Unit - 1

SYSTEMS MODELING, CLUSTERING AND VIRTUALIZATION

Scalable Computing over the Internet

Scalable computing over the internet refers to the ability to dynamically adjust computing resources to accommodate varying workloads. This concept is central to cloud computing, where resources like processing power, storage, and network bandwidth can be scaled up or down based on demand. Here are some key aspects:

1. Elasticity
   - Resources can be added or removed in real time. For example, a web application can scale up during peak traffic and down during low usage.

2. Cloud Service Models
   - Infrastructure as a Service (IaaS): Provides virtualized computing resources over the internet (e.g., AWS EC2, Google Compute Engine).
   - Platform as a Service (PaaS): Offers a platform allowing customers to develop, run, and manage applications without dealing with the underlying infrastructure (e.g., Heroku, Google App Engine).
   - Software as a Service (SaaS): Delivers software applications over the internet on a subscription basis (e.g., Salesforce, Microsoft 365).

3. Load Balancing
   - Distributing incoming network traffic across multiple servers to ensure no single server becomes overwhelmed, enhancing performance and reliability.

4. Distributed Computing
   - Tasks are divided among multiple machines or nodes, allowing for parallel processing and improved efficiency. This is often seen in big data processing and machine learning applications.

5. Serverless Computing
   - Allows developers to build applications without managing the underlying infrastructure. The cloud provider automatically handles resource allocation based on demand (e.g., AWS Lambda, Azure Functions).

6. Microservices Architecture
   - Breaking applications into smaller, independent services that can be developed, deployed, and scaled individually. This enhances agility and scalability.

7. Cost Management

- Pay-as-you-go pricing models allow organizations to pay only for the resources they use, making it cost-effective to scale up or down as needed.

8. Global Reach
   - Cloud providers have data centers around the world, enabling low-latency access and redundancy, which is essential for scalable applications that serve global users.

9. Monitoring and Management Tools
   - Tools and services that provide insights into resource usage, performance, and application health, allowing organizations to optimize and scale effectively.

Scalable computing over the internet enables businesses to meet fluctuating demands efficiently, optimize costs, and enhance performance. As technology evolves, the approaches to achieving scalability continue to advance, further driving innovation in cloud computing and distributed systems.

Here are some real-time examples of scalable computing over the internet:

1. Netflix
   - Use Case: Streaming services.
   - Scalability: Netflix uses AWS to dynamically scale its computing resources based on viewer demand. During peak times (like new show releases), it can automatically increase server capacity to handle the load and then reduce it during off-peak times.

2. Airbnb
   - Use Case: Online marketplace for lodging and travel.
   - Scalability: Airbnb leverages cloud infrastructure to manage fluctuating user traffic, especially during high-demand periods like holidays or events. They use microservices architecture to scale individual services independently.

3. Spotify
   - Use Case: Music streaming.
   - Scalability: Spotify utilizes cloud computing to handle millions of users streaming music simultaneously. Their infrastructure can scale in real-time to accommodate varying listener numbers, especially during album releases or events.

4. Slack
   - Use Case: Team collaboration platform.
   - Scalability: Slack can scale its services to accommodate the number of active users, especially during major work hours or events. The backend is built to support real-time messaging and file sharing without performance degradation.

5. Zoom

- Use Case: Video conferencing.
- Scalability: Zoom experienced massive user growth during the pandemic. Their architecture allows them to scale up quickly to support a surge in users while ensuring video and audio quality remains high.

6. E-commerce Platforms (e.g., Amazon, eBay)
   - Use Case: Online shopping.
   - Scalability: These platforms use cloud services to manage traffic spikes during sales events (like Black Friday) by automatically provisioning additional resources to maintain site performance.

7. Google Search
   - Use Case: Search engine.
   - Scalability: Google's infrastructure is designed to handle billions of queries per day. It scales horizontally across data centers globally to ensure quick response times and reliability.

8. Dropbox
   - Use Case: Cloud storage and file synchronization.
   - Scalability: Dropbox can scale its storage and compute resources based on user demand, especially during peak usage times when many users access or upload large files.

9. Gaming Platforms (e.g., Fortnite)
   - Use Case: Online gaming.
   - Scalability: Games like Fortnite utilize cloud infrastructure to scale game servers based on player demand, ensuring smooth gameplay for millions of concurrent users.

10. Content Delivery Networks (CDNs)
    - Use Case: Speeding up content delivery.
    - Scalability: Services like Cloudflare and Akamai distribute content across a network of servers worldwide. They can scale to accommodate spikes in traffic, providing quick access to content regardless of user location.

These examples illustrate how various industries leverage scalable computing to enhance user experiences and manage resource demands efficiently.

High-Throughput Computing-HTC

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 measures high throughput or the number of tasks completed per unit of time. HTC technology needs to improve batch processing speed, and also address the acute problems of cost, energy savings, security, and reliability at many data and enterprise computing centers.

Computing Paradigm Distinctions

1. **Centralized computing**

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

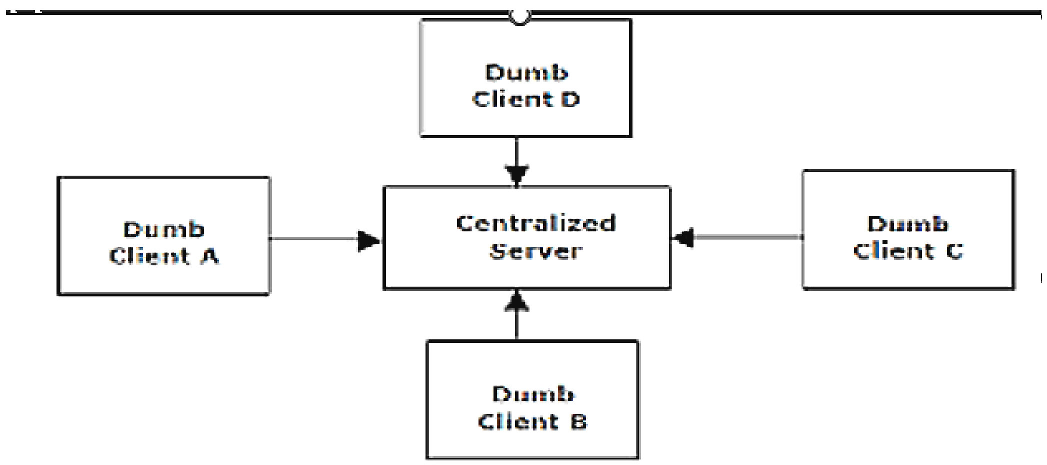


Fig: Centralized networking model

2. **Parallel computing**

- In parallel computing, all processors are either tightly coupled with centralized shared memory or loosely coupled with distributed memory.
- Inter processor communication is accomplished through shared memory or via message passing.
- A computer system capable of parallel computing is commonly known as a parallel computer.
- Programs running in a parallel computer are called parallel programs. The process of writing parallel programs is often referred to as parallel programming.

