JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY ANANTAPUR

B.Tech. III-II Sem. (I.T)	T	Tu	C
	3	1	3

(13A05607) CLOUD COMPUTING
(CBCC)

Course Objectives:

- To explain the evolving computer model called cloud computing.
- To introduce the various levels of services that can be achieved by cloud.
- To describe the security aspects in cloud.

Course Outcomes:

- Ability to create cloud computing environment
- Ability to design applications for Cloud environment

UNIT- I

Systems Modeling, Clustering and Virtualization

Distributed System Models and Enabling Technologies, Computer Clusters for Scalable Parallel Computing, Virtual Machines and Virtualization of Clusters and Data centers.

UNIT- II

Foundations

Introduction to Cloud Computing, Migrating into a Cloud, Enriching the 'Integration as a Service' Paradigm for the Cloud Era, The Enterprise Cloud Computing Paradigm.

UNIT- III

Infrastructure as a Service (IAAS) & Platform and Software as a Service (PAAS / SAAS)

Virtual machines provisioning and Migration services, On the Management of Virtual machines for Cloud Infrastructures, Enhancing Cloud Computing Environments using a cluster as a Service, Secure Distributed Data Storage in Cloud Computing.

Aneka, Comet Cloud, T-Systems', Workflow Engine for Clouds, Understanding Scientific Applications for Cloud Environments.

UNIT- IV

Monitoring, Management and Applications

An Architecture for Federated Cloud Computing, SLA Management in Cloud Computing, Performance Prediction for HPC on Clouds, Best Practices in Architecting Cloud Applications in the AWS cloud, Building Content Delivery networks using Clouds, Resource Cloud Mashups.

UNIT- V

Governance and Case Studies

Organizational Readiness and Change management in the Cloud age, Data Security in the Cloud, Legal Issues in Cloud computing, Achieving Production Readiness for Cloud Services.

UNIT I

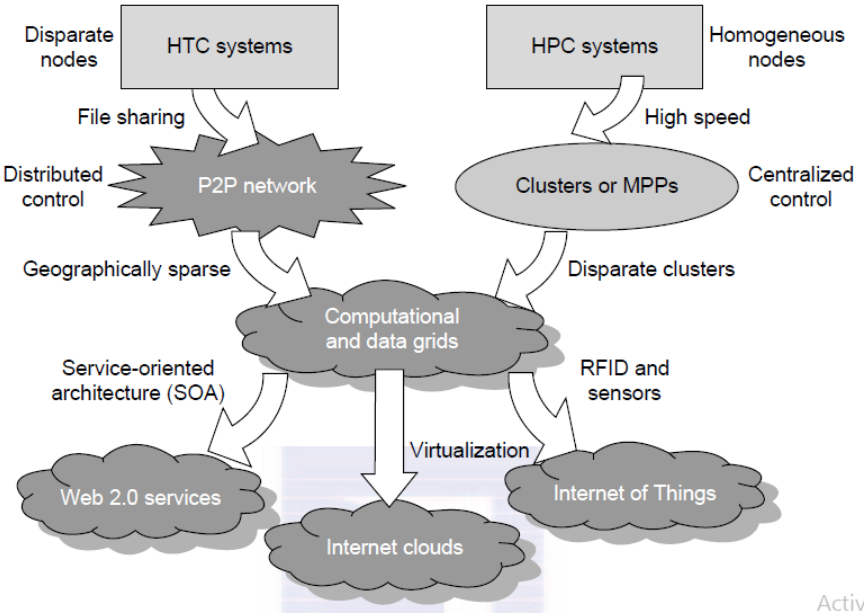
- Distributed System Models and Enabling Technologies
- Computer cluster for scalable parallel computing
- Virtual Machines
- Virtualization of clusters and data centers

Distributed System Models and Enabling Technologies:-

Scalable Computing over the Internet:-

Computing technology changes in machine architecture, operating system platform, network connectivity, and application workload. Instead of using a centralized computer to solve computational problems, a parallel and distributed computing system uses multiple computers to solve large-scale problems over the Internet. Thus, distributed computing becomes data-intensive and network-centric.

Supercomputer sites and large data centers must provide high-performance computing services to huge numbers of Internet users concurrently. Because of this high demand, the Linpack Benchmark for high-performance computing (HPC) applications is no longer optimal for measuring system performance. The emergence of computing clouds instead demands high-throughput computing (HTC) systems built with parallel and distributed computing technologies



History:

- In 1950 to 1970, a handful of mainframes, including the IBM 360 and CDC 6400, were built to satisfy the demands of large businesses and government organizations.
- From 1960 to 1980, lower-cost minicomputers such as the DEC PDP 11 and VAX Series became popular among small businesses and on college campuses.
- From 1970 to 1990, we saw widespread use of personal computers built with VLSI microprocessors.

- From 1980 to 2000, massive numbers of portable computers and pervasive devices appeared in both wired and wireless applications.
- Since 1990, the use of both HPC and HTC systems hidden in clusters, grids, or Internet clouds has proliferated.

HPC and HTC systems will demand multicore or many-core processors that can handle large numbers of computing threads per core. Both HPC and HTC systems emphasize parallelism and distributed computing. HPC and HTC systems must be able to satisfy this huge demand in computing power in terms of throughput, efficiency, scalability, and reliability. The system efficiency is decided by speed, programming, and energy factors (i.e., throughput per watt of energy consumed).

Meeting these goals requires to yield the following design objectives:

• **Efficiency** measures the utilization rate of resources in an execution model by exploiting massive parallelism in HPC. For HTC, efficiency is more closely related to job throughput, data access, storage, and power efficiency.

• **Dependability** measures the reliability and self-management from the chip to the system and application levels. The purpose is to provide high-throughput service with Quality of Service (QoS) assurance, even under failure conditions.

• **Adaptation in the programming model** measures the ability to support billions of job requests over massive data sets and virtualized cloud resources under various workload and service models.

• **Flexibility** in application deployment measures the ability of distributed systems to run well in both HPC (science and engineering) and HTC (business) applications.

Computing Paradigm:

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

Parallel computing In parallel computing, all processors are either tightly coupled with centralized shared memory or loosely coupled with distributed memory. Interprocessor communication is accomplished through shared memory or via message passing. A computer system capable of parallel computing is commonly known as a parallel computer. Programs

running in a parallel computer are called parallel programs. The process of writing parallel programs is often referred to as parallel programming.

