**UNIT – 1**

**Systems Modelling, Clustering and Virtualization**

1. **Scalable Computing Over the Internet**
   1. **Scalability:** Scalability is the capability of a system or network or process to handle a growing amount of works like database storage, software usage and so on [1]. A scalable system should be able to handle the ever-increasing data, levels of computations and should be efficient.
   2. **NOTE:** Generally, a computer uses a centralized system to solve the problems. A parallel and distributed computing system uses multiple computers to solve large scale problems over the Internet [2].
   3. **Parallel Computing:** Execution of many processes is carried out simultaneously in this case. Large problems can be divided into smaller ones, solved at the same time and integrated later.
   4. **Distributed Computing:** A distributed system is a model in which components located on connected computers (through a network) interchange/monitor their actions by passing messages. Distributed computing may refer to systems situated at different physical locations or different actions being performed on the same system.

Distributed Computing is centred on data and based on networks.

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

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

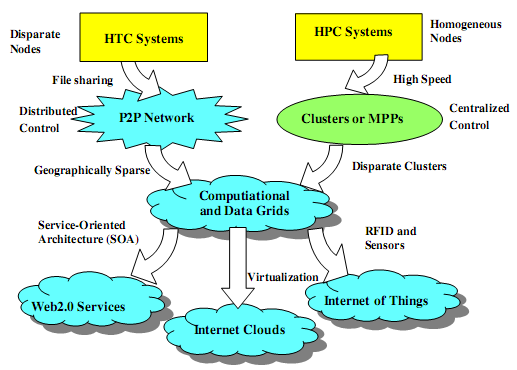


Figure 1.1 [2]: Evolutionary Trend towards parallel, distributed and cloud computing

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

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

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

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

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

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

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

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

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

1. **Radio-Frequency Identification (RFID)**: This uses electro-magnetic fields to automatically identify and track tags attached to objects [4]. These tags contain electronically stored information.
2. **Global Positioning System (GPS)**: It is a global navigation satellite system that provides the geographical location and time information to a GPS receiver [5].
3. **Internet of Things (IoT)**: It is the internetworking of different physical devices (vehicles, buildings etc.) embedded with electronic devices (sensors), software, and network connectivity [6]. Data can be collected and exchanged through this network (IoT).
   1. **Computing Paradigm Distinctions:**
4. **Centralized Computing**: All computer resources like processors, memory and storage are centralized in one physical system. All of these are shared and inter-connected and monitored by the OS.
5. **Parallel Computing**: All processors are tightly coupled with centralized shared memory or loosely coupled with distributed memory (parallel processing). Inter processor communication is achieved by message passing. This methodology is known as parallel computing.

**NOTE**: Coupling is the inter-dependence between software/hardware modules.

1. **Distributed Computing**: A distributed system consists of multiple autonomous computers with each device having its own private memory. They interconnect among themselves by the usage of a computer network. Here also, information exchange is accomplished by message passing.
2. **Cloud Computing**: An Internet Cloud of resources can either be a centralized or a distributed computing system. The cloud applies parallel or distributed computing or both. Cloud can be built by using physical or virtual resources over data centers. CC is also called as utility/ service/concurrent computing.
   1. **NOTE**: IoT is a networked connection of general objects used everyday including computers, systems and sensors. IoT is supported by Internet Clouds to access any ‘thing’ at any place at any time. Internet Computer is a larger concept that covers all computing paradigms, emphasizing on distributed and cloud computing.
   2. *Explanation on the recent surge in networks of clusters, data grids.* Internet Clouds are the result of moving desktop computing to service-oriented computing using server clusters and huge databases at data centers.

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

**Main Objectives**:

1. **Efficiency**: Efficiency is decided by speed, programming and throughput demands’ achievement.
2. **Dependability**: This measures the reliability from the chip to the system at different levels. Main purpose here is to provide good QoS (Quality of Service).
3. **Adaption in the Programming Model**: This measures the ability to support unending number of job requests over massive data sets and virtualized cloud resources under different models.
4. **Flexibility:** It is the ability of distributed systems to run in good health in both HPC (science/engineering) and HTC (business).
   1. **Degrees of ‘Parallelism’**:
5. **Bit-level parallelism** (BLP) 8 bit, 16, 32, and 64.
6. **Instruction-level parallelism** (ILP): The processor executes multiple instructions simultaneously. Ex: Pipelining, supercomputing, VLIW (very long instruction word), and multithreading.

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

**Multithreading**: Multithreading is the ability of a CPU or a single core in a [multi-core](https://en.wikipedia.org/wiki/Multi-core) processor to execute multiple [processes](https://en.wikipedia.org/wiki/Process_(computing)) or [threads](https://en.wikipedia.org/wiki/Thread_(computer_science)) concurrently, supported by the [OS](https://en.wikipedia.org/wiki/Operating_system).

1. **Data-level Parallelism** (DLP): Here, instructions are given like arrays (single instruction, multiple data SIMD). More hardware support is needed.
2. **Task-level Parallelism (TLP):** It is a process of execution where different threads (functions) are distributed across multiple processors in parallel computing environments.
3. **Job-level Parallelism (JLP):** Job level parallelism is the highest level of parallelism where we concentrate on a lab or computer center to execute as many jobs as possible in any given time period [7]. To achieve this, we purchase more systems so that more jobs are running at any one time, even though any one user's job will not run faster.
   1. **Usage of CC**: It is used in different fields for different purposes. All applications demand computing economics, web-scale data collection, system reliability, and scalable performance. Ex: Distributed transaction processing is practiced in the banking industry. Transactions represent 90 percent of the existing market for reliable banking systems. [Give an example of demonetization to increase Internet transactions.]