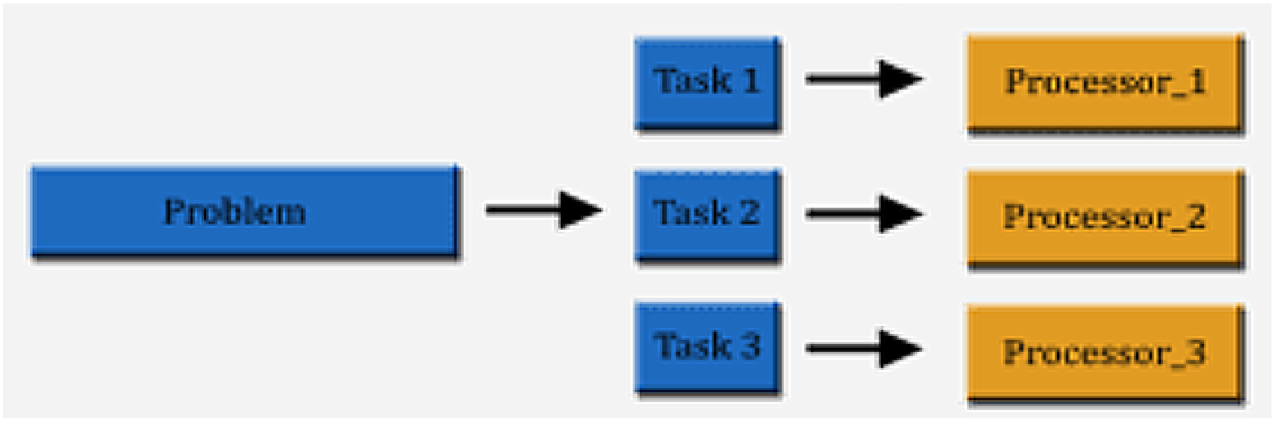


Fig: Parallel computing model

3. **Distributed computing**

- A distributed system consists of multiple autonomous computers, each having its own private memory, communicating through a computer network.
- Information exchange in a distributed system is accomplished through message passing.
- A computer program that runs in a distributed system is known as a distributed program.
- The process of writing distributed programs is referred to as distributed programming.
- Distributed computing system use multiple computers to solve large-scale problems over the Internet using a centralized computer to solve computational problems.