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

Cloud computing An Internet cloud of resources can be either a centralized or a distributed computing system. The cloud applies parallel or distributed computing, or both. Clouds can be built with physical or virtualized resources over large data centers that are centralized or distributed. cloud computing can also be called as utility computing or service computing.

Scalable Computing Trends & Paradigms:

Degree of Parallelism

Bit-level parallelism (BLP) converts bit-serial processing to word-level processing gradually. Over the years, users graduated from 4-bit microprocessors to 8-,16-, 32-, and 64-bit CPUs. This led us to the next wave of improvement, known as instruction-level parallelism (ILP), in which the processor executes multiple instructions simultaneously rather than only one instruction at a time. For the past 30 years, we have practiced ILP through pipelining, superscalar computing, VLIW (very long instruction word) architectures, and multithreading. ILP requires branch prediction, dynamic scheduling, speculation, and compiler support to work efficiently. Data-level parallelism (DLP) was made popular through SIMD (single instruction, multiple data) and vector machines using vector or array types of instructions. DLP requires even more hardware support and compiler assistance to work properly. Ever since the introduction of multicore processors and chip multiprocessors (CMPs), we have been exploring task-level parallelism (TLP).

Innovative Applications

Both HPC and HTC systems desire transparency in many application aspects. For example, data access, resource allocation, process location, concurrency in execution, job replication, and failure recovery should be made transparent to both users and system management.

Applications:

- Science and engineering
- Business, education, services industry, and health care
- Internet and web services,and government applications
- Mission-critical applications

These applications spread across many important domains in science, engineering, business, education, health care, traffic control, Internet and web services, military, and government applications.

Utility Computing:- Utility computing focuses on a business model in which customers receive computing resources from a paid service provider. All grid/cloud platforms are regarded as utility service providers. However, cloud computing offers a broader concept than utility computing. Distributed cloud applications run on any available servers in some edge networks. Major technological challenges include all aspects of computer science and engineering. For example, users demand new networkefficient processors, scalable memory and storage schemes, distributed OSes, middleware for machine virtualization, new programming models, effective resource management, and application program development.

The Hype Cycle of New Technologies:- Any new and emerging computing and information technology may go through a hype cycle, as illustrated in Figur. This cycle shows the expectations for the technology at five different stages. The expectations rise sharply from the trigger period to a high peak of inflated expectations. Through a short period of disillusionment, the expectation may drop to a valley and then increase steadily over a long enlightenment period to a plateau of productivity. The number of years for an emerging technology to reach a certain stage is marked by special symbols. The hollow circles indicate technologies that will reach mainstream adoption in two years. The gray circles represent technologies that will reach mainstream adoption in two to five years. The solid circles represent those that require five to 10 years to reach mainstream adoption, and the triangles denote those that require more than 10 years. The crossed circles represent technologies that will become obsolete before they reach the plateau.

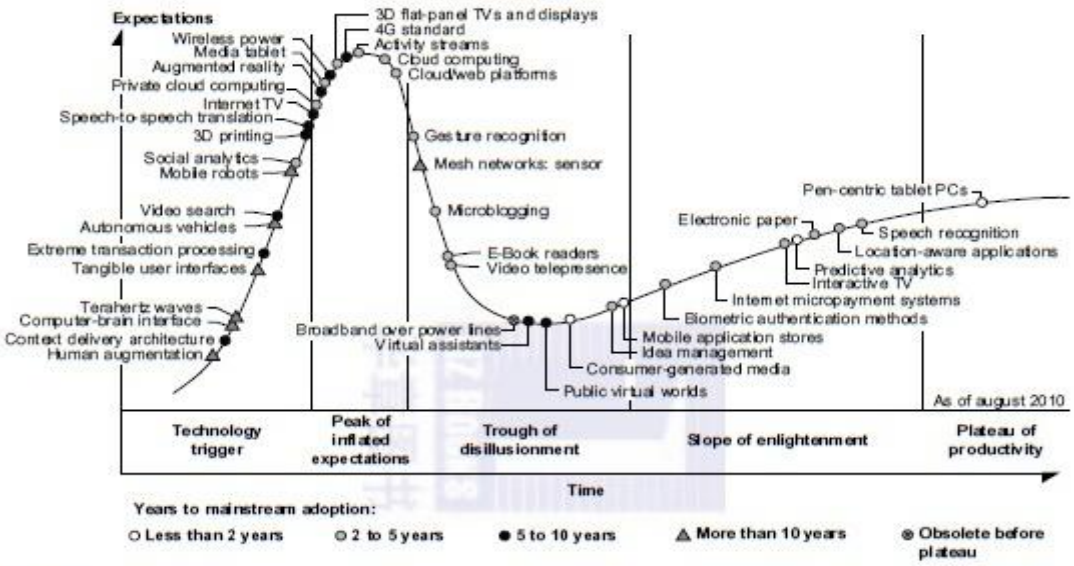


FIGURE 1.3

Hype cycle for Emerging Technologies, 2010.

Hype Cycle Disclaimer

The Hype Cycle is copyrighted 2010 by Gartner, Inc. and its affiliates and is reused with permission. Hype Cycles are graphical representations of the relative maturity of technologies, IT methodologies and management disciplines. They are intended solely as a research tool, and not as a specific guide to action. Gartner disclaims all warranties, express or implied, with respect to this research, including any warranties of merchantability or fitness for a particular purpose.

This Hype Cycle graphic was published by Gartner, Inc. as part of a larger research note and should be evaluated in the context of the entire report. The Gartner report is available at http://www.gartner.com/it/page.jsp?id=1447613.

(Source: Gartner Press Release "Gartner's 2010 Hype Cycle Special Report Evaluates Maturity of 1,800 Technologies" 7 October 2010.)

The IOTs and Cyber Physcial systems:

The traditional Internet connects machines to machines or web pages to web pages. The concept of the IoT was introduced in 1999 at MIT. The IoT refers to the networked interconnection of everyday objects, tools, devices, or computers. The dynamic connections will grow exponentially into a new dynamic network of networks, called the Internet of Things (IoT).

A cyber-physical system (CPS) is the result of interaction between computational processes and the physical world. A CPS integrates "cyber" (heterogeneous, asynchronous) with "physical" (concurrent and information-dense) objects. A CPS merges the "3C" technologies of computation, communication, and control into an intelligent closed feedback system between the physical world and the information world, a concept which is actively explored in the United States.