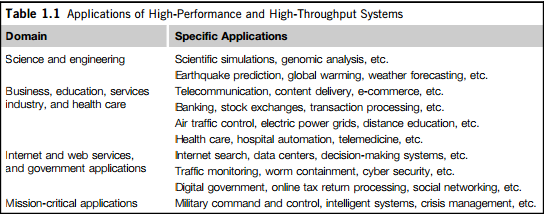


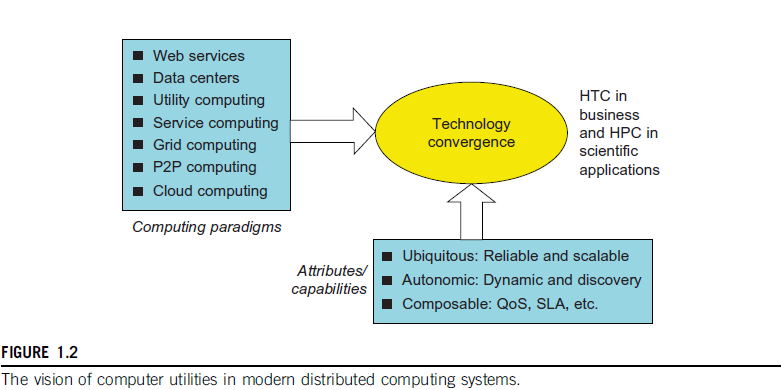
Table 1.1 [2]

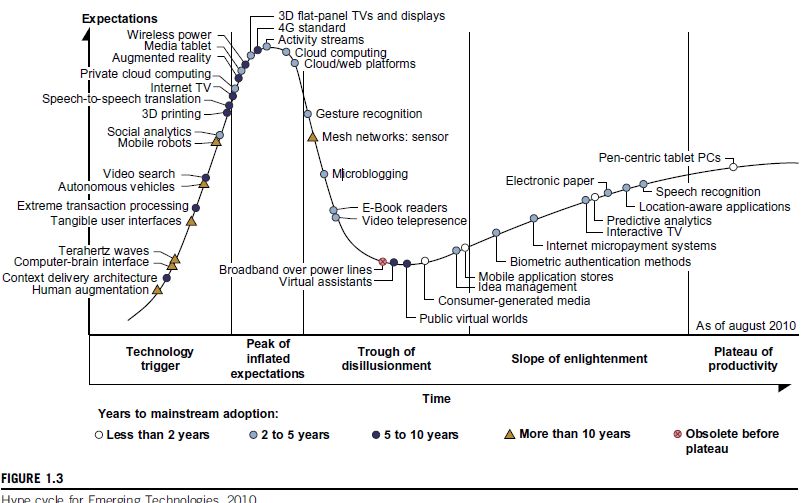
* 1. Major computing paradigms and available services/capabilities are coming together to produce a technology convergence of cloud/utility computing where both HPC and HTC are utilised to achieve objectives like reliability and scalability. They also aim to reach autonomic operations that can be self-organized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

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

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

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





* 1. **Internet of Things**: The IoT [8] refers the networked interconnection of everyday objects, tools, devices or computers. It can be seen as a wireless network of sensors that interconnect all things we use in our daily life. RFID and GPS are also used here. The IoT demands universal addressability of all the objects or things that may be steady or moving.

These objects can be interconnected, can exchange data and interact with each other by the usage of suitable applications (web/mobile). In the IoT era, CC can be used efficiently and in a secure way to provide different services to the humans, computers and other objects. Ex: Smart cities, inter-connected networks, self-controlling street lights/traffic lights etc.

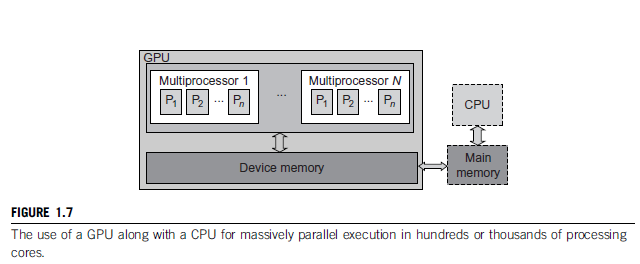
* 1. **NOTE**: CPS means cyber–physical system where physical objects and computational processes interact with each other. Ex: Wrest bands to monitor BP. CPS merges the 3Cs which are computation, communication and control to provide intelligent feedbacks between the cyber and physical worlds.

1. **Technologies for Network based Systems**
   1. **Multi-core CPUs and Multithreading Technologies**: Over the last 30 years the speed of the chips and their capacity to handle variety of jobs has increased at an exceptional rate. This is crucial to both HPC and HTC system development. Note that the processor speed is measured in MIPS (millions of instructions per second) and the utilized network bandwidth is measured in Mbps or Gbps.
   2. **Advances in CPU Processors**: The advanced microprocessor chips (by Intel, NVIDIA, AMD, Qualcomm etc.) assume a multi-core architecture with dual core, quad core or more processing cores. They exploit parallelism at different levels. Moore’s law has proven accurate at these levels. Moore's law is the observation that the number of transistors in a dense integrated circuit doubles approximately every two years.
   3. **Multi-core CPU**: A multi-core processor is a single computing component with two or more independent actual processing units (called "cores"), which are units that read and execute program instructions [9]. (Ex: add, move data, and branch). The multiple cores can run multiple instructions at the same time, increasing overall speed for programs open to parallel computing.
   4. **Many-core GPU**: (Graphics Processing Unit) Many-core processors are specialist multi-core processors designed for a high degree of parallel processing, containing a large number of simpler, independent processor cores [10]. Many-core processors are used extensively in embedded computers and high-performance computing. (Main frames, super computers).
   5. **GPU Computing:** A GPU is a graphics co-processor mounted on a computer’s graphics card to perform high level graphics tasks in video editing apps. (Ex: Intel Xeon, NVIDIA). A modern GPU chip can be built with hundreds of processing cores. These days, parallel GPUs or GPU clusters are gaining more attention.