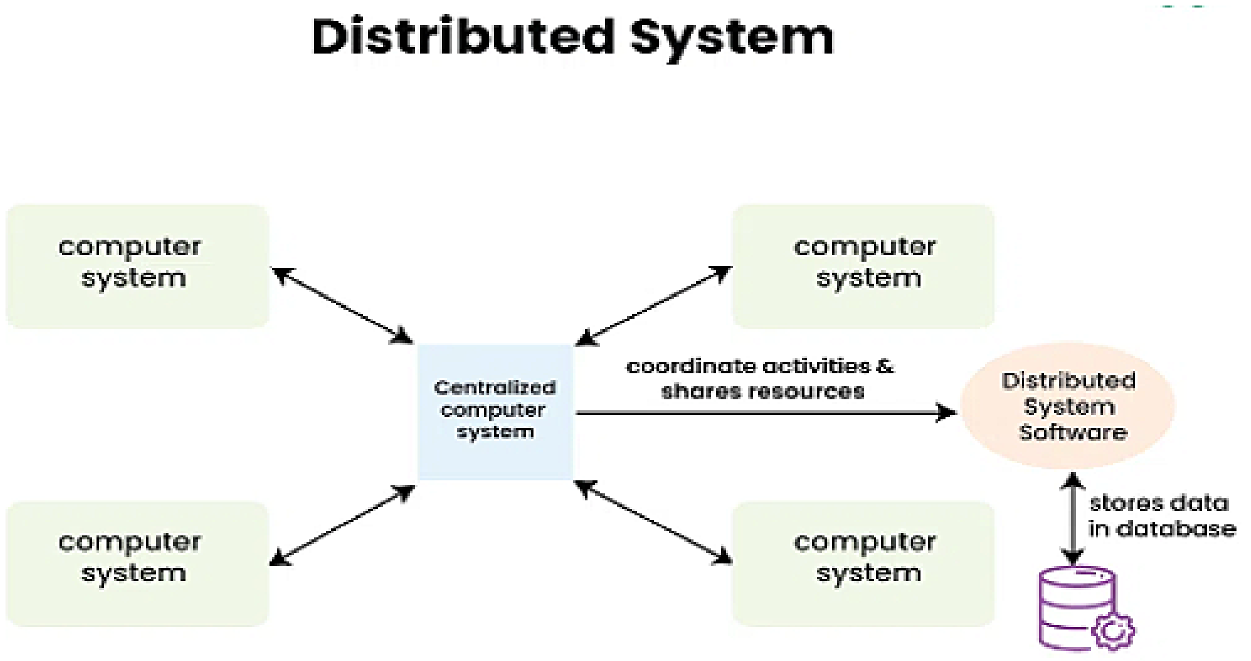


Fig: Distributed system model

4. **Cloud computing**

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

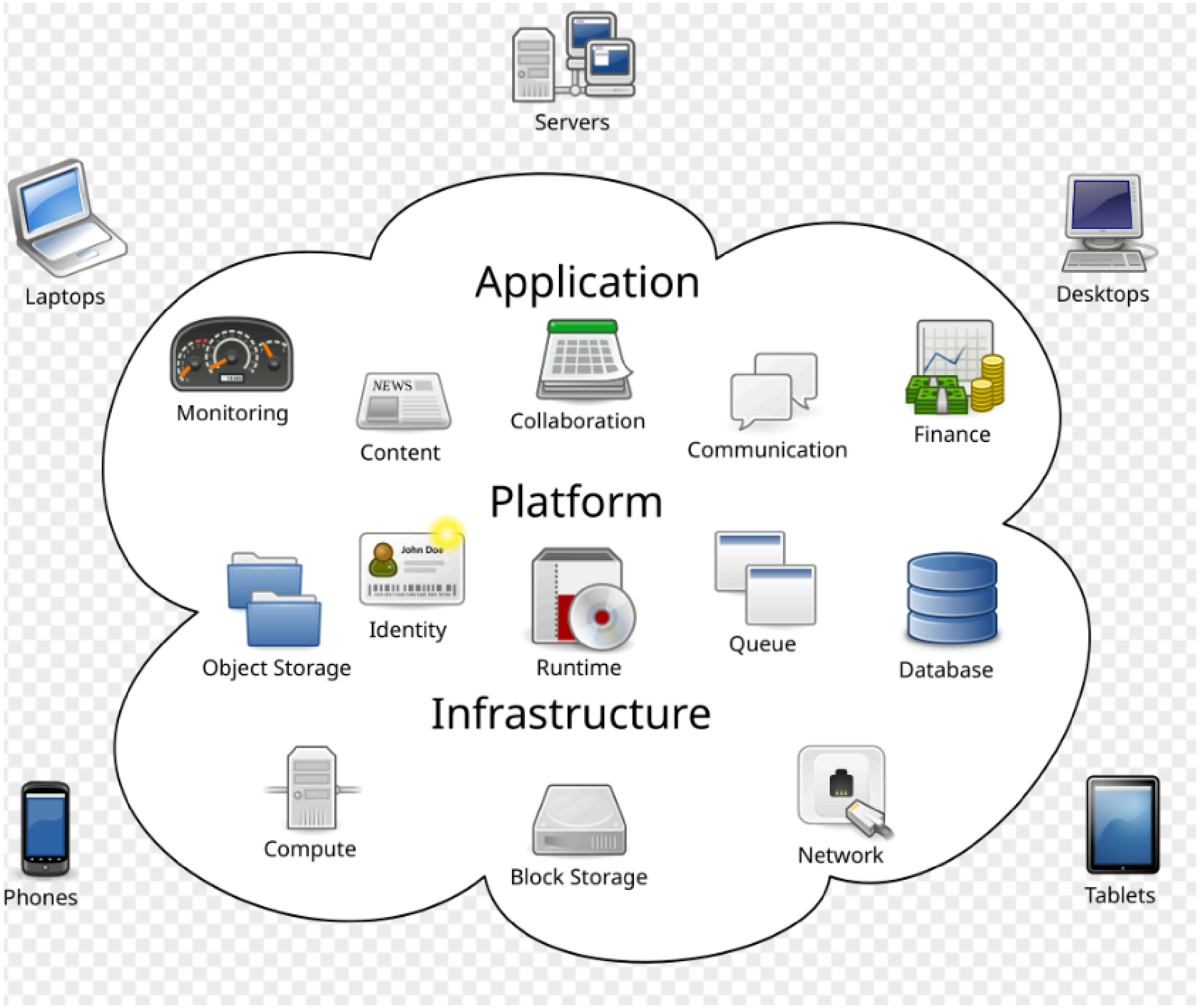


Fig: Cloud computing model

Degrees of Parallelism

1. **Bit-level parallelism (BLP):** Converts bit-serial processing to word-level processing gradually.
2. **Instruction-Level Parallelism (ILP):**

- The processor executes multiple instructions simultaneously rather than only one instruction at a time.
- ILP is executed through pipelining, super scalar computing, VLIW (very long instruction word) architectures, and multithreading.

- ILP requires branch prediction, dynamic scheduling, speculation, and compiler support to work efficiently.

3. **Data-level parallelism (DLP):**

- DLP 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.

4. **Task-level parallelism (TLP):**

- Ever since the introduction of multi core processors and chip multiprocessors (CMPs), we have been exploring TLP.
- TLP is far from being very successful due to difficulty in programming and compilation of code for efficient execution on multicore CMPs.

5. **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.
6. **The Internet of Things (IoT):**

- Traditional Internet connects machines to machines or web pages to web pages.
- IoT was introduced in 1999 at MIT
- Networked interconnection of everyday objects, tools, devices, or computers.
- A wireless network of sensors that interconnect all things in our daily life. Three communication patterns co-exist: namely H2H (human-to-human), H2T (human-to-thing),and T2T (thing-to-thing).
- Connect things (including human and machine objects) at any time and any place intelligently with low cost.
- IPv6 protocol, 2128 IP addresses are available to distinguish all the objects on Earth, including all computers and pervasive devices.
- IoT needs to be designed to track 100 trillion static or moving objects simultaneously.
- IoT demands universal addressability of all of the objects or things.
- The dynamic connections will grow exponentially into a new dynamic network of networks, called the Internet of Things (IoT).

