industry'. It should be realized that services are not manufactured/invented from time-to-time; they are only rented and improved as per the requirements.

   Clouds aim to utilize the resources of data centers virtually over automated hardware, databases, user interfaces and apps [1].

7. **Public, Private and Hybrid Clouds**: Cloud computing has evolved from the concepts of clusters, grids and distributed computing. Different resources (hardware, finance, time) are leveraged (use to maximum advantage) to bring out the maximum HTC. A CC model enables the users to share resources from anywhere at any time through their connected devices.

   Advantages of CC: Recall that in CC, the programming is sent to data rather than the reverse, to avoid large data movement, and maximize the bandwidth utilization. CC also reduces the costs incurred by the data centers, and increases the app flexibility.

   CC consists of a virtual platform with elastic resources [2] and puts together the hardware, data and software as per demand. Furthermore, the apps utilized and offered are heterogeneous.

7.1 **The Basic Architecture** of the types of clouds can be seen in Figure 4.1 [1] below.

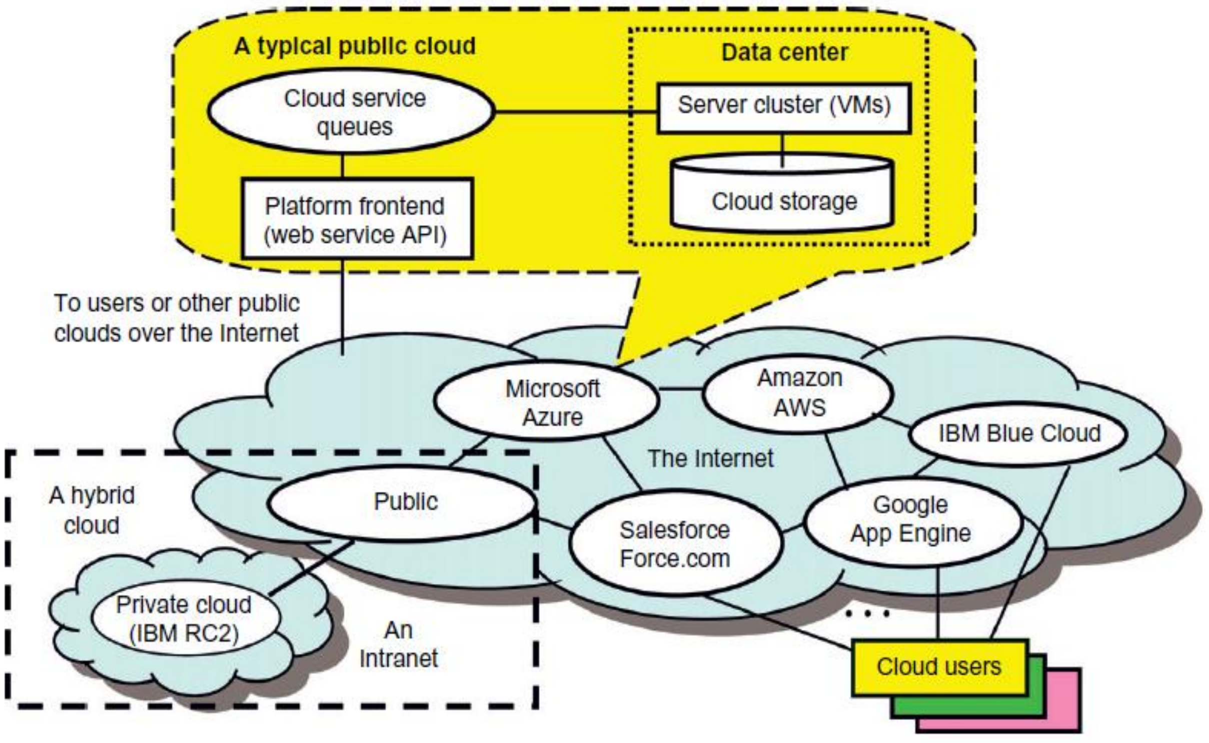


FIGURE 4.1

Public, private, and hybrid clouds illustrated by functional architecture and connectivity of representative clouds available by 2011.

7.2 **Public Clouds**: A public cloud is owned by a service provider, built over the Internet and offered to a user on payment. Ex: Google App Engine (GAE), AWS, MS-Azure, IBM Blie Cloud and Salesforce-

Force.com. All these offer their services for creating and managing VM instances to the users within their own infrastructure.

Private Clouds: A private cloud is built within the domain of an **intranet** owned by a single organization. It is client-owned and managed; its access is granted to a limited number of clients only. Private clouds offer a flexible and agile private infrastructure to run workloads within their own domains. Though private cloud offers more control, it has limited resources only.

Hybrid Clouds: A hybrid cloud is built with both public and private clouds. Private clouds can also support a hybrid cloud model by enhancing the local infrastructure with computing capacity of a public external cloud.

Summary: **Public** clouds provide standardization, preserve the investment and offer flexibility. **Private** clouds attempt to achieve customization (modify to suit the current situation), higher efficiency, resilience (capacity to recover quickly from difficulties), security and privacy. **Hybrid** clouds work in the middle with many compromises in resource sharing.

7.3 Data Center Networking Architecture: The core of a cloud is the server cluster and the cluster nodes are used as compute nodes. The scheduling of user jobs requires that virtual clusters are to be created for the users and should be granted control over the required resources. **Gateway nodes** are used to provide the access points of the concerned service from the outside world. They can also be used for security control of the entire cloud platform. It is to be noted that in physical clusters/grids, the workload is static; in clouds, the workload is dynamic and the cloud should be able to handle any level of workload on demand.

Differences between Data Centers and Super Computers: In data centers, scalability is a fundamental requirement. Note that data centers have multiple servers. Ex: MS-Chicago Data Center has 100,000 eight-core servers housed in 50 containers (2000 in each). In supercomputers, a separate **data farm** [3] is used; a data center uses disks on server nodes plus memory cache and DBs.

Data Centers and Supercomputers also possess different networking requirements. (bandwidth, routers used etc.)

NOTE: **Data Farm** => Data farming is the process of using computational experiments to ‘grow’ or increase data which can be utilized for statistical analyzing.

7.4 Cloud Development Trends: There is a good chance that private clouds will grow in the future since private clouds are more secure, and adjustable within an organization. Once they are matured and more scalable, they might be converted into public clouds. In another angle, hybrid clouds might also grow in the future.

7.5 Cloud Ecosystem and Enabling Technologies: The differences between classical computing and cloud computing can be seen in the table [1] below. In traditional computing, a user has to buy the hardware, acquire the software, install the system, test the configuration and execute the app code. The management of the available resources is also a part of this. Finally, all this process has to be revised for every 1.5 or 2 years since the used methodologies will become obsolete.

Classical Computing	**Cloud Computing**
(Repeat the following cycle every 18 months)	(Pay as you go per each service provided)
Buy and own	**Subscribe**
Hardware, system software, applications to meet peak needs	- - - -
Install, configure, test, verify, evaluate, manage	**Use** (Save about 80-95% of the total cost)
- - - -	- - - -
Use	(Finally)
- - - -	**$ - Pay for what you use**
Pay $$$$$ (High cost)	based on the QoS

On the other hand, CC follows a pay-as-you-go model [1]. Hence the cost is reduced significantly – a user doesn't buy any resources but rents them as per his requirements. All S/W and H/W resources are leased by the user from the cloud resource providers. This is advantageous for small and middle business firms which require limited amount of resources only. Finally, CC also saves power.

7.6 Cloud Design Objectives:

- Shifting computing from desktops to data centers
- Service provisioning and cloud economics
- Scalability in performance (as the no. of users increases)
- Data Privacy Protection
- High quality of cloud services (QoS must be standardized to achieve this)
- New standards and interfaces

7.7 Cost Model:

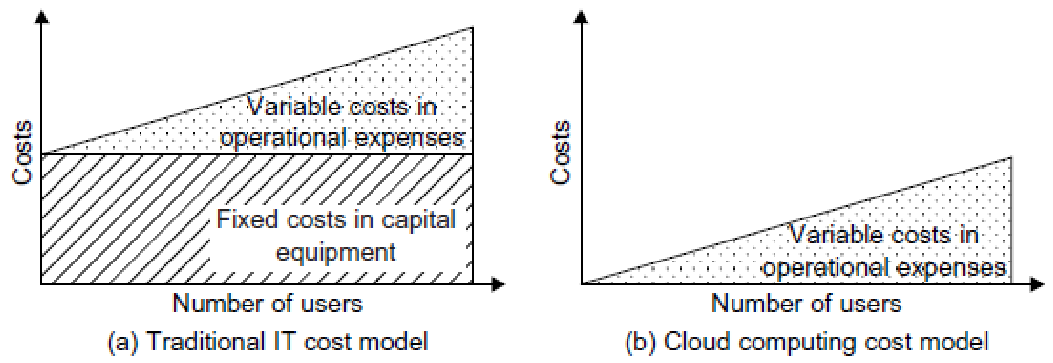


FIGURE 4.3

Computing economics between traditional IT users and cloud users.

The above Figure 4.3a [1] shows the additional costs on top of fixed capital investments in traditional computing. In CC, only pay-as-per-use is applied, and user-jobs are outsourced to data centers. To use a cloud, one has no need to buy hardware resources; he can utilize them as per the demands of the work and release the same after the job is completed.

7.8 Cloud Ecosystems

Cloud Ecosystems: With the emergence of Internet clouds, an 'ecosystem' (a complex inter-connected systems network) has evolved. This consists of users, providers and technologies. All this is based mainly on the open source CC tools that let organizations build their own IaaS. Private and hybrid clouds are also used. Ex: Amazon EC2.

An **ecosystem for private clouds** was suggested by scientists as depicted in Figure 4.4 [1].

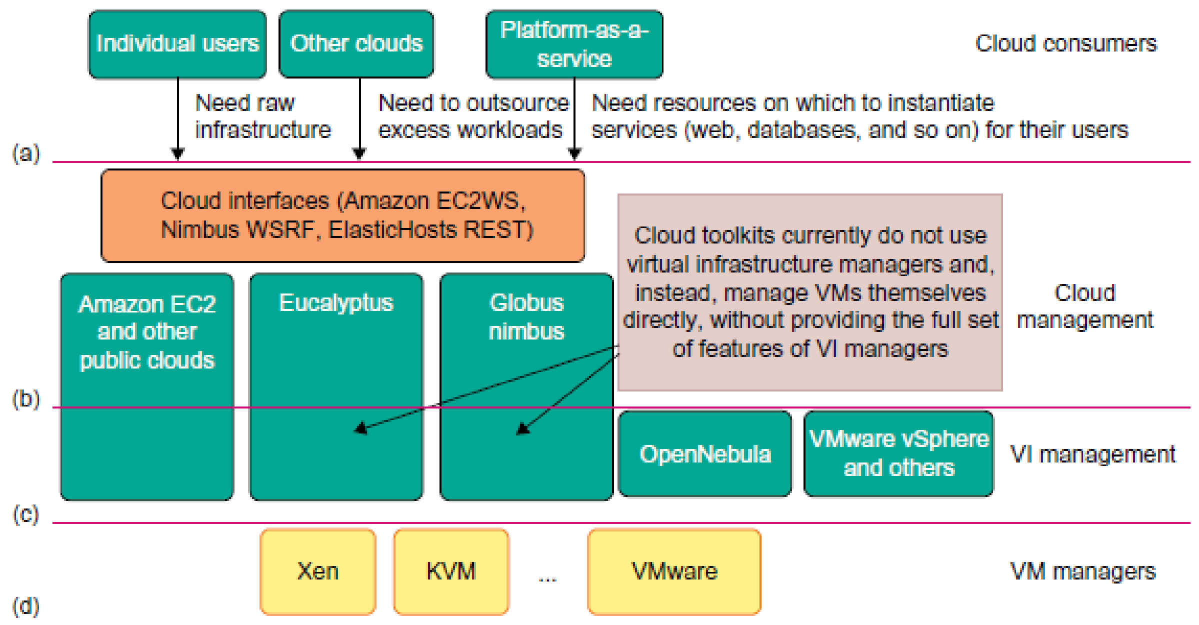

Individual users
Other clouds
Platform-as-a-service
Cloud consumers
Need raw infrastructure
Need to outsource excess workloads
Need resources on which to instantiate services (web, databases, and so on) for their users
(a)
Cloud interfaces (Amazon EC2WS, Nimbus WSRF, ElasticHosts REST)
Cloud toolkits currently do not use virtual infrastructure managers and, instead, manage VMs themselves directly, without providing the full set of features of VI managers
Amazon EC2 and other public clouds
Eucalyptus
Globus nimbus
Cloud management
(b)
OpenNebula
VMware vSphere and others
VI management
(c)
Xen
KVM
...
VMware
VM managers
(d)

FIGURE 4.4

Cloud ecosystem for building private clouds: (a) Consumers demand a flexible platform; (b) Cloud manager provides virtualized resources over an IaaS platform; (c) VI manager allocates VMs; (d) VM managers handle VMs installed on servers.

In the above suggested 4 levels, at the **user end**, a flexible platform is required by the customers. At the **cloud management level**, the VZ resources are provided by the concerned cloud manager to offer the IaaS. At the **VI management level**, the manager allocates the VMs to the available multiple clusters. Finally, at the **VM management level**, the VM managers handle VMs installed on the individual host machines.

7.9 **Increase of Private Clouds**: Private clouds influence the infrastructure and services that are utilized by an organization. Private and public clouds handle the workloads dynamically but public clouds handle them without communication dependency. On the other hand, private clouds can balance workloads to exploit the infrastructure effectively to obtain HP. The major advantage of private clouds is less security problems and public clouds need less investment.

7.10 **Infrastructure-as-a-Service (IaaS)**: A model for different services is shown in Figure 4.5 [1], as shown below. The required service is performed by the rented cloud infrastructure. On this environment, the user can deploy and run his apps. Note that user doesn't have any control over the cloud infrastructure but can choose his OS, storage, apps and network components.

Ex: Amazon EC2.

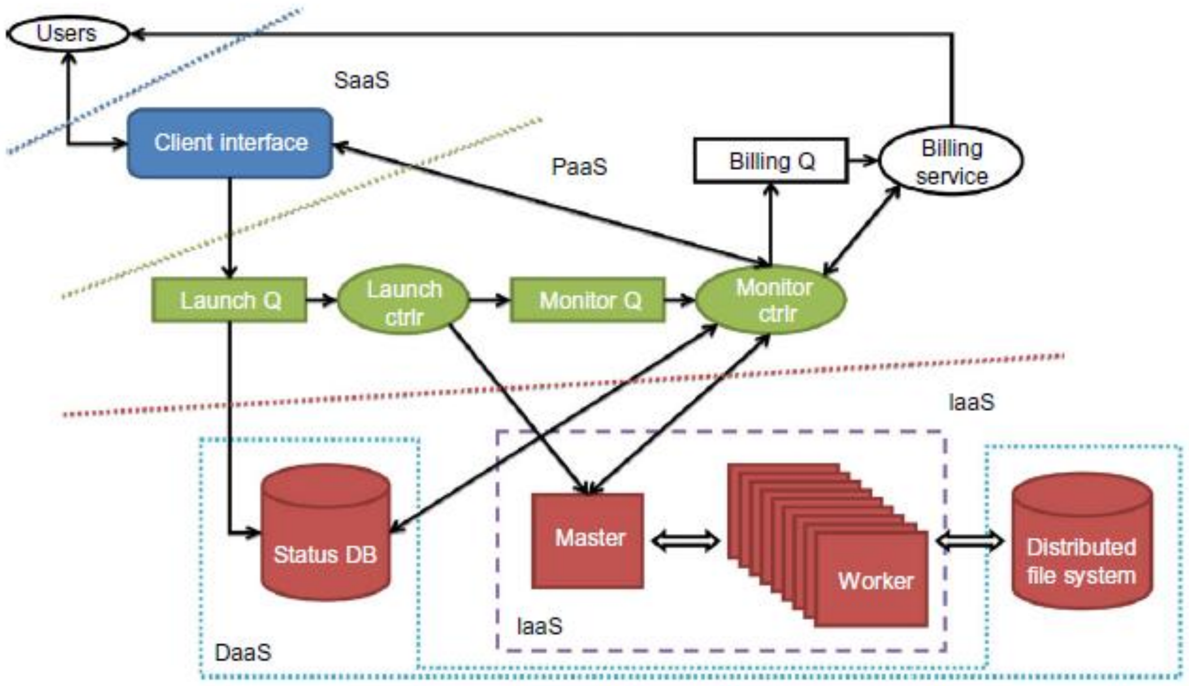


FIGURE 4.5

The IaaS, PaaS, and SaaS cloud service models at different service levels..

