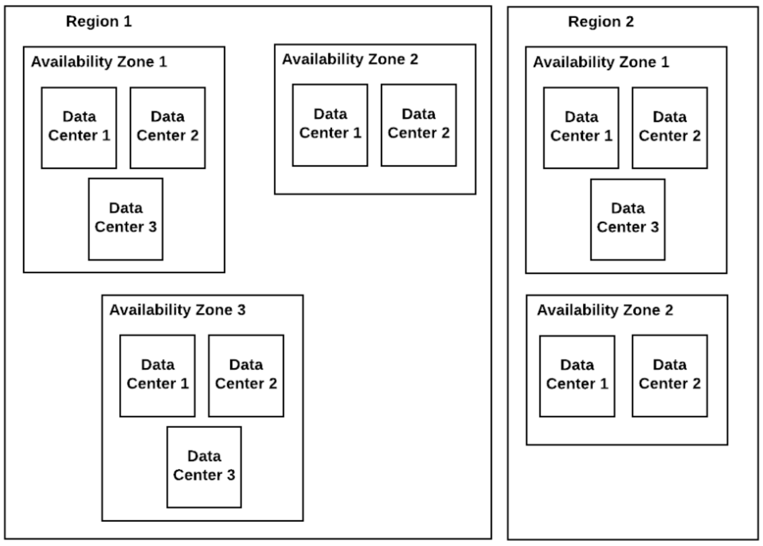
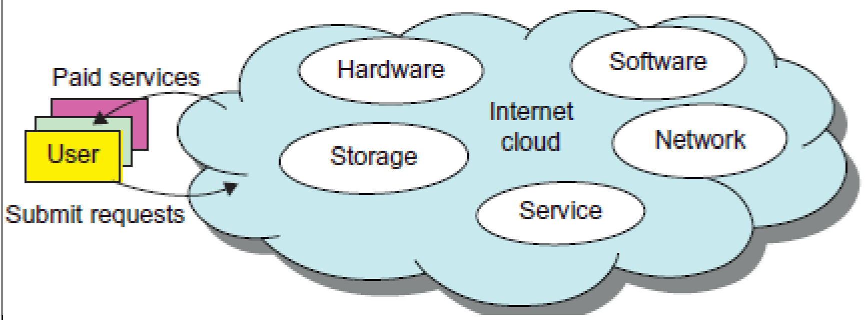
USN

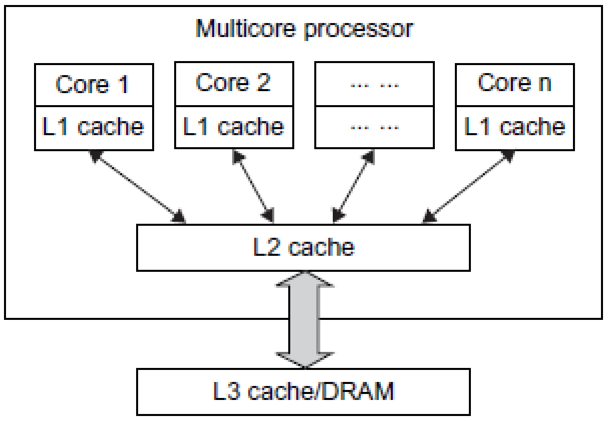
Internal Assessment Test 1 – March 2025

Sub:	Cloud Computing				Sub Code:	BCS601	Branch:	CSE
Date:	25.03.2025	Duration:	90 mins	Max Marks: 50	Sem / Sec:	6 A,B,C		OBE

Answer any FIVE FULL Questions		MARKS	CO	RBT
1 (a)	Define the following : i)HPC ii)HTC iii)Amdahl's law iv) Utility Computing i) HPC (High Performance Computing) that has massive computing power to solve problems that require speed and performance. They are made from homogeneous nodes with centralized control and are the technology for clusters. Their speed is measured in floating point operations per second (flops) and they are used in scientific modeling and simulations such as weather prediction or molecular dynamics. Eg. El Capitan is currently the world's fastest supercomputer with it's performance exceeding 2 exaflops ii) HTC (High throughput Computing) – focuses on high-flux computing that aims at more number of tasks completed per unit time rather than speed. They are usually built from disparate nodes and have distributed control. HTC technology is used in several batch processing jobs and address several problems such as cost, security, reliability, etc in enterprise servers. iii) Amdahl's law: Amdahl's Law states that the speedup factor of using the n-processor system over the use of a single processor is expressed by: $Speedup = S = \frac{T}{\alpha T + \frac{(1-\alpha)T}{N}} = \frac{1}{\alpha + \frac{(1-\alpha)}{N}}$ N : The number of processors α : The fraction of the program that is sequential (1-α): Portion of program that is parallelizable T : Total execution time The maximum speedup of n is achieved only if the sequential bottleneck α is reduced to zero or the code is fully parallelizable with α = 0. As the cluster becomes sufficiently large, that is, n →∞, S approaches 1/α, an upper bound on the speedup S. Amdahl's law teaches us that we should make the sequential bottleneck as small as possible. iv) Utility Computing : A business model wherein the customers receive computing from a paid service provider just like gas or electricity. It is also referred to as the 'pay-as-you-go' model. In a cloud model, users avail resources such as virtual machines (VM) or storage and they pay for the capacity and utilization levels. Eg. VM's are billed based on the configuration and usage per hour basis. Storage is billed based on the capacity, access frequency and number of outbound transfers.	4M	CO1	L2

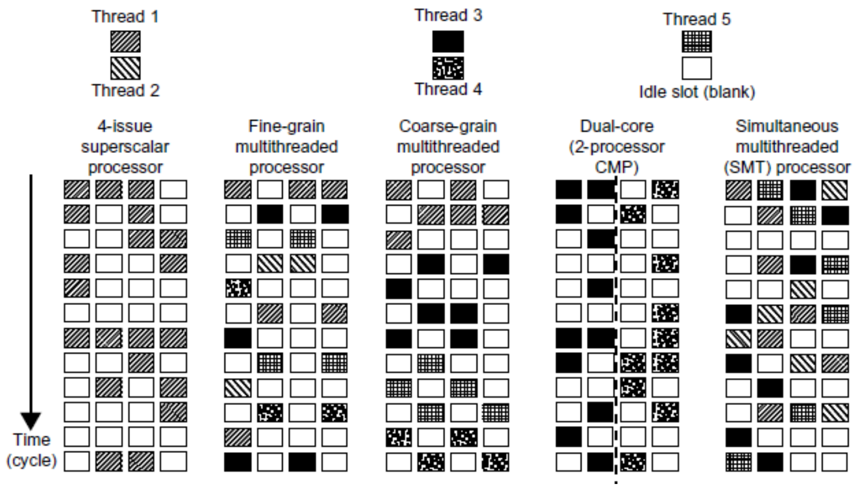
(b)	 Region 1 Availability Zone 1 Data Center 1 Data Center 2 Data Center 3 Availability Zone 2 Data Center 1 Data Center 2 Availability Zone 3 Data Center 1 Data Center 2 Data Center 3 Region 2 Availability Zone 1 Data Center 1 Data Center 2 Data Center 3 Availability Zone 2 Data Center 1 Data Center 2 Above is a diagram for regions and availability zone. Explain with respect to AWS or GCP, the following terms i) Datacenter ii)Region iii)Availability Zone iv)VM **Datacenter** : A facility that is built using a large number of servers and huge interconnection networks. They are built on economies of scale and provide services for businesses, research and governments. They are built with both HPC and HTC systems to provide good performance for compute-intensive workloads and high throughput for data-intensive workloads. Eg. Google has around 42 regions globally in over 200 countries and territories and 2 datacenters in Mumbai and Delhi in India. AWS has around 36 regions in over 200 countries with datacenters in Hyderabad and Mumbai in India. ii)**Regions** are independent geographic areas that consist of zones. A region consists of 3 or more zones housed in three or more physical datacenters. Eg. In GCP, us-central1, us-east1, india-south1 are regions iii)**Availability Zone :** a deployment area for resources within a region and should be considered a single failure domain within a region To deploy fault-tolerant applications with high availability and help protect against unexpected failures, deploy your applications across multiple zones in a region Eg. In GCP, us-central1a, us-central1b, us-central1c are various zones in the region us-central1 iv) **VM** – A virtual machine is a virtualized environment of a computer system running a guest OS. VM's can be deployed in a particular region and zone by choosing a pre-defined configuration or a custom configuration. They can also be deployed by using templates. The VM instances are broadly classified into general-purpose, compute-optimized, memory-optimized, GPU accelerated. Eg. In GCP, E2 machines are used for cost-effective general purpose workloads. An e2-standard-4 has 4vCPu's and 16 GB RAM.	4M	CO1	L2
c)	System availability = MTTF/ (MTTF+MTTR) Explain under which condition will a system be highly available using the formula given above. **MTTF** (Mean Time to Failure) and **MTTR** (Mean Time to Repair) are both parameters that determine the availability. As system will be **highly available if if has a long MTTF and a short MTTR**. Eg. if a system has MTTF = 980 hours and MTTR = 20 hours	2M	CO1	L2