7. **Cyber-Physical Systems:**

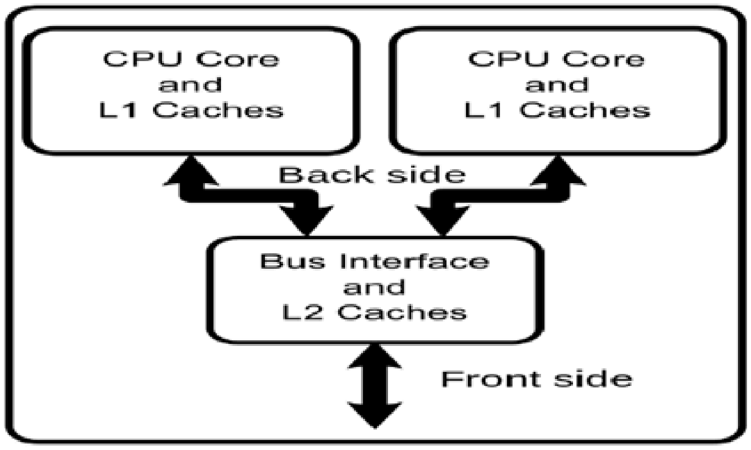
- A cyber-physical system (CPS) is the result of interaction between computational processes and the physical world.
- CPS integrates “cyber” (heterogeneous, asynchronous) with “physical” (concurrent and information-dense) objects.

Technologies for Network based systems:

1. **Multi-core CPUs and Multithreading Technologies:**

- CPUs today assume a multi-core architecture with dual, quad, six, or more processing cores.
- The clock rate increased from 10 MHz for Intel 286 to 4 GHz for Pentium 4 in 30 years.
- However, the clock rate reached its limit on CMOS chips due to power limitations. Clock speeds cannot continue to increase due to excessive heat generation and current leakage.

- LI cache is private to each core, L2 cache is shared and L3 cache or DRAM is off the chip. Examples of multi-core CPUs include Intel i7, Xeon, AMD Opteron. Each core can also be multithreaded. E.g., the Niagara II has 8 cores with each core handling 8 threads for a total of 64 threads maximum.



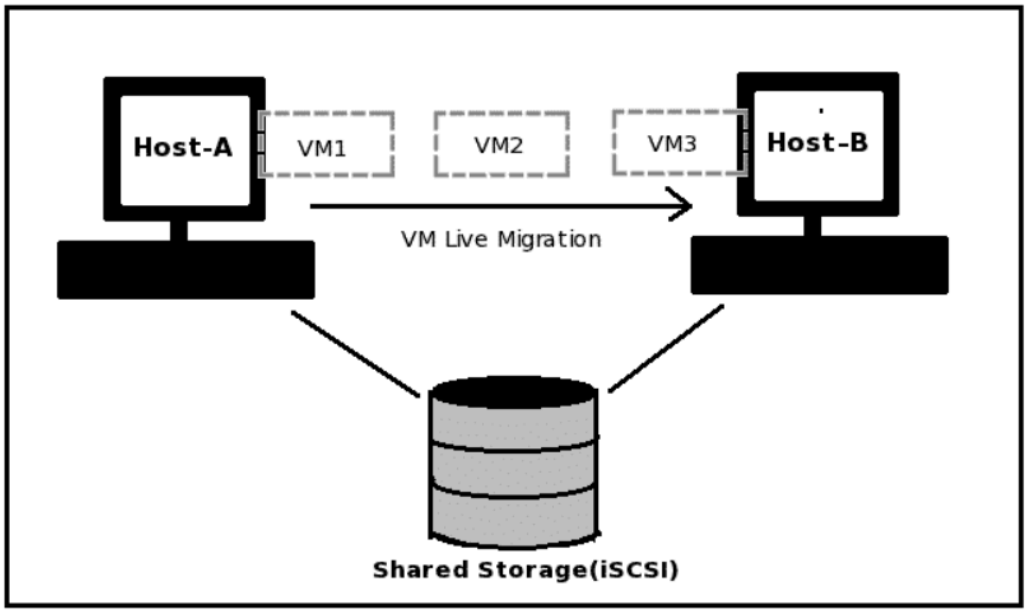
2. **Memory, Storage and WAN:**

- DRAM chip capacity increased from 16 KB in 1976 to 64 GB in 2011 for a 4x increase in capacity every 3 years. Memory access time did not improve as much.
- For hard drives, capacity increased from 260 MB in 1981 to 3 TB for the Seagate Barracuda XT hard drive in 2011 for an approximate 10x increase in capacity every 8 years.
- The "memory wall" is the growing disparity of speed between CPU and memory outside the CPU chip.
- An important reason for this disparity is the limited communication bandwidth beyond chip boundaries. From 1986 to 2000, CPU speed improved at an annual rate of 55% while memory speed only improved at 10%.
- Faster processor speed and larger memory capacity result in a wider performance gap between processors and memory. The memory wall may become an even worse problem limiting CPU performance.

System-Area Interconnects:

- A LAN is typically used to connect clients to big servers. A Storage Area Network (SAN) connects servers to network storage such as disk arrays.
- Network attached storage (NAS) connects servers directly to disk arrays. All 3 types of networks often appear in a large cluster built with commercial network components.
- A NAS is fundamentally a bunch of disks, usually arranged in a disk array. Users and servers attach to the NAS primarily using TCP/IP over Ethernet, and the NAS has its own IP address. The primary job of a NAS is to serve files, so most NAS systems offer support for Windows networking, HTTP, plus file systems and protocols such as NFS.

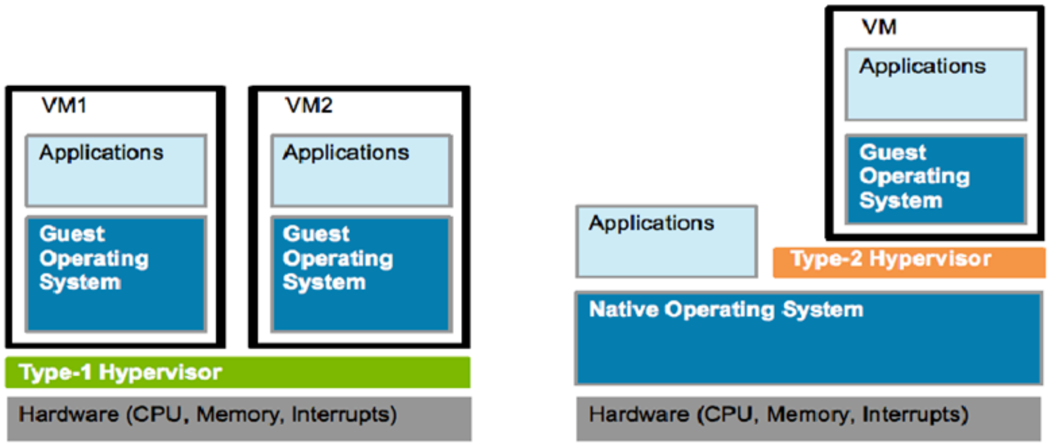
One way to loosely conceptualize the difference between a NAS and a SAN is that a NAS appears to the client OS (operating system) as a file server (the client can map network drives to shares on that server) whereas a disk available through a SAN still appears to the client OS as a disk, visible in disk and volume management utilities (along with client's local disks), and available to be formatted with a file system and mounted.
SANs allow multiple servers to share a RAID, making it appear to the server as if it were local or directly attached storage, and it cannot be accessed by individual users. A dedicated networking standard, Fiber Channel, allow blocks to be moved between servers and storage at high speed. It uses dedicated switches and a fibre-based cabling system which separates it from the day-to-day traffic. It uses the SCSI protocol for communication.



