UNIT-1
SYSTEMS MODELING, CLUSTERING AND VIRTUALIZATION

Introduction, Scalable computing over the Internet, Technologies for Network based systems. System models for Distributed and Cloud Computing, Software environments for distributed systems and clouds. Performance, Security and Energy Efficiency.

INTRODUCTION

The evolutionary changes that have occurred in parallel, distributed, and cloud computing over the past 30 years, driven by applications with variable workloads and large data sets are described here.

SCALABLE COMPUTING OVER THE INTERNET

Over the past 60 years, computing technology has undergone a series of platform and environment changes. Evolutionary 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. Here the following are discussed.

The age of internet computing**:** Computer technology has gone through five generations of development, with each generation lasting from 10 to 20 years. Successive generations are overlapped in about 10 years. For instance, from 1950 to 1970, a handful of mainframes, including the IBM 360 and CDC 6400. 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. These systems are employed by both consumers and high-end web-scale computing and information services.

The below fig. illustrates the evolution of HPC and HTC systems. On the HPC side, supercomputers (***massively parallel processors*** or MPPs) are gradually replaced by clusters of cooperative computers out of a desire to share computing resources.

On the HTC side, peer-to-peer (P2P) networks are formed for distributed file sharing and content delivery applications. A P2P system is built over many client machines. Peer machines are globally distributed in nature. P2P, cloud computing, and web service platforms are more focused on HTC applications than on HPC applications.

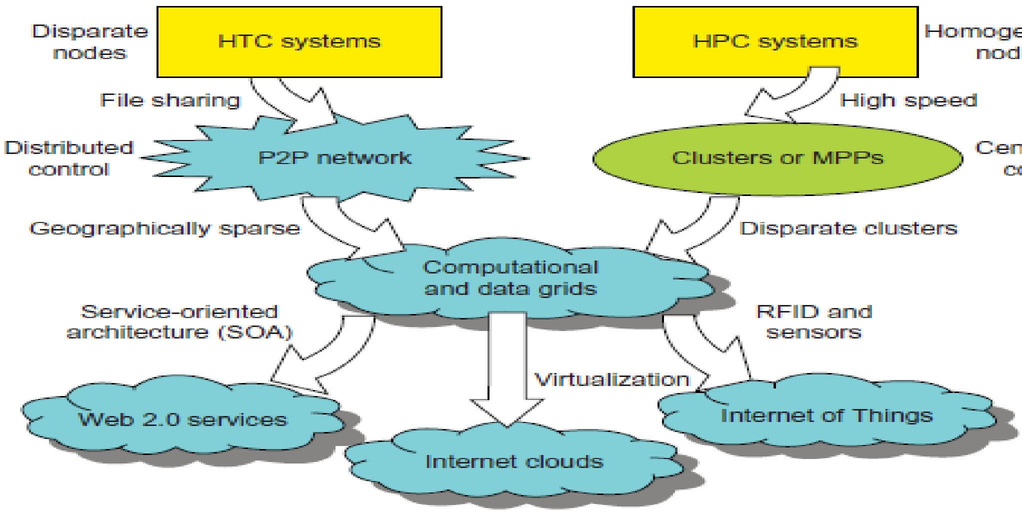


Fig. Evolutionary trend toward parallel, distributed, and cloud computing with clusters,

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

High-Throughput Computing: The development of market-oriented high-end computing systems is undergoing a strategic change from an HPC paradigm to an HTC paradigm. This HTC paradigm pays more attention to high-flux computing. The main application for high-flux computing is in Internet searches and web services by millions or more users simultaneously. The performance goal thus shifts to measure high throughput or the number of tasks completed per unit of time.

Computing Paradigm Distinctions: In general, distributed computing is the opposite of centralized computing. The field of parallel computing overlaps with distributed computing to a great extent. There exists the following types of computing

- **Centralized computing** This is a computing paradigm by which all computer resources are centralized in one physical system.
- **Parallel computing** In parallel computing, all processors are either tightly coupled with centralized shared memory or loosely coupled with distributed memory

- **Distributed computing** This is a field of computer science/engineering that studies distributed systems. A distributed system consists of multiple autonomous computers, each having its own private memory, communicating through a computer network
- **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.

Distributed System Families: In the future, both 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. Future HPC and HTC systems must be able to satisfy this huge demand in computing power in terms of throughput, efficiency, scalability, and reliability.

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.
- **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

Scalable computing trends and new paradigms: Several predictable trends in technology are known to drive computing applications many persons played a good role in this such as Jim Gray, Moore law, Gilder's law were famous among them.

Degrees of Parallelism: bit-level parallelism (BLP) converts bit-serial processing to word-level processing gradually. instruction-level parallelism (ILP), in which the processor executes multiple instructions simultaneously rather than only one instruction at a time. Data-level parallelism (DLP) was made popular through SIMD (single instruction, multiple data) and vector machines using vector or array types of instructions. DLP requires even more hardware support and compiler assistance to work properly. TLP (Task Level Parallesim) is far from being very successful due to difficulty in programming and compilation of code for efficient execution on multicore CMPs. As we move from parallel processing to distribute processing, we will see an increase in computing granularity to job-level parallelism (JLP). It is fair to say that coarse-grain parallelism is built on top of fine-grain parallelism.

Innovative Applications: Both HPC and HTC systems desire transparency in many application aspects. The highlights a few key applications that have driven the development of parallel and distributed systems over the years. These applications spread across many important domains in science, engineering, business, education, health care, traffic control, Internet and web services, military, and government applications. Almost all applications demand computing economics, web-scale data collection, system reliability, and scalable performance.

Applications of High Performance and High Throughput Systems	
Domain	**Specific Applications**
Internet and web services	Data centers, internet search, decision making systems, worm containment cyber security, traffic monitoring
Mission critical applications	Intelligent systems, crisis management, military control and command
Science and engineering	Predication of Earthquakes, Genomic analysis, Scientific simulation

The Trend toward 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.

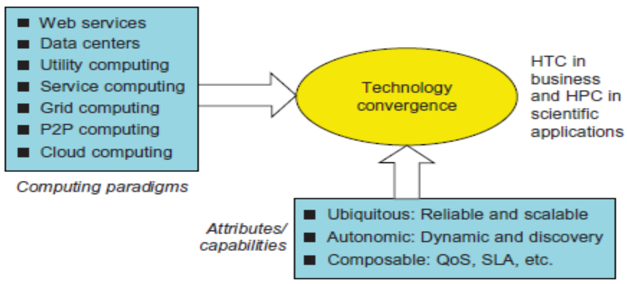


Fig: The vision of computer utilities in modern distributed computing systems

Major technological challenges include all aspects of computer science and engineering. For example, users demand new network efficient processors, scalable memory and storage schemes, distributed OSes.

The Hype Cycle of New Technologies: Any new and emerging computing and information technology may go through a hype cycle, as illustrated in the below fig. 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.