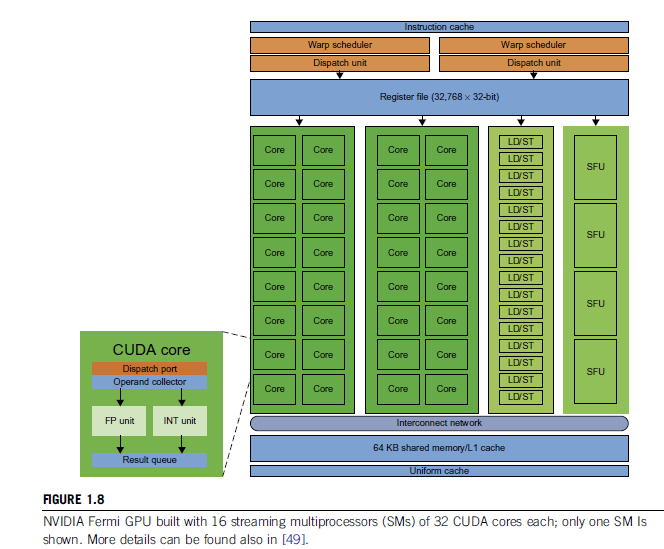
Starting as co-processors attached to the CPU, the GPUs these days possess 128 cores on a single chip (NVIDIA). Hence they have 1024 threads (128\*8) executing tasks concurrently, on a single GPU. This can be termed as massive parallelism at multicore and multi-threading levels. GPUs are not restricted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

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

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

****

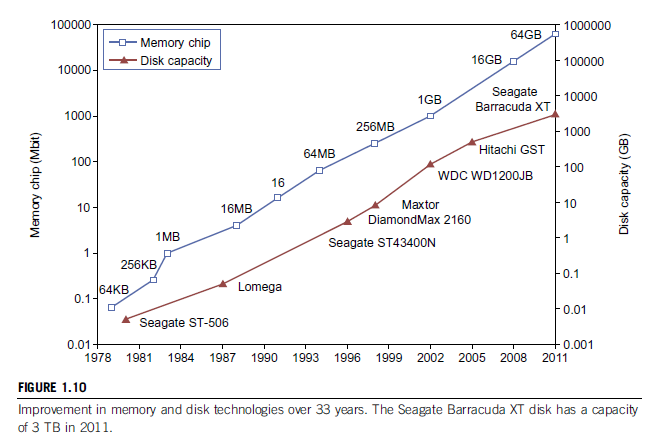
* 1. **NOTE**: Bandwidth is the bit-rate of available or consumed information capacity expressed typically in metric multiples of bits per second. Variously, bandwidth may be characterized as network bandwidth, data bandwidth, or digital bandwidth.
  2. In future, thousand-core GPUs may feature in the field of Eflops/1018 flops systems.



* 1. **Power Efficiency of the GPU**: The major benefits of GPU over CPU are power and massive parallelism. Estimation says that 60 Gflops/watt per core is needed to run an exaflops system. [One exaflops is a thousand petaflops or a quintillion, 1018, floating point operations per second]. A GPU chip requires one-tenth less of the power that a CPU requires. (Ex: CPU: 100, GPU: 90).

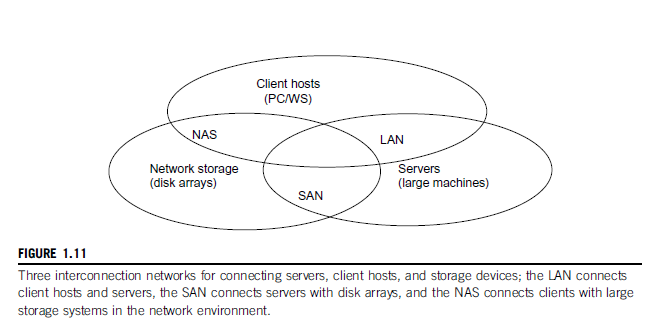
CPU is optimized (use most effectively) for latency (time between request and response) in caches and memory; GPU is optimized for throughput with explicit (open) management of on-chip memory.

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



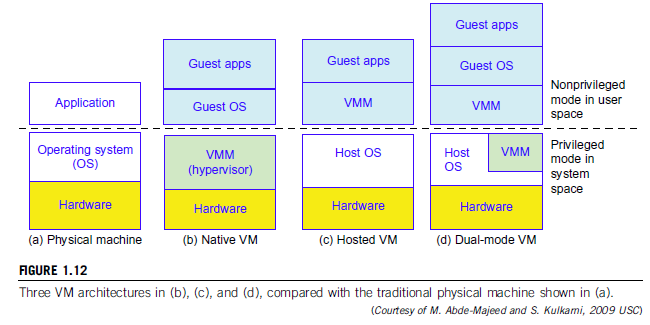
* 1. **Memory, Storage and WAN**:

1. **Memory Technology**: The upper curve in Figure 1.10 shows the growth of DRAM chip capacity from 16 KB to 64 GB. [**SRAM** is Static RAM and is 'static' because the memory does not have to be continuously refreshed like **Dynamic RAM**. SRAM is faster but also more expensive and is used inside the CPU. The traditional RAMs in computers are all DRAMs]. For hard drives, capacity increased from 260 MB to 3 TB and lately 5 TB (by Seagate). Faster processor speed and higher memory capacity will result in a wider gap between processors and memory, which is an ever-existing problem.
2. **Disks and Storage Technology:** The rapid growth of flash memory and solid-state drives (SSD) also has an impact on the future of HPC and HTC systems. An SSD can handle 300,000 to 1 million write cycles per block, increasing the speed and performance. Power consumption should also be taken care-of before planning any increase of capacity.
3. **System-Area Interconnects**: The nodes in small clusters are interconnected by an Ethernet switch or a LAN. As shown in Figure 1.11 [2], a LAN is used to connect clients to servers. A Storage Area Network (SAN) connects servers to network storage like disk arrays. Network Attached Storage (NAS) connects clients directly to disk arrays. All these types of network appear in a large cluster built with commercial network components (Cisco, Juniper). If not much data is shared (overlapped), we can build a small cluster with an Ethernet Switch + copper cables to link to the end machines (clients/servers).