3. **Virtual Machines and Virtualization Middleware:** To build clouds we need to aggregate large amounts of computing, storage, and networking resources in a virtualized manner. Specifically, clouds rely on the dynamic virtualization of CPU, memory, and I/O.
   Virtual Machines (VMs):
- The VM is built with virtual resources managed by a guest OS to run a specific application.
- Between the VMs and the host platform, a middleware layer (called the Virtual Memory Monitor (VMM) or a hypervisor) is deployed.

Type 1 (bare metal) hypervisor:
- The bare metal hypervisor runs on the bare hardware and handles all the Hardware (CPU, memory, and I/O) directly. This runs in the privileged mode.
- The guest OS could be any OS such as Linux, Windows etc.

- They provide an almost native performance to the guest OSs (VMs), generally losing only 3–4% of the Central Processing Unit's cycles to the running of the hypervisor.
- Bare-metal is great for consolidating a company's collection of servers onto a single piece of hardware.



Some examples of the leading bare-metal hypervisors are VMware's ESX(i) (proprietary), Citrix XenServer (FOS (Free & Open Source)), and KVM (kernel loaded VM) (FOS). ESX and XenServer are installed that reside directly on the hardware. KVM sits within a Linux kernel.

4. **Full Virtualization vs. Paravirtualization:**

- Full Virtualization allows the guest OS to run on the hypervisor without any modification and without knowing that it is hosted.
- Paravirtualization requires that the kernel of the guest OS is modified and compiled with hooks (an API) for the hypervisor (guest OSs must know about the hypervisor).
- The guest OS can then communicate and cooperates with the hypervisor with a potential to improving performance, though this might be marginal (load that generates system calls benefits the most).
- Windows guests can only run-on Full Virtualization, as their source is proprietary.
- Operating systems that support paravirtualization interfaces need custom kernel adjustments.
- If compiled manually (where possible), guest operating systems with hypervisor support require more maintenance and configuration.
- These additional costs and complexity, combined with the marginal performance gains, means paravirtualization remains a niche product in the server virtualization market.

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.
- A massive system is with millions of computers connected to edge networks.

- Massive systems are considered highly scalable

Massive systems are classified into four groups: Clusters, P2P networks, computing grids, and Internet clouds.

1. **Computing cluster:** A computing cluster consists of interconnected stand-alone computers which work cooperatively as a single integrated computing resource.

Cluster Architecture:

- The architecture consists of a typical server cluster built around a low-latency, high bandwidth interconnection network.
- Build a larger cluster with more nodes, the interconnection network can be built with multiple levels of Gigabit Ethernet, Myrinet, or InfiniBand switches.
- Through hierarchical construction using a SAN, LAN, or WAN, one can build scalable clusters with an increasing number of nodes.
- Cluster is connected to the Internet via a virtual private network (VPN) gateway.
- Gateway IP address locates the cluster.
- Clusters have loosely coupled node computers.
- All resources of a server node are managed by their own OS.
- Most clusters have multiple system images as a result of having many autonomous nodes under different OS control oriented instead of server-oriented

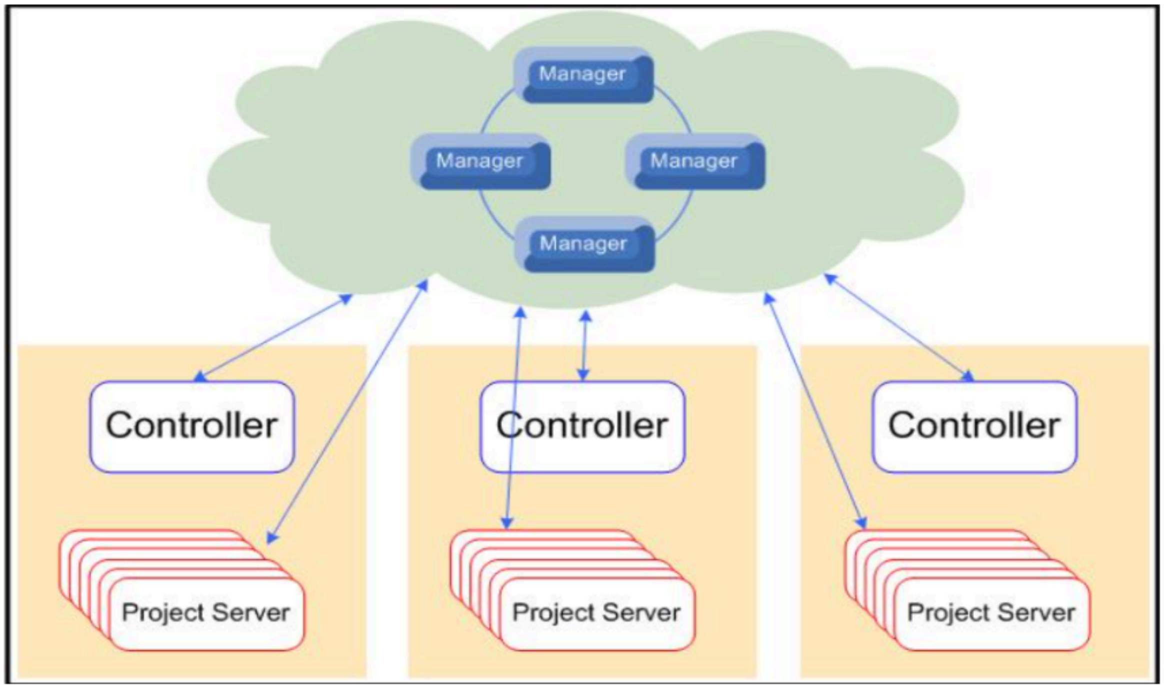


Fig: Cluster architecture

Single-System Image – Cluster:

- An ideal cluster should merge multiple system images into a single-system image (SSI)
- A cluster operating system or some middleware have to support SSI at various levels, including the sharing of CPUs, memory, and I/O across all cluster nodes.