	System availability = 980/(980+20) = 98% availability.			
2 (a)	Describe the main characteristics of cloud computing. Define cloud computing.  Paid services User Submit requests Hardware Software Internet cloud Storage Network Service • On demand Provisioning • 3 Offerings - IaaS, PaaS, SaaS ○ IaaS - servers, storage, networks, and the data center fabric ○ PaaS - middleware, databases, development tools, and some runtime support such as Web 2.0 and Java ○ SaaS - model applies to business processes, industry applications, consumer relationship management (CRM), enterprise resources planning (ERP), human resources (HR), and collaborative applications • 4 deployment models – Public, private, managed, hybrid – each with different levels of security and SLA's • Pay-as-you go model – based on usage and subscription models for SaaS • Scalability – ability to increase resources as the business grows • Security & Reliability – Security options and disaster recovery options • Service and data discovery, content/service distribution – to build applications from existing microservices, reduce latency in applications globally. The **IBM definition of cloud** : A cloud is a pool of virtualized computer resources. A cloud can host a variety of different workloads, including batch-style backend jobs and interactive and user-facing applications OR **NIST Definition** : A model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction.	4M	CO1	L2
(b)	Discuss any one out of each of the enabling technologies, system models and software models in detail i) Technologies for Network based systems (Multicore/multithreading systems, VM, GPU computing) (Any 1) **Multicore, Multithreading systems**	6M	CO1	L2



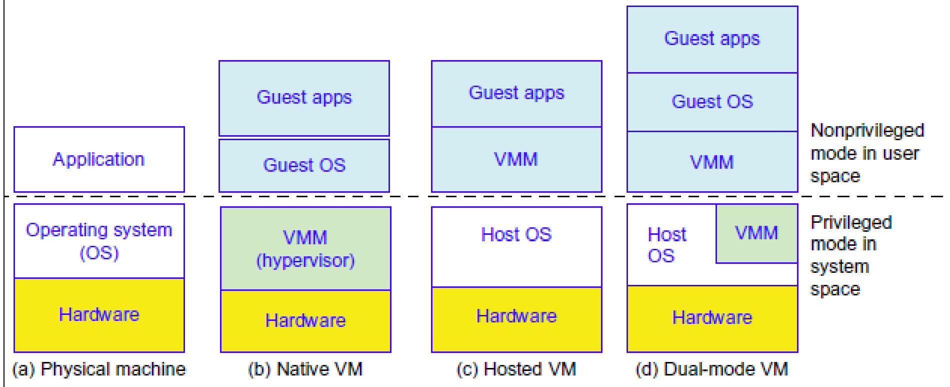
Advanced CPUs or microprocessor chips assume a multicore architecture with dual, quad, six, or more processing cores.
These processors exploit parallelism at ILP and TLP levels. The clock rate reached its limit on CMOS-based chips due to power limitations.
The ILP is highly exploited in modern CPU processors. ILP mechanisms include multiple-issue superscalar architecture, dynamic branch prediction, and speculative execution, among others
Each core is essentially a processor with its own private cache (L1 cache). Multiple cores are
housed in the same chip with an L2 cache that is shared by all cores

Multi-threading


Thread 1
Thread 2
Thread 3
Thread 4
Thread 5
Idle slot (blank)
4-issue superscalar processor
Fine-grain multithreaded processor
Coarse-grain multithreaded processor
Dual-core (2-processor CMP)
Simultaneous multithreaded (SMT) processor
Time (cycle)

Both multi-core CPU and many-core GPU processors can handle multiple instruction threads at
different magnitudes,
Fine-grain multithreading switches the execution of instructions from different threads per cycle. Course-grain multithreading executes many instructions from the same thread for quite a few cycles before switching to another thread.
The multicore CMP executes instructions from different threads completely.
The SMT allows simultaneous scheduling of instructions from different threads in the same cycle.

VM :

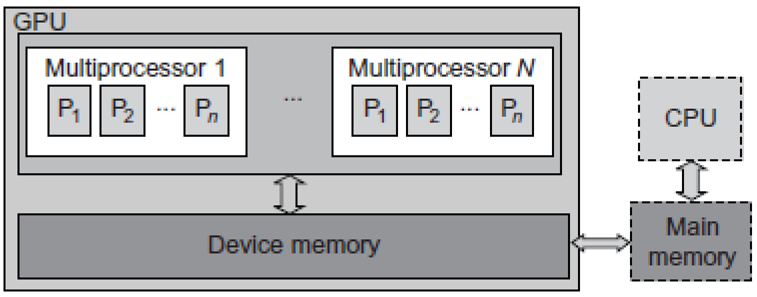


the host machine is equipped with the physical hardware, as shown at the bottom of the figure. An example is an x-86 architecture desktop running its installed Windows OS, as shown in part (a) of the figure.
The VM can be provisioned for any hardware system.
The VM is built with virtual resources managed by a guest OS to run a specific application. Between the VMs and the host platform, one needs to deploy a middleware layer called a virtual machine monitor (VMM).
Native VM installed with the use of a VMM called a hypervisor in privileged mode. For example, the hardware has x-86 architecture running the Windows system.
The guest OS could be a Linux system and the hypervisor is the XEN system developed at Cambridge University. This hypervisor approach is also called bare-metal VM, because the hypervisor handles the bare hardware (CPU, memory, and I/O) directly.
Host VM : VMM runs in nonprivileged mode. The host OS need not be modified.
Dual Mode : Part of the VMM runs at the user level and another part runs at the supervisor level. In this case, the host OS may have to be modified to some extent. Multiple VMs can be ported to a given hardware system to support the virtualization process.

GPU computing:

- GPU is a graphics coprocessor or accelerator mounted on a computer’s graphics card or video card.
- A GPU offloads the CPU from tedious graphics tasks in video editing applications. For example, the Xeon X5670 CPU has six cores.
- However, a modern GPU chip can be built with hundreds of processing cores.
- GPUs have a throughput architecture that exploits massive parallelism by executing many concurrent threads slowly, instead of executing a single long thread in a conventional microprocessor very quickly.
- NVIDIA’s CUDA model was for HPC using GPGPUs.

Modern GPUs are not restricted to accelerated graphics or video coding. They are used in HPC systems to power supercomputers with massive parallelism at multicore and multithreading levels.
GPUs are designed to handle large numbers of floating-point operations in parallel.
In a way, the GPU offloads the CPU from all data-intensive calculations, not just those that are related to video processing. Conventional



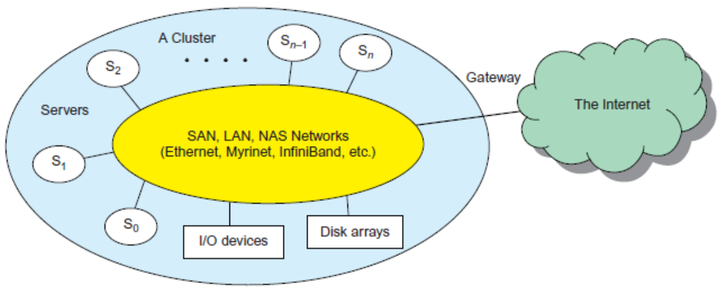
The GPU has a many-core architecture that has hundreds of simple processing cores organized as multiprocessors. Each core can have one or more threads. Essentially, the

CPU's floating-point kernel computation role is largely offloaded to the many-core GPU. The CPU instructs the GPU to perform massive data processing.