1. **WAN**: We can also notice the rapid growth of Ethernet bandwidth from 10 Mbps to 1 Gbps and still increasing. Different bandwidths are needed for local, national, and international levels of networks. It is also estimated that computers will be used concurrently in the coming future and higher bandwidth will certainly add more speed and capacity to aid the cloud/distributed computing. Note that most data centers use gigabit Ethernet as interconnect in their server clusters.
   1. **Virtual Machines and Middleware**: A typical computer has a single OS image at a time. This leads to a rigid architecture that tightly couples apps to a specific hardware platform i.e., an app working on a system might not work on another system with another OS (non-portable).

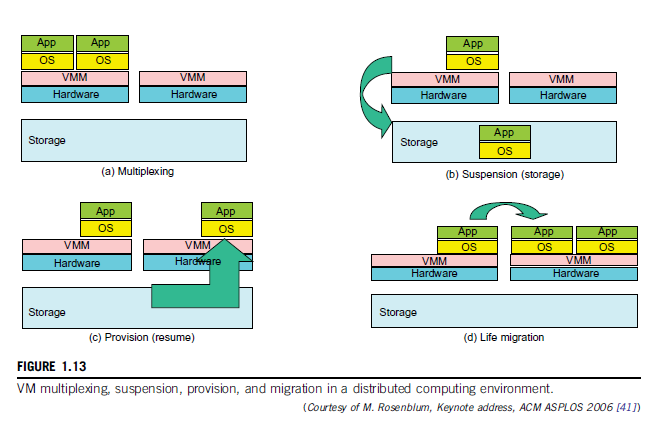
To build large clusters, grids and clouds, we need to increase the capacity of computing, storage and networking resources in a virtualized manner. A cloud of limited resources should aggregate all these dynamically to bring out the expected results.



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

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

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

****

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

**Advantages**:

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

1. **Virtual Infrastructures**: Virtual Infrastructure connects resources to distributed applications in such a way that a resource needed by an app is exactly mapped to it. This decreases the costs and increases efficiency and server response.
   1. **Data Center Virtualization for Cloud Computing**: Cloud architecture is built with products like hardware and network devices. Almost all cloud platforms use x86 (Family of Intel 8086 processors). Low-cost terabyte disks and gigabit Ethernet are used to build data centers. A data center takes into consideration the performance/price ratio instead of only speed.
2. **Data Center Growth and Cost Breakdown**: Large data centers are built with thousands of servers and smaller ones have hundreds of the same. The cost of maintaining a data center has increased and much of this money is spent on management and maintenance which did not increase with time. Electricity and cooling also consume much of the allocated finance.
3. **Low Cost Design Philosophy:** High-end switches or routers that provide high bandwidth networks cost more and do not match the financial design of cloud computing. For a fixed budget, typical switches and networks are more desirable.

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

1. **Convergence of Technologies**: CC is enabled by the convergence of technologies in four areas:

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

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

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

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

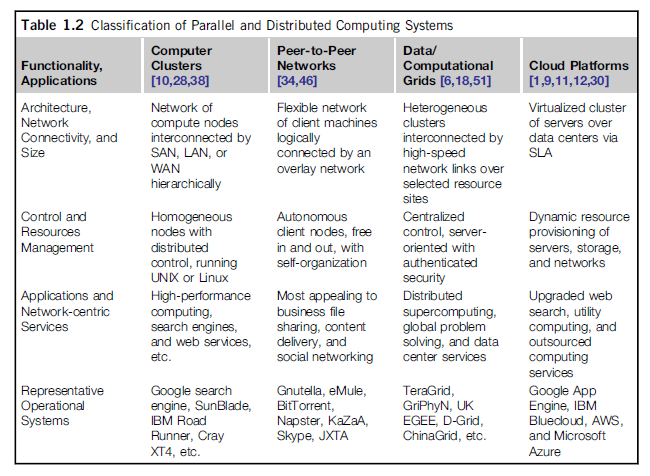
Cloud Computing is a transformative approach since it promises more results than a normal data center. The basic interaction with the information is taken up in a different approach to obtain a variety of results, by using different types of data to end up with useful analytical results.

It should also be noted that a cloud provides sources on demand at the infrastructure, platform, or software level. At platform level, MapReduce offers a new programming model that transparently handles data parallelism with natural fault tolerance capability. Iterative MapReduce extends MapReduce to support a broader range of DM algorithms.

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

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

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

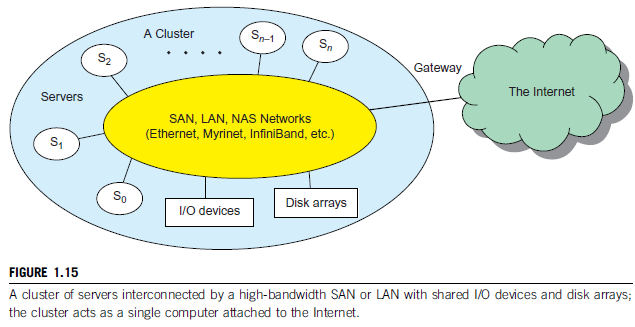


* 1. Clusters are more popular in supercomputing apps. They have laid the foundation for cloud computing. P2P are mostly used in business apps. Many grids formed in the previous decade have not been utilized per their potential due to lack of proper middleware or well-coded apps.