- 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.

Hardware, Software, and Middleware Support –Cluster:

- Clusters exploring massive parallelism are commonly known as MPPs –Massive Parallel Processing.
- 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.
- Most clusters run under the Linux OS.
- Nodes are interconnected by a high-bandwidth network
- Special cluster middleware supports are needed to create SSI or high availability (HA).
- All distributed memory to be shared by all servers by forming distributed shared memory (DSM).
- SSI features are expensive
- Achieving SSI, many clusters are loosely coupled machines
- virtual clusters are created dynamically, upon user demand.

Grid Computing:

- A web service such as HTTP enables remote access of remote web pages Computing grid offers an infrastructure that couples computers, software/middleware, special instruments, and people and sensors together.
- Enterprises or organizations present grids as integrated computing resources. They can also be viewed as virtual platforms to support virtual organizations.
- The computers used in a grid are primarily workstations, servers, clusters, and supercomputers.

Peer-to-Peer Network-P2P:

What is a peer-to-peer network?

A peer-to-peer network is a group of computers or devices that share resources and access shared resources without centralized control. In the group, there is no central authority that defines access rules. All group members have equal rights. A member cannot control another member in any form.

How does a peer-to-peer network work?

In this model, first, a group is created. Any computer or device that wants to share a resource or wants to access a shared resource joins the group. A group password can be set to block unauthorized devices from joining the group.

If a group password is set, the group is known as the private group. A group password is only required to join the group. It is not required to leave the group. Private groups are mostly used in the home or small office networks. If a group password is not set, the group is known as the public group. A device can join or leave a public group anytime. Public groups are mostly used on the Internet for sharing files.

After joining the group, a member can share resources or access shared resources. A resource can be any shareable object such as a file, hard disk, printer, etc. When a member shares a resource, the resource becomes available for all group members. The member that shares a resource is called the server, and the member that accesses a shared resource is called the client. A member can be a server, a client, or both at a time.

For example, two PCs are connected through a peer-to-peer network. A printer is attached to the first PC. A modem is attached to the second PC. Both printer and modem are shared in the network. When the first PC accesses the Internet through the shared modem of the second PC, the first PC acts as the client while the second PC acts as the server. When the second PC accesses the shared printer of the first PC, the second PC acts as the client while the first PC acts as the server.

The following image shows this example.

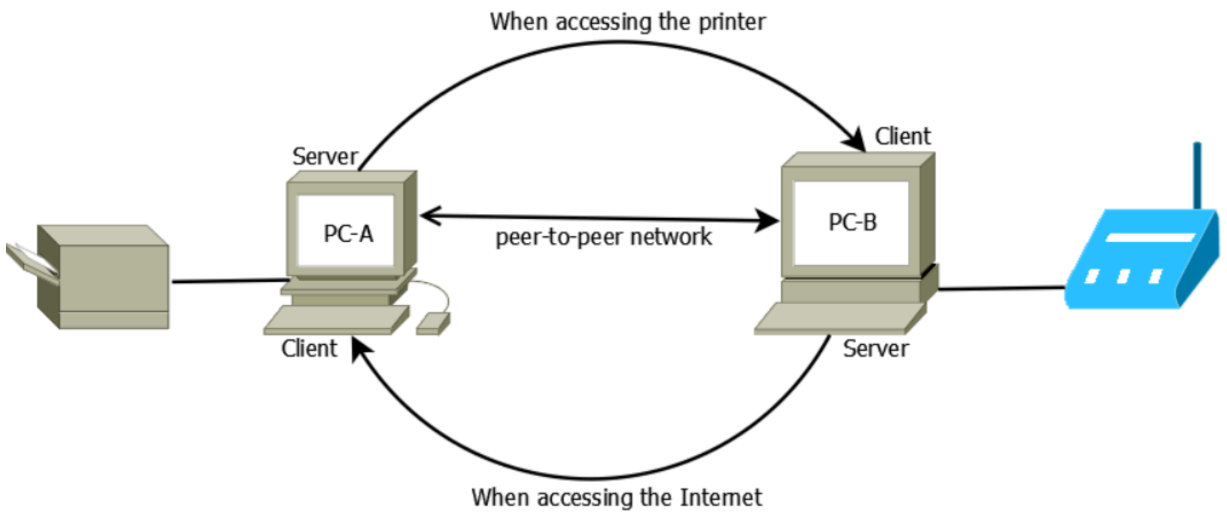


Fig: peer-to-peer example

To share a resource, a member has two options: simple sharing and advanced sharing. In simple sharing, the resource is shared with a single access rule. For example, a member can share a file with the read-only rule. In this situation, all group members can read the file.

In advanced sharing, a member may define some basic access rules for its shared resources. For this, the member creates different user accounts and defines access rules for each user account. Since user accounts and access rules both are created on the local system, the local system can check the rules before granting access to the file. If another member of the network wants to access this file, the member has to create and use the same user accounts with the same passwords.

Advantages of peer-to-peer networks

A peer-to-peer network is easier to set up. It does not require any special software or operating system. You can set up a peer-to-peer network by using any operating system. Almost all operating systems support peer-to-peer networking and include necessary components to connect an existing peer-to-peer group or to create a new peer-to-peer group.