Cyber-Physical Systems

A cyber-physical system (CPS) is the result of interaction between computational processes and the physical world. A CPS integrates "cyber" (heterogeneous, asynchronous) with "physical" (concurrent and information-dense) objects. A CPS merges the "3C"

technologies of computation, communication, and control into an intelligent closed feedback system between the physical world and the information world, a concept which is actively explored in the United States. The IoT emphasizes various networking connections among physical objects, while the CPS emphasizes exploration of virtual reality (VR) applications in the physical world.

TECHNOLOGIES FOR NETWORK-BASED SYSTEMS

TECHNOLOGIES FOR NETWORK-BASED SYSTEMS

Distributed computing system uses scalable computing model and different hardware, software and network technologies.

Multicore CPUs and Multithreading Technologies

Processor speed is measured in millions of instructions per second (MIPS) and network bandwidth is measured in megabits per second (Mbps) or gigabits per second (Gbps). The unit GE refers to 1 Gbps Ethernet bandwidth.

CPU Processors

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. CPUs executes 1 MIPS for the VAX 780 in 1978 to 1,800 MIPS for the Intel Pentium 4 in 2002, up to a 22,000 MIPS peak for the Sun Niagara 2 in 2008. The clock rate for these processors increased from 10 MHz for the Intel 286 to 4 GHz for the Pentium 4.However, the clock rate reached its limit on CMOS-based chips due to power limitations. At the time of this writing, very few CPU chips run with a clock rate exceeding 5 GHz. In other words, clock rate will not continue to improve unless chip technology matures. This limitation is attributed primarily to excessive heat generation with high frequency or high voltages.

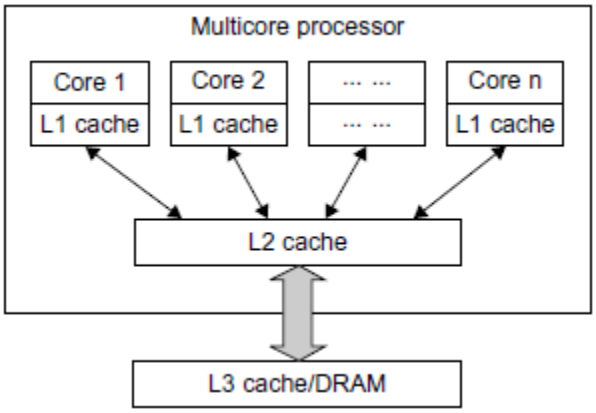
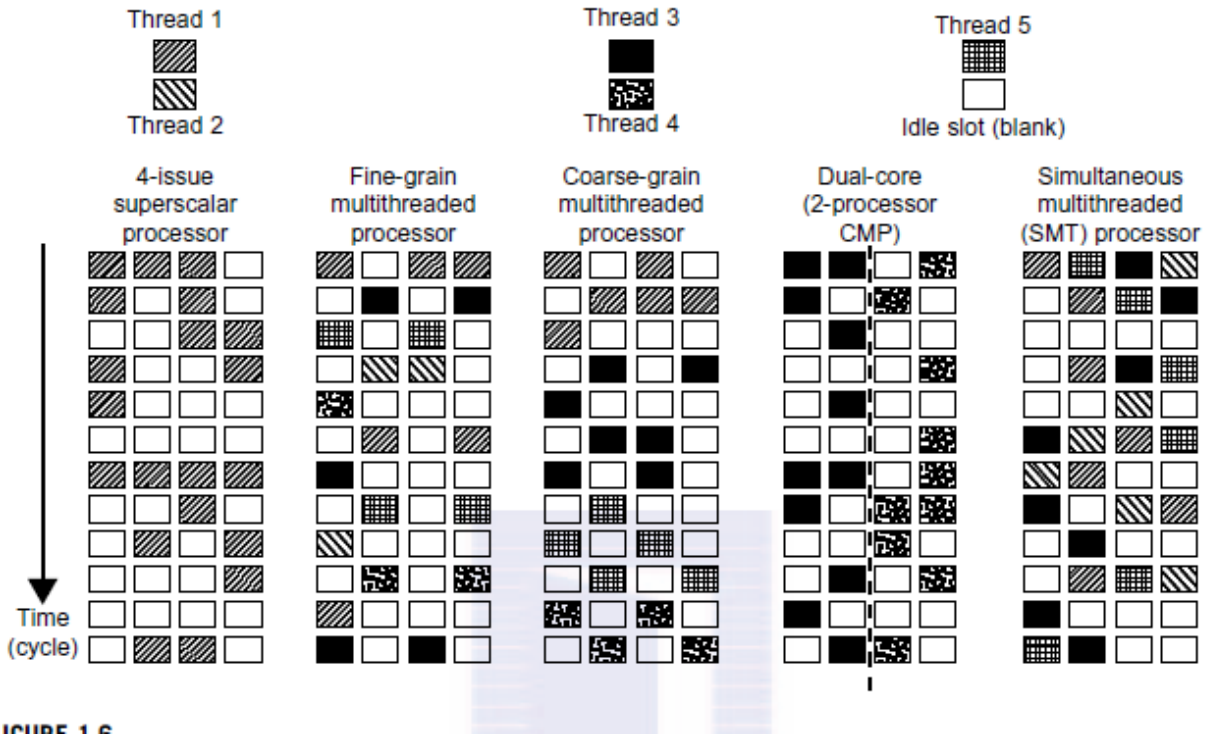


Figure shows the architecture of a typical multicore processor. Each core is essentially a processor with its own private cache (L1 cache). Multiple cores are housed in the same chip with an L2 cache that is shared by all cores. In the future, multiple CMPs could be built on the same CPU chip with even the L3 cache on the chip. Multicore and multithreaded CPUs are equipped with many high-end processors, including the Intel i7, Xeon, AMD Opteron, Sun Niagara, IBM Power 6, and X cell processors. Each core could be also multithreaded.

CPUs may increase from the tens of cores to hundreds or more in the future. But the CPU has reached its limit in terms of exploiting massive DLP due to the aforementioned memory wall problem. This has triggered the development of many-core GPUs with hundreds or more thin cores.