7.11 Platform-as-a-Service (PaaS): To develop, deploy and manage apps with provisioned resources, an able platform is needed by the users. Such a platform includes OS and runtime library support. Different PaaS offered in the current market and other details are highlighted in the Table 4.2 [1] below:

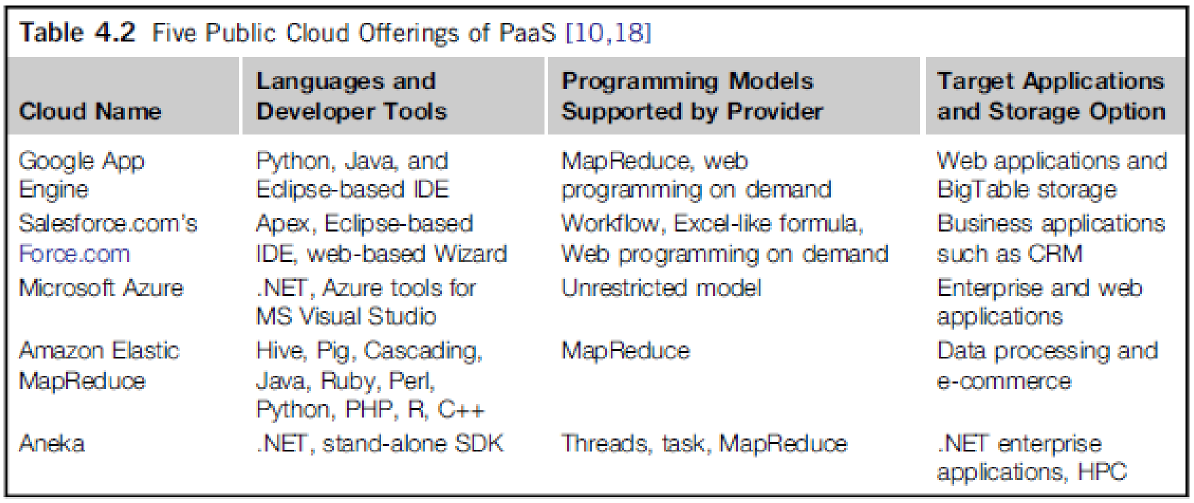


Table 4.2 Five Public Cloud Offerings of PaaS [10,18]

Cloud Name	Languages and Developer Tools	Programming Models Supported by Provider	Target Applications and Storage Option
Google App Engine	Python, Java, and Eclipse-based IDE	MapReduce, web programming on demand	Web applications and BigTable storage
Salesforce.com's Force.com	Apex, Eclipse-based IDE, web-based Wizard	Workflow, Excel-like formula, Web programming on demand	Business applications such as CRM
Microsoft Azure	.NET, Azure tools for MS Visual Studio	Unrestricted model	Enterprise and web applications
Amazon Elastic MapReduce	Hive, Pig, Cascading, Java, Ruby, Perl, Python, PHP, R, C++	MapReduce	Data processing and e-commerce
Aneka	.NET, stand-alone SDK	Threads, task, MapReduce	.NET enterprise applications, HPC

It should be noted that platform cloud is an integrated system consisting of both S/W and H/W. The user doesn't manage the cloud infrastructure but chooses the platform that is best suited to his choice of apps. The model also encourages third parties to provide software management, integration and service monitoring solutions.

7.12 Software as a Service (SaaS): This is about a browser-initiated app s/w over thousands of cloud customers. Services & tools offered by PaaS are utilized in construction and deployment of apps and management of their resources. The customer needs no investment and the provider can keep the costs low. Customer data is also stored in a cloud and is accessible through different other services. Ex: Gmail, Google docs, Salesforce.com etc.

7.13 Mixing of Cloud Services: Public clouds are more used these days but private clouds are not far behind. To utilize the resources up to the maximum level and deploy/remove the apps as per requirement, we may need to mix-up the different parts of each service to bring out a chain of connected activities. Ex: Google Maps, Twitter, Amazon ecommerce, YouTube etc.

8. **Architectural Design of Compute and Storage Clouds**: An Internet cloud [4] (CC) is envisaged (imagined) as a public cluster of servers allocated on demand to perform collective web services or distributed apps using the resources of a data center.

8.1 Cloud Platform Design Goals: The major goals of a cloud computing platform are scalability, efficiency, VZ, and reliability. A cloud platform manager receives the user requests, finds the resources, and calls the provisioning services to allocate the appropriate amount of resources for the job. Note that a manager supports both physical and virtual machines.

The platform also needs to establish an infrastructure that can obtain HPC. Scalability can be obtained by adding more data centers or servers, which leads to more efficient data distribution and, usage of less power and bandwidth.

8.2 Enabling Technologies for Clouds: The important motives behind the growth of CC are the ubiquity (present everywhere) of broadband and wireless networking, falling costs of storage, remove unneeded storage. Service-providers like Amazon and Google can make the utilization of available resources more efficient through multiplexing [5] (incorporate into an existing system), VZ, and dynamic resource provisioning. In Table 4.3 [1], the enabling of clouds is summarized.

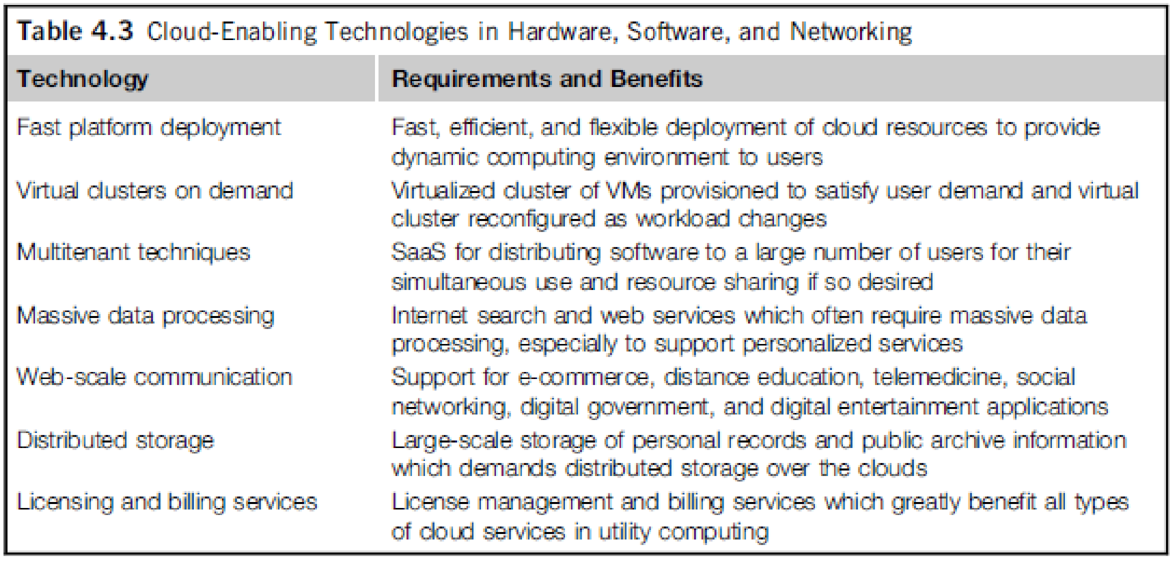


Table 4.3 Cloud-Enabling Technologies in Hardware, Software, and Networking

Technology	Requirements and Benefits
Fast platform deployment	Fast, efficient, and flexible deployment of cloud resources to provide dynamic computing environment to users
Virtual clusters on demand	Virtualized cluster of VMs provisioned to satisfy user demand and virtual cluster reconfigured as workload changes
Multitenant techniques	SaaS for distributing software to a large number of users for their simultaneous use and resource sharing if so desired
Massive data processing	Internet search and web services which often require massive data processing, especially to support personalized services
Web-scale communication	Support for e-commerce, distance education, telemedicine, social networking, digital government, and digital entertainment applications
Distributed storage	Large-scale storage of personal records and public archive information which demands distributed storage over the clouds
Licensing and billing services	License management and billing services which greatly benefit all types of cloud services in utility computing

8.3 Cloud Architecture: A generic cloud architecture can be seen Figure 4.14 [1]. The Internet Cloud is imagined as a massive cluster of servers. The different resources (space, data, and speed) of the concerned servers are allocated as per demand dynamically.

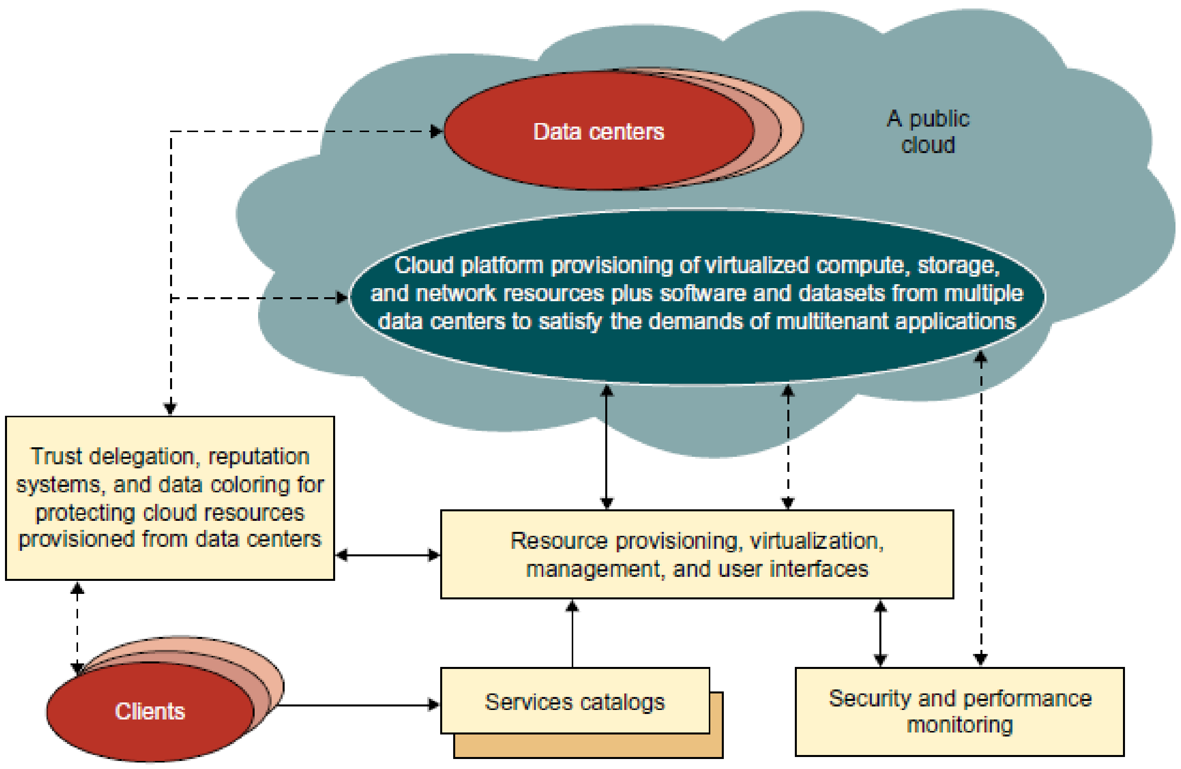

Data centers
A public cloud
Cloud platform provisioning of virtualized compute, storage, and network resources plus software and datasets from multiple data centers to satisfy the demands of multitenant applications
Trust delegation, reputation systems, and data coloring for protecting cloud resources provisioned from data centers
Resource provisioning, virtualization, management, and user interfaces
Clients
Services catalogs
Security and performance monitoring

FIGURE 4.14

A security-aware cloud platform built with a virtual cluster of VMs, storage, and networking resources over the data-center servers operated by providers.

NOTE: **Data colouring [6] (like watermarking)** protects shared data objects and ensures the security level in the cloud. These techniques safeguard multi-way authentications, enable single sign-on in the cloud, and strengthen the security for accessing confidential data in both public and private clouds.

The cloud platform demands distributed storage and different services (PaaS, IaaS and SaaS). Though the resources and services do exist and work in parallel, the user need not know about the real-work behind the screen. Any software in the cloud is a service and any service demands high amount of trust on the data retrieved from the data. Other cloud resources include storage area networks (**SAN**s), firewalls, and security devices.

The **usage and performance of granted resources** are monitored and metered by special units. The **software infrastructure of a cloud platform** must automatically handle all the resource grants and management and note the status of each node system/server when it joins/leaves the cluster. The physical location of the data center, type of power used (general/solar/hydroelectric) and cooling required are also important points.

Typically, **private clouds are easier to manage and public clouds are easier to access**. In future the clouds which utilize the best resources from both the types (hybrid) are expected to grow. Finally, security becomes a critical issue in CC to grant the success of all the services.

8.4 Cloud Architecture in Layers: Cloud architecture is developed at three layers: infrastructure, platform and app. This can be noticed in Figure 4.15 [1].

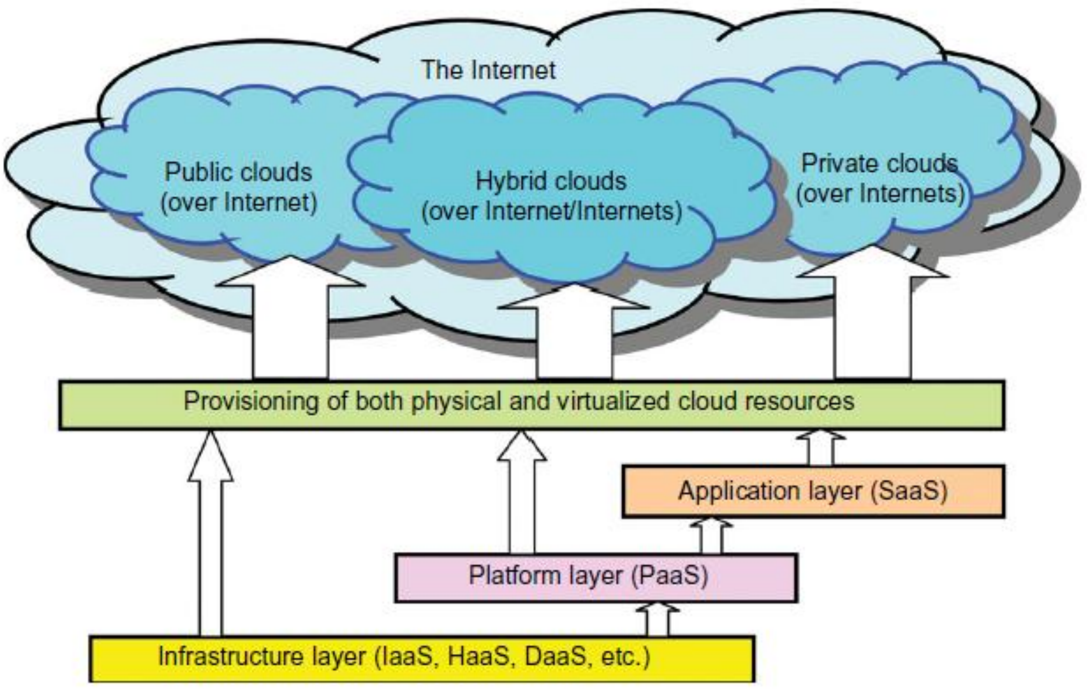

The Internet
Public clouds (over Internet)
Hybrid clouds (over Internet/Internets)
Private clouds (over Internets)
Provisioning of both physical and virtualized cloud resources
Application layer (SaaS)
Platform layer (PaaS)
Infrastructure layer (IaaS, HaaS, DaaS, etc.)

FIGURE 4.15

Layered architectural development of the cloud platform for IaaS, PaaS, and SaaS applications over the Internet.

Different VZ standards are framed and utilized in all these layers to provision the resources allocated for a cloud. The services offered to public, private and hybrid through different networking supports over the Internet and intranets.

- **Infrastructure layer** is deployed first to support the IaaS layer. It also serves as a foundation for the PaaS layer services.
- **Platform layer** itself is a foundation for the SaaS services.

The layers demand resource allocation as per demand and are granted.

- The **infrastructure layer is built with** virtualized compute, storage, and network resources. Proper utilization of these resources provides the flexibility demanded by the users. Note that **VZ demands automated provisioning** of the resources and minimum management time.
- The **platform layer** is for general purpose and repeated usage of the service resources. Proper environment is provided for the development, testing, deployment and monitoring the usage of apps. Indirectly, a virtualized cloud platform acts as a 'system middleware' between the infrastructure and application layers of a cloud.
- The **application layer** is formed with the collection of different modules of all software that are needed for the SaaS apps. The general service apps include those of information retrieval, doc processing, and authentication services. This layer also used in large-scale by the CRMs, financial transactions, and supply chain management.

Note that all the layers are built from the scratch (bottom-up) with dependence relations in between.

8.5 **NOTE**: In general, SaaS demands most work from the provider, PaaS in the middle, and IaaS demands the least. Ex: Amazon EC2. Services at app layer demand more work from the providers. Ex: Salesforce.com CRM service.