Since a peer-to-peer network does not require additional components or applications, it does not increase the setup cost. A peer-to-peer network also does not require any special networking knowledge. With basic networking knowledge, anyone can easily set up a simple peer-to-peer network.

Disadvantages of peer-to-peer networks
Peer-to-peer networks provide only basic options for advanced sharing. To use these options, a lot of setups are required. For example, suppose you have 10 computers in your network and want to use this architecture to set up a folder where users can save files.

In this case, first, you have to create 10 local user accounts on the system that holds the shared folder and defines access rules for each user account. Then you have to create the same user accounts with the same passwords on all reaming computers. You have to create and manage 100 user accounts (10x10 = 100) for a simple setup.

The following image shows this example.

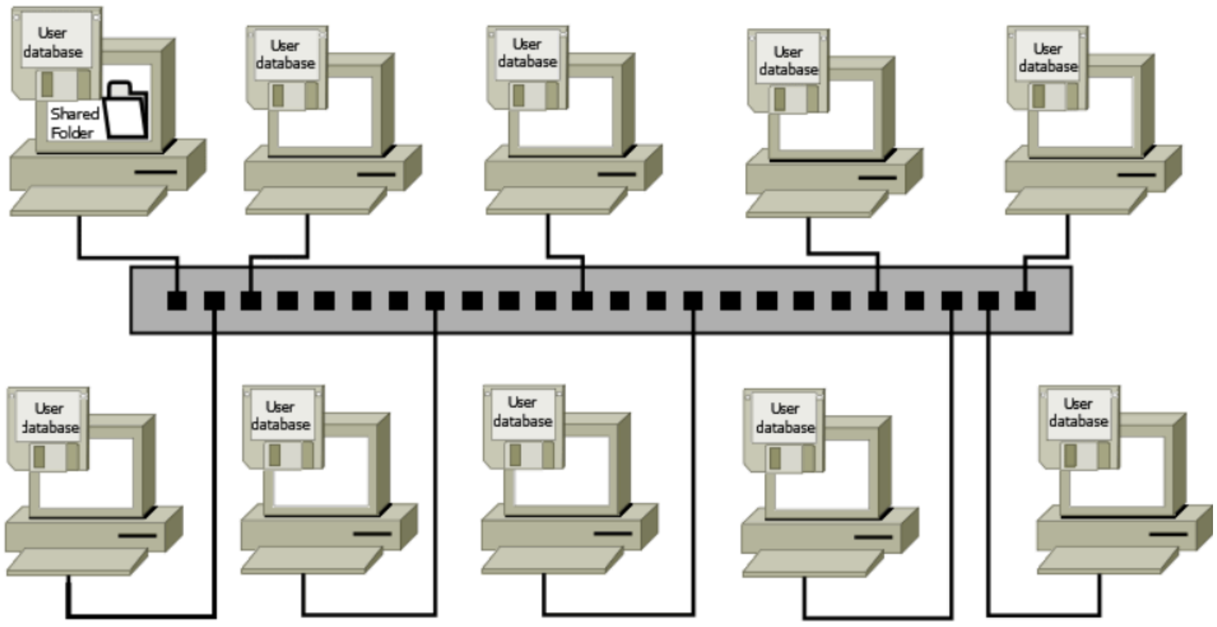


Fig: peer-to-peer network users database

Since usernames and passwords are stored locally, a user can change them without notifying you. If this happens, the user will not be able to connect to the shared folder. This can be an organizational nightmare.

Since peer-to-peer networks do not use dedicated network operating systems, the performance of these networks is less than client-server networks that use dedicated network operating systems or applications.

Peer-to-peer networks are also not considered secure networks. In a peer-to-peer network, shared resources can be easily discovered and used by unauthorized users.

In this type of network, since data is not stored in a centralized location, backing up critical data is quite a difficult task.

Where should peer-to-peer networking be used?
Peer-to-peer networking is useful in the following conditions.

- The number of computers or devices in the network is less than 15.
- Networking is mainly required for hardware sharing.
- Advanced sharing is not required.
- Additional networking features are not required.
- The administrator personally knows all users of the network.
- Data security is not the top priority.
- The above conditions are usually fulfilled in home and small office networks. Thus, peer-to-peer networking is mostly used in home and small office networks.

Peer-to-peer file sharing on the Internet
A peer-to-peer file sharing service is a service that allows users to exchange files over the Internet. There are a lot of applications that allow users to use this service. Since this service allows users to exchange files directly over the Internet, this service is mostly used to share and download copyright-protected movies, music tracks, and games. But it does not mean that this service is used only for illegal purposes. Several open-source projects allow users to download their applications through the peer-to-peer file sharing service. Downloading a file through the peer-to-peer file sharing service is easier and faster than downloading it directly.

Features of p2p network:

- P2P architecture offers a distributed model of networked systems.
- P2P network is client-oriented instead of server-oriented.
- In a P2P system, every node acts as both a client and a server.
- Peer machines are simply client computers connected to the Internet.
- All client machines act autonomously to join or leave the system freely.
- This implies that no master-slave relationship exists among the peers.
- No central coordination or central database is needed. The system is self-organizing with distributed control.
- Each peer machine joins or leaves the P2P network voluntarily.
- Only the participating peers form the physical network at any time.

- Physical network is simply an ad hoc network formed at various Internet domains randomly using the TCP/IP and NAI protocols.

Peer-to-Peer Network-Overlay network:

- 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.
- When a new peer joins the system, its peer ID is added as a node in the overlay network.
- When an existing peer leaves the system, its peer ID is removed from the overlay network automatically.
- 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.

Cloud Computing:

- A cloud is a pool of virtualized computer resources.
- A cloud can host a variety of different workloads, including batch-style backend jobs and interactive and user-facing applications."
- Cloud computing applies a virtualized platform with elastic resources on demand by provisioning hardware, software, and datasets dynamically.