The bandwidth must be matched between the on-board main memory and the on-chip GPU memory

ii) System Models (Cluster, Grid, P2P networks)(Any 1)

Clusters



A typical server cluster built around a low-latency, high bandwidth interconnection network. To build a larger cluster with more nodes, the interconnection network can be built with multiple levels of Gigabit Ethernet, Myrinet, or InfiniBand switches.
Through hierarchical construction using a SAN, LAN, or WAN, one can build scalable clusters with an increasing number of nodes.
The cluster is connected to the Internet via a virtual private network (VPN) gateway. The gateway IP address locates the cluster. The system image of a computer is decided by the way the OS manages the shared cluster resources. Most clusters have loosely coupled node computers. All resources of a server node are managed by their own OS. Thus, most clusters have multiple system images as a result of having many autonomous nodes under different OS control.
Cluster designers desire a cluster operating system or some middleware to support SSI at various levels, including the sharing of CPUs, memory, and I/O across all cluster nodes.
Special cluster middleware supports are needed to create SSI or high availability (HA). Both sequential and parallel applications can run on the cluster, and special parallel environments are needed to facilitate use of the cluster resources.
Issues :
A cluster-wide OS for complete resource sharing is not available yet. Middleware or OS extensions were developed at the user space to achieve SSI at selected functional levels.
Without this middleware, cluster nodes cannot work together effectively to achieve cooperative computing. The software environments and applications must rely on the middleware to achieve high performance.
The cluster benefits come from scalable performance, efficient message passing, high system availability, seamless fault tolerance, and cluster-wide job management

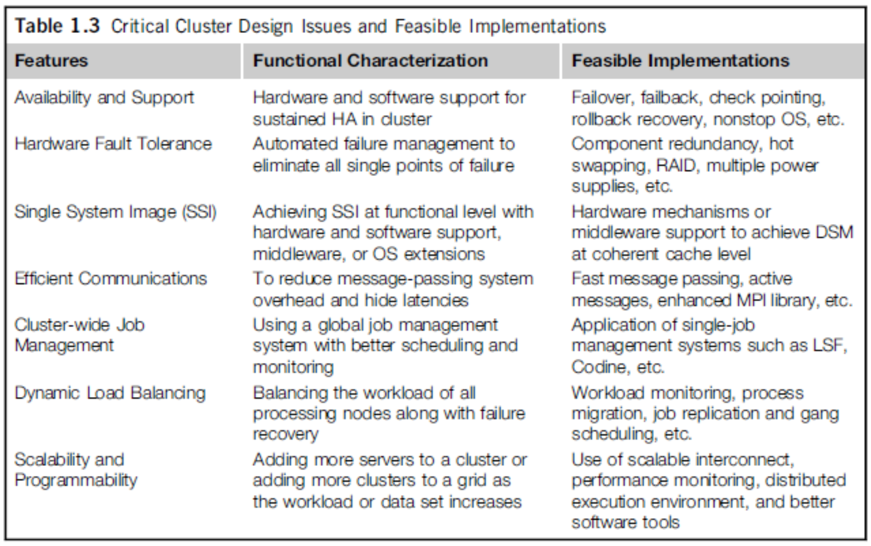
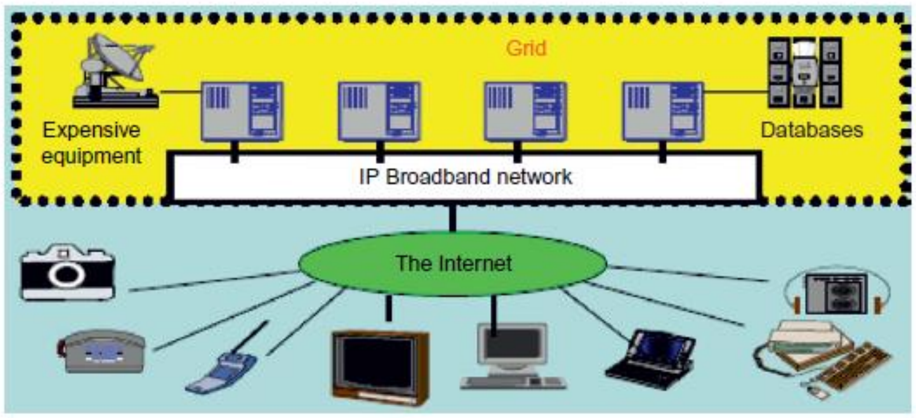


Table 1.3 Critical Cluster Design Issues and Feasible Implementations

Features	Functional Characterization	Feasible Implementations
Availability and Support	Hardware and software support for sustained HA in cluster	Failover, failback, check pointing, rollback recovery, nonstop OS, etc.
Hardware Fault Tolerance	Automated failure management to eliminate all single points of failure	Component redundancy, hot swapping, RAID, multiple power supplies, etc.
Single System Image (SSI)	Achieving SSI at functional level with hardware and software support, middleware, or OS extensions	Hardware mechanisms or middleware support to achieve DSM at coherent cache level
Efficient Communications	To reduce message-passing system overhead and hide latencies	Fast message passing, active messages, enhanced MPI library, etc.
Cluster-wide Job Management	Using a global job management system with better scheduling and monitoring	Application of single-job management systems such as LSF, Codine, etc.
Dynamic Load Balancing	Balancing the workload of all processing nodes along with failure recovery	Workload monitoring, process migration, job replication and gang scheduling, etc.
Scalability and Programmability	Adding more servers to a cluster or adding more clusters to a grid as the workload or data set increases	Use of scalable interconnect, performance monitoring, distributed execution environment, and better software tools

Grid Computing



Grid technology demands new distributed computing models, software/middleware support, network protocols, and hardware infrastructures.
Grid computing is envisioned to allow close interaction among applications running on distant
computers simultaneously
Like an electric utility power grid, a computing grid offers an infrastructure that couples computers, software/middleware, special instruments, and people and sensors together. The grid is often constructed across LAN, WAN, or Internet backbone networks at a regional, national, or global scale.
Enterprises or organizations present grids as integrated computing resources. They can also be viewed as virtual platforms to support virtual organizations. The computers used in a grid are primarily workstations, servers, clusters, and supercomputers. Personal computers, laptops, and PDAs can be used as access devices to a grid system.
The grid is built across various IP broadband networks including LANs and WANs already used by enterprises or organizations over the Internet.
Special instruments may be involved such as using the radio telescope in SETI@Home search of life in the galaxy and the austrophysics@Swineburne for pulsars. At the server end, the grid is a network. At the client end, we see wired or wireless terminal devices.
Grid technology demands new distributed computing models, software/middleware support, network protocols, and hardware infrastructures. National grid projects are followed by industrial grid platform development by IBM, Microsoft, Sun, HP, Dell, Cisco, EMC, Platform Computing, and others.

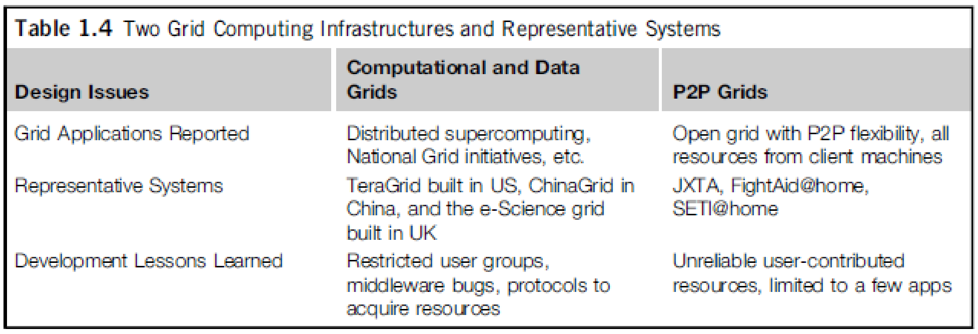
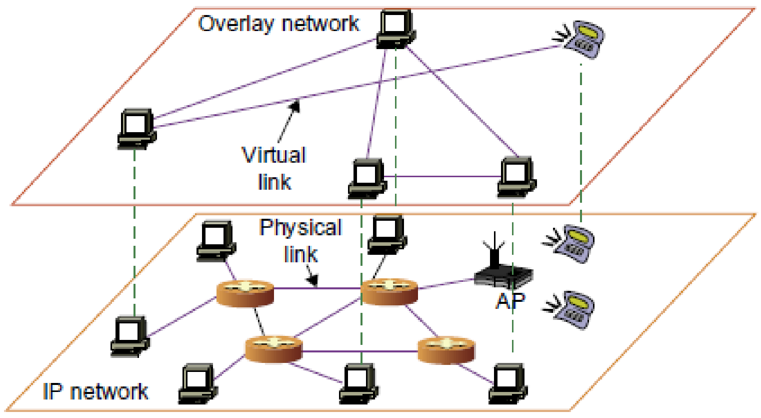


Table 1.4 Two Grid Computing Infrastructures and Representative Systems

Design Issues	Computational and Data Grids	P2P Grids
Grid Applications Reported	Distributed supercomputing, National Grid initiatives, etc.	Open grid with P2P flexibility, all resources from client machines
Representative Systems	TeraGrid built in US, ChinaGrid in China, and the e-Science grid built in UK	JXTA, FightAid@home, SETI@home
Development Lessons Learned	Restricted user groups, middleware bugs, protocols to acquire resources	Unreliable user-contributed resources, limited to a few apps

P2P Networks



An example of a well-established distributed system is the client-server architecture. In this scenario, client machines (PCs and workstations) are connected to a central server for compute, e-mail, file access, and database applications. The P2P architecture offers a distributed model of networked systems. First, a P2P network is client-oriented instead of server-oriented.