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

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

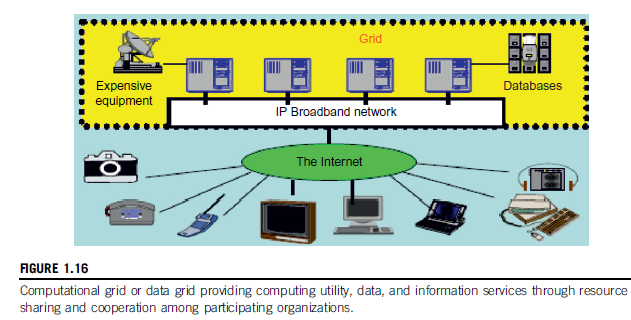
1. The Figure 1.1.5 [2] below shows the architecture of a typical server cluster that has low latency and high bandwidth network. [**Latency** is the delay from input into a system to desired outcome]. For building a large cluster, an interconnection network can be utilized using Gigabit Ethernet, Myrinet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



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

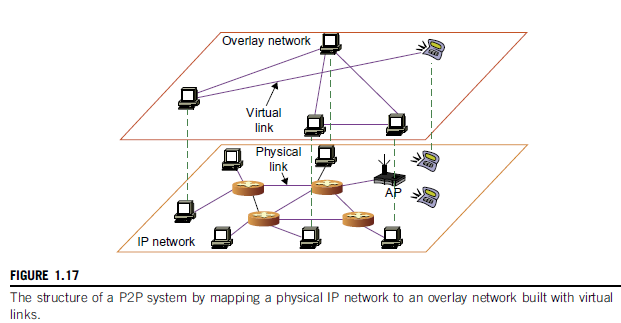
1. **Single-System Image (SSI)**: It was indicated that multiple system images should be integrated into a single-system image for a cluster. A cluster-OS is more desired these days, or a middleware to support SSI that includes sharing of CPUs, memory, I/O across all the nodes in the cluster. An SSI is an illusion (something that doesn’t exist actually) that shows the integrated resources as a single and powerful resource. SSI can be created by software or hardware. Finally, a cluster is with multiple system images is only a collection of the resources of independent computers that are loosely inter-connected.
2. **HW, SW and MW Support**: It should be noted that MPPs (Massively Parallel Processing) are clusters exploring high-level parallel processing. The building blocks here are the computer nodes (PCs, Symmetric Multi-Processors (SMPs), work stations or servers), communication software like Parallel Virtual Machine (PVM), Message Passing Interface (MPI), and a network interface card in each node. All the nodes are interconnected by high bandwidth network (Ex: Gigabit Ethernet).

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

1. **Major Cluster Design Issues**: A cluster-wide OSs or a single OS controlling the cluster virtually is not yet available. This makes the designing and achievement of SSI difficult and expensive. All the apps should rely upon the middleware to bring out the coupling between the machines in cluster or between the clusters. But it should also be noted that the major advantages of clustering are scalable performance, efficient message passing, high system availability, good fault tolerance and a cluster-wide job management which react positively to the user demands.
   1. **Grid Computing Infrastructures**: Grid computing is designed to allow close interaction among applications running on distant computers simultaneously.
2. **Computational Grids**: A computing grid provides an infrastructure that couples computers, software/hardware, sensors and others together. The grid can be constructed across LAN, WAN and other networks on a regional, national or global scale. They are also termed as *virtual platforms*. Computers, workstations, servers and clusters are used in a grid. Note that PCs, laptops and others can be viewed as access devices to a grid system. Figure 1.6 [2] below shows an example grid built by different organisations over multiple systems of different types, with different operating systems. 
3. **Grid Families**: Grid technology demands new distributed computing models, software/middleware support, network protocols, and hardware infrastructures. National grid projects are followed by industrial grid platforms by IBM, Microsoft, HP, Dell-EMC, Cisco, and Oracle. New grid service providers (GSPs) and new grid applications have emerged rapidly, similar to the growth of Internet and web services in the past two decades. Grid systems are classified in essentially two categories: computational or data grids and P2P grids. Computing or data grids are built primarily at the national level.
   1. **Peer-to-Peer Network Families**: In the basic client-server architecture, the client machines are connected to a central server for different purposes and these are essentially P2P networks. The P2P architecture offers a distributed model of networked systems. Note that P2P network is client-oriented instead of server-oriented.
4. **P2P Systems**: Here, every node acts as both a client and a server. Peer machines are those connected to the Internet; all client machines act autonomously to join or leave the P2P system at their choice. No central coordination DB is needed. The system is self-organising with distributed control.

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

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

****

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

1. **P2P Application Families**: There exist 4 types of P2P networks: distributed file sharing, collaborative platform, distributed P2P computing and others. Ex: BitTorrent, Napster, Skype, Geonome, JXTA, .NET etc.
2. **P2P Computing Challenges:** The main problems in P2P computing are those in hardware, software and network. Many hardware models exist to select from; incompatibility exists between the software and the operating systems; different network connections and protocols make it too complex to apply in real-time applications. Further, data location, scalability, performance, bandwidth etc. are the other challenges.

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

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

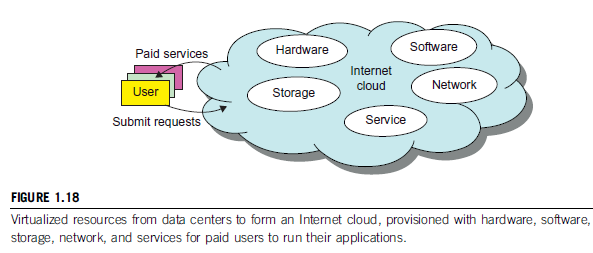
NOTE: P2P computing or networking is a distributed application architecture that partitions tasks or workloads between peers [11].