Multithreading Technology



The superscalar processor is single-threaded with four functional units. Each of the three multithreaded processors is four-way multithreaded over four functional data paths. In the dual-core processor, assume two processing cores, each a single-threaded two-way superscalar processor. Instructions from different threads are distinguished by specific shading patterns for instructions from five independent threads. Typical instruction scheduling patterns are shown here. Only instructions from the same thread are executed in a superscalar processor. 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.

GPU Computing

A 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. The world's first GPU, the GeForce 256, was marketed by NVIDIA in 1999. These GPU chips can process a minimum of 10 million polygons per second.

How GPUs Work

Early GPUs functioned as coprocessors attached to the CPU. The NVIDIA GPU has been upgraded to 128 cores on a single chip. Furthermore, each core on a GPU can handle eight threads of instructions. This translates to having up to 1,024 threads executed concurrently on a single GPU. This is true massive parallelism, compared to only a few threads that can be handled by a conventional CPU. The CPU is optimized for latency caches, while the GPU is optimized to deliver much higher throughput with explicit management of on-chip memory. 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.

GPU Programming Model

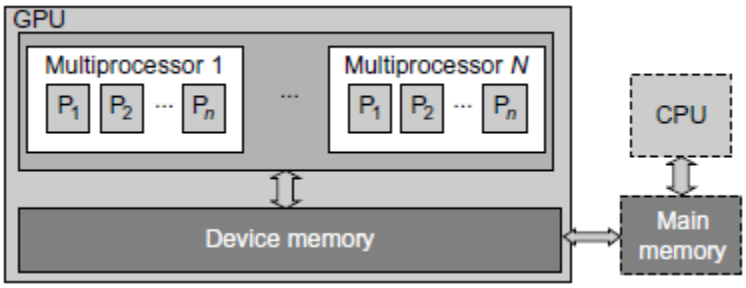
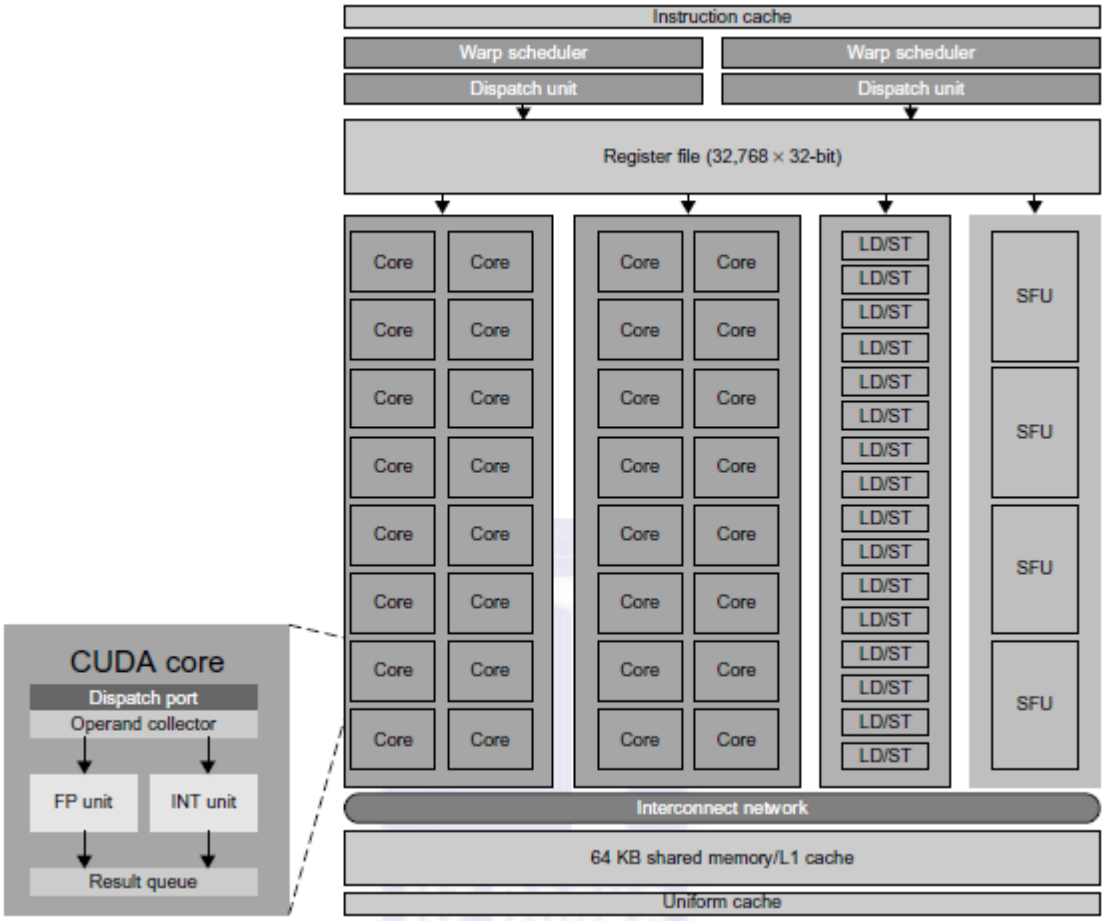


Figure shows the interaction between a CPU and GPU in performing parallel execution of floating-point operations concurrently. The CPU is the conventional multicore processor with limited parallelism to exploit. The GPU has a many-core architecture that has hundreds of simple processing cores organized 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.



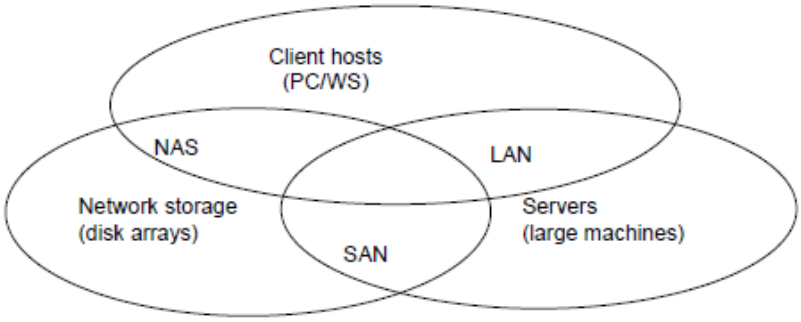
This is a streaming multiprocessor (SM) module. Multiple SMs can be built on a single GPU chip. The Fermi chip has 16 SMs implemented with 3 billion transistors. Each SM comprises up to 512 streaming processors (SPs), known as CUDA cores. The Tesla GPUs used in the Tianhe-1a have a similar architecture, with 448 CUDA cores. The Fermi GPU is a newer generation of GPU, first appearing in 2011. The Tesla or Fermi GPU can be used in desktop workstations to accelerate floating-point calculations or for building large-scale data centers.