8.6 **Market-Oriented Cloud Architecture**: This can be seen in the Figure 4.16 below.

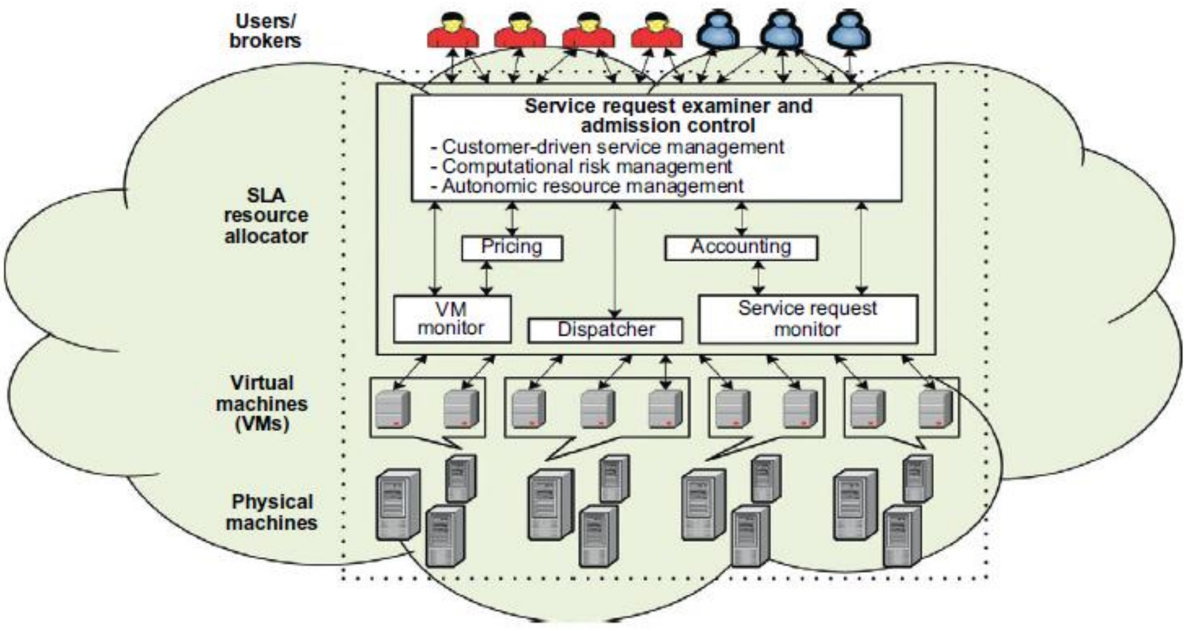

Users/ brokers
Service request examiner and admission control
- Customer-driven service management
- Computational risk management
- Autonomic resource management
SLA resource allocator
Pricing
Accounting
VM monitor
Dispatcher
Service request monitor
Virtual machines (VMs)
Physical machines

FIGURE 4.16

Market-oriented cloud architecture to expand/shrink leasing of resources with variation in QoS/demand from users.

(SLA=> Service Level Agreements)

A high level architecture can be seen in the figure for supporting market oriented resource allocation in a CC environment. The entities here are users, brokers (acting on behalf of a set of similar users), and resource allocators. When a request is made, the service request examiner comes into picture and acts as an interface between the user and the data center resources.

8.7 QoS factors: In CC, different services being offered as commercial options in the market should take into account diverse factors for every service request: time, cost, reliability, and security. The QoS requirements can’t be static and might from time to time on demand. Importance must be given to the customer, his requests and requirements – he is paying for all these. For achieving all these accomplishments in the CC market, the CRM steps into picture and plays a crucial role to satisfy each and every customer.

8.8 VZ Support and Disaster Recovery: System VZ is a much used feature in CC to improve provisioning of the resources to various services or customers. The provisioning tools, through the VMs (containers of services), try to find the best physical location wherein they plug the nodes into the data centers.

In CC, VZ also means the resources and fundamental infrastructures are virtualized. The user need not care about the computing resources, where and how they are deployed and used. The user only uses the service offers as current situation demands.

8.9 Hardware VZ: System VZ is a special kind of technique that simulates the hardware execution, utilization and provisioning methods before they can be applied in the real world of CC. VZ software is used for simulations, platform-developing for clouds, and use any kind of OS that is preferred by a developer/user. The infrastructure needed by the servers to VZ the whole data center and utilize it for CC is given below in Figure 4.17 [1].

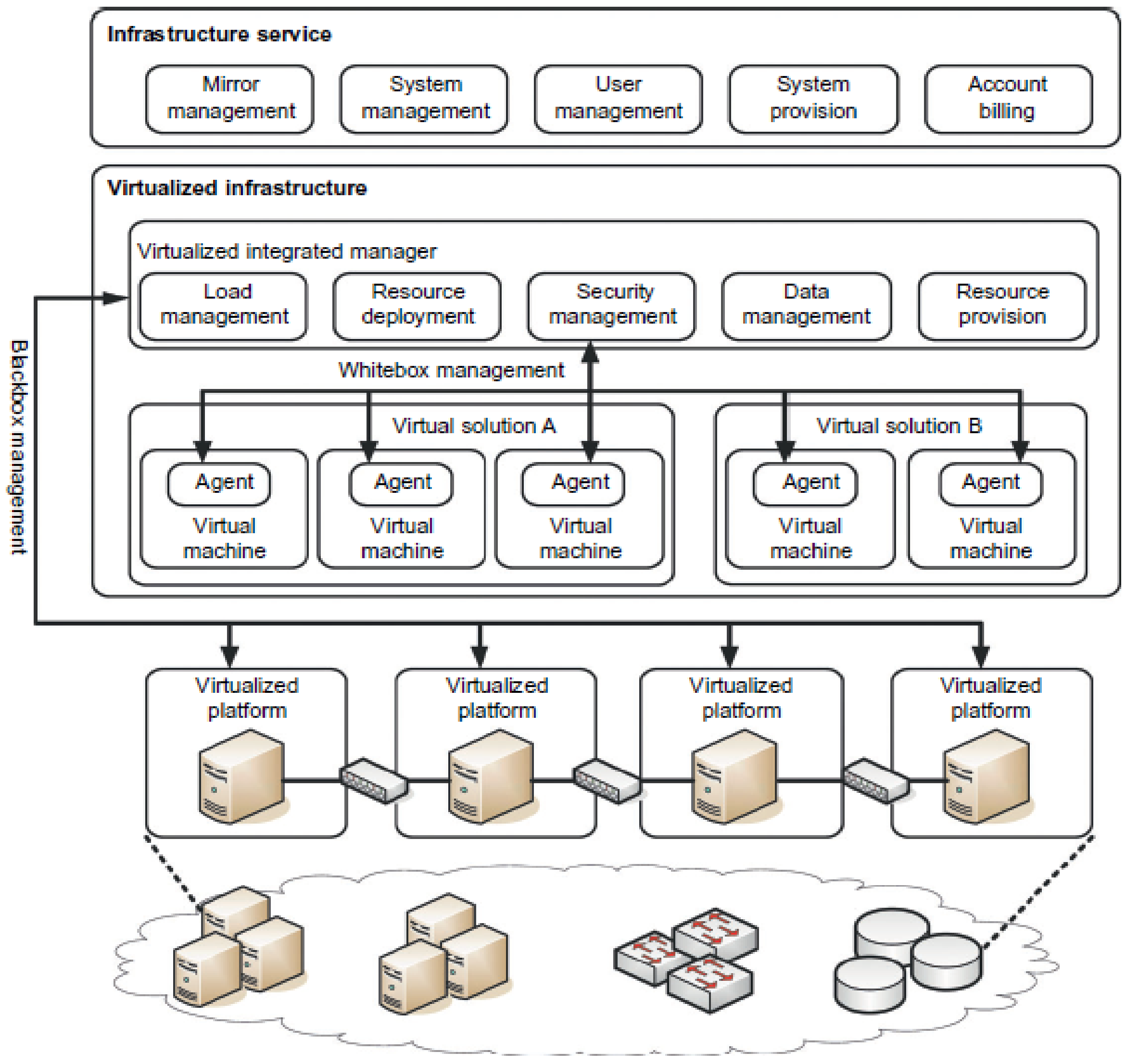


FIGURE 4.17

Virtualized servers, storage, and network for cloud platform construction.

(Courtesy of Zhong-Yuan Qin, SouthEast University, China)

8.10 **Using VMs in CC** ensures maximum flexibility for the users. A proper methodology is required for correct provisioning of the resources, distribute the burdens of space and time evenly and bring out HP. Traditional sharing of cluster resources doesn't confirm the above stated goals and an appropriate usage of all the hardware resources in all angles can be brought out by VZ of the same resources.

8.11 **VZ Support in Public Clouds**: Public clouds like AWS, MS-Azure, GAE are the famous products in the market. AWS provides extreme flexibility through VMs for the users to execute their own applications. GAE provides limited app level VZ for users since it supports only Google's services.MS provides programming level VZ for users to build their own apps.

Continuing, the VMware tools apply to workstations, servers and virtual infrastructure. The MS tools are mainly used on PCs and some servers. **The entire IT industry is changing its look** and becoming more embedded in the cloud. **VZ leads to** HA (high availability), disaster recovery, dynamic load levelling, and commendable provisioning support. Both CC and utility computing leverage (use to the maximum advantage) the benefits of VZ to increase scalability and provide an autonomous computing environment.

8.12 **VZ for IaaS**: VM technology is ubiquitous (present everywhere) enabling the users to create customised environments atop physical infrastructure. **Advantages** are: The under-utilized servers can be removed and the workload can be evenly distributed among the existing servers, VMs can run their code without conflicting with other APIs, VMs can also be used to improve security through sandbox methodology (tightly controlled set of resources) and VZ cloud platforms can isolate their performance also, increasing the QoS.

8.13 VM Cloning for Disaster Recovery: [Cloning => Make an identical copy] There exist two methods to recover from any disaster. In the first scheme, a physical machine is recovered by another physical machine. Apparently, this takes more time, energy and is more expensive. The needed hardware is to be setup, the OS is to be installed and the data recovery process has to be adjusted to other requirements too. In the other methodology, to recover a VM platform, no installation, configuration, OS setup etc. are needed – the time utilized becomes 40% less than the previous scheme.

8.14 Architectural Design Challenges:

(a) **Service Availability and Data Lock-in Problem**: If all the cloud services are functioning under a single company, that itself may be the reason of failure of the cloud. To achieve HA, it is advisable to use different services from multiple companies. Another obstacle for availability is DDoS attacks and ransomware.

Software storage and usage in a distributed manner is being done systematically, but the APIs are still vulnerable to attacks. The solution to this challenge is to standardize the APIs that are used in SaaS; all this enables the usage of a new model in public and private clouds. All this leads to '**surge computing**' where extra tasks are performed by public clouds, which can't be done in the case of private clouds.

(b) **Data Privacy and Security**: Present cloud offerings are public, but this makes them more exposed and prone to attacks. The steps that are to be taken are encrypted storage, virtual LANs, firewalls, and packet filters. The attacks that might try to intrude the cloud are malware, spyware, hijacking, DDoS, man in the middle (while migrating) and others.

(c) **Unpredictable Performance and Bottlenecks**: Multiple VMs can share CPUs and main memory in CC, but I/O sharing is difficult and cumbersome. As a solution one might try to improve the I/O architectures and operating systems to virtualize the interrupts and I/O channels. Finally, in the clouds, the data bottlenecks must be removed or widened to obtain the efficient HP.

(d) **Distributed Storage and Widespread Bugs**: DB usage is growing in CC and all of it can't be stored at a single place. Distributed storage thus comes into picture, buts also brings new problems like requirement of efficient SANs (Storage Area Network), and data durability. Simulator is a nice way to understand the problem and propose a satisfactory solution.

(e) **Cloud Scalability, Interoperability and Standardization**

(f) **Software Licensing**: Since distributed computing is widely used, any single customer's unsatisfactory usage of the concerned service may collapse the whole cloud

8.15 Public Cloud Platforms: Cloud services are provided as per demand by different companies. It can be seen in Figure 4.19 [1] that there are 5 levels of cloud players.

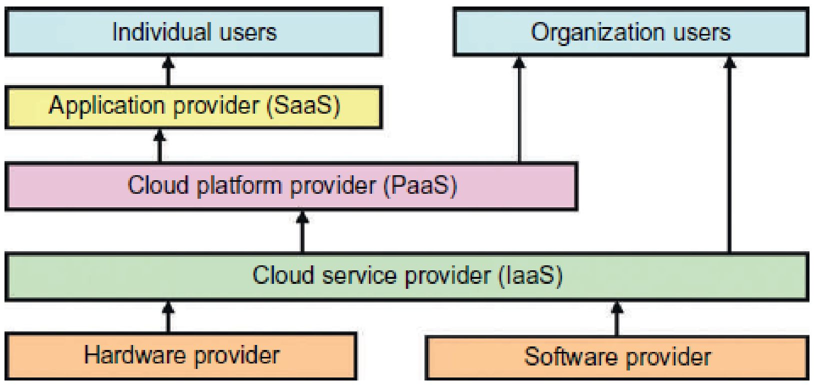

Individual users
Organization users
Application provider (SaaS)
Cloud platform provider (PaaS)
Cloud service provider (IaaS)
Hardware provider
Software provider

FIGURE 4.19

Roles of individual and organizational users and their interaction with cloud providers under various cloud service models.

The app providers at the SaaS level are used mainly by the individual users. Most business organisations are serviced by IaaS and PaaS providers. IaaS provides compute, storage, and communication resources to both app providers and organisational users. The cloud environment is defined by PaaS providers. Note that PaaS provides support both IaaS services and organisational users directly.

Cloud services depend upon machine VZ, SOA, grid infrastructure management and power efficiency. The provider service charges are much lower than the cost incurred by the users when replacing damaged servers. The Table 4.5 shows a summary of the profiles of the major service providers.

Table 4.5 Five Major Cloud Platforms and Their Service Offerings [36]

Model	IBM	Amazon	Google	Microsoft	Salesforce
PaaS	BlueCloud, WCA, RC2		App Engine (GAE)	Windows Azure	Force.com
IaaS	Ensembles	AWS		Windows Azure	
SaaS	Lotus Live		Gmail, Docs	.NET service, Dynamic CRM	Online CRM, Gifttag
Virtualization		OS and Xen	Application Container	OS level/ Hypel-V	
Service Offerings	SOA, B2, TSAM, RAD, Web 2.0	EC2, S3, SQS, SimpleDB	GFS, Chubby, BigTable, MapReduce	Live, SQL Hotmail	Apex, visual force, record security
Security Features	WebSphere2 and PowerVM tuned for protection	PKI, VPN, EBS to recover from failure	Chubby locks for security enforcement	Replicated data, rule-based access control	Admin./record security, uses metadata API
User Interfaces		EC2 command-line tools	Web-based admin. console	Windows Azure portal	
Web API	Yes	Yes	Yes	Yes	Yes
Programming Support	AMI		Python	.NET Framework	

Note: *WCA: WebSphere CloudBurst Appliance; RC2: Research Compute Cloud; RAD: Rational Application Developer; SOA: Service-Oriented Architecture; TSAM: Tivoli Service Automation Manager; EC2: Elastic Compute Cloud; S3: Simple Storage Service; SQS: Simple Queue Service; GAE: Google App Engine; AWS: Amazon Web Services; SQL: Structured Query Language; EBS: Elastic Block Store; CRM: Consumer Relationship Management.*

PKI=> Public Key Infrastructure; VPN=> Virtual Private Network

8.16 **Google App Engine (GAE)**: The Google platform is based on its search engine expertise and is applicable to many other areas (Ex: MapReduce). The **Google Cloud Infrastructure** consists of several apps like Gmail, Google Docs, and Google Earth and can support multiple no. of users simultaneously to raise the bar for HA (high availability). Other technology achievements of Google include Google File System (GFS) [like HDFS], MapReduce, BigTable, and Chubby (A Distributed Lock Service). GAE enables users to run their apps on a large number of data centers associated with Google's search engine operations. The GAE architecture can be seen in Figure 4.20 [1] below:

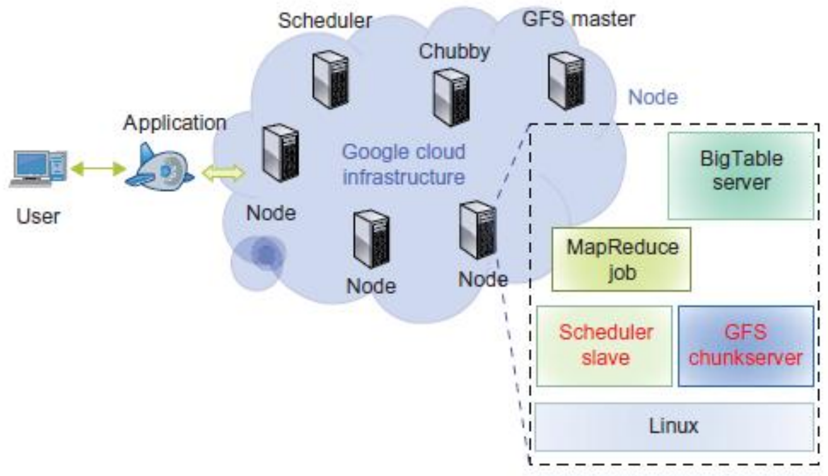


FIGURE 4.20

Google cloud platform and major building blocks, the blocks shown are large clusters of low-cost servers.