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

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

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

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



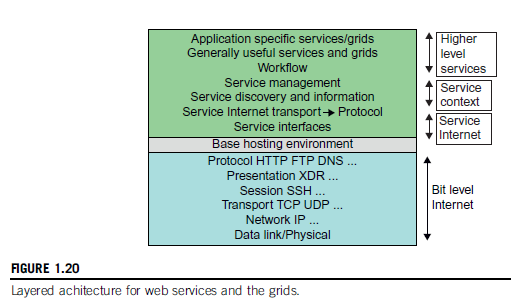
1. **Internet Clouds**: The idea in CC is to move desktop computing to a service-oriented platform using server clusters and huge DBs at data centers. CC benefits both users and providers by using its low cost and simple resources through machine virtualization. Many user applications are satisfied simultaneously by CC and finally, its design should satisfy the security norms, be trustworthy and dependable. CC is viewed in two ways: a centralized resource pool or a server cluster practising distributed computing.
2. **The Cloud Landscape**: A distributed computing system is controlled by companies or organisations. But these traditional systems encounter several bottlenecks like constant maintenance, poor utilization, and increasing costs and updates of software or hardware. To get rid of these, CC should be utilized as on-demand computing.

CC offers different types of computing as services:

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

1. Inter clouds offer four deployment models: private, public, managed and hybrid.

* **Private Cloud**: Private cloud is a type of cloud computing that delivers similar advantages to public cloud, including scalability and self-service, but through a proprietary architecture.
* **Public Cloud**: A public cloud is one based on the standard cloud computing model, in which a service provider makes resources, such as applications and storage, available to the general public over the Internet.
* **Managed Cloud**: Managed cloud hosting is a process in which organizations share and access resources, including databases, hardware and software tools, across a remote network via multiple servers in another location. [12]
* **Hybrid Cloud**: A hybrid cloud is an integrated cloud service utilising both private and public clouds to perform distinct functions within the same organisation. [13]
  1. **NOTE**: The different service level agreements (SLAs) mean that the security responsibility is shared among all the cloud providers, consumers, and the third-party cloud-enabled software service providers.
  2. **Software Environments for Distributed Systems and Clouds – SOA:** In grids that use Java/CORBA, an entity is a service or an object. Such architectures build on the seven OSI layers (APSTNDP) that provide networking abstractions. Above this we have a base service environment like .NET, Java etc. and a broker network for CORBA, which enables collaboration between systems on different operating systems, programming languages and hardware [14]. By using this base, one can build a higher level environment reflecting the special features of distributed computing. The same is reflected in the figure 1.20 [2] below.



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

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

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

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

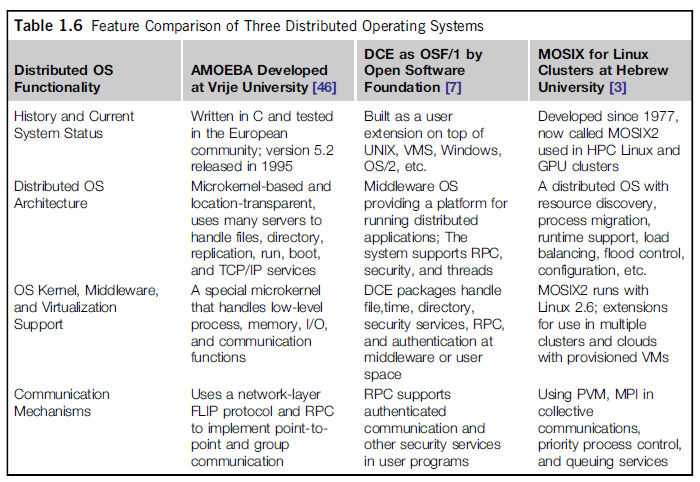
In the REST approach, simplicity is stressed upon, and difficult problems are delegated to the apps. In a web services language, REST has minimal information in the header and the message body carries the needed information. REST architectures are more useful in rapid technology environments. Above the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

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

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

**Raw Data Data Information Knowledge Wisdom Decisions**

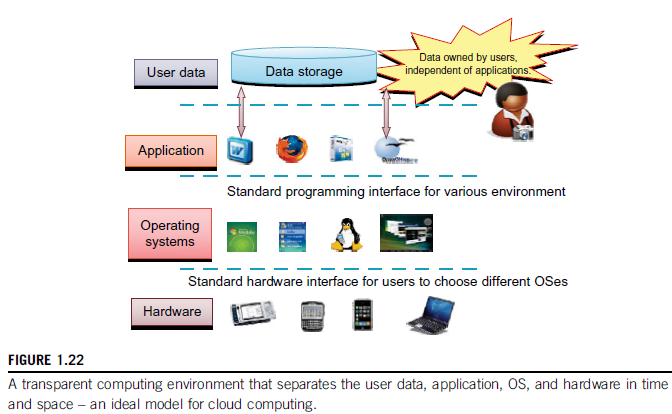
1. **Grids Vs Clouds**: Grid systems apply static resources, while a cloud stresses upon elastic resources. Differences between grid and cloud exist only in dynamic resource allocation based on virtualization and autonomic computing. A ‘grid of clouds’ can also be built and can do a better job than a pure cloud because it can support resource allocation. Grid of clouds, cloud of grids, cloud of clouds and inter-clouds are also possible.
   1. **Distributed Operating Systems**: To promote resource sharing and fast communication, it is best to have a distributed operating system that can manage the resources efficiently. In distributed systems or more generally, a network needs an operating system itself since it deals with many heterogeneous platforms. But such an OS offers low transparency to the users. It should be noted that middleware can also be used to generate resource sharing but only till we attain a certain level. The third approach is to develop a truly distributed OS to achieve highest efficiency and maximum transparency. Comparison can be seen in Table 1.6 [2].