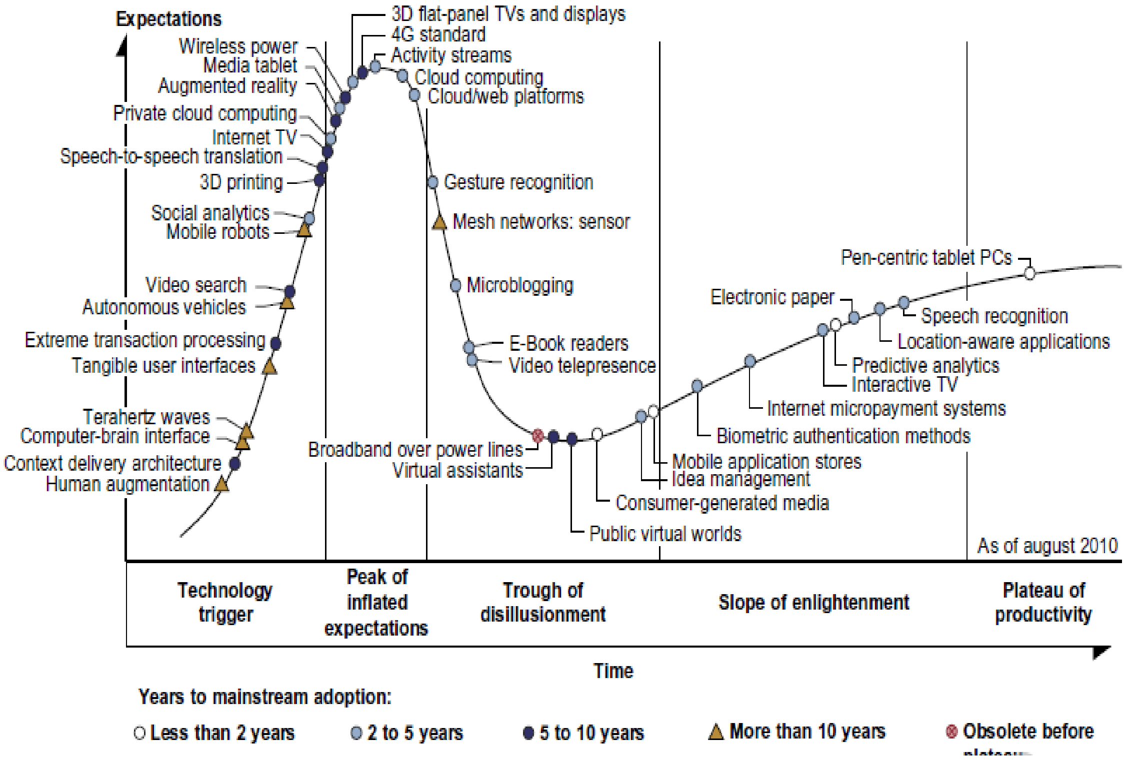


Fig: Hype cycle for Emerging Technologies

The internet of things and cyber physical system: here the following two things are discussed; these evolutionary trends emphasize the extension of the Internet to everyday objects.

The Internet of Things: 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. One can view the IoT as a wireless network of sensors.

In the IoT era, all objects and devices are instrumented, interconnected, and interacted with each other intelligently. This communication can be made between people and things or among the things themselves. Three communication patterns co-exist: namely H2H (human-to-human), H2T (human-to-thing), and T2T (thing-to-thing). Here things include machines such as PCs and mobile phones.

The dynamic connections will grow exponentially into a new dynamic network of networks, called the Internet of Things (IoT). The IoT is still in its infancy stage of development.

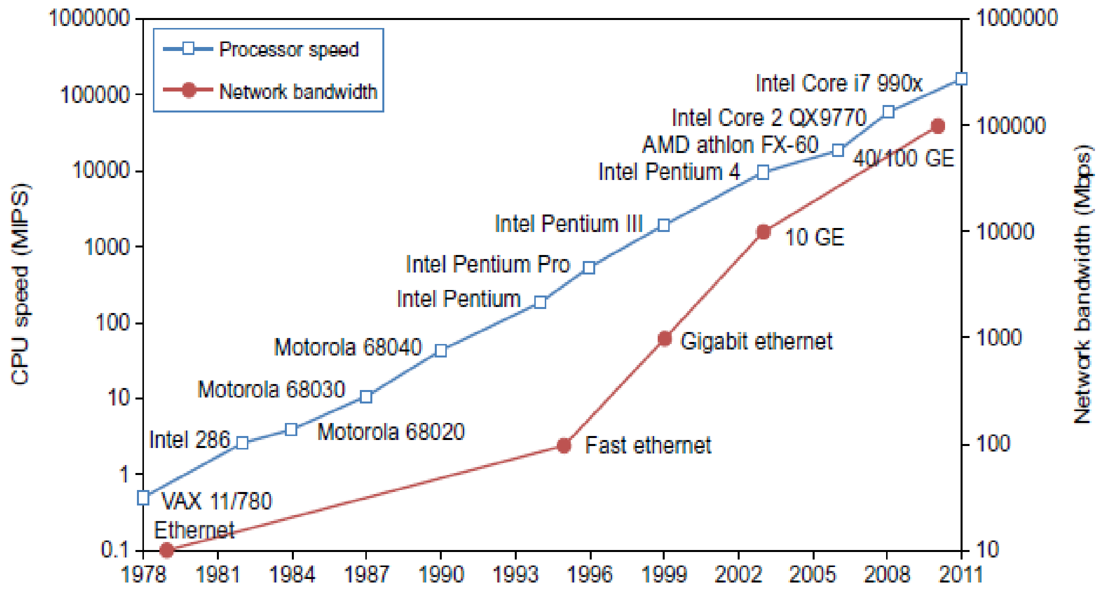
Cyber-Physical Systems : A cyber-physical system (CPS) is the result of interaction between computational processes and the physical world. A CPS integrates "cyber" (heterogeneous, asynchronous) with "physical" (concurrent and information-dense) objects. A CPS merges the "3C" technologies of computation, communication, and control into an intelligent closed feedback system between the physical world and the Information World.

TECHNOLOGIES FOR NETWORK BASED SYSTEMS

MULTICORE CPUS AND MULTITHREADING TECHNOLOGIES: Consider the growth of component and network technologies over the past 30 years. They are crucial to the development of HPC and HTC systems. In the below fig., 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).

Advances in CPU Processors : Today, advanced CPUs or microprocessor chips assume a multicore architecture with dual, quad, six, or more processing cores. These processors exploit parallelism at ILP and TLP levels. Processorspeed growth is plotted in the upper curve in the above fig. across generations of microprocessors or CMPs. We see growth from 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. As the figure shows, Moore's law has proven to be pretty accurate in this case. The clock rate for these processors increased from 10 MHz for the Intel 286 to 4 GHz for the Pentium 4 in 30 years.

Multicore CPU and Many-Core GPU Architectures : Multicore 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.



.

Fig: comparison of CPU speeds with network bandwidth

Multithreading Technology: there exist five processor categories, from left to right: a four-issue superscalar processor, a fine-grain multithreaded processor, a coarse-grain multithreaded processor, a two-core CMP, and a simultaneous multithreaded (SMT) processor. 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.

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

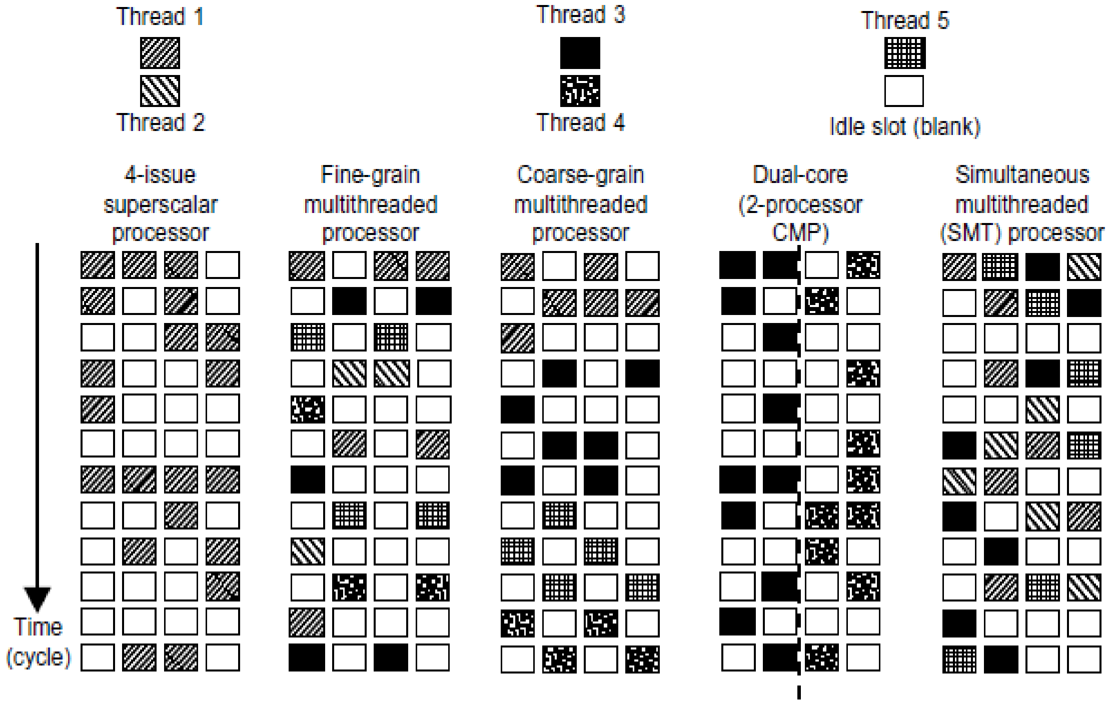


Fig: Five micro-architectures in modern CPU processors

GPU COMPUTING TO EXASCALE AND BEYOND: 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, and are used in nearly every computer on the market today. Some GPU features were also integrated into certain CPUs. Unlike CPUs, 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.

How GPUs Work: Early GPUs functioned as coprocessors attached to the CPU. Today, 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. 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.