The building blocks of Google's CC app include GFS for storing large amounts of data, the MapReduce programming framework for developers, Chubby for distributed lock services and BigTable as a storage service for accessing structural data.

GAE runs the user program on Google's infrastructure where the user need not worry about storage or maintenance of data in the servers. It is a combination of several software components but the frontend is same as ASP (Active Server Pages), J2EE and JSP.

8.17 Functional Modules of GAE:

(a) **Datastore** offers OO, distributed and structured data storage services based on BigTable techniques. This secures data management operations.
(b) **Application Runtime Environment:** It is a platform for scalable web programming and execution. (Supports the languages of Java and Python)
(c) **Software Development Kit:** It is used for local app development and test runs of the new apps.
(d) **Administration Console**: Used for easy management of user app development cycles instead of physical resource management.
(e) **Web Service Infrastructure** provides special interfaces to guarantee flexible use and management of storage and network resources.

The well-known GAE apps are the search engine, docs, earth and Gmail. Users linked with one app can interact and interface with other apps through the resources of GAE (synchronise and one login for all services).

8.18 Amazon Web Services (AWS): Amazon applies the IaaS model in providing its services. The Figure 4.21 [1] below shows the architecture of AWS:

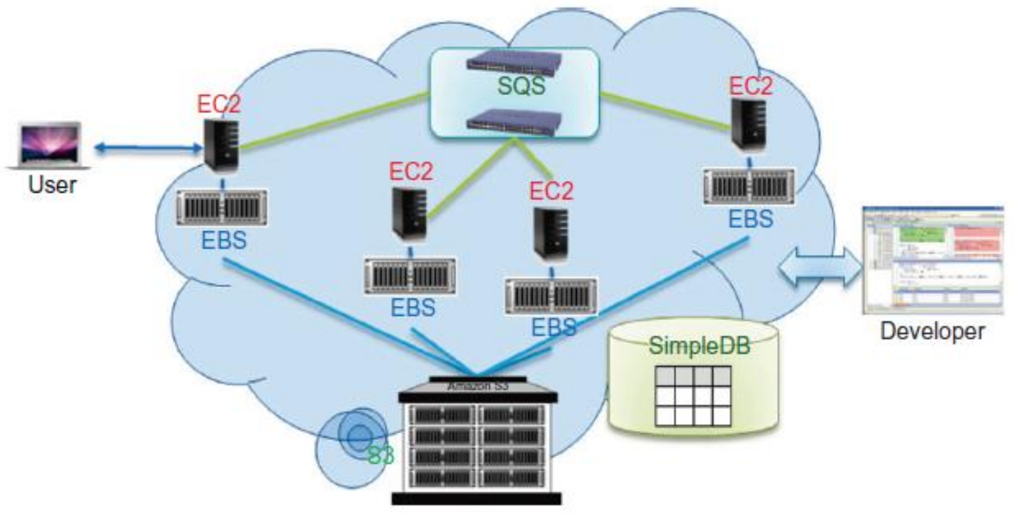


FIGURE 4.21

Amazon cloud computing infrastructure (Key services are identified here; many more are listed in Table 4.6).

EC2 provides the virtualized platforms to host the VMs where the cloud app can run.
S3 (Simple Storage Service) provides the OO storage service for the users.
EBS (Elastic Block Service) provides the block storage interface which can be used to support traditional apps.
SQS (Simple Queue Service) ensures a reliable message service between two processes.

Amazon offers a **RDS** (relational database service) with a messaging interface. The AWS offerings are given below in Table 4.6 [1].

Table 4.6 AWS Offerings in 2011

Service Area	Service Modules and Abbreviated Names
Compute	Elastic Compute Cloud (EC2), Elastic MapReduce, Auto Scaling
Messaging	Simple Queue Service (SQS), Simple Notification Service (SNS)
Storage	Simple Storage Service (S3), Elastic Block Storage (EBS), AWS Import/Export
Content Delivery	Amazon CloudFront
Monitoring	Amazon CloudWatch
Support	AWS Premium Support
Database	Amazon SimpleDB, Relational Database Service (RDS)
Networking	Virtual Private Cloud (VPC) (Example 4.1, Figure 4.6), Elastic Load Balancing
Web Traffic	Alexa Web Information Service, Alexa Web Sites
E-Commerce	Fulfillment Web Service (FWS)
Payments and Billing	Flexible Payments Service (FPS), Amazon DevPay
Workforce	Amazon Mechanical Turk

8.19 **MS-Azure**: The overall architecture of MS cloud platform, built on its own data centers, is shown in Figure 4.22 [1]. It is divided into 3 major component platforms as it can be seen. Apps are installed on VMs and Azure platform itself is built on Windows OS.

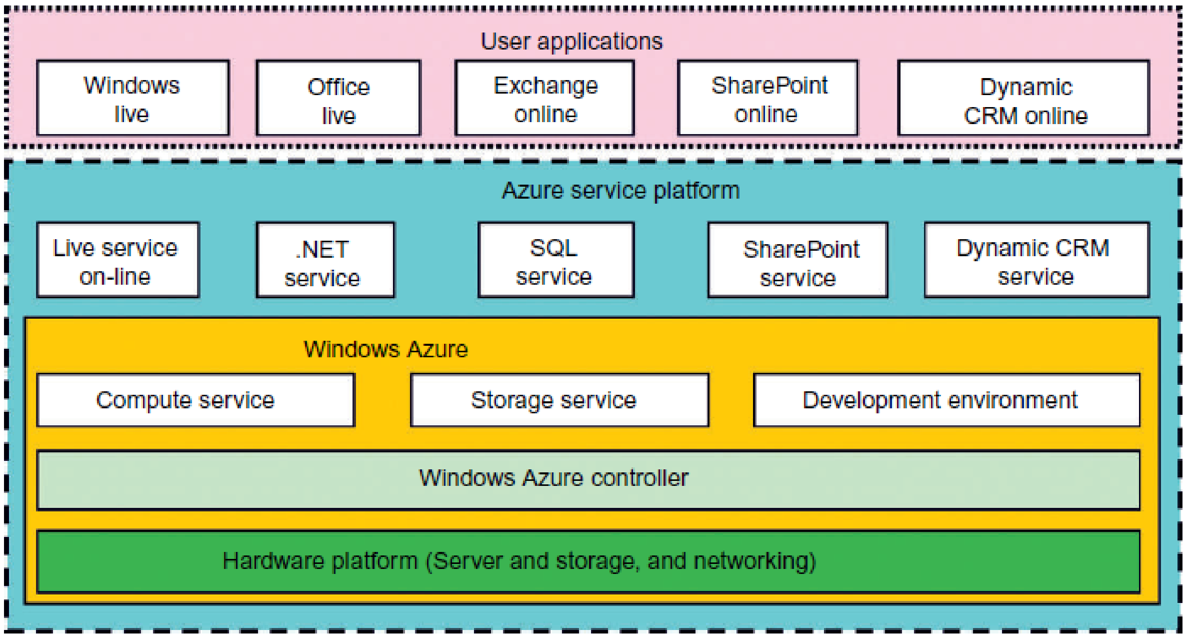


FIGURE 4.22

Microsoft Windows Azure platform for cloud computing.

- **Live Service**: Through this, the users can apply MS live apps and data across multiple machines concurrently.
- **.NET Service**: This package supports app development on local hosts and execution on cloud machines.
- **SQL Azure**: Users can visit and utilized the relational database associated with a SQL server in the cloud.
- **SharePoint Service**: A scalable platform to develop special business apps.
- **Dynamic CRM Service**: This provides a business platform for the developers to manage the CRM apps in financing, marketing, sales and promotions.

UNIT – 4

Inter-Cloud Resource Management

4. Inter-Cloud Resource Management

4.1 **Extended CC Services**: This can be viewed in Figure 4.23 [1]:

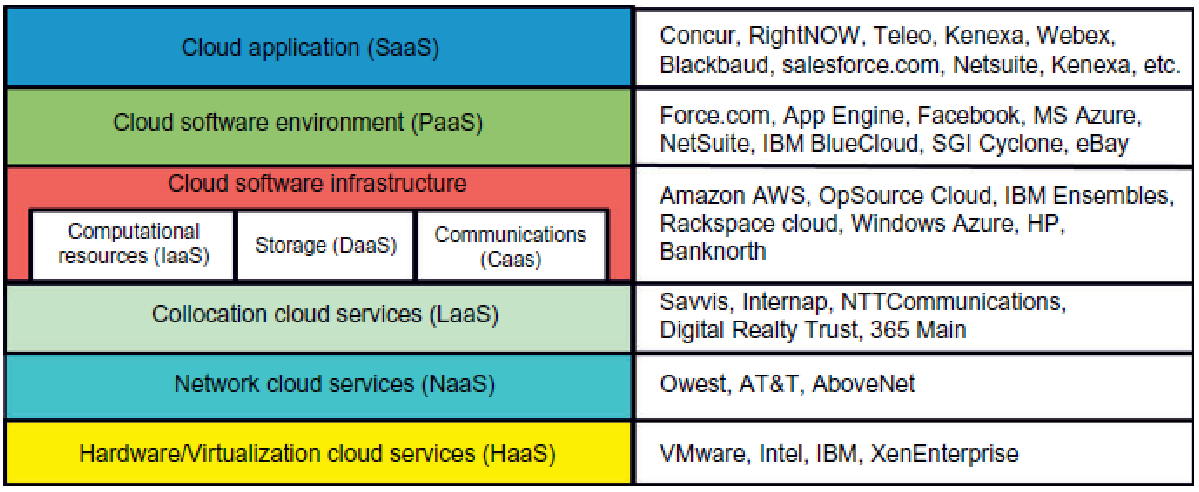

Cloud application (SaaS)
Concur, RightNOW, Teleo, Kenexa, Webex, Blackbaud, salesforce.com, Netsuite, Kenexa, etc.
Cloud software environment (PaaS)
Force.com, App Engine, Facebook, MS Azure, NetSuite, IBM BlueCloud, SGI Cyclone, eBay
Cloud software infrastructure
Computational resources (IaaS)
Storage (DaaS)
Communications (Caas)
Amazon AWS, OpSource Cloud, IBM Ensembles, Rackspace cloud, Windows Azure, HP, Banknorth
Collocation cloud services (LaaS)
Savvis, Internap, NTTCommunications, Digital Realty Trust, 365 Main
Network cloud services (NaaS)
Owest, AT&T, AboveNet
Hardware/Virtualization cloud services (HaaS)
VMware, Intel, IBM, XenEnterprise

FIGURE 4.23

A stack of six layers of cloud services and their providers.

The top three service layers are SaaS, PaaS and IaaS. The bottom three layers are related to physical requirements and are as Hardware as a Service (HaaS), Network as a Service (NaaS), Location as a Service (LaaS), and Security as a Service (SaaS).

Table 4.7 Cloud Differences in Perspectives of Providers, Vendors, and Users

Cloud Players	IaaS	PaaS	SaaS
IT administrators/cloud providers	Monitor SLAs	Monitor SLAs and enable service platforms	Monitor SLAs and deploy software
Software developers (vendors)	To deploy and store data	Enabling platforms via configurators and APIs	Develop and deploy software
End users or business users	To deploy and store data	To develop and test web software	Use business software

Table 4.7 [1] shows that cloud players are into three classes.

4.2 **Software Stack for CC**: A software stack [7] is a group of programs that work in tandem (in order) to produce a common goal. It may also refer to any set of apps that works in a specific order toward a common goal. Ex: Like a set in maths or a cluster in DM. The system has to be designed to meet goals like HT, HA, and fault tolerance. Physical or virtual servers can be used making the platform more flexible and be able to store and utilize large amount of data.

4.3 **Resource Provisioning and Platform Deployment:**

1. **Provisioning of Compute Resources (VMs)**: The provisioning of resources like CPU, memory, and bandwidth are distributed among the users as per the service level agreements (SLAs) signed before the start of the work. The problem here is the ever-changing levels of requests from the user, power management and conflicts in the SLAs.

Efficient VM provisioning depends on the cloud architecture and management of cloud infrastructures. Resource provisioning also demands fast discovery of services and data in the provided infrastructure. Ex: Efficient installation of VMs, live VM migration, and fast recovery from failures. Providers like Amazon, IBM and MS-Azure use VM templates, automation of provisioning and power-efficient schemes.

2. **Resource Provisioning Methods**:
   - **(a) Demand-Driven Resource Provisioning**: This method adds or removes computing instances based on the current utilization level for the allocated resources. This method automatically allocates two processors for the user app, if the user utilizes more than 60% of time for an extended period. That is, if **the resource utilization has crossed a threshold** of the concerned resource, **extra resources will be allocated**. This methodology is implemented by Amazon in EC2.
   - **(b) Event-Driven Resource Provisioning**: This scheme adds or removes machine instances based on an event like festival season. At this time, the no. of users peaks and so does the traffic. This anticipation results in good QoS and customer satisfaction.
   - **(c) Popularity-Driven Resource Provisioning:** In this method, The Internet searches for popularity of certain apps and creates extra instances if the popularity has risen.
   - **(d) Dynamic Resource Deployment:** This can be implemented to achieve scalability in performance through efficient allocation of resources at every place in the grid as the situation demands. To achieve this, we need an **inter-grid gateway** (IGG) between different grids that allocates the resources from a local cluster to deploy apps by **requesting the VMs**, **enacting** (endorse) **the leases**, and **deploying the VMs as per requests**.

     The Inter-Grid provides and allocates a **distributed virtual environment (DVE**). It is a virtual cluster of VMs that runs in isolation from other virtual clusters. This process is carried out by a component called DVE manager. Received massages are handled in parallel in a thread pool. All these methodologies are depicted in Figure 4.26.

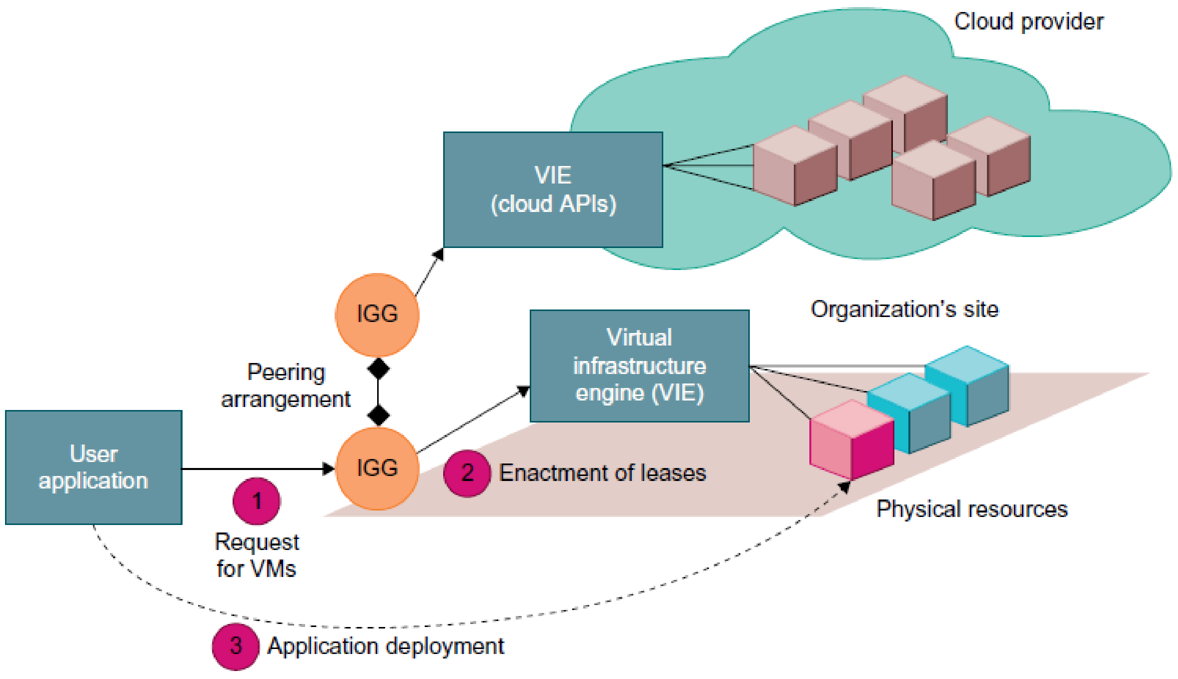


FIGURE 4.26

Cloud resource deployment using an IGG (intergrid gateway) to allocate the VMs from a Local cluster to interact with the IGG of a public cloud provider.