Memory, Storage, and Wide-Area Networking

System-Area Interconnects

The nodes in small clusters are mostly interconnected by an Ethernet switch or a local area network (LAN). As Figure 1.11 shows, a LAN typically is used to connect client hosts to big servers. A storage area network (SAN) connects servers to network storage such as disk arrays. Network attached storage (NAS) connects client hosts directly to the disk arrays. All three types of networks often appear in a large cluster built with commercial network components. If no large distributed storage is shared, a small cluster could be built with a

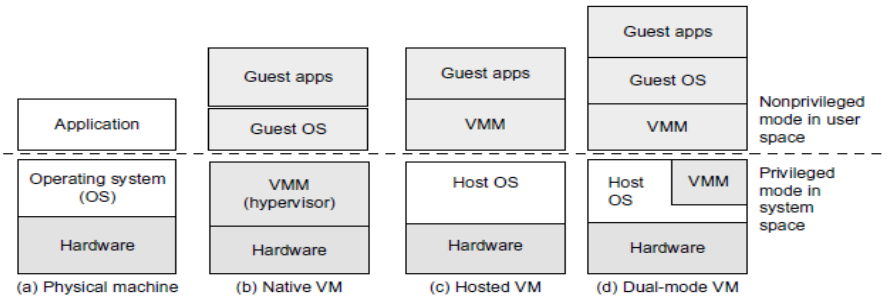
multiport Gigabit Ethernet switch plus copper cables to link the end machines. All three types of networks are commercially available.



Virtual Machines and Virtualization Middleware:-

A conventional computer has a single OS image. This offers a rigid architecture that tightly couples application software to a specific hardware platform. Some software running well on one machine may not be executable on another platform with a different instruction set under a fixed OS. Virtual machines (VMs) offer novel solutions to underutilized resources, application inflexibility, software manageability, and security concerns in existing physical machines. To build large clusters, grids, and clouds, we need to access large amounts of computing, storage, and networking resources in a virtualized manner. We need to aggregate those resources, and hopefully, offer a single system image. In particular, a cloud of provisioned resources must rely

on virtualization of processors, memory, and I/O facilities dynamically.



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

DISTRIBUTED SYSTEM MODELS

A massively parallel and distributed computing system or in short a massive system is built over a large number of autonomous computer nodes. These node machines are interconnected by system-area networks (SAN), local-are networks (LAN), or wide-area networks (WAN) in a hierarchical manner.

Massive systems are considered highly scalable to reach a web-scale connectivity, either physically or logically.

we classify massive systems into four classes:

- Clusters
- P2P Networks
- Computing Grids
- Internet Clouds

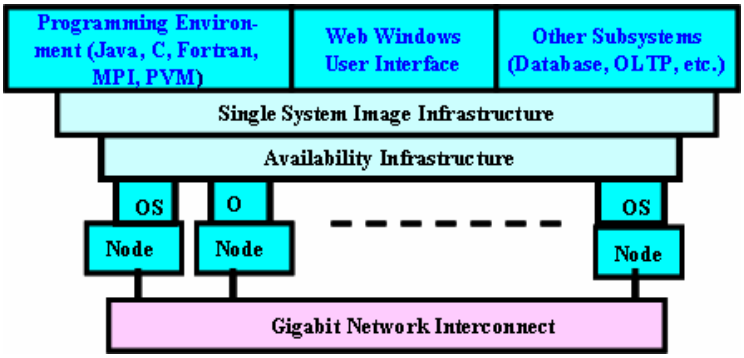
CLUSTERS:-

A computing cluster is built by a collection of interconnected stand-alone computers, which work cooperatively together as a single integrated computing resource. To handle heavy workload with large datasets.

The architecture of a typical sever cluster built around a low-latency and high-bandwidth interconnection network. This network can be as simple as a SAN (e.g.Myrinet) or a LAN (e.g. Ethernet). To build a larger cluster with more nodes, the InterNetworking can be built with multiple levels of Gigabit Ethernet, Myrinet, or InfiniBand switches. Through hierarchical construction using SAN, LAN, or WAN, one can build scalable clusters with increasing number of nodes. The whole cluster is connected to the Internet via a VPN gateway. The gateway IP address could be used to locate the cluster over the cyberspace.

Clusters have loosely-coupled node computers. All resources of a server node is managed by its own OS. Thus, most clusters have multiple system images coexisting simultaneously.We need an idealized cluster operating system or some middlware to support SingleSystemImage at various levels, including the sharing of all CPUs, memories, and I/O across all computer nodes attached to the cluster.

A cluster of servers (S_1, S_2,…,S_n) interconnected by a high-bandwidth system-area or local-area network with shared I/O devices and disk arrays. The cluster acts as a single computing node attached to the Internet throught a gateway.



A cluster with multiple system images is nothing but a collection of independent computers. above figure shows the hardware and software architecture of a typical cluster system. Each node computer has its own operating system. On top of all operating systems, we deploy some two layers of middleware at the user space to support the high availability and some SSI features for shared resources or fast MPI communications.

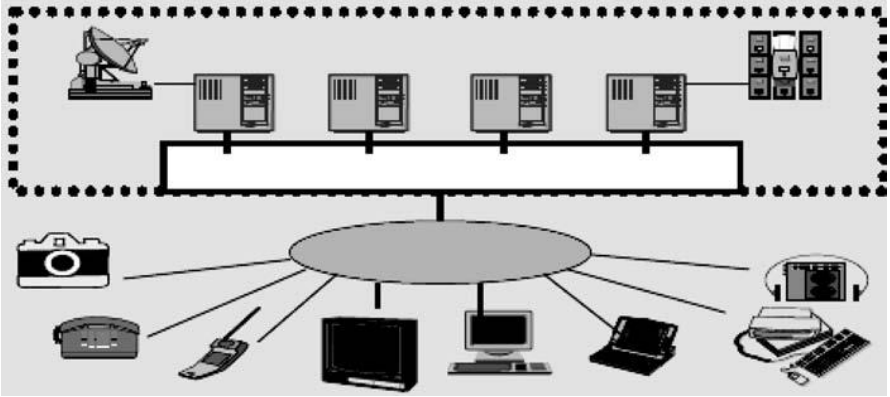
Cluster Computing Model

- Architecture:- Network of compute nodes interconnected by SAN, LAN, or WAN hierarchically
- Control and Resources Management:- Homogeneous nodes with distributed control, running UNIX or Linux.
- Applications:- High-performance computing, search engines, and web services, etc.