****

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

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

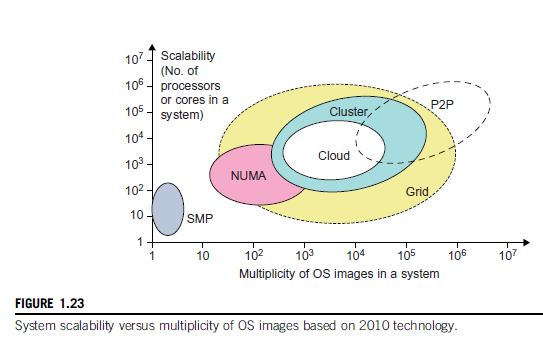
Transparency in programming environments that handle user data, OS, and hardware plays a key role in the success of clouds. This concept is divided into 4 levels as seen below [2]: Data, app, OS, and hardware. Users will be able to chose the OS they like as well as the app they like – this is the main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



* 1. **Message-Passing Interface (MPI)**: MPI is a library of sub-programs that can be called from C or FORTRAN to write parallel programs running on a distributed system. The goal here is to represent clusters, grid systems, and P2P systems with upgraded web services and other utility apps. Distributed programming can also be supported by Parallel Virtual Machine (PVM).
  2. **MapReduce**: it is a web programming model for scalable data processing on large data clusters. It is applied mainly in web-scale search and cloud computing apps. The user specifies a Map function to generate a set of intermediate key/value pairs. Then the user applies a Reduce function to merge all intermediate values with the same (intermediate) key. MapReduce is highly scalable to explore high degrees of parallelism at different job levels and can handle terabytes of data on thousands of client machines. Many MapReduce programs can be executed simultaneously. Ex: Google’s clusters.
  3. **Hadoop Library**: Hadoop enables users to write and run apps over vast amounts of distributed data. Users can easily scale Hadoop to store and process Petabytes of data in the web space. The package is economical (open source), efficient (high level of parallelism) and is reliable (keeps multiple data copies).
  4. **Open Grid Services Architecture**: OGSA is driven by large-scale distributed computing apps. These apps must provide take into account high degree of resource and data sharing. The key features here are: distributed executed environment, public key infrastructure (PKI) services, trust management and security problems in grid computing.

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

* 1. **Performance Metrics**: In a distributed system, system throughput is measured in MIPS, Tflops (Tera Floating point Operations per Second) or Transactions per Second (TPS). Other measures also exist: job response and network latency. An interconnection network with low latency and high bandwidth is preferred. The key factors to be considered for performance are OS boot time, compile time, I/O data rate, and the runtime support system used.
  2. **Dimensions of Scalability**: System scaling can increase or decrease resources depending on different practical factors.
* **Size Scalability**: This targets higher performance or more functionality by increasing the machine size (cache, processors, memory etc.). We can determine the size scalability by counting the number of processors installed. That is more processors => more ‘size’.
* **Software Scalability**: Upgrades in OS/compilers, adding mathematical libraries, installing new apps, and using more user friendly environments are the factors considered in determining software scalability.
* **Application Scalability**: This refers to matching problem size scalability (increasing data) with machine size scalability (effectively use the resources to obtain the best result possible).
* **Technology Scalability**: Here, systems that can adapt to changes in different aspects of technology like component or network are considered. Three aspects play an important role here: time, space and heterogeneity. Time is concerned with processors, motherboard, power supply packaging and cooling. All these have to be upgraded between 3 to 5 years. Space is related to packaging and energy concerns. Heterogeneity refers to the use of hardware components or software packages from different vendors; this affects scalability the most.
  1. **Scalability versus OS Image Count**: In Figure 1.23 [2], scalable performance is estimated against the multiplicity of OS images in distributed systems. Note that scalable performance means we can ever increase the speed of the system by adding more servers of processors, or by enlarging memory size and so on. The OS image is counted by the no. of independent OS images observed in a cluster, grid, P2P network or the cloud.



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

* 1. **Amdahl’s Law**: Consider the execution of a given program on a uniprocessor workstation with a total execution time of T minutes. Say the program is running in parallel with other servers on a cluster of many processing nodes. Assume that a fraction α of the code must be executed sequentially (sequential bottleneck). Hence, (1-α) of the code can be compiled for parallel execution by *n* processors. The total execution time of the program is calculated by *αT + (1-α) T/n* where the first term is for sequential execution time on a single processor and the second term is for parallel execution time on *n* parallel nodes.

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

Speedup S= *T/[αT + (1-α) T/n]* = 1/[ *α + (1-α)/n]* ---- (1.1)

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

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

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

*E = S/n = 1/[α n + 1-α]* ---- (1.2)

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

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

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

*S’ = W’/W = [αW + (1-α) nW]/W = α+ (1-α) n* ---- (1.3)

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

E’ = S’/n = *α/n+ (1-α)* ---- (1.4)

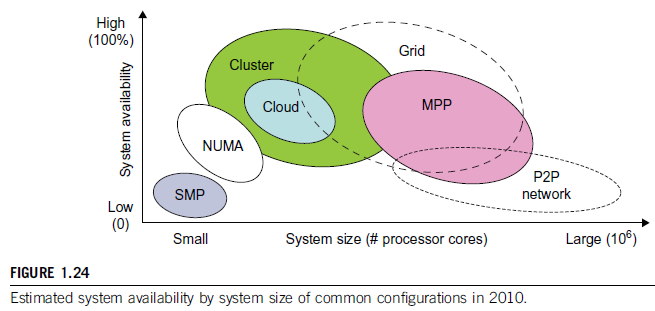
Taking previous workload values into consideration, efficiency can be improved for a 256-node cluster to E’ = 0.25/256 + (1-0.25) = 0.751. For a fixed workload Amdahl’s law must be used and for scaled problems users should apply Gustafson’s law.

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

* 1. **System Availability**: High availability (HA) is needed in all clusters, grids, P2P networks and cloud systems. A system is highly available if it has a long mean time to failure (MTTF) and a short mean time to repair (MTTR).