3. **Provisioning of Storage Resources**: The data in CC is stored in the clusters of the cloud provider and can be accessed anywhere in the world. Ex: email. For data storage, distributed file system, tree structure file system, and others can be used. Ex: GFS, HDFS, MS-Cosmos. This method provides a

convenient coding platform for the developers. The storage methodologies and their features can be found in Table 4.8 [1].

Table 4.8 Storage Services in Three Cloud Computing Systems

Storage System	Features
GFS: Google File System	Very large sustainable reading and writing bandwidth, mostly continuous accessing instead of random accessing. The programming interface is similar to that of the POSIX file system accessing interface.
HDFS: Hadoop Distributed File System	The open source clone of GFS. Written in Java. The programming interfaces are similar to POSIX but not identical.
Amazon S3 and EBS	S3 is used for retrieving and storing data from/to remote servers. EBS is built on top of S3 for using virtual disks in running EC2 instances.

POSIX => Portable OS Interface
EBS => Elastic Block Storage
EC2 => Elastic Compute Cloud
S3 => Amazon Simple Storage Service

4.4 **Virtual Machine Creation & Management**: Figure 4.27 [1] shows the interactions among VM managers for cloud creation and management.

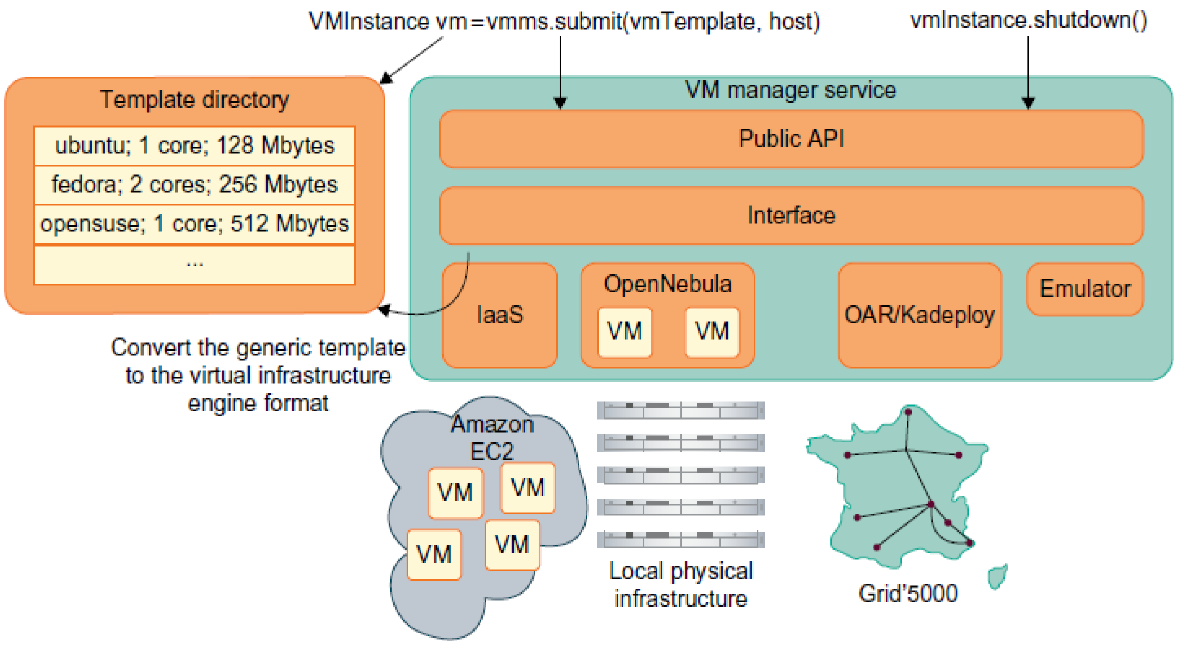

VMInstance vm=vmms.submit(vmTemplate, host)
vmInstance.shutdown()
Template directory
ubuntu; 1 core; 128 Mbytes
fedora; 2 cores; 256 Mbytes
opensuse; 1 core; 512 Mbytes
...
VM manager service
Public API
Interface
IaaS
OpenNebula
VM
VM
OAR/Kadeploy
Emulator
Convert the generic template to the virtual infrastructure engine format
Amazon EC2
VM
VM
VM
VM
Local physical infrastructure
Grid'5000

FIGURE 4.27

Interactions among VM managers for cloud creation and management; the manager provides a public API for users to submit and control the VMs.

(a) **Independent Service Management**: By using independent service providers, the cloud apps can run different services at the same time. Some other services are used for providing data other than the compute or storage services.

(b) **Running Third Party Apps**: IN this case, the cloud platforms have to provide support for apps constructed by third-party app providers. The concerned APIs are in the form of services provided by another company. (Ex: Dropbox + Gmail + User).

(c) **VM Manager**: It is a link between the gateway and resources. The physical resources aren't shared directly, but in a virtualized method. The VMs themselves become the actual resources. Ex: OpenNebula (an OS). Users submit VMs on physical machines using hypervisors, which enables the running of several operating systems on the same host concurrently.

(d) **VM Templates**: A VM template is analogous (similar) to the configuration of a computer and contains the description for a VM. Information provided is:

- The no. of processors allocated to the VM
- Memory required by a VM
- The kernel used by the VM's OS
- The disk image containing the VM's file system
- The price per hour

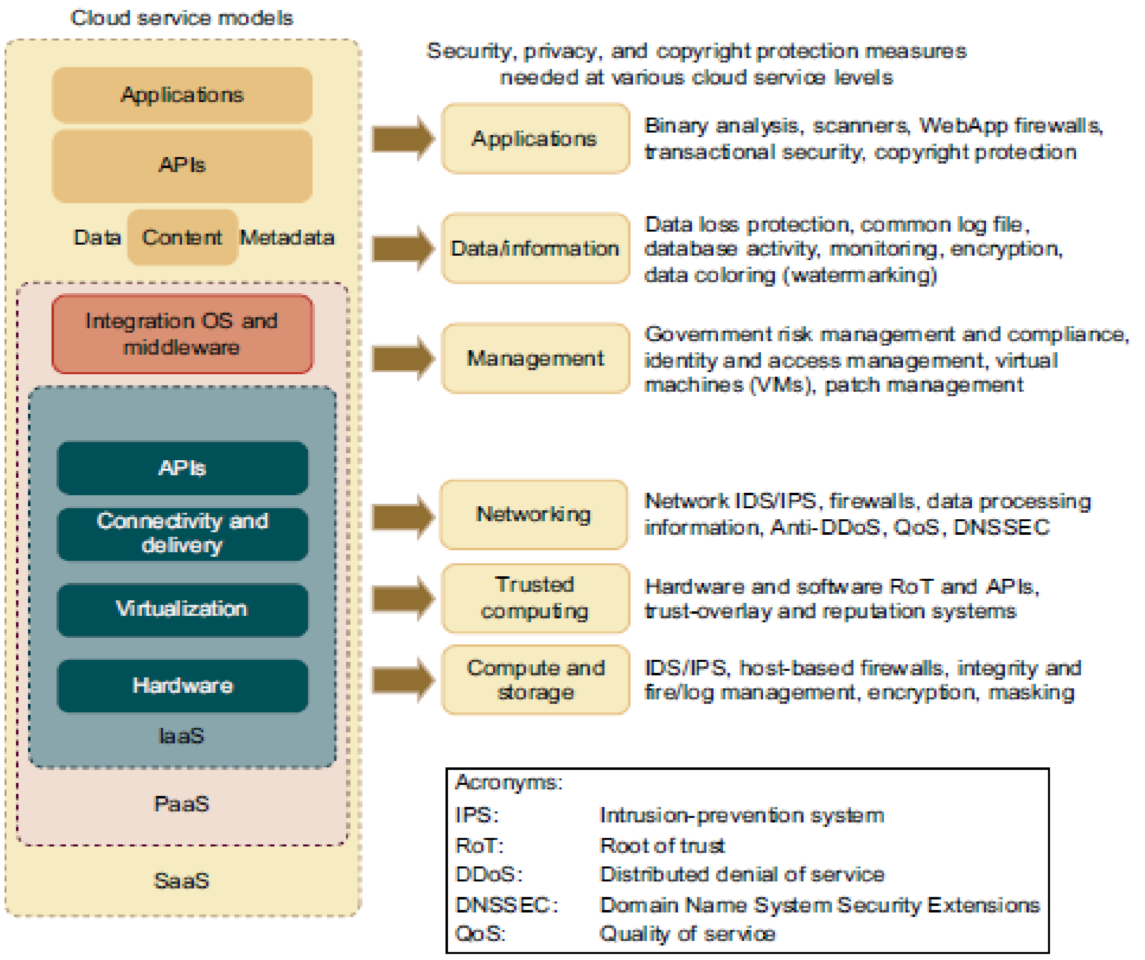
The gateway administrator provides the VM template information and can add, update and delete the templates at any time. Before starting an instance, scheduler gives the network configuration and address of the host. The MAC and IP addresses are also allocated. It also contains the path to the disk image storage.

(e) **Distributed VM Management**: A distributed VM manager requests for VMs and gets their status and obtains a list containing the IP addresses of the VMs with secure shell (SSH) tunnels. The managers also obtains the template to be used by the VM, schedules the task for the VM, sets up the tunnel, and executes the tasks for each of the VM.

5. **Cloud Security and Trust Management**: Lacking of trust between service providers and clients has been a major problem in the field and much more since the advent of ecommerce. Cloud platforms are a concern for some users for lack of privacy protection, security assurance, and so on. All these can be solved with a technical approach.

5.1 **Cloud Security Defence Strategies**:

5.2 **Basic Cloud Security**: The basic cloud security enforcements are: security measures in data centers (like biometric readers, CCTV, man-traps etc.), fault-tolerant firewalls, IDS Intrusion Detection System), data encryption, strict password policies, and so on. The Figure 4.31 [1] shows the security measures at various levels:


Cloud service models
Applications
APIs
Data
Content
Metadata
Integration OS and middleware
APIs
Connectivity and delivery
Virtualization
Hardware
IaaS
PaaS
SaaS
Security, privacy, and copyright protection measures needed at various cloud service levels
Applications
Binary analysis, scanners, WebApp firewalls, transactional security, copyright protection
Data/information
Data loss protection, common log file, database activity, monitoring, encryption, data coloring (watermarking)
Management
Government risk management and compliance, identity and access management, virtual machines (VMs), patch management
Networking
Network IDS/IPS, firewalls, data processing information, Anti-DDoS, QoS, DNSSEC
Trusted computing
Hardware and software RoT and APIs, trust-overlay and reputation systems
Compute and storage
IDS/IPS, host-based firewalls, integrity and fire/log management, encryption, masking

Acronyms:

IPS:	Intrusion-prevention system
RoT:	Root of trust
DDoS:	Distributed denial of service
DNSSEC:	Domain Name System Security Extensions
QoS:	Quality of service

(a) Cloud service models (b) Security, privacy, and copyright protection measures

FIGURE 4.31

Cloud service models on the left (a) and corresponding security measures on the right (b); the IaaS is at the innermost level, PaaS is at the middle level, and SaaS is at the outermost level, including all hardware, software, datasets, and networking resources.

5.3 **Cloud Defence Methods**: Virtualization enhances cloud security, but VMs add an additional layer of software that might lead to a single point of failure. So the VMs should be isolated in their deployment and work – the failure of one VM will not affect another. The Table 4.9 [1] below lists the protection schemes to secure public clouds and data centers.

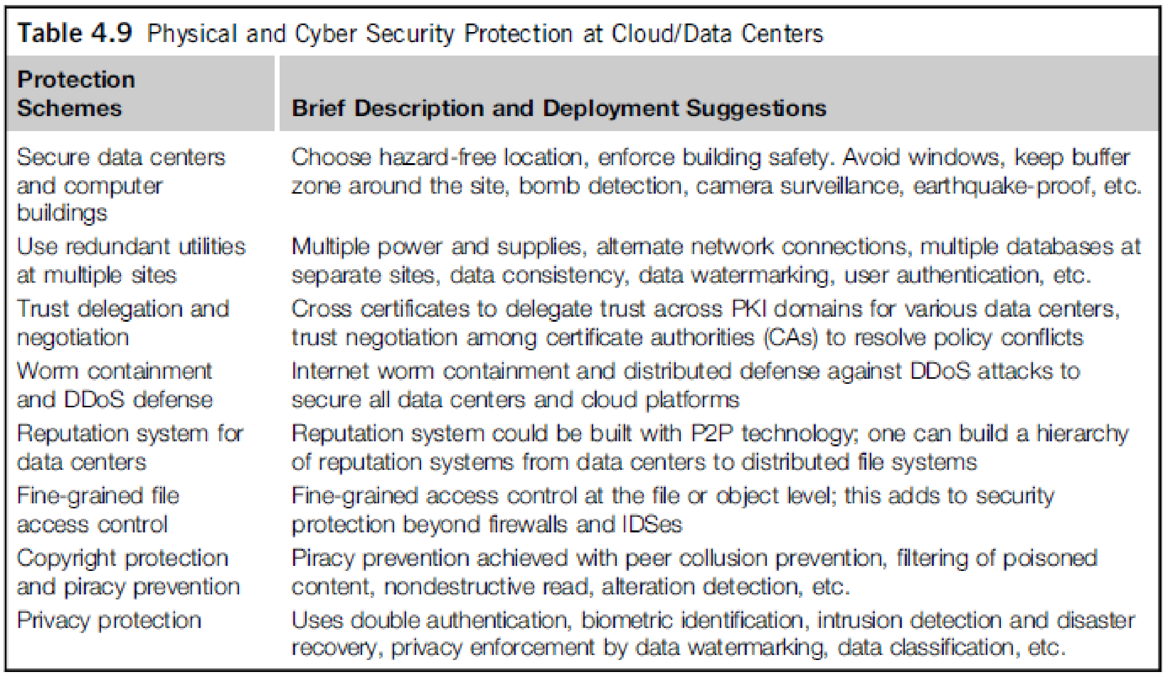


Table 4.9 Physical and Cyber Security Protection at Cloud/Data Centers

Protection Schemes	Brief Description and Deployment Suggestions
Secure data centers and computer buildings	Choose hazard-free location, enforce building safety. Avoid windows, keep buffer zone around the site, bomb detection, camera surveillance, earthquake-proof, etc.
Use redundant utilities at multiple sites	Multiple power and supplies, alternate network connections, multiple databases at separate sites, data consistency, data watermarking, user authentication, etc.
Trust delegation and negotiation	Cross certificates to delegate trust across PKI domains for various data centers, trust negotiation among certificate authorities (CAs) to resolve policy conflicts
Worm containment and DDoS defense	Internet worm containment and distributed defense against DDoS attacks to secure all data centers and cloud platforms
Reputation system for data centers	Reputation system could be built with P2P technology; one can build a hierarchy of reputation systems from data centers to distributed file systems
Fine-grained file access control	Fine-grained access control at the file or object level; this adds to security protection beyond firewalls and IDSes
Copyright protection and piracy prevention	Piracy prevention achieved with peer collusion prevention, filtering of poisoned content, nondestructive read, alteration detection, etc.
Privacy protection	Uses double authentication, biometric identification, intrusion detection and disaster recovery, privacy enforcement by data watermarking, data classification, etc.

5.4 **Defence against DDoS Flooding attacks**: A DDoS defence system must be designed to cover multiple network domains in a cloud platform. The DDoS causes an abnormal surge in the network traffic by a hidden attacker which leads of the crash of the service/website or disk exhaustion or connection saturation.

5.5 **Data and Software Protection Techniques**:
(a) Data Integrity and Privacy Protection
(b) Data Colouring and Cloud Watermarking
(c) Data Lock-in Problems and Solutions: Once the data is moved into the cloud, users cannot easily extract their data and programs from the cloud servers to run on another platform. This is known as data lock-in. The solution possible here is to build platform-independent APIs where migration from one platform to another is easier.

6. **Service-Oriented Architecture**: SOA is concerned about how to design a software system that makes use of services or apps through their interfaces. These apps are distributed over the networks. The World Wide Web Consortium (W3C) defines SOA as a form of distributed architecture characterized by:

- **Logical View**: The SOA is an abstracted, logical view of actual programs, DBs etc. defined in terms of the operations it carries out. The service is formally defined in terms of messages exchanged between providers and requests.
- **Message Orientation**
- **Description Orientation**

7. **Services and Web Services**: In an SOA concept, the s/w capabilities are delivered & consumed through loosely coupled and reusable services using messages. 'Web Service' is a self-contained modular application designed to be used by other apps across the web. This can be seen in Figure 5.2 [1].