***GPU Programming Model* : the below fig** 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.

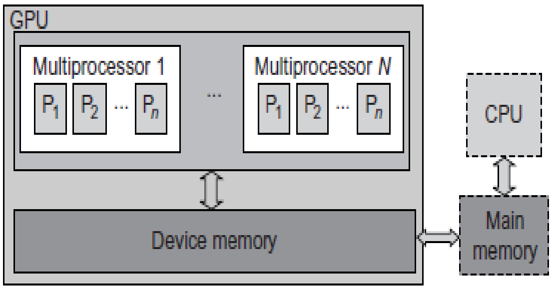
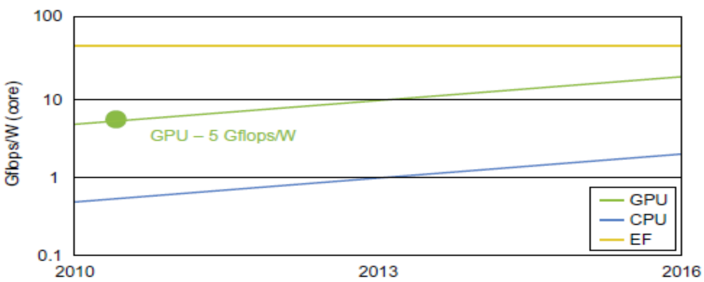


Fig: The use of a GPU along with a CPU for massively parallel execution in hundreds or thousands of processing cores.

Power Efficiency of the GPU: Bill Dally of Stanford University considers power and massive parallelism as the major benefits of GPUs over CPUs for the future. By extrapolating current technology and computer architecture, it was estimated that 60 Gflops/watt per core is needed to run an exaflops system. the below fig compares the CPU and GPU in their performance/power ratio measured in Gflops/ watt per core.



In 2010, the GPU had a value of 5 Gflops/watt at the core level, compared with less than 1 Gflop/watt per CPU core. This may limit the scaling of future

Fig: The GPU Performance

supercomputers. However, the GPUs may close the gap with the CPUs. Data movement dominates power consumption. One needs to optimize the storage hierarchy and tailor the memory to the applications.

MEMORY, STORAGE AND WIDE AREA NETWORKING

Memory Technology : The upper curve in the below fig. plots the growth of DRAM chip capacity from 16 KB in 1976 to 64 GB in 2011. This shows that memory chips have experienced a 4x increase in capacity every three years. Memory access time did not improve much in the past. In fact, the memory wall problem is getting worse as the processor gets faster. For hard drives, capacity increased from 260 MB in 1981 to 250 GB in 2004. The Seagate Barracuda XT hard drive reached 3 TB in 2011. This represents an approximately 10x increase in capacity every eight years. The capacity increase of disk arrays will be even greater in the years to come.

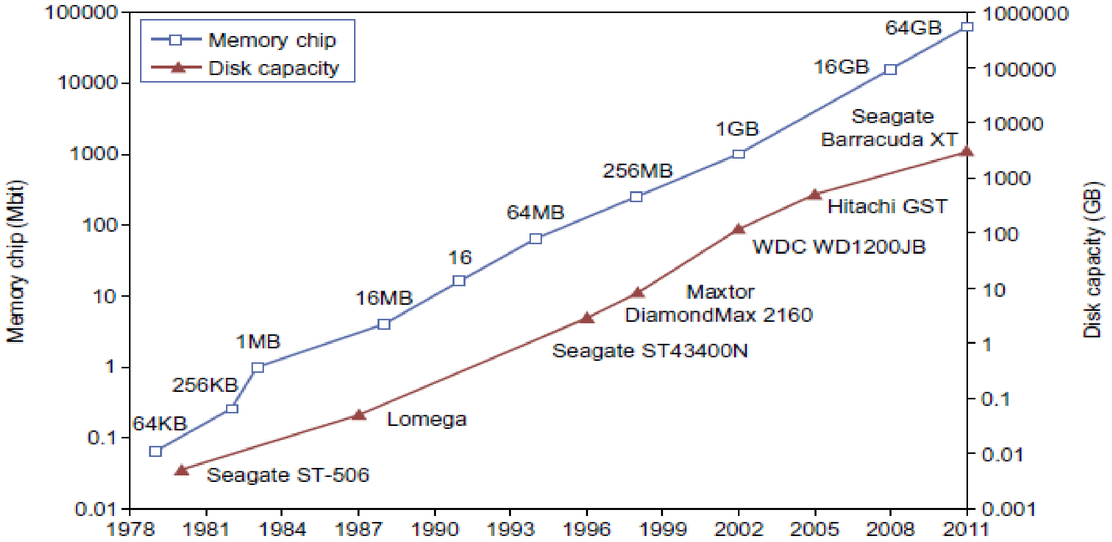


Fig: Improvement in memory and disk technologies over 33 years. The Seagate Barracuda XT disk has a capacity of 3 TB in 2011.

Disks and Storage Technology : disks or disk arrays have exceeded 3 TB in capacity. The lower curve in the above fig. shows the disk storage growth in 7 orders of magnitude in 33 years. The rapid growth of flash memory and solid-state drives (SSDs) also impacts the future of HPC and HTC systems. Flash and SSD will demonstrate impressive speedups in many applications. Eventually, power consumption, cooling, and packaging will limit large system development. Power increases linearly with respect to clock frequency and quadratic ally with respect to voltage applied on chips. Clock rate cannot be increased indefinitely

System-Area Interconnects: 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.

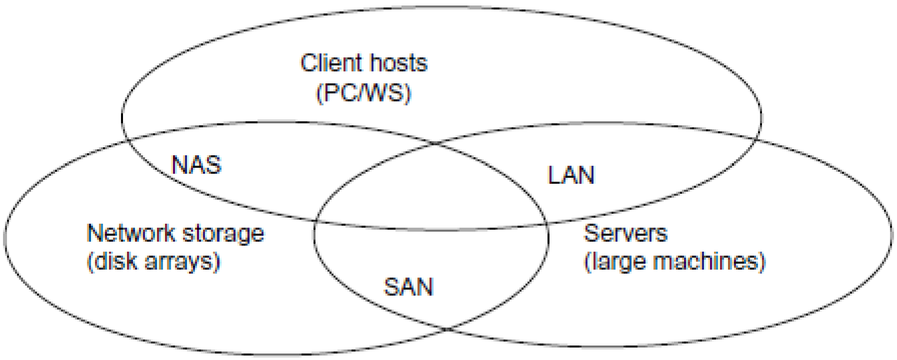


Fig: Three interconnection networks for connecting servers, client hosts, and storage devices; the LAN connects client hosts and servers,

Wide-Area Networking: An increase factor of two per year on network performance was reported, which is faster than Moore's law on CPU speed doubling every 18 months. The implication is that more computers will be used concurrently in the future. High-bandwidth networking increases the capability of building massively distributed systems. The IDC 2010 report predicted that both Infiniti Band and Ethernet will be the two major interconnect choices in the HPC arena.

VIRTUAL MACHINES AND VIRTUALIZATION MIDDLEWARE: Virtual machines (VMs) offer novel solutions to underutilized resources, application inflexibility, software manageability, and security concerns in existing physical machines.

Virtual Machines: 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). The below fig. shows a 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.

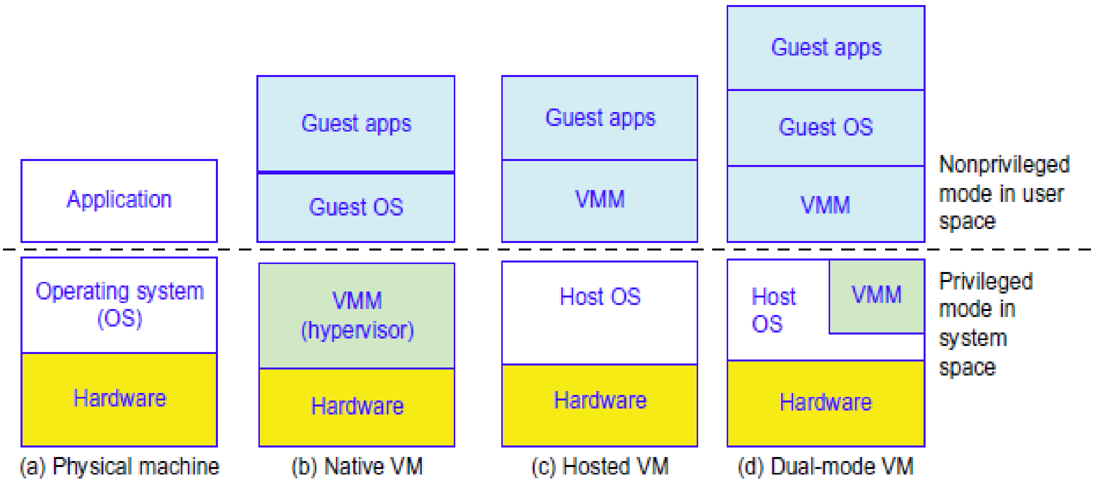


Fig: Three VM architectures in (b), (c), and (d), compared with the traditional physical machine shown in (a)

VM Primitive Operations: These VM operations enable a VM to be provisioned to any available hardware platform. They also enable flexibility in porting distributed application executions. Furthermore, the VM approach will significantly enhance the utilization of server resources. Multiple server functions can be consolidated on the same hardware platform to achieve higher system efficiency. This will eliminate server sprawl via deployment of systems as VMs, which move transparency to the shared hardware.