In a P2P system, every node acts as both a client and a server, providing part of the system resources. Peer machines are simply client computers connected to the Internet. All client machines act autonomously to join or leave the system freely. This implies that no master-slave relationship exists among the peers. No central coordination or central database is needed.
Initially, the peers are totally unrelated. Each peer machine joins or leaves the P2P network voluntarily. Only the participating peers form the physical network at any time. Unlike the cluster or grid, a P2P network does not use a dedicated interconnection network. The physical network is simply an ad hoc network formed at various Internet domains randomly using the TCP/IP and NAI protocols.
Data items or files are distributed in the participating peers. Based on communication or file-sharing needs, the peer IDs form an overlay network at the logical level. This overlay is a virtual network formed by mapping each physical machine with its ID, logically, through a virtual mapping

There are two types of overlay networks: unstructured and structured. An unstructured overlay network is characterized by a random graph. There is no fixed route to send messages or files among the nodes. Often, flooding is applied to send a query to all nodes in an unstructured overlay, thus resulting in heavy network traffic and nondeterministic search results. Structured overlay networks follow certain connectivity topology and rules for inserting and removing nodes (peer IDs) from the overlay graph. Routing mechanisms are developed to take advantage of the structured overlays.

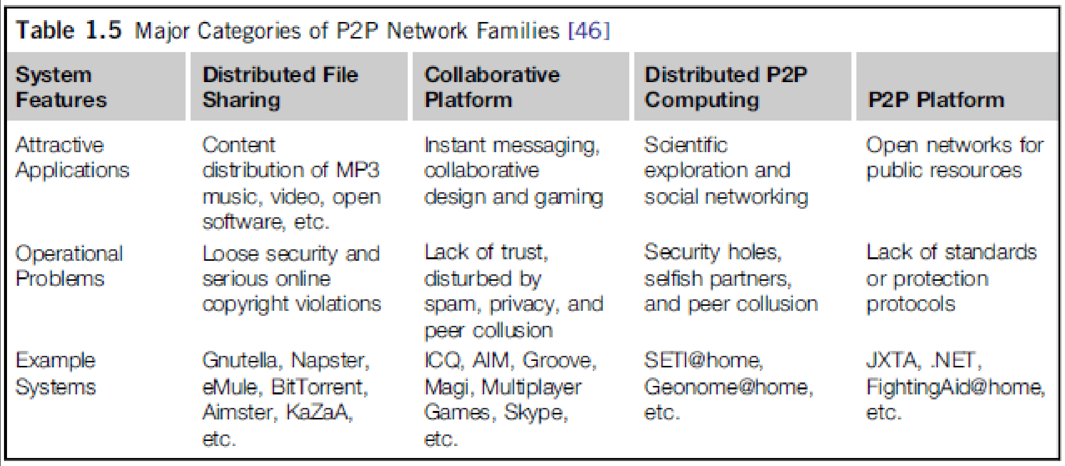


Table 1.5 Major Categories of P2P Network Families [46]

System Features	Distributed File Sharing	Collaborative Platform	Distributed P2P Computing	P2P Platform
Attractive Applications	Content distribution of MP3 music, video, open software, etc.	Instant messaging, collaborative design and gaming	Scientific exploration and social networking	Open networks for public resources
Operational Problems	Loose security and serious online copyright violations	Lack of trust, disturbed by spam, privacy, and peer collusion	Security holes, selfish partners, and peer collusion	Lack of standards or protection protocols
Example Systems	Gnutella, Napster, eMule, BitTorrent, Aimster, KaZaA, etc.	ICQ, AIM, Groove, Magi, Multiplayer Games, Skype, etc.	SETI@home, Geonome@home, etc.	JXTA, .NET, FightingAid@home, etc.

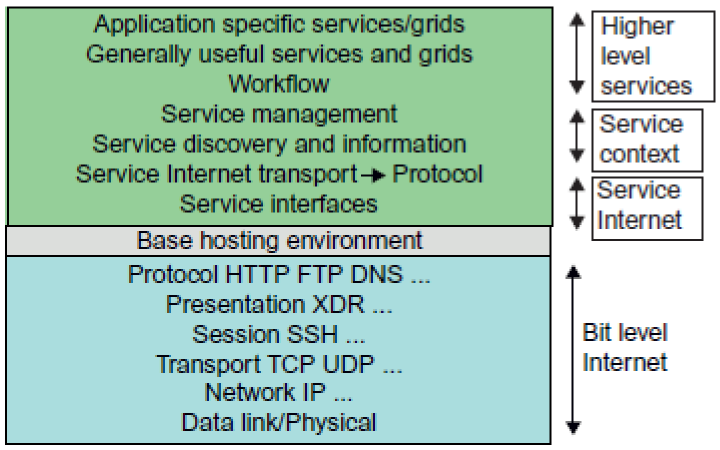
There are too many hardware models and architectures to select from; incompatibility exists between software and the OS; and different network connections and protocols make it too complex to apply in real applications. We need system scalability as the workload increases. System scaling is directly related to performance and bandwidth. P2P networks do have these properties. Data location is also important to affect collective performance.
Data locality, network proximity, and interoperability are three design objectives in distributed P2P applications P2P performance is affected by routing efficiency and self-organization by participating peers.

Fault tolerance, failure management, and load balancing are other important issues in using overlay networks. Lack of trust among peers poses another problem. By replicating data in multiple peers, one can easily lose data in failed nodes. On the other hand, disadvantages of P2P networks do exist. Because the system is not centralized, managing it is difficult.
In addition, the system lacks security. Anyone can log on to the system and cause damage or abuse.
Further, all client computers connected to a P2P network cannot be considered reliable or virus-free.

P2P networks are reliable for a small number of peer nodes. They are only useful for applications that require a low level of security and have no concern for data sensitivity.

iii) Software Models (SOA, Distributed OS, parallel and distributed programming model)(Any 1)

SOA



In grids/web services, Java, and CORBA, an entity is, respectively, a service, a Java object, and a CORBA distributed object in a variety of languages The entity interfaces correspond

to the Web Services Description Language (WSDL), Java method, and CORBA interface definition language (IDL) specifications in these example distributed systems.
These interfaces are linked with customized, high-level communication systems: SOAP, RMI, and IIOP in the three examples.
These communication systems support features including particular message patterns (such as Remote Procedure Call or RPC), fault recovery, and specialized routing.
Often, these communication systems are built on message-oriented middleware (enterprise bus) infrastructure such as Web- Sphere MQ or Java Message Service (JMS) which provide rich functionality and support virtualization of routing, senders, and recipients.
In the case of fault tolerance, the features in the Web Services Reliable Messaging (WSRM) framework mimic the OSI layer capability (as in TCP fault tolerance) modified to match the different abstractions (such as messages versus packets, virtualized addressing) at the entity levels.
Security is a critical capability that either uses or reimplements the capabilities seen in concepts such as Internet Protocol Security (IPsec) and secure sockets in the OSI layers.
Both web services and REST systems have very distinct approaches to building reliable interoperable systems. In web services, one aims to fully specify all aspects of the service and its environment. This specification is carried with communicated messages using Simple Object Access Protocol (SOAP).
The hosting environment then becomes a universal distributed operating system with fully distributed capability carried by SOAP messages. This approach has mixed success as it has been hard to agree on key parts of the protocol and even harder to efficiently implement the protocol by software such as Apache Axis.

In the REST approach, one adopts simplicity as the universal principle and delegates most of the difficult problems to application (implementation-specific) software. In a web services language,
REST has minimal information in the header, and the message body (that is opaque to generic message processing) carries all the needed information. REST architectures are clearly more appropriate for rapid technology environments.

service-oriented architecture (SOA) has evolved over the years. SOA applies to building grids, clouds, grids of clouds, clouds of grids, clouds of clouds (also known as interclouds), and systems of systems in general.