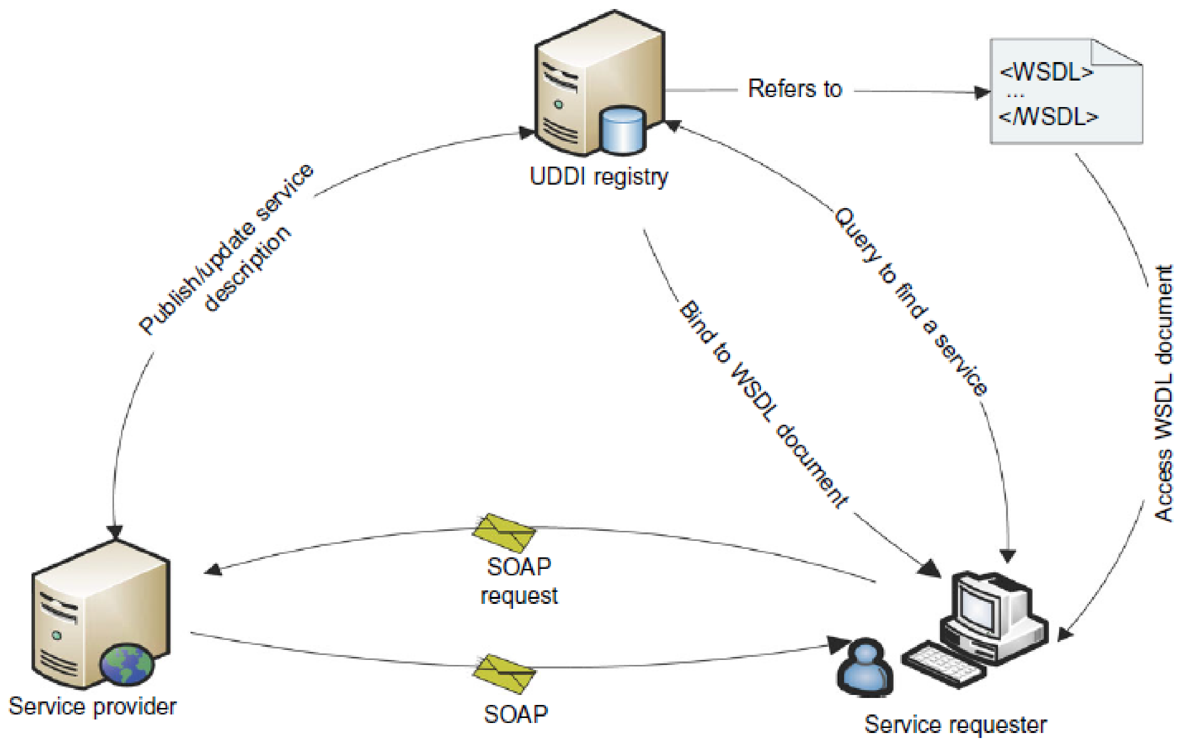


FIGURE 5.2

A simple web service interaction among provider, user, and the UDDI registry.

WSDL => Web Services Description Language
UDDI => Universal Description, Discovery and Integration
SOAP => Simple Object Access Protocol

7.1 **SOAP**: This provides a standard packaging structure for transmission of XML documents over various IPs. (HTTP, SMTP, FTP). A SOAP message consists of an **envelope** (root element), which itself contains a **header**. It also had a **body** that carries the payload of the message.
7.2 **WSDL**: It describes the interface and a set of operations supported by a web service in a standard format.
7.3 **UDDI**: This provides a global registry for advertising and discovery of web services by searching for names, identifiers, categories.

7.4 Since SOAP can combine the strengths of XML and HTTP, it is useful for heterogeneous distributed computing environments like grids and clouds

8. **Enterprise Multitier Architecture**: This is a kind of client/server architecture application processing and data management are logically separate processes. As seen below in Figure 5.4 [1], it is a three-tier information system where each layer has its own important responsibilities.

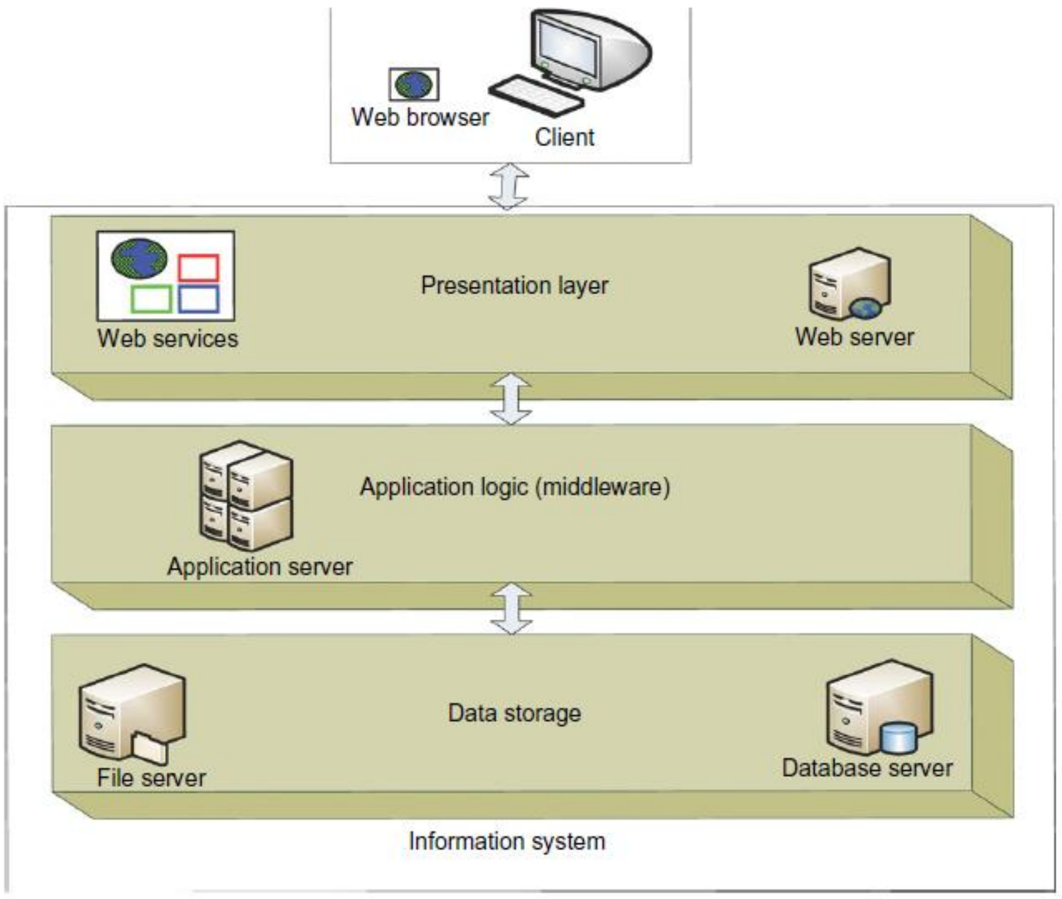


FIGURE 5.4

Three-tier system architecture.

Presentation Layer: Presents information to external entities and allows them to interact with the system by submitting operations and getting responses.

Application Logic (Middleware): These consist of programs that implement actual operations requested by the client. The middle tier can also be used for user authentication and granting of resources, thus removing some load from the servers.

Resource Management Layer (Data Layer): It deals with the data sources of an information system.

9. **OGSA Grid**: Open Grid Services Architecture is intended to
   - Facilitate the usage of resources across heterogeneous environments
   - Deliver best QoS
   - Define open interfaces between diverse resources
   - Develop inter-operable standards

10. OGSA architecture falls into seven broad areas, as shown in Figure 5.5 [1].
    Infrastructure Services, Execution Management Services, Data Management Services, Resource Management Services, Security Services, Security Services, Information Services and Self-management Services (automation).

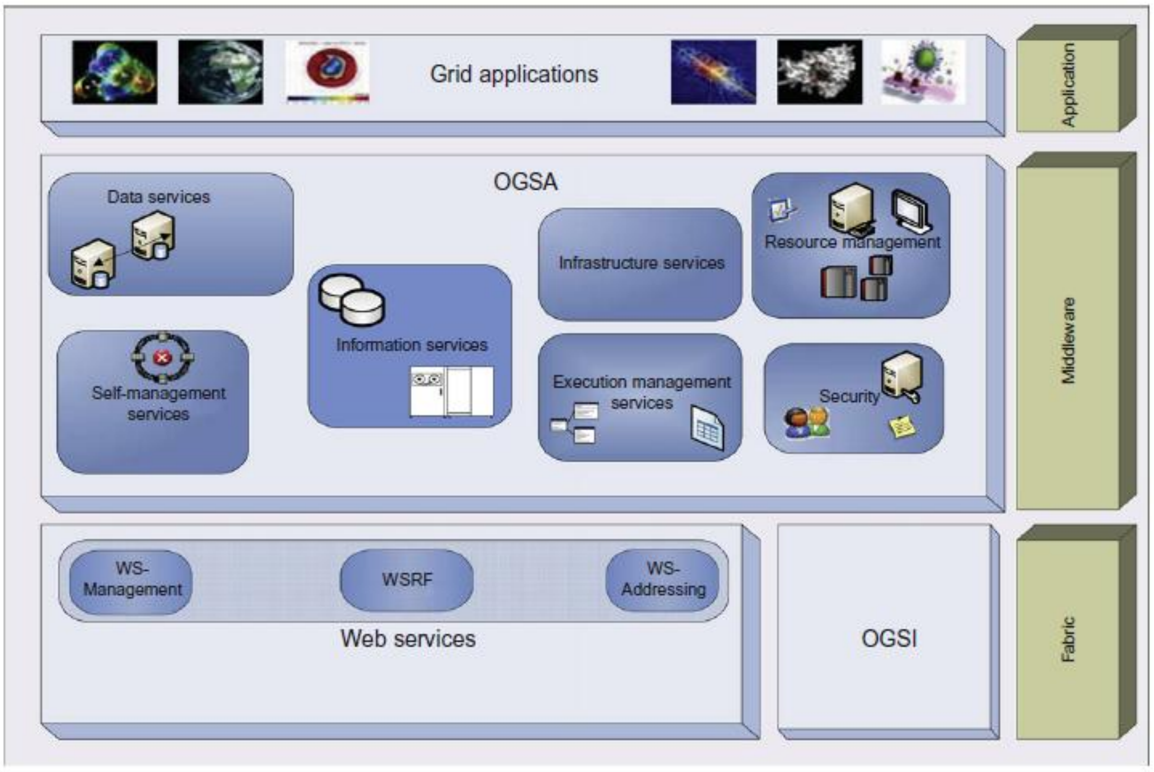


FIGURE 5.5

The OGSA architecture.

11. **Message-Oriented Middleware**:

11.1 **Enterprise Bus**: Figure 5.6 [1]

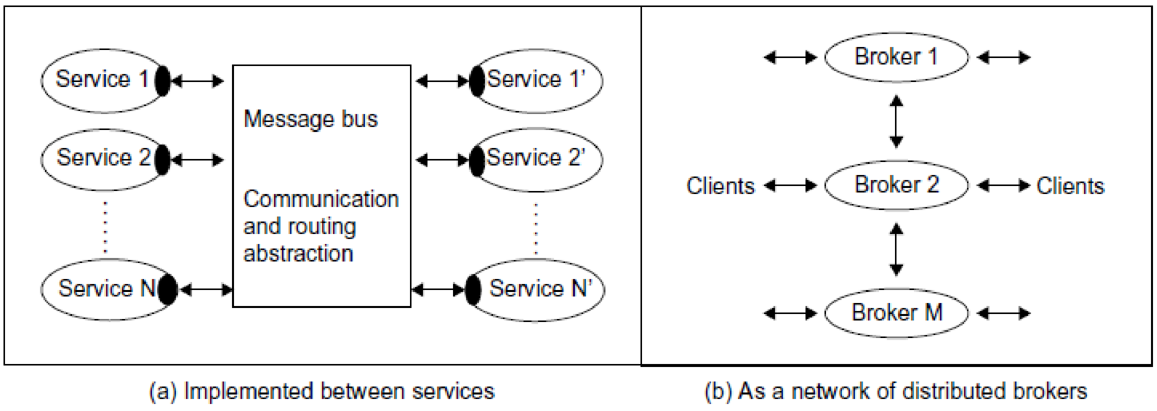

Service 1
Service 2
Service N
Message bus
Communication and routing abstraction
Service 1'
Service 2'
Service N'
(a) Implemented between services
Broker 1
Clients
Broker 2
Clients
Broker M
(b) As a network of distributed brokers

FIGURE 5.6

Two message bus implementations between services or using a broker network.

Enterprise Service Bus (ESB) refers to the case where the bus supports the integration of many components in different styles as shown above. No source and destination channel is opened but only messages are induced from different services. A message bus is shown linking the services by receiving and sending messages but this methodology can work with any software or hardware.

11.2 **Queuing and Message Systems**: The best known standard in this field is the Java Message Service (JMS) which specifies a set of interfaces utilized in communication queuing systems. Advanced Message Queuing Protocol (AMQP) specifies a set of wire formats for communications.

References

- Kai Hwang et al, Distributed and Cloud Computing – From Parallel Processing to the Internet of Things, Morgan Kaufmann, Elsevier, 2012.
- https://www.techopedia.com/definition/26598/elastic-computing-ec
- https://en.wikipedia.org/wiki/Data_farming
- http://www.webopedia.com/TERM/C/cloud_computing.html
- http://searchnetworking.techtarget.com/definition/multiplexing
- https://www.researchgate.net/post/What_is_data_coloring
- https://www.techopedia.com/definition/27268/software-stack

UNIT – 5

Virtual clusters and resource management

10. **Virtual Clusters and Resource Management**: A physical cluster is a collection of physical servers that are interconnected. The issues that are to be dealt with here are: live migration of VMs, memory and file migrations and dynamic deployment of virtual clusters.

When a general VM is initialized, the administrator has to manually write configuration information; this increases his workload, particularly when more and more VMs join the clusters. As a solution to this, a service is needed that takes care of the configuration information (capacity, speed etc.) of the VMs. The best example is Amazon's **Elastic Compute Cloud (EC2)**, which provides elastic computing power in a cloud.

Most VZ platforms like VMware ESX Server, and XenServer support a **bridging mode** which allows all *domains to appear* on the network *as individual hosts*. Through this mode, VMs can communicate with each other freely through the virtual network and configure automatically.

10.1 Physical versus Virtual Clusters: Virtual Clusters are built with VMs installed at one or more physical clusters. The VMs in a virtual cluster are interconnected by a virtual network across several physical networks. The concept can be observed in Figure 3.18 [1].

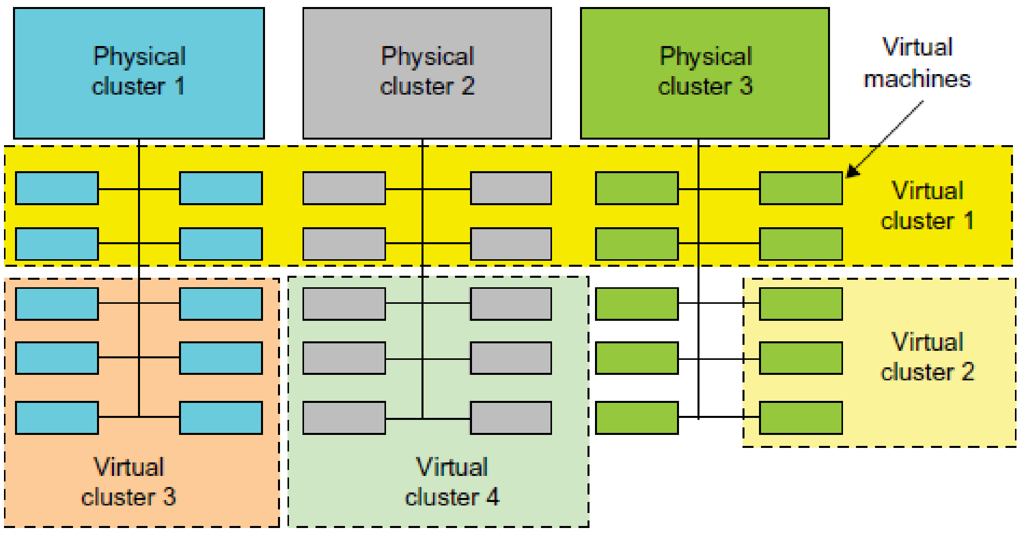


FIGURE 3.18

A cloud platform with four virtual clusters over three physical clusters shaded differently.

10.2 The provisioning of VMs to a virtual cluster is done dynamically to have the following properties:

- The virtual cluster nodes can be either physical or virtual (VMs) with different operating systems.
- A VM runs with a guest OS that manages the resources in the physical machine.