Virtual Infrastructures: Virtual infrastructure is what connects resources to distributed applications. It is a dynamic mapping of system resources to specific applications. The result is decreased costs and increased efficiency and responsiveness. Virtualization for server consolidation and containment is a good example of this.

DATA CENTER VIRTUALIZATION FOR CLOUD COMPUTING : Here the basic architecture and design considerations of data centers. Cloud architecture is built with commodity hardware and network devices. Almost all cloud platforms choose the popular x86 processors.

Data Center Growth and Cost Breakdown : A large data center may be built with thousands of servers. Smaller data centers are typically built with hundreds of servers. The cost to build and maintain data center servers has increased over the years. According to a 2009 IDC report (see below fig.), typically only 30 percent of data center costs goes toward purchasing IT equipment (such as servers and disks), 33 percent is attributed to the chiller, 18 percent to the uninterruptible power supply (UPS), 9 percent to computer room air conditioning (CRAC), and the remaining 7 percent to power distribution, lighting, and transformer costs.

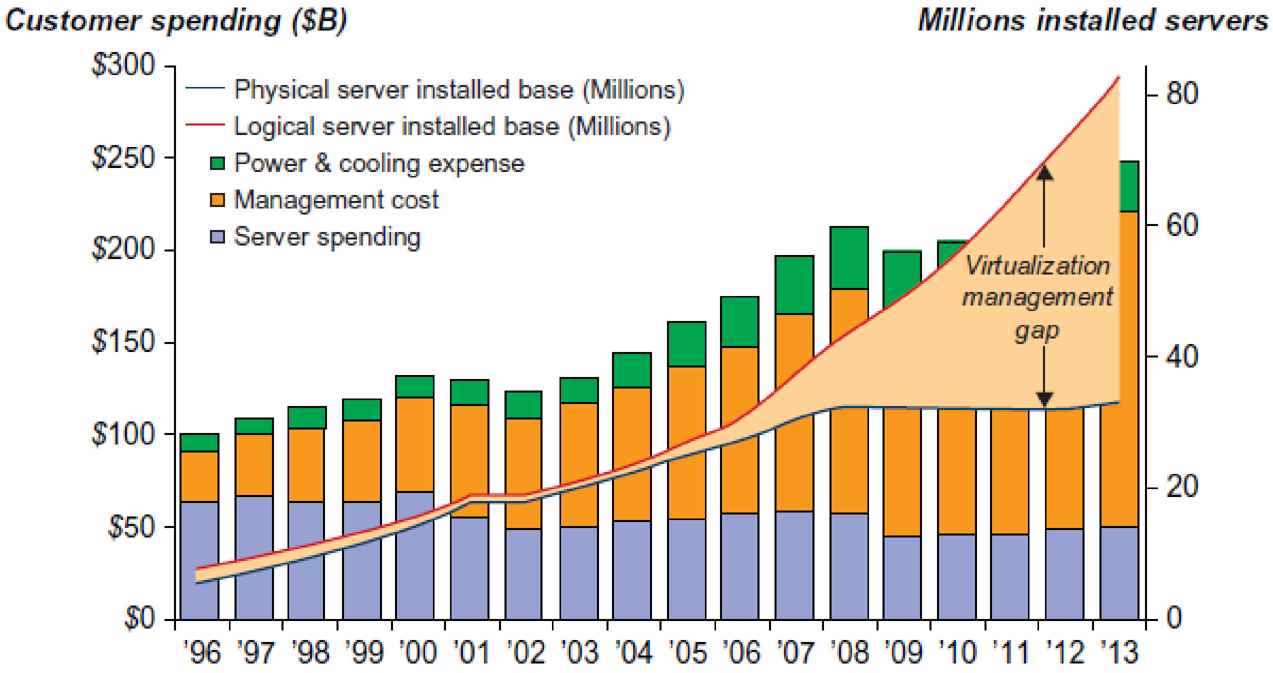


Fig: **Growth and cost breakdown of data centers over the years**

Low-Cost Design Philosophy: High-end switches or routers may be too cost-prohibitive for building data centers. Thus, using high-bandwidth networks may not fit the economics of cloud computing. The software layer handles network traffic balancing, fault tolerance, and expandability. Currently, nearly all cloud computing data centers use Ethernet as their fundamental network technology

Convergence of Technologies: cloud computing is enabled by the convergence of technologies in four areas:

(1) Hardware virtualization and multi-core chips

(2) Utility and grid computing

(3) SOA, Web 2.0, and WS mashups, and

(4) Autonomic computing and data center automation

SYSTEM MODELS FOR DISTRIBUTED AND CLOUD COMPUTING

Distributed and cloud computing systems are built over a large number of autonomous computer nodes. These node machines are interconnected by SANs, LANs, or WANs in a hierarchical manner. the below table shows the classification of massive systems are classified into four groups: clusters, P2P networks, computing grids, and Internet clouds over huge data centers.

Classification of Parallel and Distributed Computing systems				
Functionality applications	**Computer clusters**	**Peer to peer networks**	**Data / computational Grids**	**Cloud platforms**
Control and Resources management	Homogeneous nodes with distributed running and control	Autonomous client nodes, free out and in.	Server oriented authentication, centralized control	Resources management of network, servers and storages dynamically
Architecture, Network connectivity and size	Network of compute nodes which are interconnected by SAN, LAN	Flexible network of client machine logically connected by an overlay network	Heterogeneous clusters inter connected by high speed networks	Virtualized cluster of servers over data centers with SLA
Applications and Network Centric Services	Search engines, high performance computing	Most appealing to business file sharing	Global problem solving and data center services.	Outsourced computing services, Utility computing, upgraded web search

CLUSTERS OF COOPERATIVE COMPUTERS: A computing cluster consists of interconnected stand-alone computers which work cooperatively as a single integrated computing resource

Cluster Architecture : the below fig,.shows the architecture of a typical server cluster built around a low-latency, 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 interconnection network can be built with multiple levels of Gigabit Ethernet, Myrinet, or Infini Band switches The cluster is connected to the Internet via a virtual private network (VPN) gateway. The gateway IP address locates the cluster. The system image of a computer is decided by the way the OS manages the shared cluster resources.

Single-System Image: As said by Greg Pfister has indicated that an ideal cluster should merge multiple system images into a single-system image (SSI). Cluster designers desire a cluster operating system or some middleware to support SSI at various levels, including the sharing of CPUs, memory, and I/O across all cluster nodes. An SSI is an illusion created by software or hardware that presents a collection of resources as one integrated, powerful resource. SSI makes the cluster appear like a single machine to the user. A cluster with multiple system images is nothing but a collection of independent computers

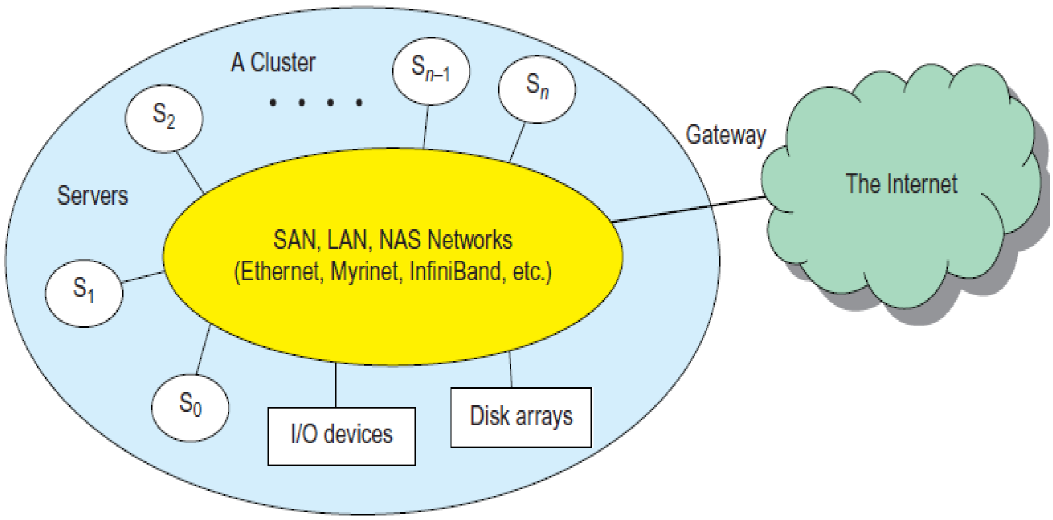


Fig: **A cluster of servers interconnected by a high-bandwidth SAN or LAN**

Hardware, Software, and Middleware Support : Clusters exploring massive parallelism are commonly known as MPPs. Almost all HPC clusters in the Top500 list are also MPPs. The building blocks are computer nodes (PCs, workstations, servers, or SMP), special communication software such as PVM or MPI, and a network interface card in each computer node. 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.