Distributed OS
Tanenbaum [26] identifies three approaches for distributing resource management functions in a distributed computer system.
The first approach is to build a network OS over a large number of heterogeneous OS platforms. Such an OS offers the lowest transparency to users, and is essentially a distributed file system, with independent computers relying on file sharing as a means of ommunication.
The second approach is to develop middleware to offer a limited degree of resource sharing, similar to the MOSIX/OS developed for clustered systems .
The third approach is to develop a truly distributed OS to achieve higher use or system transparency.

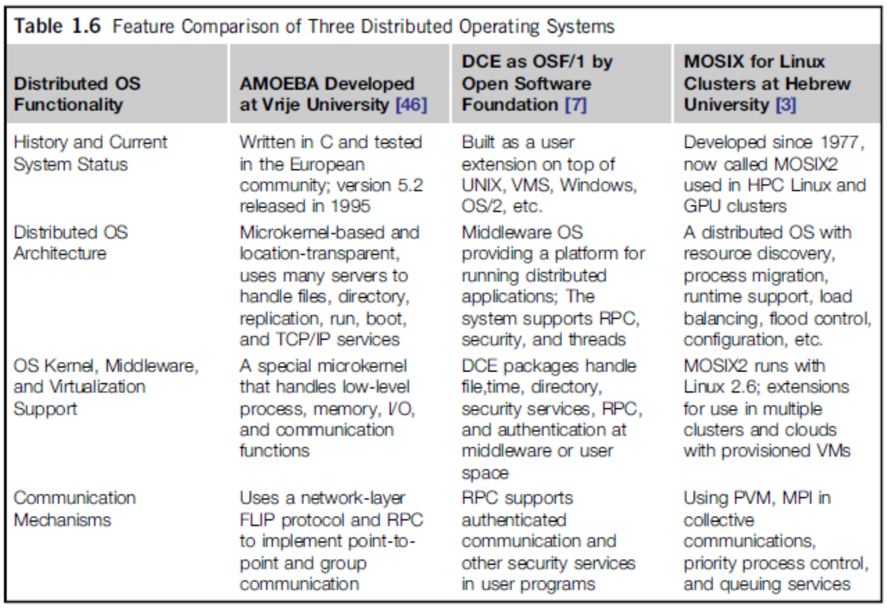


Table 1.6 Feature Comparison of Three Distributed Operating Systems

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

Parallel and Distributed Programming Model

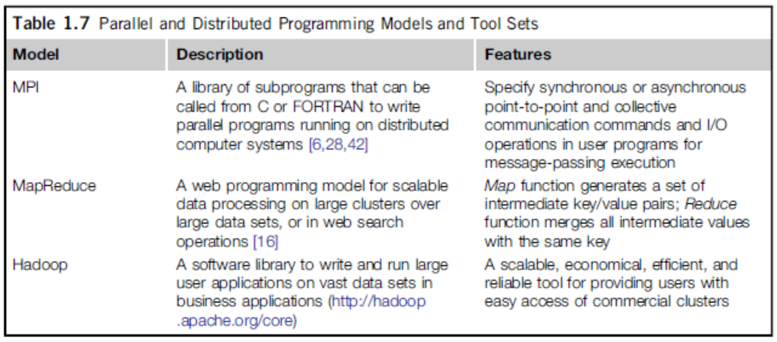


Table 1.7 Parallel and Distributed Programming Models and Tool Sets

Model	Description	Features
MPI	A library of subprograms that can be called from C or FORTRAN to write parallel programs running on distributed computer systems [6,28,42]	Specify synchronous or asynchronous point-to-point and collective communication commands and I/O operations in user programs for message-passing execution
MapReduce	A web programming model for scalable data processing on large clusters over large data sets, or in web search operations [16]	*Map* function generates a set of intermediate key/value pairs; *Reduce* function merges all intermediate values with the same key
Hadoop	A software library to write and run large user applications on vast data sets in business applications (http://hadoop.apache.org/core)	A scalable, economical, efficient, and reliable tool for providing users with easy access of commercial clusters

MPI is the most popular programming model for message-passing systems. Google's MapReduce and BigTable are for effective use of resources from Internet clouds and data centers. Service clouds demand extending Hadoop, EC2, and S3 to facilitate distributed computing over distributed storage systems.

Message-Passing Interface (MPI)

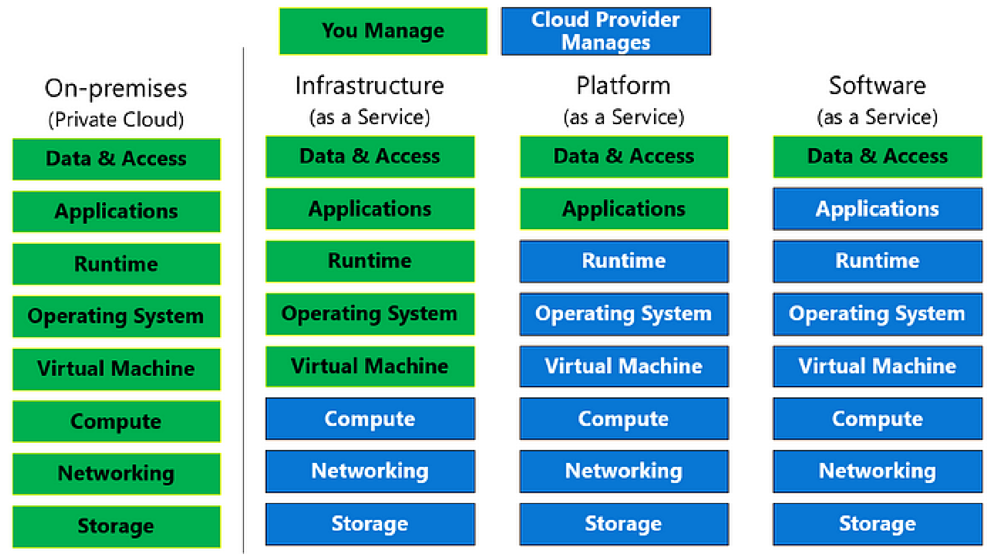
This is the primary programming standard used to develop parallel and concurrent programs to run on a distributed system. MPI is essentially a library of subprograms that can be called from C or FORTRAN to write parallel programs running on a distributed system. The idea is to embody clusters, grid systems,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

Besides MPI, distributed programming can be also supported with low-level primitives such as the Parallel Virtual Machine (PVM).

MapReduce: This is a web programming model for scalable data processing on large clusters over large data sets

The model is applied mainly in web-scale search and cloud computing applications. The 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.

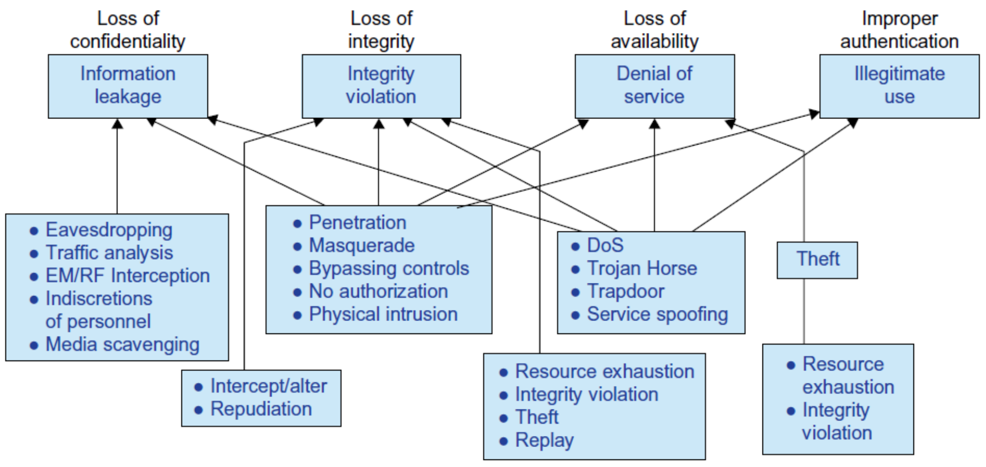
Hadoop Library: Hadoop offers a software platform that was originally developed by a Yahoo! group. The package enables users to write and run applications over vast amounts of distributed data. Users can easily scale Hadoop to store and process petabytes of data in the web space. Also, Hadoop is economical in that it comes with an open source version of MapReduce that minimizes overhead in task spawning and massive data communication. It is efficient, as it processes data with a high degree of parallelism across a large number of