- The purpose of using VMs is to consolidate multiple functionalities on the same server.
- VMs can be replicated in multiple servers to promote parallelism, fault tolerance and disaster discovery.
- The no. of nodes in a virtual cluster can grow or shrink dynamically.
- The failure of some physical nodes will slow the work but the failure of VMs will cause no harm (fault tolerance is high).

10.3 **NOTE**: Since system virtualization is widely used, the VMs on virtual clusters have to be effectively managed. The virtual computing environment should provide high performance in virtual cluster deployment, monitoring large clusters, scheduling of the resources, fault tolerance and so on.

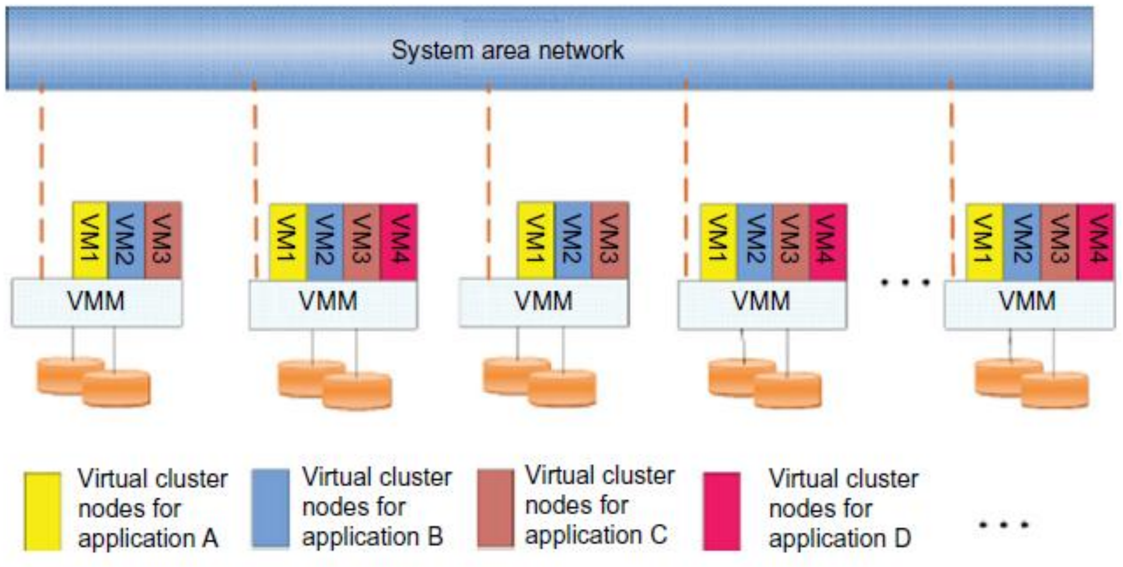

System area network
VM1
VM2
VM3
VM4
VMM
Virtual cluster nodes for application A
Virtual cluster nodes for application B
Virtual cluster nodes for application C
Virtual cluster nodes for application D
• • •

FIGURE 3.19

The concept of a virtual cluster based on application partitioning.

10.4 Figure 3.19 [1] shows the concept of a virtual cluster based on app partitioning. The different colours represent nodes in different virtual clusters. The storage images (SSI) from different VMs from different clusters is the most important concept here. Software packages can be pre-installed as templates and the users can build their own software stacks. Note that the boundary of the virtual cluster might change since VM nodes are added, removed, or migrated dynamically.

10.5 **Fast Deployment and Effective Scheduling**: The concerned system should be able to

- Construct and distribute software stacks (OS, libraries, apps) to a physical node inside the cluster as fast as possible
- Quickly switch runtime environments from one virtual cluster to another.

NOTE: **Green Computing**: It is a methodology that is environmentally responsible and an eco-friendly usage of computers and their resources. It is also defined as the study of designing, manufacturing, using and disposing of computing devices in a way that reduces their environmental impact.

10.6 Engineers must concentrate upon the point the available resources are utilized in a cost and energy-reducing manner to optimize the performance and throughput. Parallelism must be put in place wherever needed and virtual machines/clusters should be used for attaining this goal. Through this, we can reduce the overhead, attain load balancing and achieve scale-up and scale-down mechanisms on the virtual clusters. Finally, the virtual clusters must be clustered among themselves again by mapping methods in a dynamical manner.

10.7 High Performance Virtual Storage: A template must be prepared for the VM construction and usage and distributed to the physical hosts. Software packages that reduce the time for customization (getting used to) and switching of environment. Users should be identified by their profiles that are stored in data blocks. All these methods increase the performance in virtual storage. Ex: Dropbox

Steps to deploy (arrange/install) a group of VMs onto a target cluster:

- Preparing the disk image (SSI)
- Configuring the virtual machines
- Choosing the destination nodes
- Executing the VM deployment commands at every host

10.8 NOTE: A **template** is a disk image/SSI that hides the distributed environment from the user. It may consist of an OS and some apps. Templates are chosen by the users as per their requirements and can implement COW (Copy on Write) format. A new COW backup file is small and easy to create and transfer, thus reducing space consumption.

It should be noted that every VM is configured with a name, disk image, network settings, and is allocated a CPU and memory. But this might be cumbersome if the VMs are many in number. The process can be simplified by configuring similar VMs with pre-edited profiles. Finally, the deployment principle should be able to fulfil the VM requirement to balance the workloads.

10.9 Live VM Migration Steps: Normally in a cluster built with mixed modes of host and guest systems, the procedure is to run everything on the physical machine. When a VM fails, it can be replaced by another VM on a different node, as long as they both run the same guest OS. This is called a **failover** (a procedure by which a system automatically transfers control to a duplicate system when it detects a fault or failure) of a physical system to a VM. Compared to a physical-physical failover, this methodology has more flexibility. It also has a drawback – a VM must stop working if its host node fails. This can be lessened by migrating from one node to another for a similar VM. The live migration process is depicted in Figure 3.20 [1].

10.10 Managing a Virtual Cluster: There exist four ways.

(a) We can use a **guest-based manager**, by which the cluster manager resides inside a guest OS. Ex: A Linux cluster can run different guest operating systems on top of the Xen hypervisor.

(b) We can bring out a **host-based manager** which itself is a cluster manager on the host systems. Ex: VMware HA (High Availability) system that can restart a guest system after failure.

(c) An **independent cluster manager**, which can be used on both the host and the guest – making the infrastructure complex.

(d) Finally, we might also use an **integrated cluster (manager)**, on the guest and host operating systems; here the manager must clearly distinguish between physical and virtual resources.

NOTE: The virtual cluster management schemes are greatly enhanced if the VM life migration is enabled with minimum overhead.

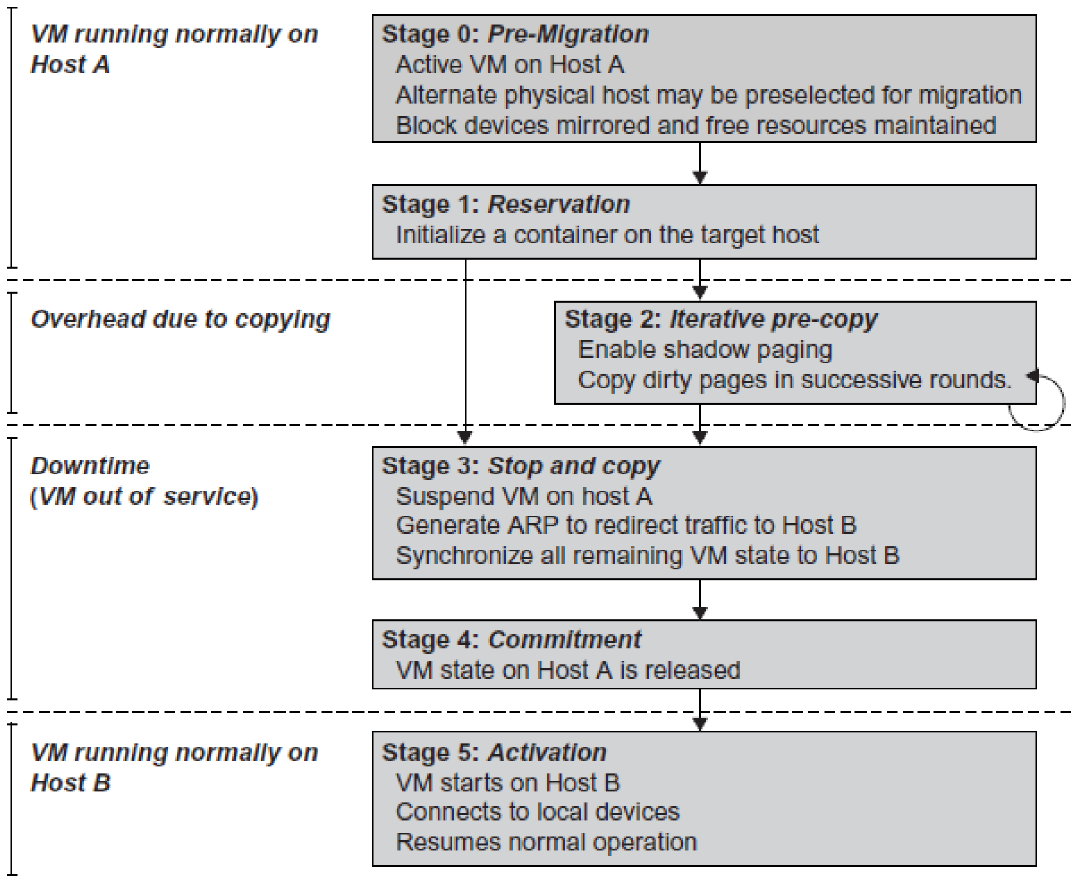

VM running normally on Host A
Stage 0: Pre-Migration
Active VM on Host A
Alternate physical host may be preselected for migration
Block devices mirrored and free resources maintained
Stage 1: Reservation
Initialize a container on the target host
Overhead due to copying
Stage 2: Iterative pre-copy
Enable shadow paging
Copy dirty pages in successive rounds.
Downtime
(VM out of service)
Stage 3: Stop and copy
Suspend VM on host A
Generate ARP to redirect traffic to Host B
Synchronize all remaining VM state to Host B
Stage 4: Commitment
VM state on Host A is released
VM running normally on Host B
Stage 5: Activation
VM starts on Host B
Connects to local devices
Resumes normal operation

FIGURE 3.20

Live migration process of a VM from one host to another.

10.11 Virtual clusters are generally used where fault tolerance of VMs on the host plays an important role in the total cluster strategy. These clusters can be applied in grids, clouds and HPC platforms. The HPC is obtained by dynamical finding and usage of resources as per requirement, and less migration time & bandwidth that is used.

10.12 A VM can be in one of the following states:

(a) **Inactive State**: This is defined by the VZ platform, under which the VM is **not** enabled.

(b) **Active State**: This refers to a VM that has been instantiated at the VZ platform to perform a task.
(c) **Paused State**: A VM has been instantiated but disabled temporarily to process a task or is in a waiting state itself.
(d) **Suspended State**: A VM enters this state if its machine file and virtual resources are stored back to the disk.

10.13 Live Migration Steps: This consists of 6 steps.

(a) Steps 0 and 1: Start migration automatically and checkout load balances and server consolidation.
(b) Step 2: Transfer memory (transfer the memory data + recopy any data that is changed during the process). This goes on iteratively till changed memory is small enough to be handled directly.
(c) Step 3: Suspend the VM and copy the last portion of the data.
(d) Steps 4 and 5: Commit and activate the new host. Here, all the data is recovered, and the VM is started from exactly the place where it was suspended, but on the new host.

10.14 Virtual Clusters are being widely used to use the computing resources effectively, generate HP, overcome the burden of interaction between different OSs and make different configurations to coexist.

10.15 Memory Migration: This is done between the physical host and any other physical/virtual machine. The techniques used here depend upon the guest OS. MM can be in a range of megabytes to gigabytes. The Internet Suspend-Resume (ISR) technique exploits temporal locality since the memory states are may have overlaps in the suspended/resumed instances of a VM. **Temporal locality (TL)** refers to the fact that the memory states differ only by the amount of work done since a VM was last suspended.

To utilize the TL, each file is represented as a tree of small sub-files. A copy of this tree exists in both the running and suspended instances of the VM. The advantage here is usage of tree representation of a file and caching ensures that the changed files are only utilized for transmission.

10.16 File System Migration: To support VM migration from one cluster to another, a consistent and location-dependent view of the file system is available on all hosts. Each VM is provided with its own virtual disk to which the file system is mapped to. The contents of the VM can be transmitted across the cluster by inter-connections (mapping) between the hosts. But migration of an entire host (if required) is not advisable due to cost and security problems. We can also provide a global file system across all host machines where a VM can be located. This methodology removes the need of copying files from one machine to another – all files on all machines can be accessed through network.

It should be noted here that the actual files are not mapped or copied. The VMM accesses only the local file system of a machine and the original/modified files are stored at their respective systems only. This decoupling improves security and performance but increases the overhead of the VMM – every file has to be stored in virtual disks in its local files.

10.17 Smart Copying ensures that after being resumed from suspension state, a VM doesn't get a whole file as a backup. It receives only the changes that were made. This technique reduces the amount of data that has to be moved between two locations.

10.18 Network Migration: A migrating should maintain open network connections. It should not depend upon forwarding mechanisms (mediators) or mobile mechanisms. Each VM should be assigned a unique IP or MAC (Media Access Control) [7] addresses which is different from that of the host machine. The mapping of the IP and MAC addresses to their respective VMs is done by the VMM.

If the destination of the VM is also on the same LAN, special messages are sent using MAC address that the IP address of the VM has moved to a new location. If the destination is on another network, the migrating OS can keep its original Ethernet MAC address and depend on the network switch [9] to detect its move to a new port [8].

10.19 Note that **live migration means** moving a VM from one physical node to another while keeping its OS environment and apps intact. All this process is carried out by a program called **migration daemon**. This capability provides efficient online system maintenance, reconfiguration, load balancing, and improved fault tolerance. The recently improved mechanisms are able to migrate without suspending the concerned VM.

10.20 There are two **approaches in live migration**: pre copy and post copy.

(a) In **pre copy**, which is manly used in live migration, all memory pages are first transferred; it then copies the modified pages in the last round iteratively. Here, performance 'degradation' will occur because migration will be encountering dirty pages (pages that change during networking) [10] all around in the network before getting to the right destination. The iterations could also increase, causing another problem. To encounter these problems, check-pointing/recovery process is used at different positions to take care of the above problems and increase the performance.

(b) In **post-copy**, all memory pages are transferred only once during the migration process. The threshold time allocated for migration is reduced. But the downtime is higher than that in pre-copy.

NOTE: **Downtime** means the time in which a system is out of action or can't handle other works.

Ex: Live migration between two Xen-enabled hosts: Figure 3.22 [1]
CBC Compression=> Context Based Compression
RDMA=> Remote Direct memory Access

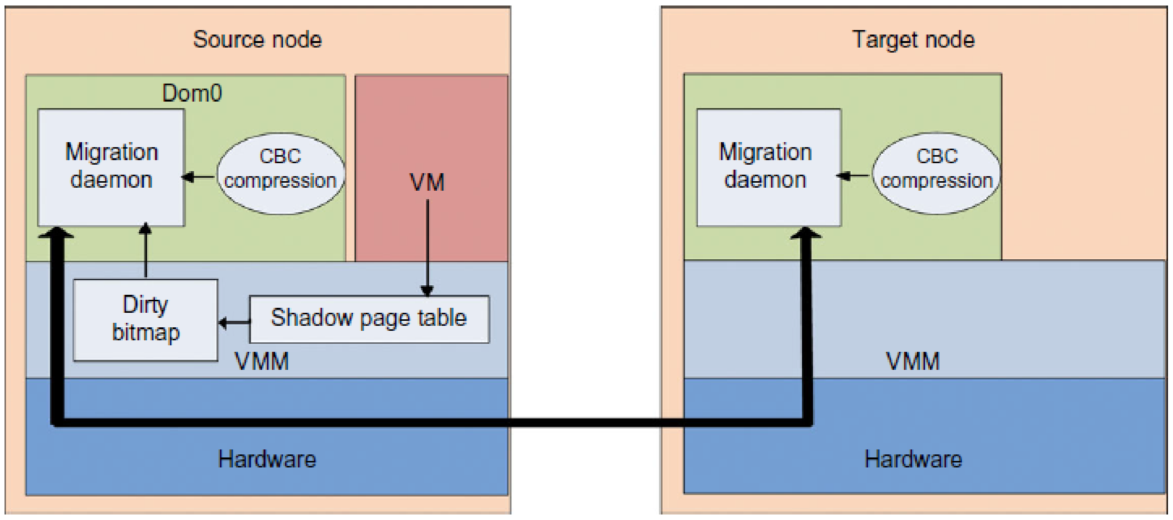


FIGURE 3.22

Live migration of VM from the Dom0 domain to a Xen-enabled target host.