Major Cluster Design Issues : The cluster benefits come from scalable performance, efficient message passing, high system availability, seamless fault tolerance, and cluster-wide job management, as summarized in the below table.

GRID COMPUTING INFRASTRUCTURES: Internet services such as the Telnet command enables a local computer to connect to a remote computer. A web service such as HTTP enables remote access of remote web pages. Grid computing is envisioned to allow close interaction among applications running on distant computers simultaneously.

Critical clusters design issues and feasible implementations		
Features	**Functional characterization**	**Feasible implementation**
Dynamic load balancing	Workload is balanced for all processing nodes with failure recovery	Job replications, process migration monitoring workloads,
Efficient Communications	Message passing system reduction	Active messaging, fast message passing.
Support and Availability	Software and hardware support for High Availability	Point checking, failover, failback, rollback recovery.
Single System Image (SSI)	SSI can be achieved with the help of both software and hardware support.	Hardware mechanisms
Hardware fault tolerance	Automated failure management in order to eliminate all single points of failure	Multiple power supply, RAID, component redundancy.

Computational Grids: the below fig. shows an example computational grid built over multiple resource sites owned by different organizations. The resource sites offer complementary computing resources, including workstations, large servers, a mesh of processors, and Linux clusters to satisfy a chain of computational needs.

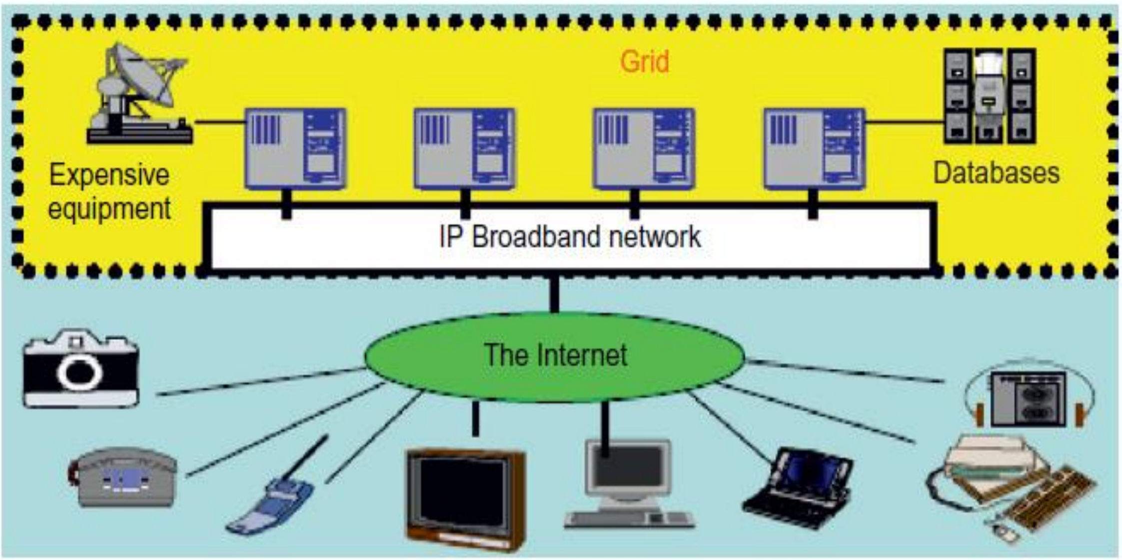


Fig: **Computational grid or data grid providing computing utility, data, and information services**

The grid is built across various IP broadband networks including LANs and WANs already used by enterprises or organizations over the Internet. The grid is presented to users as an integrated resource pool as shown in the upper half of the figure.

Grid Families: Grid technology demands new distributed computing models, software/middleware support, network protocols, and hardware infrastructures. National grid projects are followed by industrial grid platform development by IBM, Microsoft, Sun, HP, Dell, Cisco, EMC, Platform Computing, and others. New grid service providers (GSPs) and new grid applications have emerged rapidly. Grid systems are classified in essentially two categories: computational or data grids and P2P grids, which are shown in the below table.

Two Grid Computing Infrastructure and Representative systems		
Design issues	**Computational and data grids**	**P2P Grids**
Development Lessons Learned	Restricted user groups, middleware bugs, protocols related to acquire resources	Limited to few aps,Unreliable user-contributed resources.
Representative systems	Different grids which are built in UK,US and china	Fight Aid @home, SETI @home
Grid Application Reported	National Grid initiatives, Distributed supercomputing	All resources from client machines. Open grid with P2P flexibility

PEER TO PEER NETWORK FAMILIES: The P2P architecture offers a distributed model of networked systems. First, a P2P network is client-oriented instead of server-oriented.

P2P Systems: 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. The below fig shows the architecture of a P2P network at two abstraction levels. 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.

Overlay Networks: 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 as shown

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.

P2P Application Families: Based on application, P2P networks are classified into four groups, as shown in the below table. This includes many popular P2P networks such as Gnutella, Napster, and BitTorrent, among others. Collaboration P2P networks include MSN or Skype chatting, instant messaging, and collaborative design, among others. The third family is for distributed P2P computing in specific applications.

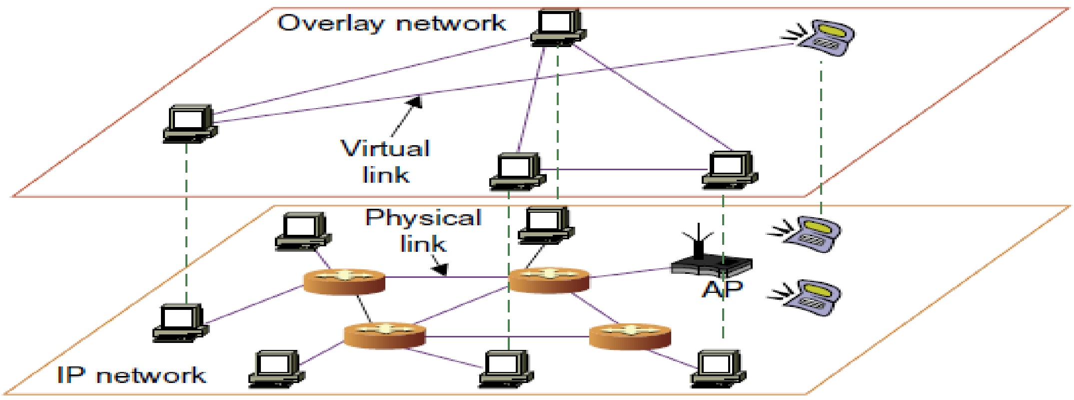


Fig: The structure of a P2P system by mapping a physical IP network to an overlay network built with virtual links.

Major categories of PP Network Families				
System Features	**Distributed File sharing**	**Collaborative Platform**	**Distributed P2P computing**	**P2P platform**
Operational problems	Copyright violation and loose security	Lagging of trust in Privacy ,spam, peer collusion	Peer collusion, selfish partners and security holes	Lagging of Protection protocol standards
Example systems	Bit torrent, Aimster,KaZaA	Multiplayer games,skype	Geonome@home, SETI @ home	.NET, fighting Aid @ home
Attractive applications	Distribution of content among MP2, video, open software	Collaboration of design and gaming	Exploration of Scientific and social networking	Open networks related to public resources

P2P Computing Challenges: P2P computing faces three types of heterogeneity problems in hardware, software, and network requirements.