	commodity nodes, and it is reliable in that it automatically keeps multiple data copies to facilitate redeployment of computing tasks upon unexpected system failures. **Open Grid Services Architecture (OGSA):** The development of grid infrastructure is driven by large-scale distributed computing applications. These applications must count on a high degree of resource and data sharing Key features include a distributed execution environment, Public Key Infrastructure (PKI) services using a local certificate authority (CA), trust management, and security policies in grid computing. **Globus Toolkits and Extensions** Globus is a middleware library jointly developed by the U.S. Argonne National Laboratory and USC Information Science Institute over the past decade. This library implements some of the OGSA standards for resource discovery, allocation, and security enforcement in a grid environment. The Globus packages support multisite mutual authentication with PKI certificates.			
3 (a)	Explain with a neat diagram and examples the various service models of IaaS, PaaS, and SaaS  You Manage Cloud Provider Manages On-premises (Private Cloud) Infrastructure (as a Service) Platform (as a Service) Software (as a Service) Data & Access Applications Runtime Operating System Virtual Machine Compute Networking Storage **Infrastructure as a Service (IaaS)** This model puts together infrastructures demanded by users—namely servers, storage, networks, and the data center fabric. The user can deploy and run on multiple VMs running guest OSes on specific applications. The user does not manage or control the underlying cloud infrastructure, but can specify when to request and release the needed resources. • **Platform as a Service (PaaS)** This model enables the user to deploy user-built applications onto a virtualized cloud platform. PaaS includes middleware, databases, development tools, and some runtime support such as Web 2.0 and Java. The platform includes both hardware and software integrated with specific programming interfaces. The provider supplies the API and software tools (e.g., Java, Python, Web 2.0, .NET). The user is freed from managing the cloud infrastructure. • **Software as a Service (SaaS)** This refers to browser-initiated application software over thousands of paid cloud customers. The SaaS model applies to business processes, industry applications, consumer relationship management (CRM), enterprise resources planning (ERP), human resources (HR), and collaborative applications. On the customer side, there is no upfront investment in servers or software licensing. On the provider side, costs are rather low, compared with conventional hosting of user applications	5M	CO1, CO3	L2
(b)	This is the answer given by Eli Lilly Systems engineer.	5M	CO1	L2

What are the data security issues that you're confronting with cloud, especially in working with external scientists or other companies?

D.P.: We have the same data security issues that everyone else has faced including: encryption of data in transit and at rest, limiting access to machines by using keys only, multi-tenant exposure, increased threat vectors from a large shared environment on the public internet and host of others.

Discuss various network security threats in cloud data center with a neat diagram.


Loss of confidentiality
Information leakage
Loss of integrity
Integrity violation
Loss of availability
Denial of service
Improper authentication
Illegitimate use
• Eavesdropping • Traffic analysis • EM/RF Interception • Indiscretions of personnel • Media scavenging
• Penetration • Masquerade • Bypassing controls • No authorization • Physical intrusion
• DoS • Trojan Horse • Trapdoor • Service spoofing
Theft
• Intercept/alter • Repudiation
• Resource exhaustion • Integrity violation • Theft • Replay
• Resource exhaustion • Integrity violation

Network viruses have threatened many users in widespread attacks. These incidents have created a worm epidemic by pulling down many routers and servers, and are responsible for the loss of billions of dollars in business, government, and services.
Information leaks lead to a loss of confidentiality.
Loss of data integrity may be caused by user alteration, Trojan horses, and service spoofing attacks. A denial of service (DoS) results in a loss of system operation and Internet connections.
Lack of authentication or authorization leads to attackers' illegitimate use of computing resources. Open resources such as data centers, P2P networks, and grid and cloud infrastructures could become the next targets. Users need to protect clusters, grids, clouds, and P2P systems.
Otherwise, users should not use or trust them for outsourced work. Malicious intrusions to these systems may destroy valuable hosts, as well as network and storage resources.
Internet anomalies found in routers, gateways, and distributed hosts may hinder the acceptance of these public-resource computing services.

Three security requirements are often considered: confidentiality, integrity, and availability for most Internet service providers and cloud users.

Collusive piracy is the main source of intellectual property violations within the boundary of a P2P network. Paid clients (colluders) may illegally share copyrighted content files with unpaid clients (pirates). Online piracy has hindered the use of open P2P networks for commercial content delivery.

Three generations of network defense technologies have appeared in the past. In the first generation, tools were designed to prevent or avoid intrusions. These tools usually manifested themselves as access control policies or tokens, cryptographic systems, and so forth. However, an intruder could always penetrate a secure system because there is always a weak link in the security provisioning process. The second generation detected intrusions in a timely manner to exercise remedial actions.
These techniques included firewalls, intrusion detection systems (IDSes), PKI services, reputation systems, and so on. The third generation provides more intelligent responses to intrusions.

Security infrastructure is required to safeguard web and cloud services. At the user level, one needs to perform trust negotiation and reputation aggregation over all users. At the application end, we need to establish security precautions in worm containment and intrusion detection against virus, worm, and distributed DoS (DDoS) attacks.

4 (a) What is virtualization? List the merits and demerits of virtualization | 4M | CO2 | L2

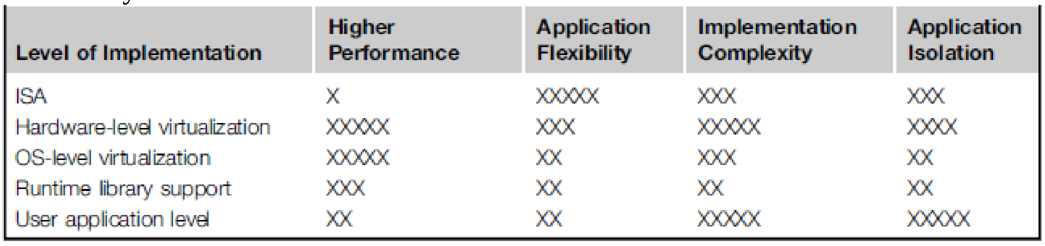
Virtualization is a computer architecture technology by which multiple virtual machines (VMs) are multiplexed in the same hardware machine.

Merits :

- virtualization techniques can be applied to enhance the use of compute engines, networks, and storage.
- virtualization technology has been revitalized as the demand for distributed and cloud computing
- Isolation of VMs
- Server consolidation and energy saving
- Optimized use of resources

Demerits :

- Performance overhead
- Increased implementation complexity
- Security issues

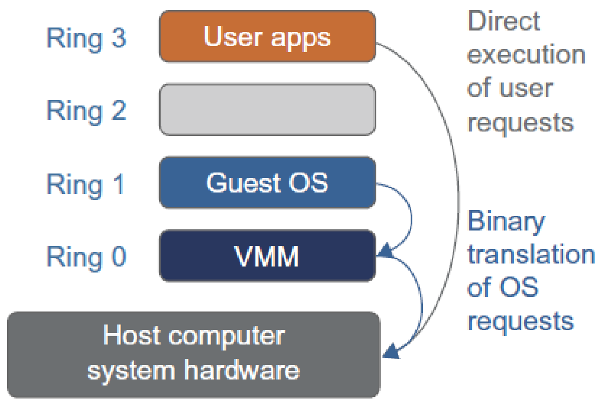


Level of Implementation	Higher Performance	Application Flexibility	Implementation Complexity	Application Isolation
ISA	X	XXXXX	XXX	XXX
Hardware-level virtualization	XXXXX	XXX	XXXXX	XXXX
OS-level virtualization	XXXXX	XX	XXX	XX
Runtime library support	XXX	XX	XX	XX
User application level	XX	XX	XXXXX	XXXXX

(b) Compare and contrast Type1, Type 2 virtualization with a neat diagram. Give examples of each type of virtualization. | 6M | CO2 | L2