GRID COMPUTING

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.

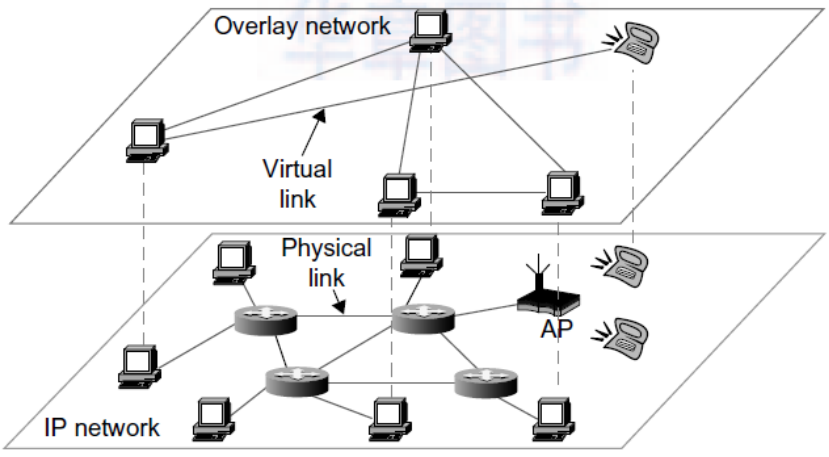


Cluster Computing Model

- Architecture:- Network of compute nodes interconnected by SAN, LAN, or WAN hierarchically
- Control and Resources Management:- Homogeneous nodes with distributed control, running UNIX or Linux.
- Applications:- High-performance computing, search engines, and web services, etc.

PEER TO PEER COMPUTING MODEL:-

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. The system is self-organizing with distributed control. P2P network does not use a dedicated interconnection network.



In a P2P system 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 as shown in Figure. When a new peer joins the system, its peer ID is added as a node in the overlay network. When an existing peer leaves the system, its peer ID is removed from the overlay network automatically. Therefore, it is the P2P overlay network that characterizes the logical connectivity among the peers.

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.

- Architecture:- Flexible network of client machines logically connected by an overlay network.
- Control and Resources Management:- Autonomous client nodes, free in and out, with self organization
- Applications:- Most appealing to business file sharing, content delivery, and social networking.

INTERNET CLOUDS

Definition:- 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.

Cloud Computing Service Models:-

Infrastructure as a Service (IaaS) This model puts together infrastructures demanded by users—namely servers, storage, networks, and the data center fabric. The user can deploy and run on multiple VMs running guest OSes on specific applications. The user does not manage or control the underlying cloud infrastructure, but can specify when to request and release the needed resources.

Platform as a Service (PaaS) This model enables the user to deploy user-built applications onto a virtualized cloud platform. PaaS includes middleware, databases, development tools, and some runtime support such as Web 2.0 and Java. The platform includes both hardware and software integrated with specific programming interfaces. The provider supplies the API

and software tools (e.g., Java, Python, Web 2.0, .NET). The user is freed from managing the cloud infrastructure.

Software as a Service (SaaS) This refers to browser-initiated application software over thousands of paid cloud customers. The SaaS model applies to business processes, industry applications, consumer relationship management (CRM), enterprise resources planning (ERP), human resources (HR), and collaborative applications. On the customer side, there is no upfront investment in servers or software licensing. On the provider side, costs are rather low, compared with conventional hosting of user applications.

Cloud Deployment Models:-

◦ Private cloud. The cloud infrastructure is operated for a private organization. It may be managed by the organization or a third party, and may exist on premise or off premise.

◦ Community cloud. The cloud infrastructure is shared by several organizations and supports a specific community that has communal concerns (e.g., mission, security requirements, policy, and compliance considerations). It maybe managed by the organizations or a third party, and may exist on premise or off premise.

◦ Public cloud. The cloud infrastructure is made available to the general public or a large industry group and is owned by an organization selling cloud services.

◦ Hybrid cloud. The cloud infrastructure is a composition of two or more clouds (private, community, or public) that remain unique entities, but are bound together by standardized or proprietary technology, that enables data and application portability

Cloud Computing Characterstics:-

Flexibility/Elasticity: users can rapidly provision computing resources, as needed, without human interaction. Capabilities can be rapidly and elastically provisioned, in some cases automatically, to quickly scale out or up.

Scalability of infrastructure: new nodes can be added or dropped from the network as can physical servers, with limited modifications to infrastructure set up and software. Cloud architecture can scale horizontally or vertically, according to demand.

Broad network access. Capabilities are available over the network and accessed through standard mechanisms that promote use by heterogeneous platforms (e.g., mobile phones, laptops, and PDAs).

Location independence. There is a sense of location independence, in that the customer generally has no control or knowledge over the exact location of the provided resources, but may be able to specify location at a higher level of abstraction (e.g., country, state, or datacenter).

Reliability improves through the use of multiple redundant sites, which makes cloud computing suitable for business continuity and disaster recovery.

COMPUTER CLUSTERS FOR SCALABLE COMPUTING

A computer cluster is a collection of interconnected stand-alone computers which can work together collectively and cooperatively as a single integrated computing resource pool.

We classify clusters using six orthogonal attributes: scalability, packaging, control, homogeneity, programmability, and security.

Scalability

Clustering of computers is based on the concept of modular growth. To scale a cluster from hundreds of uniprocessor nodes to a supercluster with 10,000 multicore nodes is a nontrivial task. The scalability could be limited by a number of factors, such as the multicore chip technology, cluster topology, packaging method, power consumption, and cooling scheme applied. The purpose is to achieve scalable performance constrained by the aforementioned factors. We have to also consider other limiting factors such as the memory wall, disk I/O bottlenecks, and latency tolerance, among others.