11. **VZ for Data Centre Automation:** Data Centres have been built and automated recently by different companies like Google, MS, IBM, Apple etc. By utilizing the data centres and the data in the same, VZ is moving towards mobility, reduced maintenance time, and increasing the number of virtual clients. Other factors that influence the deployment and usage of data centres are high availability (HA), backup services, and workload balancing.

11.1 **Server Consolidation in Data Centres**: In data centers, heterogeneous workloads may run at different times. The two types here are

(a) **Chatty (Interactive) Workloads**: These types may reach the peak at a particular time and may be silent at some other time. Ex: WhatsApp in the evening and the same at midday.

(b) **Non-Interactive Workloads**: These don't require any users' efforts to make progress after they have been submitted. Ex: HPC

The data center should be able to handle the workload with satisfactory performance both at the peak and normal levels.

It is common that much of the resources of data centers like hardware, space, power and cost are under-utilized at various levels and times. To come out of this disadvantage, one approach is to use the methodology of **server consolidation**. This improves the server utility ratio of hardware devices by reducing the number of physical servers. There exist two types of server consolidation: (a) **Centralised and Physical Consolidation** (b) **VZ based server consolidation**. The second method is widely used these days, and it has some advantages.

- Consolidation increases hardware utilization
- It enables more agile provisioning of the available resources
- The total cost of owning and using data center is reduced (low maintenance, low cooling, low cabling etc.)
- It enables availability and business continuity – the crash of a guest OS has no effect upon a host OS.

11.2 **NOTE**: To automate (VZ) data centers one must **consider several factors** like resource scheduling, power management, performance of analytical models and so on. This improves the utilization in data centers and gives high performance. Scheduling and reallocation can be done at different levels at VM level, server level and data center level, but generally **any one (or two) level is used** at a time.

The schemes that can be considered are:

(a) **Dynamic CPU allocation**: This is based on **VM utilization** and app level **QoS** [11] (Quality of Service) metrics. The CPU should adjust automatically according to the demands and workloads to deliver the best performance possible.

(b) **Another scheme** uses two-level resource management system to handle the complexity of the requests and allocations. The resources are allocated automatically and autonomously to bring down the workload on each server of a data center.

Finally, we should efficiently balance the power saving and data center performance to achieve the HP and HT also at different situations as they demand.

11.3 **Virtual Storage Management**: VZ is mainly lagging behind the modernisation of data centers and is the bottleneck of VM deployment. The CPUs are rarely updates, the chips are not replaced and the host/guest operating systems are not adjusted as per the demands of situation.

Also, the storage methodologies used by the VMs are not as fast as they are expected to be (nimble). Thousands of such VMs may flood the data center and their lakhs of images (SSI) may lead to data center collapse. Research has been conducted for this purpose to bring out an efficient storage and reduce the size of images by storing parts of them at different locations. The solution here is Content Addressable Storage (CAS). Ex: Parallax system architecture (A distributed storage system). This can be viewed at Figure 3.26 [1], P25.

11.4 Note that Parallax itself runs as a user-level application in the VM storage, providing Virtual Disk Images (VDIs). A **VDI** can accessed in a transparent manner from any host machine in the Parallax cluster. It is a core abstraction of the storage methodology used by Parallax.

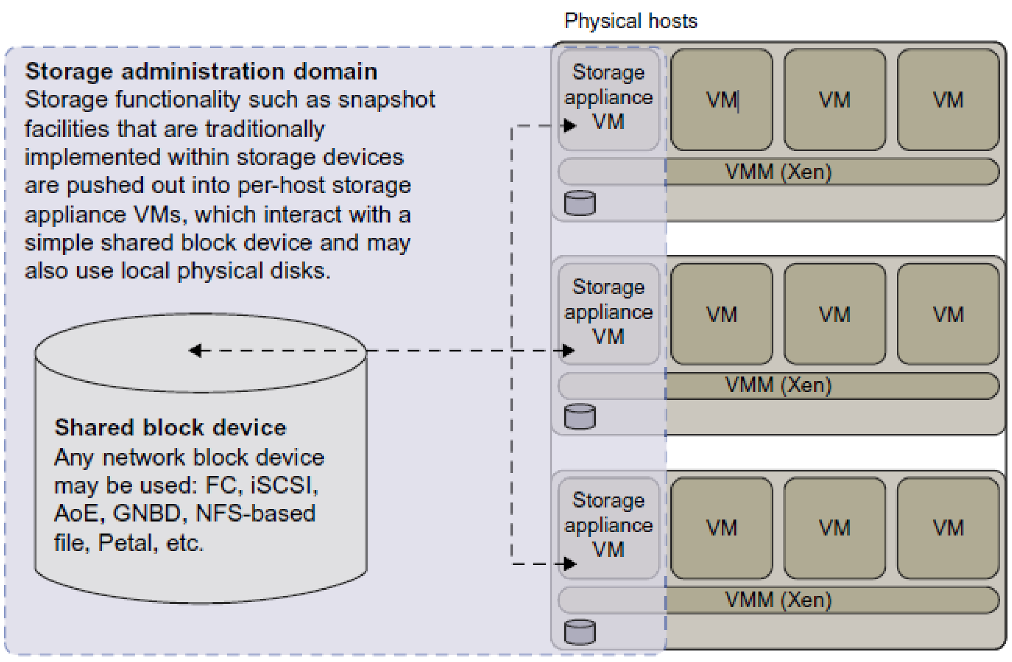

Physical hosts
Storage administration domain
Storage functionality such as snapshot facilities that are traditionally implemented within storage devices are pushed out into per-host storage appliance VMs, which interact with a simple shared block device and may also use local physical disks.
Shared block device
Any network block device may be used: FC, iSCSI, AoE, GNBD, NFS-based file, Petal, etc.
Storage appliance VM
VM
VM
VM
VMM (Xen)
Storage appliance VM
VM
VM
VM
VMM (Xen)
Storage appliance VM
VM
VM
VM
VMM (Xen)

FIGURE 3.26

Parallax is a set of per-host storage appliances that share access to a common block device and presents virtual disks to client VMs.

11.5 **Cloud OS for VZ Data Centers**: VI => Virtual Infrastructure managers Types can be seen in Table 3.6 [1].

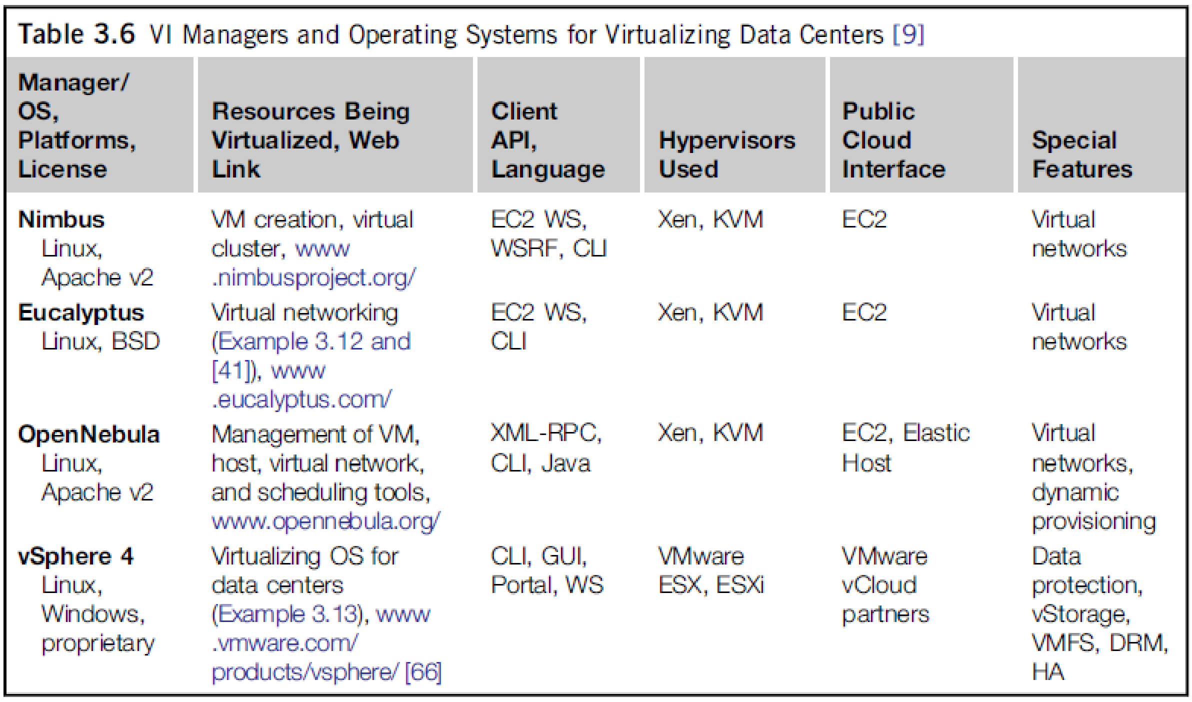


Table 3.6 VI Managers and Operating Systems for Virtualizing Data Centers [9]

Manager/ OS, Platforms, License	Resources Being Virtualized, Web Link	Client API, Language	Hypervisors Used	Public Cloud Interface	Special Features
Nimbus Linux, Apache v2	VM creation, virtual cluster, www .nimbusproject.org/	EC2 WS, WSRF, CLI	Xen, KVM	EC2	Virtual networks
Eucalyptus Linux, BSD	Virtual networking (Example 3.12 and [41]), www .eucalyptus.com/	EC2 WS, CLI	Xen, KVM	EC2	Virtual networks
OpenNebula Linux, Apache v2	Management of VM, host, virtual network, and scheduling tools, www.opennebula.org/	XML-RPC, CLI, Java	Xen, KVM	EC2, Elastic Host	Virtual networks, dynamic provisioning
vSphere 4 Linux, Windows, proprietary	Virtualizing OS for data centers (Example 3.13), www .vmware.com/ products/vsphere/ [66]	CLI, GUI, Portal, WS	VMware ESX, ESXi	VMware vCloud partners	Data protection, vStorage, VMFS, DRM, HA

11.6 EC2 => Amazon Elastic Compute Cloud
WS => Web Service
CLI => Command Line Interface
WSRF => Web Services Resource Framework
KVM => Kernel-based VM
VMFS => VM File System
HA => High Availability

11.7 Example of Eucalyptus for Virtual Networking of Private Cloud: It is an open-source software system intended for IaaS clouds. This is seen in Figure 3.27 [1].

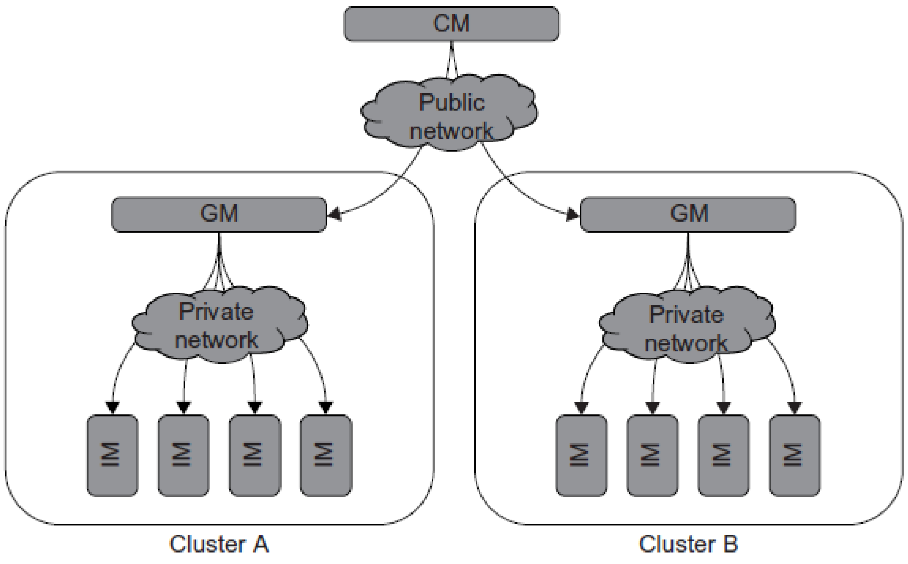


FIGURE 3.27

Eucalyptus for building private clouds by establishing virtual networks over the VMs linking through Ethernet and the Internet.

11.8 Instance Manager (IM): It controls execution, inspection and terminating of VM instances on the host machines where it runs.

Group Manager (GM): It gathers information about VM execution and schedules them on specific IMs; it also manages virtual instance network.

Cloud Manager (CM): It is an entry-point into the cloud for both users and administrators. It gathers information about the resources, allocates them by proper scheduling, and implements them through the GMs.

11.9 Trust Management in VZ Data Centers: As a recollect, a VMM (hypervisor) is a layer between the host OS and the hardware to create 1 or more VMs on a single platform. A VM encapsulates the guest OS and its current state and can transport it through the network as a SSI. At this juncture, in the network transportation, any intruders may get into the image or the concerned hypervisor itself and pose danger to both the image and the host system. Ex: A subtle problem lies in reusing a random number for cryptography.

11.10 VM-based Intrusion Detection: **Intrusions** are unauthorized access to a computer from other network users. An **intrusion detection system** (IDS), which is built on the host OS can be divided into two types: host-based IDS (HIDS) and a network-based IDS (NIDS).

VZ based IDS can isolate each VM on the VMM and work upon the concerned systems without having contacts with the other. Any problem with a VM will not pose problems for other VMs. Also, a VMM audits the hardware allocation and usage for the VMs regularly so as to notice any abnormal changes. Still yet, the host and guest OS are fully isolated from each other. A methodology on these bases can be noticed in Figure 3.29 [1].

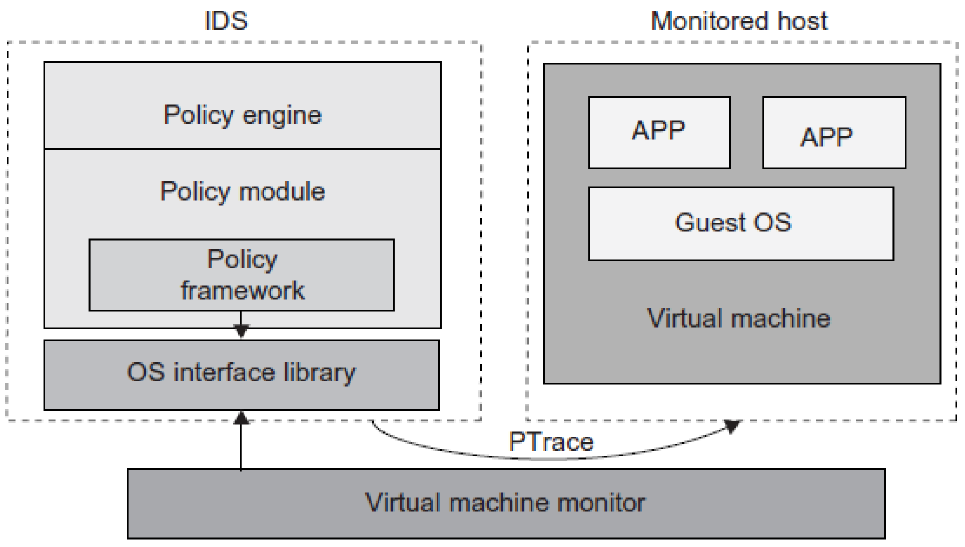


FIGURE 3.29

The architecture of livewire for intrusion detection using a dedicated VM.

The above figure proposes the concept of granting IDS runs only on a highly-privileged VM. Notice that policies play an important role here. A **policy framework** can monitor the events in different guest operating systems of different VMs by using an OS interface library to determine which grant is secure and which is not.

It is difficult to determine which access is intrusion and which is not without some time delay. Systems also may use access 'logs' to analyze which is an intrusion and which is secure. The IDS log service is based on the OS kernel and the UNIX kernel is hard to break; so even if a host machine is taken over by the hackers, the IDS log book remains unaffected.

The security problems of the cloud mainly arise in the transport of the images through the network from one location to another. The VMM must be used more effectively and efficiently to deny any chances for the hackers.

****

References

- Kai Hwang et al, Distributed and Cloud Computing – From Parallel Processing to the Internet of Things, Morgan Kaufmann, Elsevier, 2012.
- https://en.wikipedia.org/wiki/Multiplexing
- http://cloudacademy.com/blog/container-virtualization/
- https://en.wikipedia.org/wiki/Library_(computing)#Shared_libraries
- https://en.wikipedia.org/wiki/Binary_translation
- https://pubs.vmware.com/vsphere-51/topic/com.vmware.vsphere.resmgmt.doc/GUID-C25A8823-F595-4322-BD0D-4FD5B081F877.html
- http://searchnetworking.techtarget.com/definition/MAC-address
- http://searchnetworking.techtarget.com/definition/port
- https://en.wikipedia.org/wiki/Network_switch
- https://en.wikipedia.org/wiki/Live_migration
- http://searchunifiedcommunications.techtarget.com/definition/QoS-Quality-of-Service