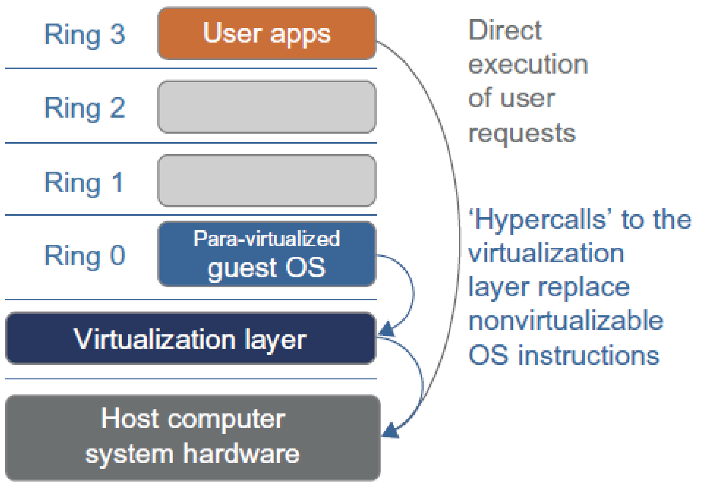
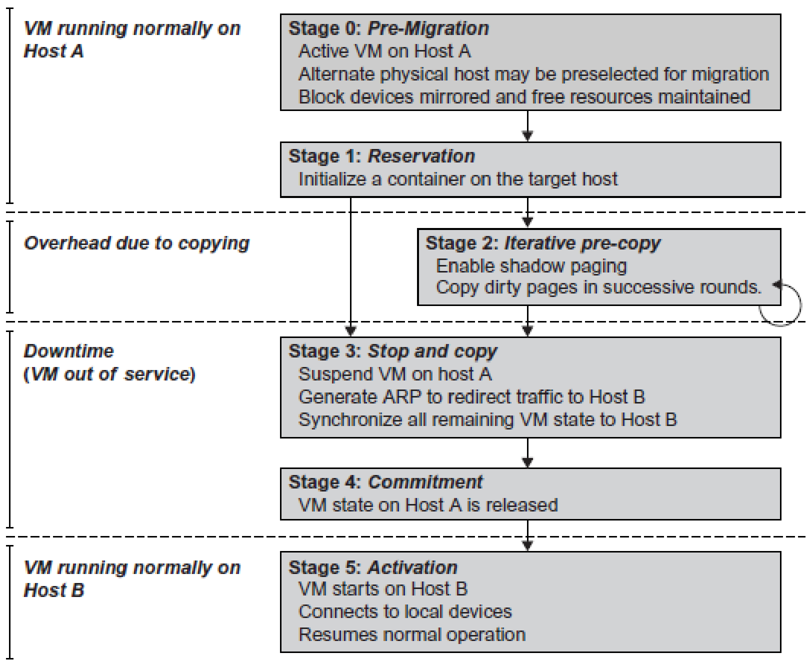
Type 1 : Bare-Metal or full virtualization

With full virtualization, noncritical instructions run on the hardware directly while critical instructions are discovered and replaced with traps into the VMM to be emulated by software. Both the hypervisor and VMM approaches are considered full virtualization

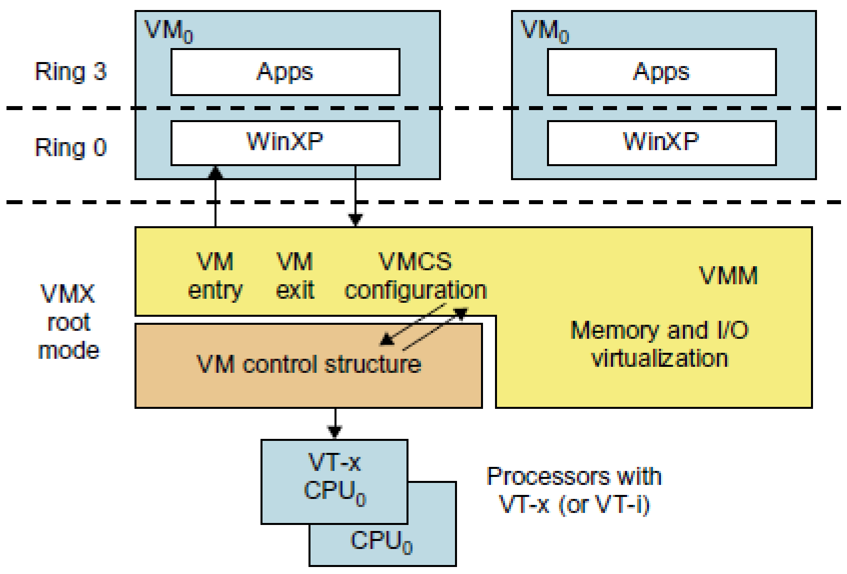
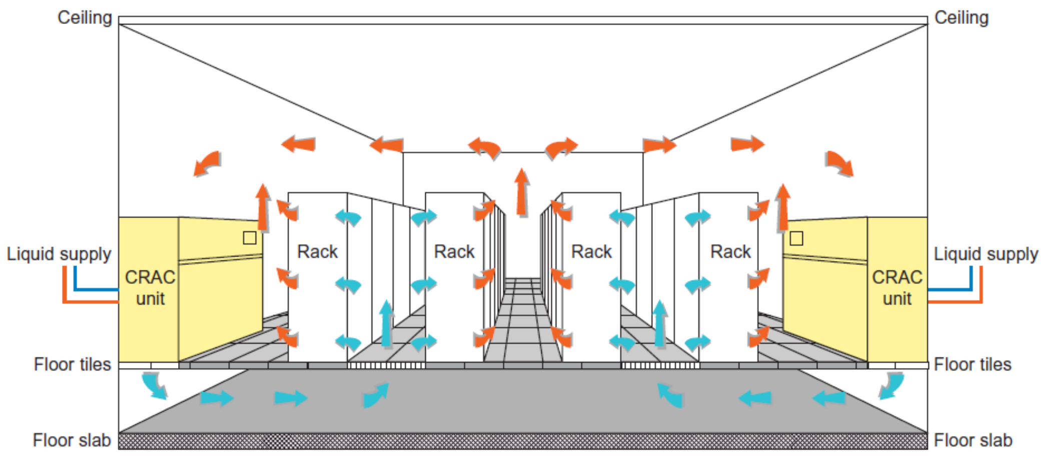


Ring 3
User apps
Ring 2
Ring 1
Guest OS
Ring 0
VMM
Host computer system hardware
Direct execution of user requests
Binary translation of OS requests

VMM scans the instruction stream and identifies the privileged, control- and ehavior-sensitive instructions. When these instructions are identified, they are trapped into the VMM, which emulates the behavior of these instructions. The method used in this emulation is called binary translation. Therefore, full virtualization combines binary translation and direct execution. The guest OS is completely decoupled from the underlying hardware. Consequently, the guest OS is unaware that it is being virtualized.
The performance of full virtualization may not be ideal, because it involves binary translation which is rather time-consuming.

	Eg. VMware ESXi, Microsoft Hyper-V, KVM **Host based virtualization** An alternative VM architecture is to install a virtualization layer on top of the host OS. This host OS is still responsible for managing the hardware. The guest OSes are installed and run on top of the virtualization layer. Dedicated applications may run on the VMs. Certainly, some other applications can also run with the host OS directly. This host-based architecture has some distinct advantages, as enumerated next. First, the user can install this VM architecture without modifying the host OS. The virtualizing software can rely on the host OS to provide device drivers and other low-level services. This will simplify the VM design and ease its deployment. Second, the host-based approach appeals to many host machine configurations. Compared to the hypervisor/VMM architecture, the performance of the host-based architecture may also be low. When an application requests hardware access, it involves four layers of mapping which downgrades performance significantly. When the ISA of a guest OS is different from the ISA of the underlying hardware, binary translation must be adopted. Although the host-based architecture has flexibility, the performance is too low to be useful in practice. Eg. Oracle VM VirtualBox, VMware Workstation, Microsoft Virtual PC.  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			
5 (a)	Explain live migration with a neat diagram.  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	6M	CO2	L2

	The migration copies the VM state file from the storage area to the host machine. This is useful for failover flexibility. live migration of a VM consists of the following six steps: *Steps 0 and 1: Start migration.* This step makes preparations for the migration, including determining the migrating VM and the destination host. Although users could manually make a VM migrate to an appointed host, in most circumstances, the migration is automatically started by strategies such as load balancing and server consolidation. *Steps 2: Transfer memory.* Since the whole execution state of the VM is stored in memory, sending the VM's memory to the destination node ensures continuity of the service provided by the VM. All of the memory data is transferred in the first round, and then the migration controller recopies the memory data which is changed in the last round. These steps keep iterating until the dirty portion of the memory is small enough to handle the final copy. Although precopying memory is performed iteratively, the execution of programs is not obviously interrupted. *Step 3: Suspend the VM and copy the last portion of the data.* The migrating VM's execution is suspended when the last round's memory data is transferred. Other nonmemory data such as CPU and network states should be sent as well. During this step, the VM is stopped and its applications will no longer run. This "service unavailable" time is called the "downtime" of migration, which should be as short as possible so that it can be negligible to users. *Steps 4 and 5:* Commit and activate the new host. After all the needed data is copied, on the destination host, the VM reloads the states and recovers the execution of programs in it, and the service provided by this VM continues. Then the network connection is redirected to the new VM and the dependency to the source host is cleared. The whole migration process finishes by removing the original VM from the source host.			
(b)	In x86 "not all of X86's **sensitive** instructions are **privileged** instructions" Why is this an issue in virtualization and what are the techniques in hardware-assisted virtualization for CPU? A CPU architecture is virtualizable if it supports the ability to run the VM's privileged and unprivileged instructions in the CPU's user mode while the VMM runs in supervisor mode. When the privileged instructions including control- and behavior-sensitive instructions of a VM are xecuted, they are trapped in the VMM. RISC CPU architectures can be naturally virtualized because all control- and behavior-sensitive instructions are privileged instructions. On the contrary, x86 CPU architectures are not primarily designed to support virtualization. This is because about 10 sensitive instructions, such as SGDT and SMSW, are not privileged instructions. When these instructions execute in virtualization, they cannot be trapped in the VMM. Intel and AMD add an additional mode called privilege mode level (some people call it Ring-1) to x86 processors. Therefore, operating systems can still run at Ring 0 and the hypervisor can run at Ring -1. All the privileged and sensitive instructions are trapped in the hypervisor automatically. This technique removes the difficulty of implementing binary translation of full virtualization. It also lets the operating system run in VMs without modification.	4M	CO2	L2