Packaging

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

Control

A cluster can be either controlled or managed in a centralized or decentralized fashion. A compact cluster normally has centralized control, while a slack cluster can be controlled either way. In a centralized cluster, all the nodes are owned, controlled, managed, and administered by a central operator. In a decentralized cluster, the nodes have individual owners. For instance, consider a cluster comprising an interconnected set of desktop workstations in a department, where each workstation is individually owned by an employee. The owner can reconfigure, upgrade, or even shut down the workstation at any time. This

lack of a single point of control makes system administration of such a cluster very difficult. It also calls for special techniques for process scheduling, workload migration, checkpointing, accounting, and other similar tasks.

Homogeneity

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

Security

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

- Being exposed, intracluster communication is not secure, unless the communication subsystem performs additional work to ensure privacy and security.
- Outside communications may disrupt intracluster communications in an unpredictable fashion. For instance, heavy BBS traffic may disrupt production jobs.
- Standard communication protocols tend to have high overhead.

CLUSTER DESIGN ISSUES

Scalable Performance:- This refers to the fact that scaling of resources (cluster nodes, memory capacity, I/O bandwidth, etc.) leads to a proportional increase in performance. Of course, both scale-up and scale-down capabilities are needed, depending on application demand or cost-effectiveness considerations. Clustering is driven by scalability. One should not ignore this factor in all applications of cluster or MPP computing systems.

Single-System Image (SSI):- A set of workstations connected by an Ethernet network is not necessarily a cluster. A cluster is a single system. For example, suppose a workstation has a 300 Mflops/second processor, 512 MB of memory, and a 4 GB disk and can support 50 active users and 1,000 processes. By clustering 100 such workstations, can we get a single

system that is equivalent to one huge workstation, or a megastation, that has a 30 Gflops/second processor, 50 GB of memory, and a 400 GB disk and can support 5,000 active users and 100,000 processes? This is an appealing goal, but it is very difficult to achieve. SSI techniques are aimed at achieving this goal.

Availability Support:- Clusters can provide cost-effective HA capability with lots of redundancy in processors, memory, disks, I/O devices, networks, and operating system images. However, to realize this potential, availability techniques are required.

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

Internode Communication:-Because of their higher node complexity, cluster nodes cannot be packaged as compactly as MPP nodes. The internode physical wire lengths are longer in a cluster than in an MPP. This is true even for centralized clusters. A long wire implies greater interconnect network latency. But more importantly, longer wires have more problems in terms of reliability, clock skew, and cross talking. These problems call for reliable and secure communication protocols, which increase overhead. Clusters often use commodity networks (e.g., Ethernet) with standard protocols such as TCP/IP.

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

Cluster Family Classification

Based on application demand, computer clusters are divided into three classes:

Compute clusters These are clusters designed mainly for collective computation over a single large job. A good example is a cluster dedicated to numerical simulation of weather conditions.The compute clusters do not handle many I/O operations, such as database services. When a single compute job requires frequent communication among the cluster nodes, the cluster must share a dedicated network, and thus the nodes are mostly homogeneous and tightly coupled. This type of clusters is also known as a Beowulf cluster.
When the nodes require internode communication over a small number of heavy-duty nodes, they are essentially known as a computational grid. Tightly coupled compute clusters are designed for supercomputing applications. Compute clusters apply middleware such as a message-passing interface (MPI) or Parallel Virtual Machine (PVM) to port programs to a wide variety of clusters.

High-Availability:- Clusters HA (high-availability) clusters are designed to be fault-tolerant and achieve HA of services. HA clusters operate with many redundant nodes to sustain faults or failures. The simplest HA cluster has only two nodes that can fail over to each other. Of course, high redundancy provides higher availability. HA clusters should be designed to avoid all single points of failure. Many commercial HA clusters are available for various operating systems.

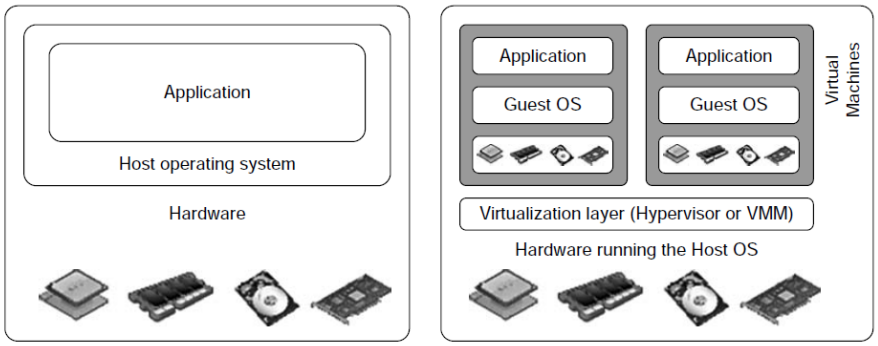
Load-balancing:- clusters These clusters shoot for higher resource utilization through load balancing among all participating nodes in the cluster. All nodes share the workload or function as a single virtual machine (VM). Requests initiated from the user are distributed to all node computers to form a cluster. This results in a balanced workload among different machines, and thus higher resource utilization or higher performance. Middleware is needed to achieve dynamic load balancing by job or process migration among all the cluster nodes.

VIRTUAL MACHINES & VIRTUALIZATION OF CLUSTERS & DATA CENTERS

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

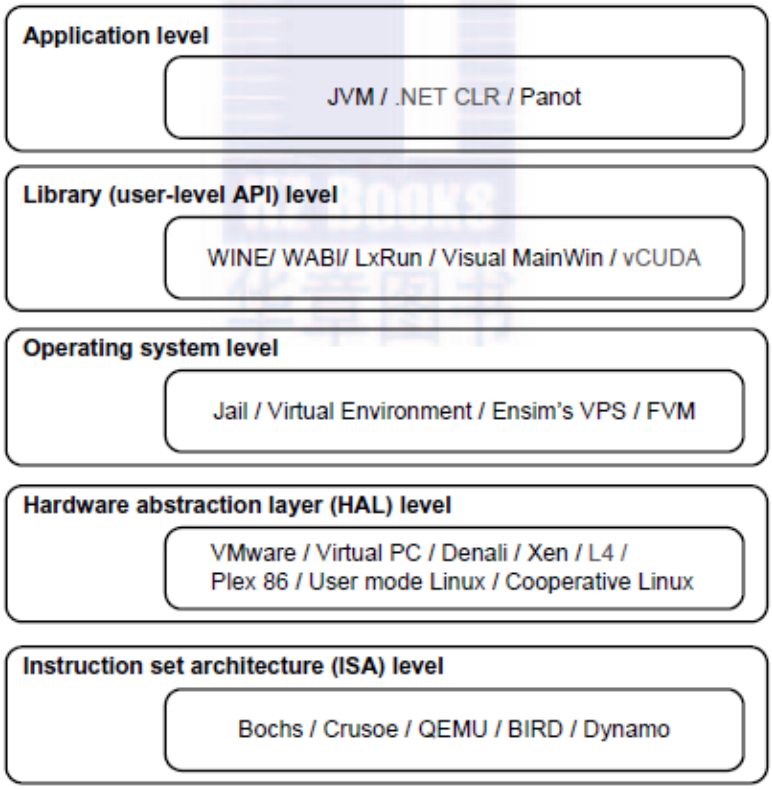
The purpose of a VM is to enhance resource sharing by many users and improve computer performance in terms of resource utilization and application flexibility. Hardware resources (CPU, memory, I/O devices, etc.) or software resources (operating system and software libraries) can be virtualized in various functional layers.