CLOUD COMPUTING OVER THE INTERNET: "A cloud is a pool of virtualized computer resources. A cloud can host a variety of different workloads, including batch-style backend jobs and interactive and user-facing applications, it consists of the following.

Internet Clouds: Cloud computing applies a virtualized platform with elastic resources on demand by provisioning hardware, software, and data sets dynamically

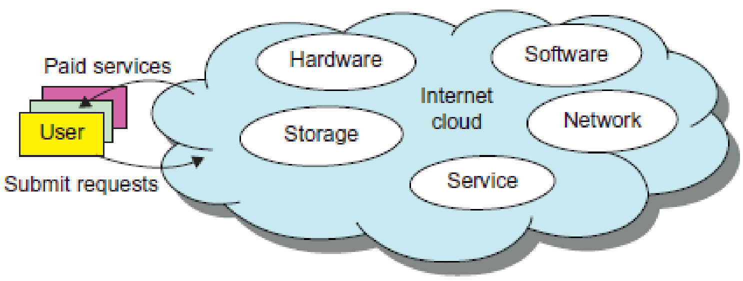


Fig: Virtualized resources from data centers to form an Internet cloud

As shown in the above fig**.** The idea is to move desktop computing to a service-oriented platform using server clusters and huge databases at data centers. Cloud computing leverages its low cost and simplicity To benefit both users and providers. The cloud ecosystem must be designed to be secure, trustworthy, and dependable. Some computer users think of the cloud as a centralized resource pool.

The Cloud Landscape **:** Traditionally, a distributed computing system tends to be owned and operated by an autonomous administrative domain (e.g., a research laboratory or company) for on-premises computing needs the below fig.shows various service models in the cloud which are as follows.

- **Infrastructure as a Service (IaaS):** This model puts together infrastructures demanded by users—namely servers, storage, networks, and the data center fabric.
- **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.

- **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.

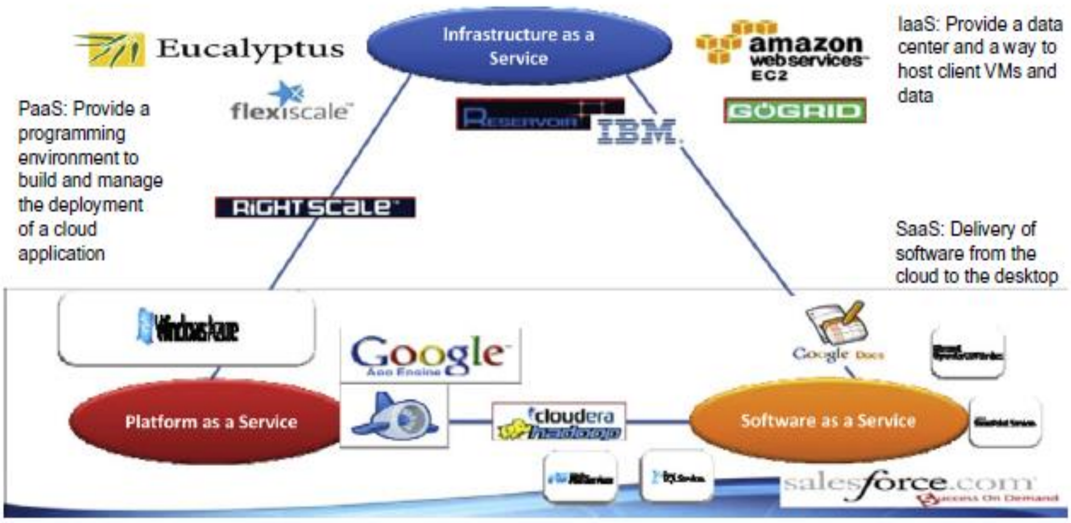


Fig: Three cloud service models in a cloud landscape of major providers

SOFTWARE ENVIRONMENTS FOR DISTRIBUTED SYSTEMS AND CLOUDS

Here popular software environments for using distributed and cloud computing systems are discussed.

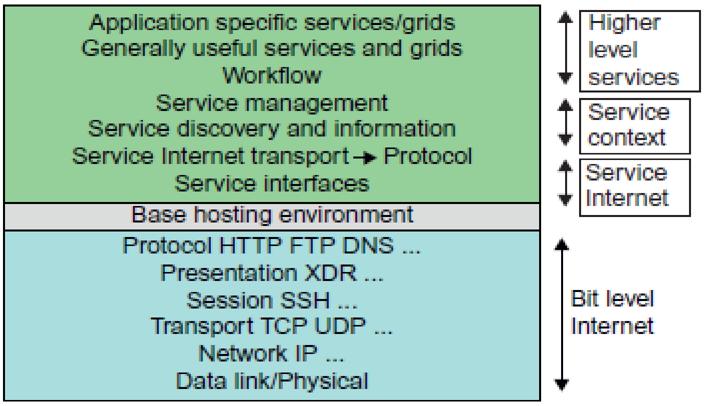
- **Service Oriented Architecture (SOA)**
- **Trends toward Distributed Operating Systems**
- **parallel and Distributed Programming models**

SERVICE ORIENTED ARCHITECTURE (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. These architectures build on the traditional seven Open Systems Interconnection (OSI) layers that provide the base networking abstractions.

Layered Architecture for Web Services and Grids: 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.

Web Services and Tools: Loose coupling and support of heterogeneous implementations make services more attractive than distributed objects.



The above fig corresponds to two choices of service architecture: web services or REST (Representation state transfer) systems. 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.

The Evolution of SOA: As shown in the below fig. 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 inter clouds), and systems of systems in general. A large number of sensors provide data-collection services, denoted in the figure as SS (sensor service). A sensor can be a ZigBee device, a Bluetooth device, a WiFi access point, a personal computer, a GPA, or a wireless phone, among other things. Raw data is collected by sensor services. All the SS devices interact with large or small computers, many forms of grids, databases, the compute cloud, the storage cloud, the filter cloud, the discovery cloud, and so on. Filter services (fs in the figure) are used to eliminate unwanted raw data, in order to respond to specific requests from the web, the grid, or web services.

Grids versus Clouds: The boundary between grids and clouds are getting blurred in recent years. In general, a grid system applies static resources, while a cloud emphasizes elastic resources. For some researchers, the differences between grids and clouds are limited only in dynamic resource allocation based on virtualization and autonomic computing. One can build a grid out of multiple clouds. This type of grid can do a better job than a pure cloud, because it can explicitly support negotiated resource allocation. Thus one may end up building with a system of systems: such as a cloud of clouds, a grid of clouds, or a cloud of grids, or inter-clouds as a basic SOA architecture.

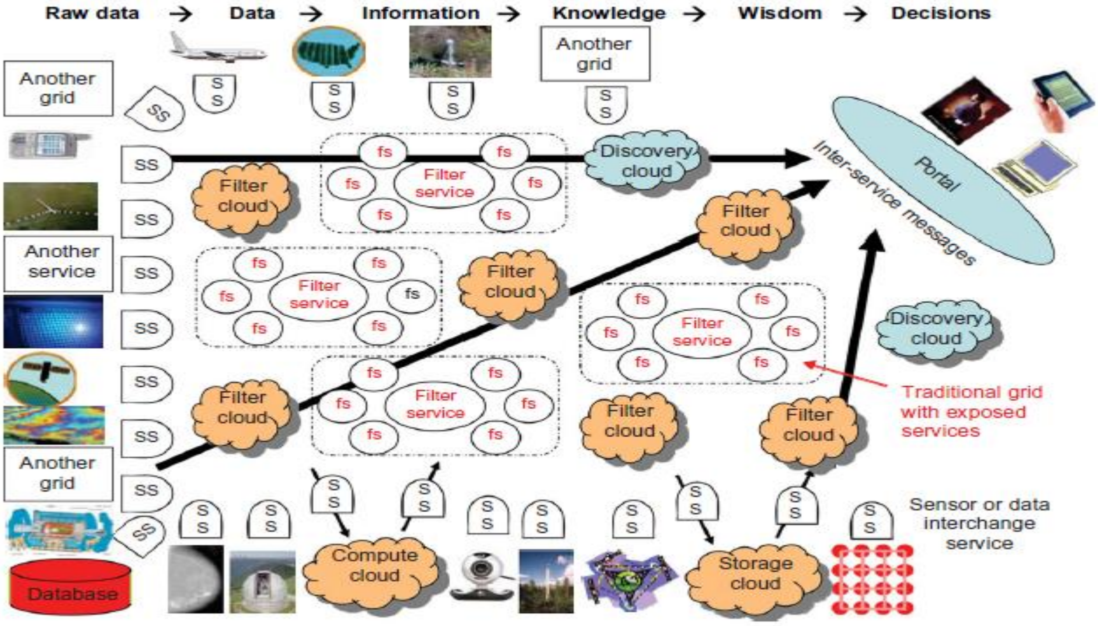


Fig: The evolution of SOA:

TRENDS TOWARD DISTRIBUTED OPERATING SYSTEMS: The computers in most distributed systems are loosely coupled. Thus, a distributed system inherently has multiple system images. This is mainly due to the fact that all node machines run with an independent operating system.