*System Availability = MTTF/(MTTF + MTTR)* ---- (1.5)

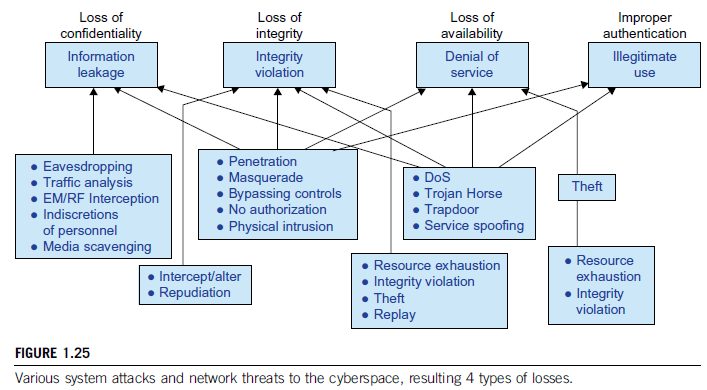
System availability depends on many factors like hardware, software and network components. Any failure that will lead to the failure of the total system is known as a *single point of failure*. It is the general goal of any manufacturer or user to bring out a system with no single point of failure.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



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

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

* 1. **Threats to networks and systems**:



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

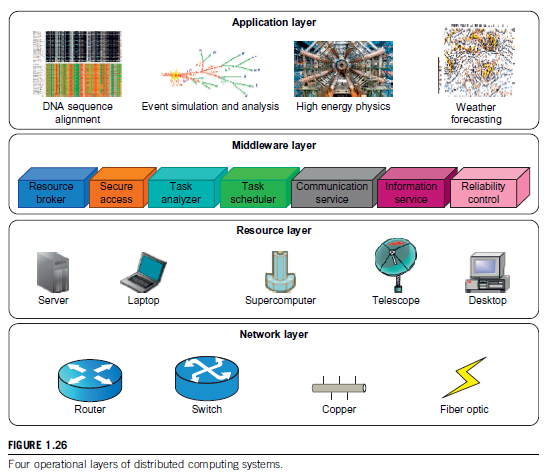
* 1. **Security Responsibilities**: The main responsibilities include confidentiality, integrity and availability for most Internet service providers and cloud users. In the order of SaaS, PaaS and IaaS, the providers increase/transfer security control to the users. IN brief, the SaaS model relies on the cloud provider for all the security features. On the other hand, IaaS wants the users to take control of all security functions, but their availability is still decided by the providers. Finally, the PaaS model divides the security aspects in this way: data integrity and availability is with the provider while confidentiality and privacy control is the burden of the users.
  2. **Copyright Protection**: Collusive (secret agreement) piracy is the main source of copyright violation within the boundary of a P2P network. Clients may illegally share their software, allotted only to them, with others thus triggering piracy. One can develop a proactive (control the situation before damage happens) content poisoning scheme to stop colluders (conspirers) and pirates, detect them and stop them to proceed in their illegal work.
  3. **System Defence Technologies**: There exist three generations of network defence. In the first generation, tools were designed to prevent intrusions. These tools established themselves as access control policies, cryptographic systems etc. but an intruder can always slip into the system since there existed a weak link every time. The second generation detected intrusions in a timely manner to enforce remedies. Ex: Firewalls, intrusion detection systems (IDS), public key infrastructure (PKI) services (banking, e-commerce), reputation systems etc. The third generation provides more intelligent responses to intrusions.
  4. **Data Protection Infrastructure**: Security infrastructure is required to protect web and cloud services. At the user level, one needs to perform trust negotiation and reputation aggregation over all users. At the app end, we need to establish security precautions and intrusion detection systems to restrain virus, worm, malware, and DDoS attacks. Piracy and copyright violations should also be detected and contained. These can be studied in detail later when the three types of clouds are encountered and the general services offered by the cloud are discussed.
  5. **Energy Efficiency in Distributed Computing**: The primary goals in parallel and distributed computing systems are HP and HT and also performance reliability (fault tolerance and security). New challenges encountered in this area (distributed power management-DPM) these days include energy efficiency, workload and resource outsourcing. In the forth-coming topics, the energy consumption issues in servers and HPC systems are discussed.

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

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

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

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

****

* 1. **Application Layer**: Most apps in different areas like science, engineering, business, financial etc. try to increase the system’s speed or quality. By introducing energy-conscious applications, one should try to design the usage and consumption in a planned manner such that the apps manage to use the new multi-level and multi-domain energy management methodologies without reducing the performance. For this goal, we need to identify a relationship between the performance and energy consumption areas (correlation). Note that these two factors (compute and storage) are surely correlated and affect completion time.
  2. **Middleware layer**: The middleware layer is a connection between application layer and resource layer. This layer provides resource broker, communication service, task analyzer & scheduler, security access, reliability control, and information service capabilities. It is also responsible for energy-efficient techniques in task scheduling. In distributed computing system, a balance has to be brought out between efficient resource usage and the available energy.
  3. **Resource Layer**: This layer consists of different resources including the computing nodes and storage units. Since this layer interacts with hardware devices and the operating systems, it is responsible for controlling all distributed resources. Several methods exist for efficient power management of hardware and OS and majority of them are concerned with the processors.

Dynamic power management (**DPM**) and dynamic voltage frequency scaling (**DVFS**) are the two popular methods being used recently. In DPM, hardware devices can switch from idle modes to lower power modes. In DVFS, energy savings are obtained based on the fact that power consumption in CMOS [15] (Complementary Metal-Oxide Semiconductor) circuits have a direct relationship with frequency and the square of the voltage supply. [***P = 0.5 CV2f***] Execution time and power consumption can be controlled by switching among different voltages and frequencies.

* 1. **Network Layer**: The main responsibilities of the network layer in distributed computing are routing and transferring packets, and enabling network services to the resource layer. Energy consumption and performance are to measured, predicted and balanced in a systematic manner so as to bring out energy-efficient networks. Two challenges exist here:
* The models should represent the networks systematically and should possess a full understanding of interactions among time, space and energy.
* New and energy-efficient algorithms have to be developed to rope in the advantages to the maximum scale and defend against the attacks.

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

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

 ---- (1.6)

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

**References**

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