The Cloud Landscape

Infrastructure as a Service (IaaS):

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

Platform as a Service (PaaS):

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

Software as a Service (SaaS):

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

SOFTWARE ENVIRONMENTS FOR DISTRIBUTED SYSTEMS AND CLOUDS

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.
- On top of this we have a base software environment, which would be
- .NET or Apache Axis for web services,
- the Java Virtual Machine for Java, and a broker network for CORBA.
- On top of this base environment, one would build a higher-level environment reflecting the special features of the distributed computing environment.
- SOA applies to building grids, clouds, grids of clouds, clouds of grids, clouds of clouds (also known as interclouds),
- SS (sensor service) : A large number of sensors provide data-collection services (ZigBee device, a Bluetooth device, Wi-Fi access point, a personal computer, a GPA, or a wireless Phone etc..,)
- Filter services : to eliminate unwanted raw data, in order to respond to specific requests from the web, the grid, or web services.

Layered Architecture for Web Services and Grids:

- Entity Interfaces
- Java method interfaces correspond to the Web Services Description Language (WSDL)
- CORBA interface - definition language (IDL) specifications
- These interfaces are linked with customized, high-level communication systems: SOAP, RMI, and IIOP.
- These communication systems support features including particular message patterns (such as Remote Procedure Call or RPC), fault recovery, and specialized routing.
- Communication systems are built on message-oriented middleware (enterprise bus) infrastructure such as Web-Sphere MQ or Java Message Service (JMS).

Web Services and Tools

REST approach:

- Delegates most of the difficult problems to application (implementation-specific) software. In a web services language.
- minimal information in the header, and the message body (that is opaque to generic message processing) carries all the needed information.
- Architectures are clearly more appropriate for rapid technology environments.
- REST can use XML schemas but not those that are part of SOAP; "XML over HTTP" is a popular design choice in this regard.
- Above the communication and management layers, we have the ability to compose new entities or distributed programs by integrating several entities together.

CORBA and Java:

- The distributed entities are linked with RPCs, and the simplest way to build composite applications is to view the entities as objects and use the traditional ways of linking them together.
- For Java, this could be as simple as writing a Java program with method calls replaced by Remote Method Invocation (RMI),
- CORBA supports a similar model with a syntax reflecting the C++ style of its entity (object) interfaces.

PERFORMANCE, SECURITY, AND ENERGY EFFICIENCY:

Performance Metrics:

- 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).
- System overhead is often attributed to OS boot time, compile time, I/O data rate, and the runtime support system used.
- Other performance-related metrics include the QoS for Internet and web services; System availability and dependability; and security resilience for system Défense against network attacks

Dimensions of Scalability: Any resource upgrade in a system should be backward compatible with existing hardware and software resources. System scaling can increase or decrease resources depending on many practical factors.

1. **Size scalability:**

- This refers to achieving higher performance or more functionality by increasing the machine size.
- The word "size" refers to adding processors, cache, memory, storage, or I/O channels. The most obvious way to determine size scalability is to simply count the number of processors installed.
- Not all parallel computer or distributed architectures are equally size scalable.

- For example, the IBM S2 was scaled up to 512 processors in 1997. But in 2008, The IBM Blue Gene/L system scaled up to 65,000 processors.

2. **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.

Some software upgrades may not work with large system configurations.

- Testing and fine-tuning of new software on larger systems is a nontrivial job.

3. **Application scalability:**

- This refers to matching problem size scalability with machine size scalability.
- Problem size affects the size of the data set or the workload increase.
- Instead of Increasing machine size, users can enlarge the problem size to enhance system efficiency or cost-effectiveness.

4. **Technology scalability:**

- This refers to a system that can adapt to changes in building technologies, such as the component and networking technologies
- When scaling a system design with new technology one must consider three aspects: time, space, and heterogeneity.
- Time refers to generation scalability. When changing to new-generation processors, one must consider the impact to the motherboard, power supply, packaging and cooling, and so forth. Based on past experience, most systems upgrade their commodity processors every three to five years.
- Space is related to packaging and energy concerns. Technology scalability demands harmony and portability among suppliers.
- Heterogeneity refers to the use of hardware components or software packages from different vendors. Heterogeneity may limit the scalability.

Amdahl's Law: Take a program that processes a known quantity of input data. Programs like this often have a sequential set of statements that are difficult to parallelize and a part that can be parallelized.Let P be the portion of the code that can be parallelized. If we throw N computing units at the problem instead of a single thread, the speedup factor S that we can expect is below.

$$S = 1 / (1 - P) + (P/N)$$

Gustafson's Law:

- To achieve higher efficiency when using a large cluster, we must consider scaling the problem size to match the cluster capability. This leads to the following speedup law proposed by John Gustafson(1988), referred to as scaled-workload speedup.
- Let W be the workload in a given program.

- When using an n-processor system, the user scales the workload to $W' = \alpha W + (1 - \alpha)nW$. 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:

$$s' = w'/w = [\alpha w + (1 - \alpha)nw]/w = \alpha + (1 - \alpha)n$$

Network Threats and Data Integrity:

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 performance reliability (e.g., fault tolerance and security).However, these systems recently encountered new challenging issues including energy efficiency, and workload and resource outsourcing

Energy Consumption of Unused Servers:

- To run a server farm (data center) a company has to spend a huge amount of money 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:
- In addition to identifying unused/under-utilized servers for energy savings, it is also necessary to apply appropriate techniques to decrease energy consumption in active distributed systems with negligible influence on their performance.

Application Layer: Until now, most user applications in science, business, engineering, and financial areas tend to increase a system's speed or quality. By introducing energy-aware applications, the challenge is to design sophisticated multilevel and multi-domain energy management applications without hurting performance.

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 analyser, task scheduler, security access, reliability control, and information service capabilities. It is also responsible for applying energy-efficient techniques, particularly in task scheduling.

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, itis responsible for controlling all distributed resources in distributed computing systems. Dynamic power management (DPM) and dynamic voltage-frequency scaling (DVFS) are two popular methods incorporated into recent computer hardware systems. In DPM, hardware devices, such as the CPU, have the capability to switch from idle mode to one or more lower power modes. In DVFS, energy savings are achieved based on the fact that the power consumption in CMOS circuits has a direct relationship with frequency and the square of the voltage supply.

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.