Distributed Operating System: Tanenbaum 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 communication. **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

Amoeba versus DCE: DCE is a middleware-based system for distributed computing environments. The Amoeba was academically developed at Free University in the Netherlands. The Open Software Foundation (OSF) has pushed the use of DCE for distributed computing.

MOSIX2 for Linux Clusters: MOSIX2 is a distributed OS, which runs with a virtualization layer in the Linux environment. This layer provides a partial single-system image to user applications. MOSIX2 supports both sequential and parallel applications, and discovers resources and migrates software processes among Linux nodes. MOSIX2 can manage a Linux cluster or a grid of multiple clusters.

Transparency in Programming Environments**:** The user data, applications, OS, and hardware are separated into four levels. Data is owned by users, independent of the applications. The OS provides clear interfaces, standard programming interfaces, or system calls to application programmers.

Features comparison of three distributed operating systems

Distributed OS Functionality	**AMOEBA Developed at Vrije University**	**DCE as OSF/1 by Open Software Foundation**	**MOSIX for Linux Clusters at Hebrew University**
Distributed OS Architecture	Uses many servers to handle files	System supports RPC, threads and security	Process migration, load balancing
Virtualization support, middleware, OS kernel	A special microkernel for handling low level process	Directory, file, time and security services	Usage of multiple cluster in cloud
Current system status and history	Tested in European community	On the top of UNIX, VMX user extension is built	Used in LINUX, HOC and GPU Clusters

In future cloud infrastructure, the hardware will be separated by standard interfaces from the OS. Thus, users will be able to choose from different OSes on top of the hardware devices they prefer to use. To separate user data from specific application programs, users can enable cloud applications as SaaS.

PARALLEL AND DISTRIBUTED PROGRAMMING MODELS: four programming models for distributed computing with expected scalable performance and application flexibility are discussed here. The below table summarizes three of these models, along with some software tool sets developed in recent years.

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

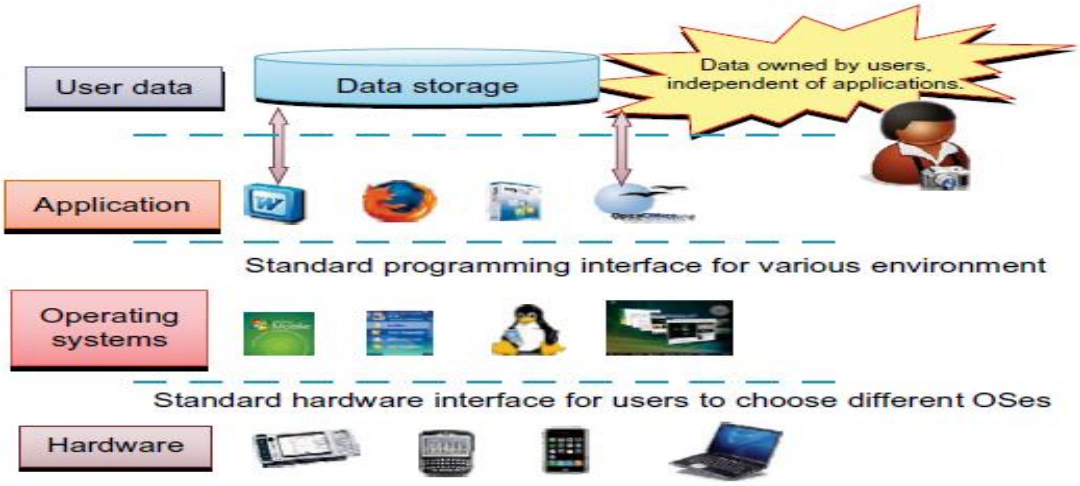


Fig: an ideal model for cloud computing.

Standards	Service Functionalities	Key features and security infrastructure
IBM Grid Tool Bock	Linux and Aix which are built on the top of Globus Toolkit	Grants Access, Grid services, Uses Simple CA
OGSA (Open Grid Services Architectures) Standard	Provides common grid service standard for general public use	Dynamic policies, multiple security mechanisms
Globus Toolkits	Globus security infrastructure (GSI) and general security services	SSL, Proxy, delegation and GSS API for message confidentiality and integrity

Table: Grid Standards and Toolkits for Scientific and Engineering Applications

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. A typical MapReduce computation process can handle terabytes of data on tens of thousands or more client machine

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.

***Open Grid Services Architecture (OGSA)*:** OGSA as a common standard for general public use of grid services. Genesis II is a realization of OGSA. 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.

PERFORMANCE, SECURITY AND ENERGY EFFICIENCY

Here the fundamental design principles along with rules of thumb for building massively distributed computing systems. Coverage includes scalability, availability, programming models, and security issues in clusters, grids, P2P networks, and Internet clouds are discussed.

- **performance Metrics and Scalability Analysis**
- **Fault Tolerance and System Availability**
- **Network Threats and Data integrity**
- **Energy Efficiency in distributed Computing**

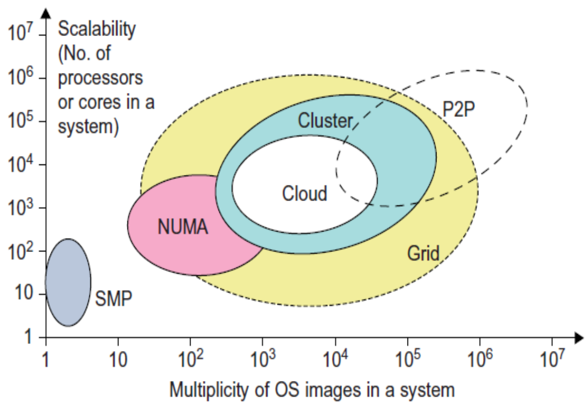
PERFORMANCE METRICS AND SCALABILITY ANALYSIS: Performance metrics are needed to measure various distributed systems which are described as follows,

Performance Metrics : CPU speed and network bandwidth are measures in MIPS a, to estimate processor and network performance. In a distributed system, performance is attributed to a large number of factors. System throughput is often measured in MIPS, Tflops (tera floating-point operations per second), or TPS (transactions per second). Other measures include job response time and network latency.

Dimensions of Scalability: The following dimensions of scalability are characterized in parallel and distributed systems:

- **Size scalability**: This refers to achieving higher performance or more functionality by increasing the machine size.
- **Software scalability:** This refers to upgrades in the OS or compilers, adding mathematical and engineering libraries, porting new application software, and installing more user-friendly programming environments.
- **Application scalability** :This refers to matching problem size scalability with machine size scalability
- **Technology scalability** This refers to a system that can adapt to changes in building technologies, such as the component and networking technologies.

Scalability versus OS Image Count: scalable performance is estimated against the multiplicity of OS images in distributed systems deployed up to 2010.



Scalable performance implies that the system can achieve higher speed by adding more processors or servers, enlarging the physical node's memory size, extending the disk capacity, or adding more I/O channels. The OS image is counted by the number of independent OS images observed in a cluster, grid,

Fig: System scalability versus multiplicity of OS images based on 2010 technology

P2P network, or the cloud. SMP and NUMA are included in the comparison. NUMA (nonuniform memory access) machines are often made out of SMP nodes with distributed, shared memory. A NUMA machine can run with multiple operating systems, and can scale to a few thousand processors communicating with the MPI library.

Amdahl's Law: Consider the execution of a given program on a uni processor workstation with a total execution time of T minutes. Now, let's say the program has been parallelized or partitioned for parallel execution on a cluster of many processing nodes. Assume that a fraction α of the code must be executed sequentially, called the sequential bottleneck. Amdahl's Law states that the speedup factor of using the n-processor system over the use of a single processor is expressed by

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

Problem with Fixed Workload: In Amdahl's law, we have assumed the same amount of workload for both sequential and parallel execution of the program with a fixed problem size or data set. This was called fixed-workload speedup. To execute a fixed workload on n processors, parallel processing may lead to a system efficiency defined as follows

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

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

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

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

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

FAULT TOLERANCE AND SYSTEM AVAILABILITY: In addition to performance, system availability and application flexibility are two other important design goals in a distributed computing system.