	 VM$_0$ Apps WinXP VM$_0$ Apps WinXP Ring 3 Ring 0 VMX root mode VM entry VM exit VMCS configuration VMM Memory and I/O virtualization VM control structure VT-x CPU$_0$ CPU$_0$ Processors with VT-x (or VT-i) Intel calls the privilege level of x86 processors the VMX Root Mode. In order to control the start and stop of a VM and allocate a memory page to maintain the CPU state for VMs, a set of additional instructions is added. Generally, hardware-assisted virtualization should have high efficiency. However, since the transition from the hypervisor to the guest OS incurs high overhead switches between processor modes, it sometimes cannot outperform binary translation.			
6 a)	Write short notes on i) Cloud design objectives ii) cooling system for a datacenter iii) datacenter interconnection networks Cloud Design Objectives 1. Shifting computing from desktops to data centers 2. Service provisioning and cloud economics – sign SLA with consumers, pay-as-you-go pricing 3. Scalability in performance – scale as number of users increases 4. Data privacy protection 5. High quality of cloud services 6. New standards and interfaces – to address vendor lockin Cooling System  Ceiling Ceiling Liquid supply Liquid supply CRAC unit CRAC unit Rack Rack Rack Rack Floor tiles Floor tiles Floor slab Floor slab • raised floors for hiding cables, power lines, and cooling supplies • under-floor area is often used to route power cables to racks, but its primary use is to distribute cool air to the server rack • CRAC (computer room air conditioning) unit pressurizes the raised floor plenum by blowing cold air into the plenum	6M	CO3	L2

- Racks are arranged in long aisles that alternate between cold aisles and hot aisles to avoid mixing hot and cold air
- hot air produced by the servers circulates back to the intakes
- incoming coolant is at 12–14°C

Data Interconnection networks

network design must meet five special requirements: **low latency**, **high bandwidth**, **low cost**, **message-passing interface (MPI) communication support**, and **fault tolerance**

- **Application Traffic Support** : point-to-point and collective MPI communications must be supported
- **Network expandability** : network should be designed to support load balancing and data movement among the servers
  - server containers while each container contains several hundred or even thousands of server nodes are just plugged in the power supply
  - efficient and reduces the cost of purchasing and maintaining servers
- **Fault Tolerance and graceful degradation**:
  - Network should tolerate link or switch failures
  - Packet forwarding should avoid using broken links
  - Hot-swappable components are desired in case of network failures
  - There should be not critical points / single point of failure
  - Two network layers : lower layer is close to end servers, upper layer establishes backbone connections among server groups / clusters.
  - hierarchical interconnection approach appeals to building data centers with modular containers
- Switch-centric Data-Center Design - switch-centric network, the switches are used to connect the server nodes, no modifications to servers are needed

b)	Compare using a tabular column *any two* popular PaaS offerings in terms of languages/developer tools, programming models supported by the developer and target applications and storage options. Eg. (GAE, SalesForce.com, Microsoft Azure, AWS MapReduce)	4M	CO3	L2

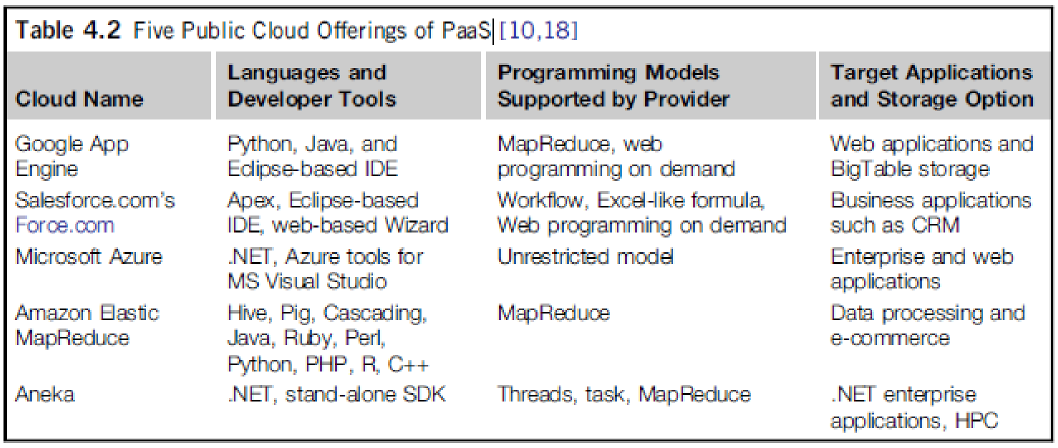


Table 4.2 Five Public Cloud Offerings of PaaS [10,18]

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

CI CCI HoD

CO-PO and CO-PSO Mapping																			
Course Outcomes		**Blooms Level**	**Modules covered**	**PO1**	**PO2**	**PO3**	**PO4**	**PO5**	**PO6**	**PO7**	**PO8**	**PO9**	**PO10**	**PO11**	**PO12**	**PSO1**	**PSO2**	**PSO3**	**PSO4**
CO1	Describe various cloud computing platforms and service providers.	**L2**	**1**	3	2	-	-	-	3	-	-	-	-	-	-	-	2	-	2
CO2	Illustrate the significance of various types of virtualization.	**L2**	**2**	3	2	-	-	2	-	-	-	-	-	-	-	-	2	-	2
CO3	Identify the architecture, delivery models and industrial platforms for cloud computing based applications.	**L2**	**3**	3	2	-	-	2	-	3	-	-	-	-	-	-	2	-	2
CO4	Analyze the role of security aspects in cloud computing.	**L2**	**4**	3	2	-	-	-	3	-	-	-	-	-	-	-	2	-	2
CO5	Demonstrate cloud applications in various fields using suitable cloud platforms..	**L2**	**5**	3	2	-	-	3	3	-	-	-	-	-	-	-	2	-	2

CO PO Mapping

COGNITIVE LEVEL	REVISED BLOOMS TAXONOMY KEYWORDS
L1	List, define, tell, describe, identify, show, label, collect, examine, tabulate, quote, name, who, when, where, etc.
L2	summarize, describe, interpret, contrast, predict, associate, distinguish, estimate, differentiate, discuss, extend
L3	Apply, demonstrate, calculate, complete, illustrate, show, solve, examine, modify, relate, change, classify, experiment, discover.
L4	Analyze, separate, order, explain, connect, classify, arrange, divide, compare, select, explain, infer.
L5	Assess, decide, rank, grade, test, measure, recommend, convince, select, judge, explain, discriminate, support, conclude, compare, summarize.

PROGRAM OUTCOMES (PO), PROGRAM SPECIFIC OUTCOMES (PSO)				CORRELATION LEVELS	
PO1	Engineering knowledge	PO7	Environment and sustainability	0	No Correlation
PO2	Problem analysis	PO8	Ethics	1	Slight/Low
PO3	Design/development of solutions	PO9	Individual and team work	2	Moderate/ Medium
PO4	Conduct investigations of complex problems	PO10	Communication	3	Substantial/ High
PO5	Modern tool usage	PO11	Project management and finance		
PO6	The Engineer and society	PO12	Life-long learning		
PSO1	Develop applications using different stacks of web and programming technologies				
PSO2	Design and develop secure, parallel, distributed, networked, and digital systems				
PSO3	Apply software engineering methods to design, develop, test and manage software systems.				
PSO4	Develop intelligent applications for business and industry				