Levels of Virtualization: The below figure shows architecture of a computer system before and after virtualization, where VMM stands for virtual machine monitor.



(a) Traditional computer (b) After virtualization

The main function of the software layer for virtualization is to virtualize the physical hardware of a host machine into virtual resources to be used by the VMs, exclusively. This can be implemented at various operational levels which includes the following levels.



At the ISA level, virtualization is performed by emulating a given ISA by the ISA of the host machine. For example, MIPS binary code can run on an x86-based host machine with the help of ISA emulation. With this approach, it is possible to run a large amount of legacy binary code written for various processors on any given new hardware host machine. Instruction set emulation leads to virtual ISAs created on any hardware machine.

The basic emulation method is through code interpretation. An interpreter program interprets the source instructions to target instructions one by one. One source instruction may require tens or hundreds of native target instructions to perform its function. Obviously, this process is relatively slow. For better performance, dynamic binary translation is desired. This approach translates basic 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. Instruction set emulation requires binary translation and optimization. A virtual instruction set architecture (V-ISA) thus requires adding a processor-specific software translation layer to the compiler.

Hardware Abstraction Level

Hardware-level virtualization is performed right on top of the bare hardware. On the one hand, this approach generates a virtual hardware environment for a VM. On the other

hand, the process manages the underlying hardware through virtualization. The idea is to virtualize a computer's resources, such as its processors, memory, and I/O devices. The intention is to upgrade the hardware utilization rate by multiple users concurrently. The idea was implemented in the IBM VM/370 in the 1960s. More recently, the Xen hypervisor has been applied to virtualize x86-based machines to run Linux or other guest OS applications.

Operating System Level

This refers to an abstraction layer between traditional OS and user applications. OS-level virtualization creates isolated containers on a single physical server and the OS instances to utilize the hardware and software in data centers. The containers behave like real servers. OS-level virtualization is commonly used in creating virtual hosting environments to allocate hardware resources among a large number of mutually distrusting users. It is also used, to a lesser extent, in consolidating server hardware by moving services on separate hosts into containers or VMs on one server.

Library Support Level

Most applications use APIs exported by user-level libraries rather than using lengthy system calls by the OS. Since most systems provide well-documented APIs, such an interface becomes another candidate for virtualization. Virtualization with library interfaces is possible by controlling the communication link between applications and the rest of a system through API hooks. The software tool WINE has implemented this approach to support Windows applications on top of UNIX hosts. Another example is the vCUDA which allows applications executing within VMs to leverage GPU hardware acceleration.

User-Application Level

Virtualization at the application level virtualizes an application as a VM. On a traditional OS, an application often runs as a process. Therefore, application-level virtualization is also known as process-level virtualization. The most popular approach is to deploy high level language (HLL) VMs. In this scenario, the virtualization layer sits as an application program on top of the operating system, and the layer exports an abstraction of a VM that can run programs written and compiled to a particular abstract machine definition. Any program written in the HLL and compiled for this VM will be able to run on it. The Microsoft .NET CLR and Java Virtual Machine (JVM) are two good examples of this class of VM. Other forms of application-level virtualization are known as application isolation,

application sandboxing, or application streaming. The process involves wrapping the application in a layer that is isolated from the host OS and other applications. The result is an application that is much easier to distribute and remove from user workstations. An example is the LANDesk application virtualization platform which deploys software applications as self-contained, executable files in an isolated environment without requiring installation, system modifications, or elevated security privileges.

VMM Design Requirements and Providers

Virtual Machine Monitor (VMM) is a layer between real hardware and traditional operating systems and it manages the hardware resources of a computing system. Each time programs access the hardware the VMM captures the process. The VMM acts as a traditional OS. One hardware component, such as the CPU, can be virtualized as several virtual copies. Therefore, several traditional operating systems which are the same or different can sit on the same set of hardware simultaneously.

There are three requirements for a VMM:-

- VMM should provide an environment for programs which is essentially identical to the original machine.
- Programs run in this environment should show, at worst, only minor decreases in speed.
- VMM should be in complete control of the system resources.

The hardware resource requirements, such as memory, of each VM are reduced, but the sum of them is greater than that of the real machine installed. The latter qualification is required because of the intervening level of software and the effect of any other VMs concurrently existing on the same hardware. Obviously, these two differences pertain to performance, while the function a VMM provides stays the same as that of a real machine. However, the identical environment requirement excludes the behavior of the usual time-sharing operating system from being classed as a VMM.

VMM should demonstrate efficiency in using the VMs. Compared with a physical machine, no one prefers a VMM if its efficiency is too low. Traditional emulators and complete software interpreters (simulators) emulate each instruction by means of functions or macros. Such a method provides the most flexible solutions for VMMs. However, emulators or simulators are too slow to be used as real machines. 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 software intervention by the VMM.

Complete control of these resources by a VMM includes the following aspects:

(1) The VMM is responsible for allocating hardware resources for programs;

(2) it is not possible for a program to access any resource not explicitly allocated to it; and

(3) it is possible under certain circumstances for a VMM to regain control of resources already allocated.

Virtualization Tools:

Depending on the position of the virtualization layer, there are several classes of VM architectures, namely the hypervisor architecture, paravirtualization, and host-based virtualization. The hypervisor supports hardware-level virtualization on bare metal devices like CPU, memory, disk and network interfaces. The hypervisor software sits directly between the physical hardware and its OS.