System Availability: HA (high availability) is desired in all clusters, grids, P2P networks, and cloud systems. A system is highly available if it has a long mean time to failure (MTTF) and a short mean time to repair (MTTR). System availability is formally defined as follows:

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

NETWORK THREATS AND DATA INTEGRITY: Clusters, grids, P2P networks, and clouds demand security and copyright protection if they are to be accepted in today's digital society, which can be achieved by the following methods.

Threats to Systems and Networks: 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 which are described in the below fig.

Security Responsibilities: Three security requirements are often considered: confidentiality, integrity, and availability for most Internet service providers and cloud users. In the order of SaaS, PaaS, and IaaS, the providers gradually release the responsibility of security control to the cloud users.

Copyright Protection: 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. One can develop a proactive content poisoning scheme to stop colluders and pirates from alleged copyright infringements in P2P file sharing. Pirates are detected in a timely manner with identity-based signatures and time stamped tokens.

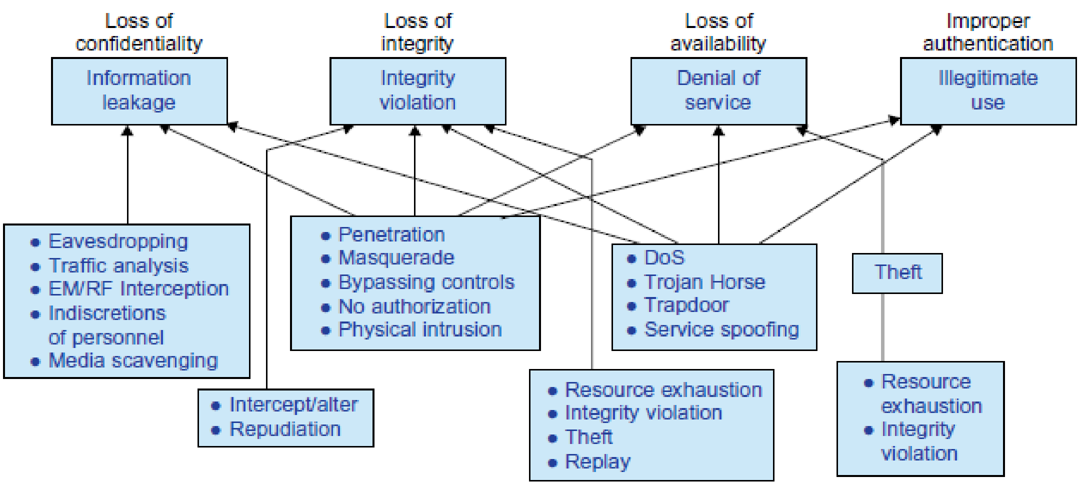


Fig: Various system attacks and network threats to the cyberspace, resulting 4 types of losses

System Defense Technologies: 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. 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.

Data Protection Infrastructure: 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.

ENERGY EFFICIENCY IN DISTRIBUTED COMPUTING: Primary performance goals in conventional parallel and distributed computing systems are high performance and high throughput, considering some form of money performance reliability.

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

Reducing Energy in Active Servers**:** it is also necessary to apply appropriate techniques to decrease energy consumption in active distributed systems with negligible influence on their performance. Power management issues in distributed computing platforms can be categorized into four layers as shown in the below fig.

Application Layer**:** most user applications in science, business, engineering, and financial areas tend to increase a system's speed or quality an application's energy consumption depends strongly on the number of instructions needed to execute the application and the number of transactions with the storage unit (or memory). These two factors (compute and storage) are correlated and they affect completion time.

Middleware Layer: The middleware layer acts as a bridge between the application layer and the resource layer. This layer provides resource broker, communication service, task analyzer, task scheduler, security access, reliability control, and information service capabilities.

Resource Layer: The resource layer consists of a wide range of resources including computing nodes and storage units. This layer generally interacts with hardware devices and the operating system; therefore, it is responsible for controlling all distributed resources in distributed computing systems.

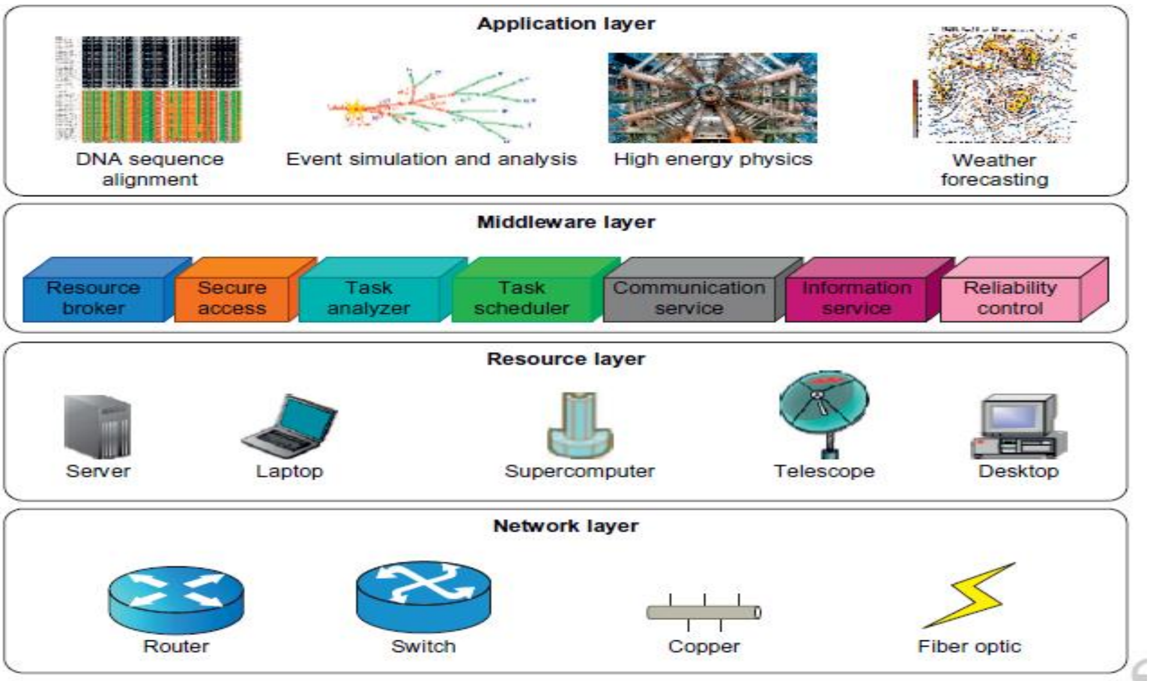


Fig: Four operational layers of distributed computing systems.

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

- The models should represent the networks comprehensively as they should give a full understanding of interactions among time, space, and energy.
- New, energy-efficient routing algorithms need to be developed.

DVFS Method for Energy Efficiency**:** The DVFS ((dynamic voltage-frequency scaling) method enables the exploitation of the slack time (idle time) typically incurred by inter task relationship.

$$\begin{cases} E = C_{eff} f v^2 t \\ f = K \dfrac{(v - v_t)^2}{v} \end{cases}$$

PARAMETER \ COMPUTING	CLUSTER	GRID	CLOUD
Resource utilization	Aggregation	Segregation	Consolidation
Running Process	Same process runs on all computers.	Jobs divided into sub jobs	Depends on the service provision
Operating System	All nodes must run on same OS	No restriction to make on the OS	No restriction to make on the OS
Job Execution	Depends on job scheduling	Execution move towards the idle processor	Self-managed
Suitable for apps	Cascading	Not suitable	On demand services provisioning
Location of nodes	Same location	Distributed	Does not matter
Homo/heterogeneity	Homogenous	Heterogenous	heterogeneous
Virtualization	None	None	applied
Transparency	Yes	Yes	Yes
Security	High	High	moderate
Interoperability	Yes	Yes	No
Implementation	Easy	Difficult	Difficult
Management	Easy	Difficult	Difficult
Resource management	Centralized	Distributed	Both centralized and distributed
Internet	Not required	Required	Required
Application domains	Industrial sector, research centers, healthcare, centers which offer services on nation wide level	Industrial sector, research centers, healthcare, centers which offer services on nation wide level	Banking, insurance, weather forecasting, space exploration, business, IaaS, PaaS, SaaS

Fig: showing the comparison of Cluster, Grid and Cloud computing